Laila Dybkjær
Wolfgang Minker
Editors

TEXT, SPEECH AND LANGUAGE TECHNOLOGY SERIES 39

Recent Trends in Discourse and Dialogue

Springer

Recent Trends in Discourse and Dialogue

Text, Speech and Language Technology

VOLUME 39

The titles published in this series are listed on www.springer.com.

Recent Trends in Discourse and Dialogue

Edited by

Laila Dybkjær

Natural Interactive Systems Laboratory
Brøndby
Denmark

Wolfgang Minker

Institute of Information Technology
University of Ulm
Germany

Springer

Library of Congress Control Number: 2008920669

ISBN 978-1-4020-6820-1 (HB)
ISBN 978-1-4020-6821-8 (e-book)

Published by Springer,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

www.springer.com

Printed on acid-free paper

Contents

Preface

This book edition highlights recent trends and important issues that still remain only partially solved or even unsolved within the broad field of discourse and dialogue. The field is discussed and illustrated both from an overall spoken (multimodal) dialogue system perspective as well as from a more component-related perspective. Issues discussed include, for example, discourse and dialogue modelling in research versus industrial spoken dialogue systems, evaluation, miscommunication and error handling, grounding, statistical and corpus-based approaches to discourse and dialogue modelling, data analysis, and corpus annotation and annotation tools.

We believe that jointly this collection of chapters provides a good picture of how far we are today within discourse and dialogue and of important challenges ahead. On this background we hope that computer scientists, engineers, and others who work in the broad area of discourse and dialogue, no matter if from an academic or industrial perspective, may benefit from the book and find it useful to their own work. Graduate students and Ph.D. students focusing on topics in discourse and dialogue may also find the book interesting and profit from reading it.

This book edition is based on a selected subset of papers from the successful 6th SIGdial Workshop on Discourse and Dialogue held in Lisbon, Portugal, in September 2005 in conjunction with the 9th Eurospeech (Interspeech) conference. SIGdial is a special interest group on discourse and dialogue sponsored jointly by the two parent organisations ACL (Association for Computational Linguistics) and ISCA (International Speech Communication Association).

The annual two-day SIGdial workshops normally attract a considerable amount of papers. The workshop in 2005 attracted 80 submissions which is the highest submission rate to date for the SIGdial workshop series. Twenty-eight papers were accepted for the workshop and of these 10 were invited for publication in this book along with a paper by an invited speaker, i.e. a total of 11 papers.

All workshop papers were extended and revised before they were submitted as book chapters. Each chapter has subsequently been reviewed by two external reviewers and further improved on the basis of their comments.

We would like to thank all those who contributed to and helped us in preparing the book. In particular we would like to express our gratitude to the following external reviewers for their valuable comments and criticism on the submitted drafts of the book chapters: Jan Alexandersson, Niels Ole Bernsen, André Berton, Rolf Carlson, Mark Core, Els den Os, Hans Dybkjær, Silke Goronzy, Joakim Gustafson, Ludwig Hitzenberger, Masato Ishizaki, Lars Bo Larsen, Esther Levin, Johanna Moore, Tim Paek, Patrick Paroubek, Norbert Reithinger, Laurent Romary, Candace Sidner, Ronnie Smith, Marc Swerts, and Nick Webb. We are also grateful to colleagues at the Institute of Information Technology in Ulm and at NISLab in Odense for their support in editing the book.

Laila DYBKJÆR

Wolfgang MINKER

Contributing Authors

Dan Bohus is currently a Researcher in the Adaptive Systems and Interaction group at Microsoft Research. He has recently obtained his Ph.D. degree in Computer Science from Carnegie Mellon University (2007). He is interested in developing techniques that allow systems to act robustly in the real world, and to acquire knowledge online and learn from their own experiences (without explicit developer supervision). His other interests include: mixed-initiative interaction, grounding, multimodal systems, and multi-participant dialogue. In previous work, Dan Bohus has developed RavenClaw, an open-source, task-independent dialogue management framework deployed in a number of systems spanning multiple domains and interaction types.

Matthias Denecke is a Research Associate at the NTT Communication Science Laboratories, NTT Corp, Japan. He received his doctorate degree from Karlsruhe University, Germany, in 2002 on Generic Interaction Patterns for Task Oriented Dialogue Systems. His research interests include statistical methods for dialogue systems, rapid prototyping of natural language applications, and natural language processing in general.

Kate Forbes-Riley is a Research Associate at the Learning Research and Development Center, University of Pittsburgh, USA, and a Lecturer in the Computer Science Department at Dartmouth College, USA. She obtained a Ph.D. in Computational Linguistics in 2003, and an MSE in Computer and Information Science in 2001, at the University of Pennsylvania. Her current research concerns emotion prediction and adaptation in a spoken dialogue tutorial system, and her research interests span the study of discourse and dialogue.

Jonathan Gratch (http://www.ict.usc.edu/∼gratch) is a Research Associate Professor of Computer Science at the University of Southern California (USC), Project Leader at the Institute for Creative Technologies and co-director of USC's Computational Emotion Group. He completed his Ph.D. in Computer Science at the University of Illinois in Urban-Champaign in 1995 with a focus on machine learning, planning and cognitive science. His research addresses the creation of virtual humans (artificially intelligent agents embodied in a human-like graphical body) and cognitive modelling. He studies the

relationship between cognition and emotion, the cognitive processes underlying emotional responses, and the influence of emotion on decision making and physical behaviour. He has worked on a number of applications of virtual agents, including considerable experience in the research and development of automated and semi-automated agents in training. He is a member of the American Association for Artificial Intelligence (AAAI) and the International Society for Research on Emotion (ISRE). Dr. Gratch is the author of over 100 technical articles.

Alex Gruenstein is a Ph.D. student in the Computer Science and Artificial Intelligence Laboratory at the Massachusetts Institute of Technology, USA. He received BS and MS degrees in Symbolic Systems from Stanford University. His research interests include: modelling multimodal dialogue context to improve fragment interpretation and language model accuracy, automatically segmenting spoken dialogue, and developing general tools for rapid development of dialogue systems.

Stefan W. Hamerich is working on dialogue research and tools for the speech dialogue systems division of Harman/Becker. He received a master's degree in Computer Science (speech and language oriented AI) from the University of Hamburg, Germany in 2001. In 1999 and 2000 he was a working student at the IBM European Speech Research group in Heidelberg, Germany. In his sparetime he currently works on his Ph.D. thesis dealing with user-friendly speech dialogue systems for complex domains in automotive environments.

Juan M. Huerta is a Researcher at the IBM T. J. Watson Research Center, in Yorktown Heights New York, USA, His research interests comprise dialogue management, techniques for rapid design and development of spoken dialogue systems, robust natural language understanding, and contact center automation and optimization. He received his Ph.D. from Carnegie Mellon University in 2000, a MS from Boston University in 1993, and a BS from ITESM, Mexico in 1992, all in Electrical Engineering. During 1994 and 1995 he was a research technical specialist at Dragon Systems, in Newton MA.

Diane Litman is both Professor of Computer Science, and Research Scientist with the Learning Research and Development Center, at the University of Pittsburgh. Previously, she was a member of the Artificial Intelligence Principles Research Department, AT&T Labs – Research (formerly Bell Laboratories), and an Assistant Professor of Computer Science at Columbia University. Her research spans many areas of spoken dialogue systems, including: prosodic analysis of misrecognitions, corrections, and affect; dialogue tutorial systems; system adaptation and optimization via reinforcement learning and supervised machine learning; and empirical evaluation.

Stacy Marsella (http://www.isi.edu/∼marsella) is a Project Leader at the University of Southern California's Information Sciences Institute (USC/ISI) and a Research Associate Professor of Computer Science at the University of Southern California. Dr. Marsella received his Ph.D. from Rutgers University in 1993. His research interests include multi-agent systems, computational models of emotion, modelling social interaction and group behaviour as well as the use of simulation in education. Dr. Marsella has developed numerous agent-based systems across a range of applications, including military training applications, exploratory social simulations for modelling psychological operations, language training and health interventions. Dr. Marsella has numerous publications spanning research in artificial intelligence, cognitive science, psychology, political science and the arts.

Michael McTear is Professor of Knowledge Engineering at the University of Ulster, Northern Ireland. His main research interests are spoken dialogue systems, natural language processing, user modelling, and language acquisition. He has published several books and journal papers in these areas. His most recent book Spoken Dialogue Technology: Toward the Conversational User Interface was published by Springer Verlag in 2004.

Sebastian Möller is Professor for Usability and Fellow at Deutsche Telekom Laboratories, Berlin University of Technology, Germany. He received his Ph.D. (Dr.-Ing.) from Ruhr-Universität Bochum in 1999 for his work on the assessment and prediction of speech quality in telecommunications, and got the qualification to be a professor (venia legendi) at that university in 2004, with a book on the quality of telephone-based spoken dialogue systems. His main research interests are speech processing, speech technology, communication acoustics, quality assessment, and the usability of telecommunication services. Since 1997, he has taken part in the standardization activities of the International Telecommunication Union (ITU-T).

John Niekrasz is a Research Engineer at the Computational Semantics Laboratory at the Center for the Study of Language and Information, Stanford University. He is a former student of the Stanford University Symbolic Systems Program and subsequently the Centre for Speech Technology Research at the University of Edinburgh, and he has been working in the field of natural language technology since 1998. His research interests include methods for automatic understanding and annotation of multiparty spoken discourse, with a focus on statistical methods that are robust to large amounts of uncertainty.

Roberto Pieraccini has worked for more than 20 years on the research, technological, and commercial aspects of spoken language human–machine communication. He was a researcher at CSELT (now Loquendo, Turin, Italy),

Bell Laboratories (Murray Hill, NJ), and AT&T Labs (Florham Park, NJ). He led the Natural Dialog group at SpeechWorks International and the Conversational Systems Technology department at IBM T. J. Watson Research. He is now Chief Technology Officer at SpeechCycle Inc.

Pascal Poupart is an Assistant Professor in the David R. Cheriton School of Computer Science at the University of Waterloo in Waterloo, Canada. He received a Ph.D. in Computer Science (2005) from the University of Toronto in Toronto, Canada, an MSc in Computer Science (2000) from the University of British Columbia in Vancouver, Canada and a BSc in Mathematics and Computer Science (1998) from McGill University in Montreal, Canada. His research focuses on the development of decision theoretic planning and machine learning techniques for autonomous and adaptive systems in various domains including spoken dialogue systems, language processing, assistive technologies and health informatics.

Matthew Purver is a Research Engineer in the Computational Semantics and Dialogue Systems Laboratory at the Center for the Study of Language and Information, Stanford University. He received his Ph.D. in computational linguistics from the University of London in 2004. His research interests include theoretical and practical aspects of dialogue processing, including spoken dialogue systems, context modelling, repair and clarification, and the automatic understanding of multi-party discourse.

Alex Rudnicky is currently a Principal Systems Scientist in the Computer Science Department at Carnegie Mellon University and on the faculty of the Language Technologies Institute. His research spans many areas of spoken language processing, including contributions to dialogue management, language generation and recovery from misunderstanding. He is a recipient of the Allen Newell Award for Research Excellence. He currently serves on the boards of the Applied Voice Input-Output Society (AVIOS) and of SIGdial.

Gabriel Skantze is a Ph.D. student at the Department for Speech Music and Hearing and the Centre for Speech Technology, KTH, Sweden. He received an MA in Cognitive Science from Linköping University in 2000. His research is focussed on error handling and robustness in conversational systems, including design, implementation, and user studies.

William Swartout is Director of Technology for USC's Institute for Creative Technologies (ICT) and a research professor of computer science at USC. He received his Ph.D. and MSc in Computer Science from MIT and his bachelor's degree from Stanford University. Dr. Swartout's particular research interests include virtual humans, explanation and text generation, knowledge acquisition, knowledge representation, knowledge sharing, education, intelligent

agents and the development of new AI architectures. Dr. Swartout is a Fellow of the American Association for Artificial Intelligence (AAAI), has served on the Board of Councilors of the AAAI and is past chair of SIGART.

David Traum (http://www.ict.usc.edu/∼traum) is a Research Scientist at ICT and a Research Assistant Professor of Computer Science at the University of Southern California. He completed his Ph.D. in Computer Science at the University of Rochester in 1994. His research focuses on collaboration and dialogue communication between agents, including both human and artificial agents. Of primary interest is the interaction between the individual cognitive functioning and the social fabric, and the relationship between task-related and communicative actions. He has engaged in theoretical, implementational, and empirical approaches to the problem, studying human–human natural language and multimodal dialogue, as well as building a number of dialogue systems to communicate with human users. Dr. Traum is author of over 100 technical articles, has served on many conference program committees, and is currently the president of SIGDIAL, the international special interest group in discourse and dialogue.

Yu-Fang H. Wang is a Member of the Dialogue Research Group within Harman/Becker Automotive system's speech division. She received her master's degree in computational linguistics from the Saarland University in 1997. Having done her thesis on Chinese natural language generation, she continued the project as a visiting researcher at Jiaotong University in Shanghai. She then joined Philips Speech Processing in Aachen, Germany, to gain experience in spoken dialogue and speech recognition. Her areas of interest are man-machine usability issues such as user-friendly dialogues for multimodal interfaces, as well as the interaction of dialogue and recognition, for example employability of confidence values for dialogue development.

Jason D. Williams is a Senior Member of Technical Staff at AT&T Labs – Research in Florham Park, New Jersey, USA. He received a BSE in Electrical Engineering from Princeton University in 1998, and at Cambridge University he received an M Phil in Computer Speech and Language Processing in 1999 under a Churchill Scholarship and a Ph.D. in Information Engineering in 2006 under a Gates Scholarship. His main research interests are dialogue management, the design of spoken language systems, and planning under uncertainty. He has previously held positions at Tellme Networks, Edify Corporation (now Intervoice), and McKinsey & Company's Business Technology Office.

Norihito Yasuda is a Researcher at the NTT Communication Science Laboratories, NTT Corp, Japan. He received his master's degree from Kyoto University, Graduate School of Human and Environmental Studies in 1999.

His research interest is in spoken dialogue systems and natural language processing.

Steve Young is Professor of Information Engineering at Cambridge University and Head of the Information Engineering Division. His main research interests lie in the area of spoken language systems including speech recognition, speech synthesis and dialogue management. He was Editor of Computer Speech and Language from 1993 to 2004. He is a Fellow of the Royal Academy of Engineering, the Institution of Electrical Engineers (IEE) and the RSA. He is a member of the British Computer Society (BCS) and the US Institute of Electrical and Electronics Engineers (IEEE). In 2004, he was a recipient of an IEEE Signal Processing Society Technical Achievement Award.

Trends and Challenges in Discourse and Dialogue

Laila Dybkjær
Natural Interactive Systems Laboratory
Denmark
laila@nis.sdu.dk

Wolfgang Minker
Institute of Information Technology
University of Ulm, Germany
wolfgang.minker@uni-ulm.de

Knowledge on discourse and dialogue empowers the development of increasingly advanced dialogue managers and spoken dialogue systems, be they unimodal or multimodal, for use in stationary or mobile environments. During the last 15–20 years significant progress has been made in the technologies which form part of spoken (multimodal) dialogue systems, including dialogue or interaction management. Commercial dialogue systems in the early and mid-1990s were basically telephone-based and had very small vocabularies and simple dialogue management. Research systems were also mostly unimodal and included, e.g., train timetable information systems with larger vocabularies and more complex dialogue management. Systems like these matured and have now been in commercial use for several years.

Research systems moved on to address new challenges by including, e.g., other modalities in addition to speech, looking at applications for use in mobile environments, such as in-car systems, and, quite recently, trying to move beyond purely task-oriented dialogue. As part of this development process, technologies already in use have continuously been improved and new technologies have been taken up to enable increasingly powerful and diverse applications.

Nevertheless, the achievement of the ultimate goal – i.e., full natural interactivity where we communicate with systems in the same way we communicate with humans – is still far ahead. Many problems addressed already still remain unsolved or only partially solved. There are, of course, also many problems which eventually have been solved, thus contributing to the technology advances that enable the construction of increasingly sophisticated systems. This again leads to a demand for finding solutions to new problems in order for

this development to proceed. Actually new problems tend to appear at an even faster pace than old problems are solved due to the ever-increasing system complexity and diversity.

In the following we discuss current trends in discourse and dialogue in the light of important problems that are being addressed today and in relation to the chapters in this book (Section 1). As part of the discussion we also point to challenges ahead. In Section 2 we provide a brief overview of each of the eleven chapters that are included. Finally, Section 3 addresses readership.

1. Trends and Challenges

Based on the chapters in this book and our own experience we now present a number of important issues which, we believe, provide a good picture of current trends in discourse and dialogue and some of the challenges we are faced with.

1.1 System Variability and the Need for Discourse and Dialogue Modelling

The requirements to discourse and dialogue modelling vary across applications. It is therefore no surprise that most chapters in this book discuss discourse and dialogue in the context of some particular application. Chapter 2 by Wang and Hamerich concerns the development of a commercial, in-car, speech-controlled MP3 player. Chapter 3 by Traum, Swartout, Gratch, and Marsella deals with advanced discourse and dialogue modelling for a multimodal negotiation trainer in a military environment. In an evaluation context Chapter 4 by Möller addresses spoken dialogue for controlling domestic devices, such as lamps, video recorder, and answering machine. Chapter 6 by Bohus and Rudnicky investigates non-understanding recovery strategies in a phone-based conference room reservation system. In Chapter 7 by Skantze discourse and dialogue modelling issues, e.g., grounding and miscommunication, for a speech-based, pedestrian city navigation and guidance system are discussed. Chapter 8 by Williams, Poupart, and Young has its focus on statistical approaches to improving dialogue management using air ticket purchase as an example application. Chapter 9 by Denecke and Yasuda deals with machine learning to decide if users of restricted domain question-answering systems should be prompted for more information. The example system used can provide information about various aspects of travel destinations in Japan. Chapter 10 by Gruenstein, Niekrasz, and Purver discusses multiparty meeting discourse data. The kind of application aimed at is a system that can understand and process multiparty meetings and, e.g., summarise them and draw attention to particular issues. Chapter 11 by Forbes-Riley and Litman, finally, explores cognitive aspects related to the development of tutoring systems.

Some of these systems are in themselves not new but they serve as test beds to explore issues which still challenge us, e.g., aspects of evaluation, miscommunication and how to recover from it, or some kind of improvement of dialogue management. Other systems are very interesting as applications, such as the negotiation trainer and the tutoring system. Indeed these system are also used to explore the modelling of issues like social behaviour or student certainty. However, at the same time they take a step towards going beyond task-orientation.

As it appears from the above and as we shall also discuss in the following subsections, there are still challenges related to what we – at least from a research point of view – may consider relatively simple applications. The amount of new challenges increases dramatically the more – in all respects – diverse systems we want to develop. There seems to be a trend towards building more and more human-like systems that can act as tutors, companions, friends, secretaries or whatever. Successful construction of such systems poses huge demands on not least discourse and dialogue modelling.

1.2 Interaction Design

Interaction design is an issue that continues to challenge us when building spoken (multimodal) dialogue systems. The major reason for this is that users cannot just interact with a system in whichever way they want and expect to be understood. Today's systems are still far from being able to handle fully spontaneous conversational dialogue.

The solution to this problem is in a sense straightforward, i.e., we have to work within the existing limitations while at the same time – at least on the research side – we continue to push the borders for what the systems can do.

Chapter 1 by Pieraccini and Huerta discusses some of the differences in dialogue model design and interaction design between commercial and research applications, respectively. Commercial applications normally cover well-defined and limited task domains. Within the chosen task domain a commercial system tries to cater for every possible kind of input, including error situations, to ensure completeness and robustness. Full natural language interaction is not required for the user to feel that s/he has a natural dialogue with such a system. However, to induce this feeling in users, it is necessary to carefully craft and control the user–system interaction via the system's output so that the user naturally stays within the limits of the system. This again requires knowledge on what users perceive as natural interaction within the chosen task domain. Chapter 2 by Wang and Hamerich addresses this issue in the context of a commercial, in-car MP3 player.

While commercial applications strive to be complete within their delimited task domains, research systems usually have other goals, such as exploring or

testing new techniques, or serving to show that something is feasible, or that one approach works better than another. The negotiation trainer presented in Chapter 3 by Traum, Swartout, Gratch, and Marsella is a good example of a research system which, on the one hand, is very limited because it is far from trying to handle its entire task domain completely. On the other hand, the handling of the selected training scenario requires knowledge and techniques that we are only at the beginning to explore.

Skilful interaction design has always been a challenge and remains so, among other things because users tend to behave very differently and sometimes quite unpredictably. It is therefore hard to predict how they would like to interact with a new application. Regarding new challenges, Chapter 3 illustrates some of those we have to face, the more human-like interactions we want to model. Such new challenges include, among others, modelling of social behaviour (Chapter 3), cognitive parameters like certainty (Chapter 11), and new aspects of dialogue strategies (Chapter 3).

1.3 Evaluation

Evaluation is an old issue but has moved more and more into focus over the years. In particular evaluation of usability aspects is being researched today and continues to challenge us.

Evaluation may be quantitative or qualitative, and objective or subjective. Quantitative and objective measures are preferred because they are much easier to interpret and compare than, in particular, subjective, qualitative data. Nevertheless, subjective, qualitative data remains crucial since there is no way in which to safely predict how user's will receive a system based on quantitative or qualitative, objective data.

Several models have been developed to predict usability of unimodal, task-oriented, spoken dialogue systems based on some set of parameters (Chapter 4) but none of them seem to produce results which strongly correlate with results from subjective evaluation data. Thus, it is perhaps characteristic that the recommendations provided by the International Telecommunication Union (ITU) and presented in Chapter 4 contain both subjective evaluation methods for telephone-based spoken dialogue services as well as a set of interaction parameters which address system performance and which can be objectively measured. As the investigation presented in Chapter 4 by Möller shows, there is only a weak correlation between the parameters extracted from log files from user–system interactions and the subjective data obtained by asking users. Both kinds of data provide a separate contribution to the evaluation of a system.

Also the more sophisticated PARADISE model is far from providing results that can replace subjective user data (Chapter 4). A major problem seems to be that there are very many parameters which may affect user satisfaction and

how users perform with, and perceive, a system. Precisely which they all are and with which weight they contribute in a given system context is unknown. This means that so far we do not know exactly what makes users happy. In all cases users' interaction with the (possibly simulated) system continues to be crucial to collect objective as well as subjective data to evaluate system usability. This trend is illustrated in Chapters 2, 3, 7 and 11. In these chapters data are collected with users for the evaluation of an in-car MP3 player, a negotiation trainer, a system for pedestrian city navigation and guidance, and a tutoring system, respectively.

There are still gaps in our knowledge on usability evaluation of unimodal, task-oriented systems but many more new challenges are appearing these years. The increasing sophistication of spoken (multimodal) dialogue systems, the use of different modalities, and the development of mobile applications and of non-task-oriented systems are all creating new evaluation challenges. Open questions include, e.g., which are the relevant parameters and questions to ask users when we evaluate non-task-oriented systems, how do we figure out which preferences and priorities users have (just asking them may not be enough) and how do we evaluate new systems and system types in the light of these, and how do we evaluate the use of emotions and cognitive parameters? Moreover, we need more knowledge on long-term effects which again requires evaluation.

1.4 Miscommunication

Handling of miscommunication is not a new issue but it remains a challenge. It is well known that miscommunication should be avoided if at all possible. However, no matter the precautions we take in the design of a dialogue system, miscommunication will inevitably occur from time to time, both in terms of non-understandings, misunderstandings, and misinterpretations.

Most work on miscommunication so far has focused on speech recognition accuracy, cf. Chapter 5 by McTear. Although speech recognition is a fairly frequent source of miscommunication, it is not the only one. For example, out-of-grammar formulations or requests for non-existing functionality are other sources. However, no matter what caused the problem, it is in all cases the task of the dialogue manager to handle miscommunication in a reasonable way which is not always straightforward to do.

Non-understandings are always detected immediately, contrary to misunderstandings and misinterpretations which are sometimes only detected much later in a dialogue, or not at all (Chapter 5). There are many ways in which to try to recover from a non-understanding error, e.g., by just repeating output, providing help on what the user can say, or moving on in the dialogue and trying to get the missing information later. However, not all ways are equally

good in all situations as discussed by Bohus and Rudnicky in Chapter 6 that reports on an empirical study of non-understanding errors in the context of a conference room reservation system, and on ten different recovery strategies.

Feedback supports immediate detection of misunderstandings and misinterpretations. Depending on what caused the misunderstanding, immediate correction may be more or less difficult to handle. Sometimes a simple rephrasing may suffice. Sometimes, if the user, e.g., addressed some out-of-domain issue which is then being misunderstood rather than not understood, successful correction is much more difficult. If a misunderstanding or misinterpretation is only detected much later in a dialogue, correction becomes a more complex action. Depending on how the correction is phrased, it may be difficult for the system to figure out what should be corrected. Moreover, not only must the new piece of information provided by the user replace the relevant old one. It also needs to be checked if information has already been obtained that is dependent on the corrected information. For instance, if a person changes the destination when trying to book a flight ticket, date and time may also have to be changed.

If possible, commercial systems tend to connect to a human operator if the system is not able to recover from a miscommunication after three trials. If this possibility does not exist, the system may, e.g., refer to a website for further information or it will have to deal with the problem on its own (Chapter 2) like most research systems do. For good reasons research systems normally do not use an operator solution. Rather, they provide, e.g., suggestions on what the user could try to do, such as talk about something different, or the system takes the initiative itself and starts talking about something else (Chapter 3).

The previous paragraphs give an impression of how far we are today in handling miscommunication but also of where the challenges are. In fact part of the solution to miscommunication problems may lie in improved interaction design to avoid – better than today – that users are led to believe that the system has more or other functionality than is actually the case. Still, miscommunication situations will not entirely disappear. To become better at handling these, we need, among other things, further empirical investigations in broad domain contexts of miscommunication recovery strategies. Moreover, as suggested in Chapter 5, the spoken dialogue community may to a larger extent than so far, begin to draw on contributions from disciplines, such as social psychology, conversation analysis, and sociolinguistics. The more we try to model humans and also go beyond task-oriented dialogue, the more relevant such contributions are likely to become. Furthermore, with new modalities involved, the handling of miscommunication not only caused by speech but also by other sources, such as facial expression or the combination of speech and gesture, is a challenge that must be handled.

1.5 Interaction Data

Interaction data are crucial in many respects when building spoken (multimodal) dialogue systems. They are used to investigate aspects of how humans interact with each other or with a (simulated) system as well as to evaluate, e.g., how well users perform with a (simulated) system, how they perceive the system, and to which extent they like it.

As already pointed out in Section 1.3 several of the chapters in this book report on data collected for evaluation purposes. There are also several chapters that use data collection for development purposes. Chapter 2 includes data collection with a prototype to investigate what users find natural when interacting with an MP3 player. Chapter 3 reports on human–human data collected from role-play sessions to study negotiation strategies. In Chapter 6 human–system data have been collected to study non-understanding and error recovery strategies. Chapter 10 by Gruenstein, Niekrasz, and Purver has its full focus on annotation tools and annotation of multiparty human–human meeting data. Such tools and annotations support interaction studies which again may support and improve the development of systems. Chapter 11 by Forbes-Riley and Litman reports on both human–human and human–system tutorial dialogues which may be compared and form the basis for improvement of the tutoring system.

Data will continue to be an important source of information for driving the development and evaluation of spoken (multimodal) dialogue systems. A problem about data is that they are laborious and time-consuming to collect and analyse. The existence of annotation tools may greatly facilitate and speed up the data analysis and information extraction process. However, as is apparent from Chapter 10, the construction of such tools is a project in itself. Another related challenge which comes prior to the use of an annotation tool, is to find out and decide on which phenomena we actually want to look for in the data, i.e., the phenomena we need knowledge about to build better systems. Having decided on that, we must construct annotation schemes if there are no suitable existing schemes. Then we can start to use an annotation tool if one is available.

2. Overview of the Individual Chapters

In the previous section, we have already briefly touched upon all chapters in this book. In the following we provide a slightly more detailed overview of the contents of each chapter. Very roughly we may divide the chapters into the following categories although many chapters address aspects from more than one category and all chapters deal with discourse and dialogue aspects.

- Spoken dialogue systems (Chapters 1–3)
- Evaluation (Chapter 4)
- Miscommunication (Chapters 5–6)

- Dialogue management (Chapters 7–9)
- Corpus work (Chapters 10–11)

2.1 Spoken Dialogue Systems

Chapters 1–3 deal with differences and challenges in commercial and research systems with a focus on discourse and dialogue aspects.

In Chapter 1 **Pieraccini and Huerta** discuss spoken dialogue systems from a commercial as well as a research point of view in terms of characteristics and main differences.

While commercial systems have a primary focus on robustness, usability and cost effectiveness, research systems aim at achieving natural interaction by allowing unconstrained input and full mixed-initiative dialogue. Commercial systems to a considerable degree build on standards and draw on various tools and platforms. They are characterised by constrained and directed dialogue, limited initiative, and a demand for completeness, i.e., the system behaviour must be completely and explicitly specified regarding every possible situation that may occur. Thus the typical dialogue model can be described as a graph with user input type on the edges and dialogue states with pre-defined prompts in the nodes. By contrast research systems typically operate with an unrestricted and open-ended input space which makes completeness impossible.

The different focus points of commercial and research systems influence the form of the dialogue manager. Commercial systems typically use a finite state-based dialogue manager. On the research side dialogue management for more complex applications is being investigated. Various kinds of inference-based dialogue managers have been developed, e.g., agenda-driven dialogue managers, and there is research in dialogue managers based on statistical learning of dialogue strategy. However, prototypes based on such dialogue management approaches have not yet demonstrated the level of usability needed for commercial use.

Wang and Hamerich describe in Chapter 2 an in-car speech-controllable MP3 player. The system goes beyond current audio systems that only accept a few spoken commands like “next title” or “previous CD”. The presented system allows, e.g., the selection of track, album, artist, genre, or composer via spoken input which becomes important with the rapidly increasing number of titles that can be stored on MP3 devices.

The described type of speech-enabled in-car MP3 player is not yet available as a commercial product but, as it appears from this chapter, prototypes are being built to obtain experience that will support the eventual development of an actual product. Two successive multimodal prototypes are presented and a

third is reported to be under development building on experience from the two previous ones.

There are several challenges related to the development of in-car systems in general, e.g., that speech recognition must work in noisy environments and that computing resources are limited compared to server-based solutions. For a speech-enabled MP3 player there are a number of additional challenges to cope with. A major one is the problem of pronunciation of foreign titles. Sometimes a title even mixes words from several languages. Another challenge is – like for new applications in general – the interaction design. What does the user find natural and intuitive? The described prototypes served among other things to explore this issue.

In Chapter 3 **Traum, Swartout, Gratch and Marsella** address challenges in non-team negotiation dialogue. The system context is a multimodal negotiation dialogue trainer for military people who can train their negotiation skills by engaging in dialogue with a virtual human doctor.

Focus is on a dialogue model originally developed for virtual humans and allowing for multimodal, multiparty conversational team interaction. This model which is based on a layered information state approach, is described, including its extension to allow for non-team negotiation dialogue. Non-team negotiation introduces a number of new issues that must be dealt with, compared to what the dialogue model used as point of departure can handle. For instance, the interlocutors do not necessarily share a common goal and have trust in each other. Moreover, they may or may not see any benefit from engaging in negotiation. Three different orientations towards negotiation are discussed, including avoidance, win-lose, and win-win. In dialogues from a number of role-play sessions with one person playing a military commander and another playing the doctor, examples of all three orientations were found. Extensions to the dialogue model to handle these different negotiation strategies are presented, in particular how to model trust, initiative-taking, and avoidance to do something.

Two example dialogues with the implemented system show unsuccessful and reasonably successful negotiation with the doctor. The implementation includes a trace facility which automatically annotates how each input turn affected the virtual doctor's trust, beliefs and choice of strategy. This supports the subsequent review of the negotiation interaction.

2.2 Evaluation

Chapter 4 is the only chapter that is entirely devoted to evaluation.

Möller addresses in this chapter the evaluation of quality of telephone-based spoken dialogue systems. Such evaluation involves both subjective and instrumental, objective measurements. Thus the chapter provides an overview of main issues involved in collecting subjective user data and presents a set of

interaction parameters that may be extracted from human–system interaction log files to obtain quantitative measures. The use of subjective evaluation and interaction parameters has been recommended by the ITU.

Subjective evaluation is frequently done by having users interact with the simulated or implemented system under controlled laboratory conditions. Before and after interaction with the system users may be asked to fill in a questionnaire which serves to collect subjective user data. A multidimensional analysis of questionnaire results carried out within the European INSPIRE project is reported and results are compared to dimensions obtained from a number of other experiments. This revealed five perceptual dimensions underlying the users' judgements in the experiments, i.e., acceptability, communication efficiency, cognitive effort, personality, and smoothness.

The chapter continues to characterise five parameter categories based on literature studies, i.e., parameters related to dialogue and communication, metacommunication, cooperativity, task, and speech input. The parameters belonging to each category are further detailed and explained in an appendix. The set of parameters were evaluated in the reported INSPIRE experiment to determine their relationship to subjective user judgements. The correlation was found to be rather low and insufficient to predict overall system quality. Several PARADISE-style quality prediction models were subsequently applied to INSPIRE data using different sets of interaction parameters. In all cases results were found insufficient to replace users' subjective judgements of quality.

2.3 Miscommunication

Chapters 5 and 6 address miscommunication. Actually Chapter 7 also to some extent addresses miscommunication but we have chosen to include it under the dialogue management category.

In Chapter 5 **McTear** discusses miscommunication in spoken dialogue systems and why we should bother about handling it. Miscommunication has been addressed by many other disciplines than spoken dialogue. However, there has been little exploitation of research results from these areas although some of the research would seem truly relevant, such as research in cross-cultural and inter-gender miscommunication, conversational analysis, and text-based natural language interfaces. Within the area of spoken dialogue, miscommunication has been dealt with mainly at the level of recognition errors although there are several other potential error sources, such as out-of-domain utterances or the use of grammar constructions not included in the system.

The best approach to miscommunication is to prevent it from occurring, to the extent possible, through good design. However, there will always be a need to keep an eye on miscommunication in order to predict it, if possible, and perhaps change dialogue strategy to avoid, or at least to detect, an error

when it occurs and then recover from it. In a multimodal system conflicting information from different modalities may add to error handling complexity.

Error handling is not easy, but errors tend to negatively affect task success. Thus various ways exist in which to deal with errors, e.g., by explicitly confirming user input to detect misunderstandings immediately or by asking the user to speak more clearly if input was not understood. How to approach error handling depends largely on the type of interaction and its context. For instance, error handling may be important in banking systems requiring precise information to achieve task success, whereas in games miscommunication may be part of the game itself and just pose an additional challenge.

In Chapter 6 **Bohus and Rudnicky** present results on non-understanding errors and the use of ten different recovery strategies from an experiment with a telephone-based mixed-initiative conference room reservation system. Forty-six subjects carried out up to ten scenarios each. Subjects were divided into two groups since two conditions (control and wizard) were to be tested. In the control condition the system randomly selected an error recovery strategy while in the wizard condition a human who could hear what users said, selected the strategy. The purpose was to enable comparative evaluation of the ten recovery strategies and to see if a non-random policy for choice of recovery strategy is likely to improve system performance.

A comparison of the performance of the ten recovery strategies in the control condition is provided. The best one turns out to be the strategy that just ignores the non-understanding and advances the dialogue trying to find other ways in which to achieve the needed information. The second best strategy provides help with examples of what can be said.

A comparison of the two test conditions is also reported. It shows that an informed recovery policy is likely to yield a better performance than a random policy. On this background the chapter continues to look at data-based prediction of the likelihood of success for each recovery strategy. Based on the predictors two policies for recovery are proposed, i.e., maximising recovery rate and maximising recovery efficiency. First results indicate that these policies may perform better than the wizard policy.

2.4 Dialogue Management

Chapters 7–9 address particular aspects of dialogue management.

Skantze describes in Chapter 7 the discourse modeller Galatea which has been developed as part of the Swedish Higgins spoken dialogue system. The Higgins task domain is pedestrian city navigation and guiding. The architecture of Higgins is described, in particular the components which communicate directly with Galatea, i.e., a semantic interpreter that provides input to Galatea, and an action manager which together with Galatea constitutes the dialogue

manager. The core tasks of Galatea are to handle resolution of ellipses and anaphora and to keep track of the grounding status of information exchanged during a dialogue. It is described how Galatea resolves ellipses and anaphora and how it models grounding information by storing confidence scores for concepts mentioned by the interlocutors during a dialogue.

The Higgins system is a testbed serving to investigate error handling techniques in spoken dialogue systems. Thus approaches to error handling used in Higgins and, in particular, in Galatea are illustrated and discussed, including early as well as late error detection, fragmentary clarification, and display of understanding.

Performance results from an evaluation of Higgins and Galatea involving 16 naive users in a scenario-based laboratory setting is reported. These results look promising no matter if recognition errors occur or not.

Williams, Poupart and Young consider Partially Observable Markov Decision Processes (POMDPs) in Chapter 8. Whereas traditional approaches to dialogue management track a single hypothesis for the state of the dialogue, POMDPs maintain a distribution over many possible states. This distribution is beneficial because it allows speech recognition errors and non-deterministic user behaviour to be tracked robustly. The chapter first provides some background information on POMDPs, and then explains dialogue modelling using POMDPs in more detail.

The POMDP approach is compared with an approach which uses Markov Decision Processes (MDPs). Unlike POMDPs, an MDP tracks a single hypothesis for the state of the dialogue. Based on a test-bed-simulated dialogue management problem from the travel domain in which a user wants to buy a ticket to travel between two cities, it is shown that the POMDP approach performs better than the MDP approach.

Using the same test-bed problem the authors also show how handcrafted dialogue managers, i.e., dialogue managers in which designers have specified actions directly, can be improved. The advantage of this approach is that the human designer can build the handcrafted dialogue managers without regard for confidence score information, and the POMDP approach will determine how to best account for confidence score information as it executes the handcrafted dialogue manager. Experiments confirm that improvements are obtained in practice.

In Chapter 9 **Denecke and Yasuda** deal with dialogue management for restricted-domain question–answering systems. Their focus is on how to decide if the user provided enough information in a question or whether and how s/he should be prompted for additional information.

The approach they propose is to let the dialogue manager choose one among a predefined set of action types. Action type selection depends on the dialogue context which is based on extracted keywords from user input and retrieved

documents in response to this input. The selected action type is communicated to an instance-based generation component which realises it by retrieving an utterance annotated with the right type from a corpus and replacing content words to fit the actual context.

The approach has been implemented in a restricted-domain question–answering system concerning information on travel destinations in Japan, such as events, food and sightseeing. The Wizard-of-Oz collection and annotation of the corpus used by the generation component is described. The representation of dialogue context as a bag-of-words and the representation of how well the retrieved documents answer the user's question is presented. Furthermore alignment is described. Alignment is used to determine whether an answer candidate should be considered a valid answer to the user's question.

Based on the mentioned representations machine learning is used to learn the dialogue management function. Support vector machines are used to classify the current dialogue state to determine the next action to be taken.

2.5 Corpus Work

Chapters 10 and 11 mainly deal with corpora, including tools for annotation and analysis and the use of corpora for developing and improving practical applications.

Gruenstein, Niekrasz and Purver present in Chapter 10 a generic set of tools for representing, annotating and analysing multiparty discourse. These tools contribute to the larger goal of building an automatic office assistant that, e.g., can summarise meetings and point to relevant documents. A step towards this goal is to achieve an understanding of how people can best be assisted in browsing or reviewing meeting contents. To this end annotation of, e.g., structure in meeting data is important.

The authors describe the architecture developed for working with multiparty discourse. This description includes a semantics-based multimodal discourse ontology, its associated ontology programming interface, and the audiovisual toolkit NOMOS for annotation of media with ontological structure. Three tools developed on top of this architecture as NOMOS plugins are presented, i.e., a topic and action item annotation tool, a feature extractor and visualiser, and a meeting browser.

The topic and action item annotation tool has been used by two annotators to do topic segmentation and annotation of action item subdialogues of two meeting corpora. The annotation process is explained and inter-annotator agreement is evaluated. The rather low agreement reflects that there are various problems related to audiovisual multiparty discourse annotation.

In Chapter 11 **Forbes-Riley and Litman** address the development of more effective spoken dialogue tutoring systems by annotating and analysing

corpora of human–human and human–computer tutorial dialogue. In the presented study focus is on student certainty and the subsequent tutor dialogue acts in the human–human corpus and what can be learned from an analysis of possible dependencies.

Both corpora concern physics problems. The human–computer corpus was collected with the ITSPOKE spoken dialogue tutoring system which is meant to be improved on the basis of what is learned in the presented study. The collection of the human–human corpus with university students and a human tutor, and the annotation of student certainty and tutor dialogue acts, such as positive or negative feedback or long or short questions in return to answers, is described.

The annotated human–human tutorial dialogues are represented as bigrams of student–tutor turns. Chi Square analysis is then used to identify statistically significant dependencies between student certainty and tutor dialogue acts. The analysis process is explained and results are shown.

The correlation between dependent bigrams and student learning is analysed to investigate how the tutor's dialogue act strategies correlate with learning. A similar analysis is performed on the human–computer corpus with different results. This indicates that a given human–human strategy may have a different relationship to learning in human–computer dialogue.

3. Readership

The primary target group of this book are academic and industrial researchers in the field of spoken dialogue and discourse. However, we believe that the book may actually be of interest to the speech and language technology community at large. Graduate students and Ph.D. students may also benefit from reading selected chapters from the book.

The chapters vary considerably regarding the level of expertise required in advance to benefit from them. However, most chapters start with a state-of-the-art description from which all readers may benefit.

Chapter 1 on research and commercial dialogue systems and Chapter 5 on miscommunication are of an introductory character, deal with quite broad topic areas, and are accessible to a wide audience. This is to a high degree also the case for Chapter 4 on evaluation of telephone-based spoken dialogue systems.

Chapter 3 on negotiation training, Chapter 6 on non-understanding recovery strategies, Chapter 7 on discourse modelling and error handling, and Chapter 10 on meeting structure annotation require some background knowledge in the addressed topic areas for the reader to fully appreciate the contents. This is also true for Chapter 11 on tutoring systems and student certainty where it will be an advantage to be familiar with Chi Square and linguistic analysis.

Chapter 8 on Partially Observable Markov Decision Processes and Chapter 9 on dialogue management in question-answering concern very specific topics and require, not least, a good deal of statistical background from the reader.

The chapters contain a number of abbreviations and acronyms. To facilitate reading we have added a list of such abbreviations and acronyms towards the end of the book right before the index.

Chapter 1

WHERE DO WE GO FROM HERE?

Research and Commercial Spoken Dialogue Systems

Roberto Pieraccini
SpeechCycle Inc.
New York, NY, USA
roberto@speechcycle.com

Juan M. Huerta
IBM T.J.Watson Research Center
Yorktown Heights, NY, USA
huerta@us.ibm.com

Abstract The spoken dialogue industry has reached a maturity characterised by a vertical structure of technology vendors, platform integrators, application developers, and hosting companies. At the same time industrial standards are pervading the underlying technology and providing higher and higher levels of interoperability. On the one hand commercial dialogue systems are largely based on a pragmatic approach which aims at usability and task completion. On the other hand, spoken dialogue research has been moving on a parallel path trying to attain naturalness and freedom of communication. However, given the constraints of the current technology, the evolution of the commercial path shows that naturalness and freedom of expression are not necessarily a prerequisite for usability. The difference between the two goals has been influencing a parallel evolution of the architectures and in particular of the dialogue management abstractions. We believe it is the time to get a high level perspective on both lines of work, and aim at a synergistic convergence.

Keywords: Spoken dialogue system; voice user interface; dialogue manager

1. Introduction: Overview of Dialogue Systems

There are different lines of research in the field of spoken dialogue. Some researchers attempt at understanding, and possibly replicating, the mechanisms of human dialogue through linguistically motivated studies on human–human

L. Dybkjær and W. Minker (eds.), Recent Trends in Discourse and Dialogue, 1–24.

corpora. Others are interested in general design principles that, once applied, would result in usable human–machine user interfaces based on speech recognition and speech synthesis technology. Then, there is spoken dialogue system engineering (McTear, 2004), which aims at developing programming styles, models, engines and tools which can be used to build effective dialogue applications. The three lines of research are, in a way, orthogonal and complementary. The focus of the first is on understanding human communication, the second on designing the interface for usable machines, and the third on building those usable machines. The topic of this chapter is concerned with the latter, namely the engineering of spoken dialogue systems. However, every discussion on the engineering of dialogue systems would be flawed if we did not take into consideration both the nature of human–human dialogue – as this is the most efficient realization of spoken dialogue available in nature – and the goal of usability. Dialogue systems can be further studied in accordance to other dimensions, for example modality (e.g., speech only, audiovisual, multimodal), input sensors (e.g., telephone, microphone, keyboard, joystick, touch screen, gesture capture), target platform (e.g., embedded, server-based), application domains (e.g., pervasive help, personal assistance, transactional systems, command and control) and many others. We believe that these dimensions are complementary to the focus of this chapter.

The goal of usability – i.e. building machines that are usable by untrained users – is often confused with that of building human-like conversational systems. This is based on the underlying tacit assumption that a machine that approximates human behaviour, from the linguistic point of view is certainly more usable than one that does not. Although possibly true in the limit, this assumption is often misleading, especially if we consider that the performance of spoken language technology[1] today is still far from near-human performance. However, most of the research carried out during the past decade has been directed towards unconstrained natural language interactions based on the assumption that *naturalness* and *freedom* of expression are the essential goals to pursue, and usability would automatically follow from having reached these goals.

The limitation of current spoken language technology is a fact we have to live with. Thus, if we undertake the goal of building usable systems with that limitation, we would find that naturalness and freedom of expression may actually hinder usability (Oviatt, 1995; Williams and Witt, 2004) for a large number of useful applications. For instance let us consider spoken language understanding technology. In spite of the advances of the past decade, even in

[1]With the term *spoken language technology* we refer to all the technologies that attempt the replication of human spoken language skills by machines, including speech recognition, spoken language understanding, speech to speech translation, speech synthesis, and text to speech.

well-defined domains, unrestricted understanding of speech is still far to be on a par with humans. So, any spoken language system that encourages free and natural user interactions is bound to a non-insignificant level of understanding errors (Sagawa et al., 2004; Bohus and Rudnicky, 2007). Moreover, as of today, there are no effective error recovery dialogue strategies[2] available for unconstrained natural language interactions. Conversely there are several types of transactional applications that achieve high usability with interactions that are not *natural* and *free*. After all call centres often adopt scripts to be followed by their customer service representatives (CSR) which do not leave much freedom to callers.[3] Most of the applications in this category are characterised by a domain model that is well understood by the user population. For instance, the model for ordering pizzas is known to most of the users: a number of pies of a certain size (small, medium, or large) with a selection of toppings (mushroom, pepperoni, etc.) The same applies to the domain of flight status information: flights can be on time, late, or cancelled, they arrive and depart daily from airports which serve one or more cities and can be identified by a number or by their itinerary and time. All of this is generally well understood by most of the users of commercial flights. Banking, stock trading, prescription ordering, and many other services belong to the same category.

Generally, when the domain model is quite simple and known by the users, as in the above cases, applications can be implemented in a structured dialogue fashion, generally referred to as *directed dialogue*. Directed dialogue, even if seemingly more restrictive from the point of view of the user, can attain much higher usability and task completion rates than free form interaction does, given the limitations of the current technology. In fact, when users are prompted to provide specific pieces of information, the system can activate grammars designed to collect exactly that information. Moreover, as discussed in Oviatt (1995), user guidance reduces user disfluencies. Thus, the combination of user direction, strict grammars, and less disfluent speech can attain quite high recognition accuracy. On the other hand, a more open interaction would increase the space of possible user expressions at each turn, often causing a reduction of the recognition accuracy. Furthermore, without direct guidance, most users will be lost and would know neither what to say nor what the capabilities and limitations of the system are.

The concept that well-structured directed dialogue strategies may outperform natural language free-form interactions was realized by speech

[2]One of the problems arising when trying to implement error recovery in unconstrained speech is the automatic detection of recognition errors. In fact, today's speech recognition confidence measures are still highly unreliable, especially when one attempts to apply them to portions of an utterance. Without viable error correction, interaction with machines may be extremely frustrating for the user.

[3]As a matter of fact, human–human flight reservation generally follows a precise script that is dictated by the order of the entries in the CSR database.

technology vendors during the early and mid-1990s. The development of a *spoken dialogue market* during those years led to the rise, in the late 1990s, of a well-structured industry of speech engines, platform and tool vendors, application developers, and hosting companies, together with an increased attention to the industrial standards. In fact several standards are today governing the speech industry, such as VoiceXML 2.0,[4] MRCP,[5] SRGS,[6] SSML,[7] CCML,[8] and EMMA.[9] Mainly driven by the VoiceXML standards, the speech and the Web world started to merge, and the benefits of this standardisation trend took a momentum amplified by the simultaneous emergence of Web standards (e.g. J2EE[10] and JSP[11]).

From a different point of view it is interesting to notice that the research community has often adopted dialogue approaches based on general principles (e.g. Grice, 1975) that once coded give machines a reasonable behaviour for reacting to different dialogue situations. Then, in order to cope with the limitations of the technology, research started falling back to more restrictive dialogue strategies. In contrast, the commercial community started from a pragmatic approach, where each interaction is practically designed in the minimal details by *voice user interface* (VUI) experts (Barnard et al., 1999).

After mastering the crafting of directed dialogue applications, the commercial community is moving now towards more free-form types of interactions. One example of that is offered by a category of applications where *directed dialogue* cannot be applied. Applications of this type are characterised by a domain model which is complex and unknown to the majority of users. Help desk applications, for instance, fall in this class. For example, a directed dialogue system for routing callers to the appropriate computer support may prompt the user with: *Is your problem related to hardware, software, or networking?* Unfortunately the vast majority of users would not know which of the three categories would apply to their problem. A solution to that would be to provide a menu that includes all possible problems, but that menu would be too long to

[4]http://www.w3.org/TR/voicexml20/.

[5]Media Resource Control Protocol: a protocol for the low level control of conversational resources like speech recognition and speech synthesis engines: http://www.ietf.org/internet-drafts/draft-shanmugham-mrcp-06.txt.

[6]Speech Recognition Grammar Specification: a language for the specification of context-free grammars with semantic attachments: http://www.w3.org/TR/speech-grammar/.

[7]Speech Synthesis Markup Language: a language for the specification of synthetic speech: http://www.w3.org/TR/2004/REC-speech-synthesis-20040907/.

[8]Call Control Markup Language: a language for the control of the computer-telephony layer: http://www.w3.org/TR/ccxml/.

[9]Extensible MultiModal Annotation: a language for the representation of semantic input in speech and multimodal systems: http://www.w3.org/TR/emma.

[10]Java Platform, Enterprise Edition is the industry standard for developing portable, robust, scalable and secure server-side Java applications: http://java.sun.com/javaee/index.jsp.

[11]Java Server Pages technology provides an effective way to create dynamic web content: http://java.sun.com/products/jsp/.

be practical. In other words, the underlying domain model, unlike that of pizza orders and flight information, is largely unknown by the user population. The solution to this problem consists in letting callers express themselves freely, and back the system with a statistical classifier able to assign user utterances to one of several predefined categories. In other words the system is charged with the burden of mapping user expressions to the domain model, and not the users themselves. This technique, known as How May I Help You (Gorin et al., 1997), statistical call routing, or statistical natural language understanding (Chu-Carroll and Carpenter, 1999; Goel et al., 2005) is not more than a very simple form of language understanding which combines the robustness of a structured approach (a limited number of categories, or routes) with the flexibility of natural language (an open prompt leading to a large number of possible user expressions). In fact, using this technology, the dialogue can still be structured in a directed dialogue manner, because the output of the interaction is going to be one of a predefined number of categories.

The goal of this chapter is to give a high level view of the domain of dialogue systems both in the research as well as in the commercial domain, with a focus on the problem of dialogue management. The rest of this chapter is organised as follows: Section 2 describes the importance of dialogue *control* and authoring *expressiveness* in commercial dialogue applications and the principle of VUI completeness. Section 3 provides a working definition of dialogue management. Section 4 gives a detailed overview of the basic architectures used for building dialogue applications. The following sections (5, 6 and 7) describe the main approaches to dialogue management, namely programmatic, finite state, and inference based. Section 8 provides an overview of the latest trends in commercial dialogue systems, and finally Section 9 reports our closing conclusions.

2. VUI Completeness

The need for a detailed control of the VUI is thus an important factor driving the architectural and engineering choices in commercial dialogue systems. We call this the *VUI-completeness* principle: the behaviour of an application needs to be completely and explicitly specified with respect to every possible situation that may arise during the interaction. No unpredictable user input should ever lead to unforeseeable behaviour. Only two outcomes are acceptable, the user task is completed as specified in the design specification, or a fallback strategy is activated (e.g. escalation to an operator).

In order to ensure that an application is *VUI-complete*, its behaviour needs to be specified for each possible situation, or class of situations. Today, a complete VUI specification is standard procedure in commercial deployments and is generally represented by a graph that describes all the possible dialogue states, complemented by tables that describe all the details of each state. Transitions

between dialogue states are described with conditions predicated on the user inputs and other pieces of information (e.g. previous user inputs, back-end response and personal user information). The precise wording of system prompts is also specified in the design, along with an indication of the type of utterances accepted at each turn. The VUI specification document is then handed to a team of developers who implement the application using the platform of choice. In some situations the VUI designers use advanced authoring tools that lead to the complete development of the application without, or with limited, intervention of software engineers and developers. In order to reduce development costs, it is thus important to guarantee a direct mapping between the formalisms and abstractions used by the VUI designers and the programming models available to the developers. This is the reason why most commercial dialogue managers follow the same abstraction utilised in the VUI specification.

Research systems, on the other hand, are typically designed to manage dialogue in situations where the product space of inputs, dialogue states, and outputs makes an explicit and exhaustive enumeration of all the possibilities impossible or impractical at best. This is due in part to the aim that the research community has towards handling unrestricted natural language input and mixed-initiative[12] dialogue control. On the other hand, an explicit enumeration (e.g., expansion into a deterministic graph) of the possible inputs as well as the possible dialogue transition states is required in commercial systems to establish a complete VUI specification that can be signed off by the client who pays for the development of the system itself. It is in general uncommon to find research systems which present full VUI-completeness.[13]

2.1 Control and Expressiveness

In order to allow developers to implement detailed VUI specifications, the programming paradigm adopted by the dialogue manager or authoring tools should allow a fine control of the system behaviour. However, a too low-level development paradigm may result in prohibitive development costs for large and complex applications. Hence the programming paradigm needs also to be expressive enough to allow implementing complex behaviour in a simple and cost effective way. These two features are often competing, since in order to guarantee more expressiveness the dialogue manager has to allow for sophisticated built-in behaviour, which may be hard to bypass if one wants to attain

[12]The term mixed-initiative is generally used to refer to those dialogue systems that allow the user, as well as the system, to change the course of the interaction at any point in the dialogue.

[13]In the DARPA Communicator evaluations, participant sites implemented systems with common requirements on the travel planning problem. Had a VUI-complete specification been set forth by the community as a joint effort, part of the evaluation would had simply consisted in verifying application compliance against the specification document. Still the user experience and the usability of the interfaces would have played an important role in the differentiation and evaluation of the systems.

a detailed control of the interface. An effective programming and authoring paradigm for dialogue systems is thus the result of a trade-off between control and expressiveness. This can be summarised by the following principle: *simple things should be simple and complex things should be possible.*[14]

3. Dialogue Management

The design of a proper dialogue management mechanism is thus at the core of dialogue system engineering. The study of better dialogue managers and proper dialogue engineering aims mainly at reducing the application development costs. But it is also a way to move towards more sophisticated human machine interactions, since it is only with proper engineering of dialogue systems that we can raise the complexity threshold that separates what is practically realizable from what is not.

There is not an agreed upon definition of what a dialogue manager is; different systems described in the literature attribute different functions to it. Some of these functions are: integrating new user input, resolving ambiguities, confirming and clarifying the current interpretation, managing contextual information, communicating with the back-end, managing speech recognition grammars, generating system outputs, etc. In fact, the minimal functionality required by a dialogue manager covers two fundamental aspects of all interactive applications: keeping track of session states and deciding what the next action for the system to take is. Of course there are many ways of coding these two functions in order to achieve a desired interactive behaviour. The rest of this chapter will describe some of them.

4. Reference Architectures: Research and Commercial

In order to describe different approaches to dialogue management, it is important first to define, at a high level, the architecture of spoken dialogue systems.

Figure 1 shows a general functional architecture of a dialogue system, mostly used in research prototypes. We refer here and in the following to telephone-based systems. However, some of the concepts expressed in this chapter can be generalised to other types of system that do not make use of telephone communication, such as embedded systems for mobile devices and for automobiles. While the description of how some of these principles apply to different non-telephony systems is beyond the scope of this chapter, we should mention that commercial systems, especially in the embedded area, are

[14]This maxim is attributed to Alan Kay.

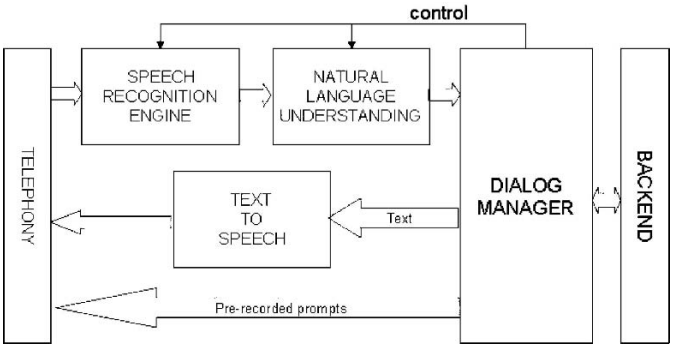


Figure 1. Functional architecture of a dialogue system mostly used in research prototypes.

moving towards multimodal applications. Applications where speech is not the only modality, but is integrated with haptic and visual interfaces, are becoming more and more common, and certainly need more sophisticated architectures than the ones described here.[15] A discussion on some of the issues related to multimodal systems can be found in Pieraccini et al. (2005).

In the most common configuration of a spoken dialogue system architecture, input speech is collected via a telephone interface and dispatched to the speech recognition engine which provides one or more recognition results (for instance the *n-best* recognition results). Each recognition result is then fed to a *natural language understanding* processor which extracts the semantics of the utterance. A formal representation of the semantics, generally a structured set of attribute-value pairs, is then passed on to the dialogue manager. The dialogue manager, based on the current utterance semantics, and on the stored contextual information derived from previous turns, decides the next *action* to take according to a *dialogue strategy*. The most obvious action performed by the system as a response to a user utterance is a system utterance, generally referred to as *prompt*, which can be generated as text and transformed into speech by a *text-to-speech* engine, or selected from a set of pre-recorded samples.[16] Other types of action performed by the dialogue manager include interactions with the back-end system, or any other type of processing required by the application.

The above described architecture has been implemented in many different forms in research. Of particular interest is the Galaxy architecture (Seneff et al.,

[15]Examples are SmartKom (http://smartkom.dfki.de/) and Embassi (http://www.embassi.de).

[16]High-quality prompts are today obtained by splicing pre-recorded phrases with TTS generated content, using concatenative speech synthesis.

1999) which was used in the DARPA Communicator[17] project and allowed the interchange of modules and plug-and-play across different research groups.

One thing to notice in the above described architecture is that the specific language models used by the speech recognition and natural language understanding engines are supposed to be constant throughout a whole session. In fact, one of the basic assumptions behind most research prototypes is that the system should be able to understand all the possible expressions defined by the language model at any point during the interaction. However it is clear that there is a correlation between the distribution of possible utterances and the dialogue state or context. Thus in order to improve system performance, the dialogue manager can change the parameters of the language model and language understanding depending on the current dialogue context. Several systems did implement this feedback loop with resulting improved performance (Xu and Rudnicky, 2000).

Commercial system architectures, instead, evolved in a different way. The basic assumption on which most of the commercial deployed systems were, and still are, based can be expressed by the following statement: properly designed prompts can effectively control the space of user expressions. Thus, based on this assumption, there is no need for the system to be able to understand, at each turn, all the possible expressions that users could say, since the user will mostly speak what is suggested by the prompts. Users are in fact *directed* (thus the term *directed dialogue*) and guided to speak exactly what the system expects. It is clear how this assumption, if true, can potentially allow the attainment of very high task completion rates by limiting the *unknowns*. Under this assumption, commercial dialogue systems provide the speech recogniser with grammars that are specifically designed for each turn of the interaction. Each grammar – typically a context-free grammar with semantic attachments – is specifically designed to accept the utterances that are expected to be possible user reactions to the specific prompt played at that particular turn. So, instead of a generic prompt like *Hello, this is XYZ flight status information line, how can I help you today?* commercial dialogue system designers use more specific prompts such as *Are you interested in arrivals or departures?* or *From which city is the flight departing?* Prompts and grammars, thus, need to be designed together.

The benefit of using restricted grammars in directed dialogue applications becomes evident when looking at the error control logic typically adopted by commercial systems. In fact even with very restricted grammars there is always a chance for the recogniser to produce erroneous interpretations, or for the user to speak utterances outside the domain. Thus in case of poor recognition

[17] http://communicator.sourceforge.net/.

scores, commercial dialogue systems *direct* users to correct presumably erroneous interpretations by using very strict prompts, such as: *I think you said Austin, is that correct? Please say yes or no*. And since the system cannot afford to confuse a yes with another phonetically similar word at this point in dialogue (misrecognitions in correction subdialogues would lead to enormous user frustration), the grammar associated with the confirmation prompt is typically restricted to yes/no utterances and a reasonable number of synonyms and command words (such as *help* and *operator*).

Early commercial dialogue systems were built using proprietary architectures based on IVR (Interactive Voice Response) platforms. Soon, the speech application development community realized the importance of industrial standards and started to create recommendations to guarantee interoperability of platforms and engines. After the introduction of VoiceXML 1.0[18] in year 2000, conversational systems started to conform to a general Web architecture, such as the one shown in Figure 2. The convergence of speech and Web technologies (the so called Voice Web) has allowed the speech industry to leverage existing Web skills and resources, and reduce the need for specialised developers.

The core of commercial dialogue systems exemplified by Figure 2 is the *voice browser* which accepts documents written in a markup language specific for speech applications, such as VoiceXML. The voice browser exchanges information with a Web server using the HTTP protocol in analogy with the browser and server in traditional visual Web applications. VoiceXML markup documents instruct the browser to activate the speech resources (speech recognition, TTS, prompt player, etc.) with a specific set of parameters, such as grammars for the speech recognition engine, prompts to be synthesised by the text-to-speech system, or audio recording to be played. Once the user's speech has been recognised, and the recognition results returned to the browser in the form of a structured set of variables, the browser sends them back to the Web

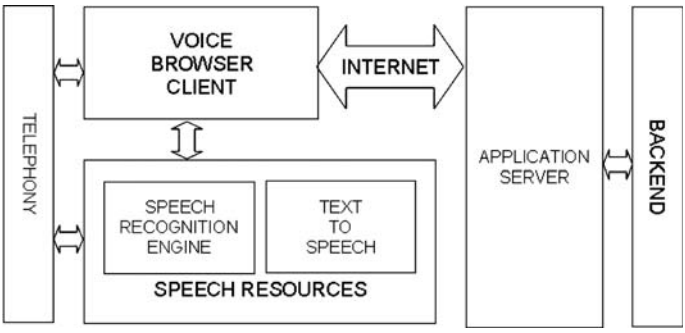


Figure 2. Typical architecture of commercial dialogue system.

[18] Voice eXtensible Markup Language.

server, together with the request for another VoiceXML document. The Web server then replies by sending the requested document to the browser, and the interaction continues in this fashion.

Using static VoiceXML documents, the dialogue manager function is actually distributed across the various VoiceXML pages, as in a static visual website, the navigation is distributed across the collection of HTML documents. In fact each document includes instructions for the browser to request the next document once the current one has been executed. All the VoiceXML documents and the corresponding resources (such as grammars, prompts, etc.) are typically stored statically on the Web server and *served*[19] to the browser upon request. However, as it happened in the visual Web world, developers found the mechanism of encoding the whole system in static VoiceXML pages quite limiting, and soon they started to write programs on the server for generating dynamic VoiceXML documents. In this case the application is actually managed by a program running on the application server, which acts as a dialogue manager and that generates dynamic VoiceXML documents upon requests by the browser. The introduction of the J2EE/JSP technology makes this process straightforward and in line with mainstream Web programming.

Generating VoiceXML dynamically on the server has the advantage of providing the developer with more powerful computational capabilities than those available on the voice browser client, and thus accommodating, in a more flexible way, the dynamic nature of sophisticated interactions and business logic. Moreover, there are security restrictions on the client browser that may prevent direct access to external resources, such as back-end databases. The evolution of server-based programming of applications brought the separation of the dialogue management functionality from the presentation (i.e. the activation of speech engines, playing of the prompts, etc.), and the realization of general purpose dialogue managers and programming models for developing speech applications on the server.

In spite of the different architectural evolution of research and commercial dialogue systems, the need for a powerful dialogue manager is felt by both communities. In the next few sections we will discuss some of the available models of dialogue manager which have been introduced in recent years.

5. Programmatic Dialogue Management

The simplest form of dialogue manager is a generic program implemented in a procedural programming language such as C++ or Java (or as a Java servlet in the case of Web-based architectures) that implements a dialogue application

[19] Voice browsers use caching strategies similar to those used by visual Web browsers. So, large grammars may be cached on the client so as to avoid significant resource provisioning latency.

without any underlying general interaction model. Early commercial dialogue applications were typically developed, on the deployment platform, as native code following a given VUI specification. Before the advent of VoiceXML and the Web programming paradigm for voice applications, IVR vendors integrated speech recognition engines directly in the platforms which had proprietary programming environments or proprietary APIs.[20]

However, building each application from scratch becomes soon an inefficient and repetitive activity. Like in all areas of software development, vendors tried to reduce the cost of application development by introducing libraries of reusable functions and interaction templates, often for internal consumption, but also as products that could be licensed to third parties. Libraries were also complemented by programming frameworks, generally in the form of sample code or templates, which could be reused and adapted to different applications.

Dialogue modules[21] developed by various speech recognition and tool providers, constitute one of the first forms of commercial reusable dialogue functions. Dialogue modules encapsulate all the low level activities required to collect one or more pieces of information from the user. That includes prompting, re-prompting in case of rejection and timeout, confirmation, disambiguation, etc. The collection procedure, including prompts, grammars, and logic for standard pieces of information, such as dates, times, social security number, credit card numbers, currency, etc., was thus encoded once and for all in pieces of reusable and configurable software. Developers could also build their own custom dialogue modules. Thus dialogue modules became, for many, the standard approach to directed dialogue. Applications were then implemented with the programming model available for the chosen platform. Each state of the dialogue flow was associated to a specific dialogue module, and the programming model of the platform was the glue used to implement the whole dialogue.

6. Finite State Control Management

The finite state control dialogue manager is an improvement on the programmatic dialogue manager. The finite state control dialogue manager implements a separation between the logic of directed dialogue and its actual specification. The logic is implemented by a finite state machine engine which is application

[20]Some platforms used GUI application development environments that were originally designed for touchtone (DTMF) applications, and then extended to handle speech recognition and TTS. Other commercial platforms allowed access to the functionality of the IVR and the speech recognition/TTS engines by exposing a proprietary API, and allowing it to be invoked by common programming languages such as C, Java, and Visual Basic.

[21]Commercialised by SpeechWorks as *Dialogue Modules* and by Nuance as *Dialogue Objects*.

independent and therefore reusable. Thus, rather than coding their own finite state machine mechanism directly in computer code, as in the programmatic model, developers had to provide a description of the finite state machine topology in terms of a graph of nodes and arcs. This can be defined as a data driven approach. Often the topology could be derived directly from the VUI specification. Then developers had to complement that with a set of custom functions required by the application. Without a separation between the finite state machine mechanism and its topology, the implementation of the dialogue state machine logic was often left to the programming skills of developers, often resulting in an unmanageable spaghetti-like nest of *if-else* or *case* statements, with increased debugging and maintenance costs, and made it impossible to build applications above a certain level of complexity.

One of the obvious advantages of the finite state control management approach is that the topology of the finite state machine is generally easier to write, debug, and maintain than the finite state machine mechanism itself. Moreover, the finite state machine engine can allow for hierarchical and modular dialogue definition (e.g. dialogues and subdialogues). Finally, the engine itself can be harnessed to verify the overall topology, check for obvious design and implementation mistakes, such as unreachable nodes and loops, and provide debugging and logging facilities. More sophisticated engines can have built-in behaviour like for instance handling specific navigation across dialogue networks, recording usage information for personalised services, implementing functions such as *back-up* and *repeat*, etc. (Pieraccini et al., 2001).

The simplest form of finite state control dialogue manager is built around the concept of *call flow* developed initially for IVR systems. In its simplest realization a call flow is a graph where the *nodes* represent prompts, and the *arcs* represent transitions conditioned on the user choice at that particular node (e.g. Figure 3). By navigating the call flow graph and by selecting the right choices, the user can reach the desired goal and complete the task. The call flow model is quite limited and breaks for complex dialogue systems since one has to explicitly enumerate all the possible choices at any node in the dialogue.

In fact the pure call flow model is inadequate to represent even modest levels of mixed-initiative, such as over-specification, when more than one piece of information is given by the user in a single utterance. For instance, if asked for the date of a flight[22] in a mixed-initiative system that allows for over-specified requests, users may instead respond with any subset of date, origin, destination, and airline. In order to be able to handle this, the simple call flow model would need to represent explicitly all the possible subsets of user choices (e.g. date, date + time, date + origin, date + origin + destination) making the design and development impractical.

[22]It looks like the spoken dialogue community has a penchant for applications related to flights. We hope to see other domains of interest in the future.

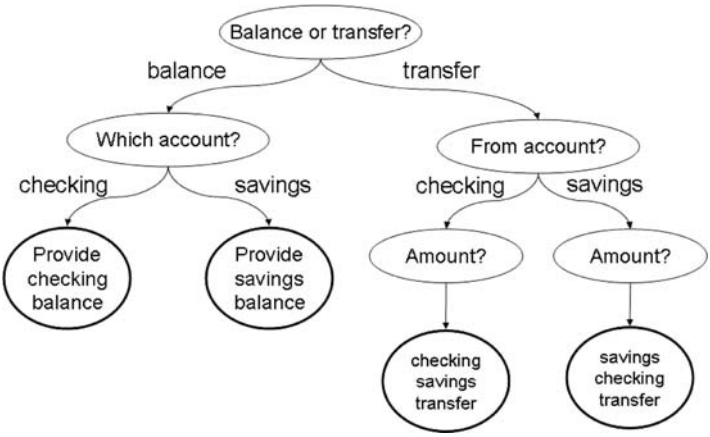


Figure 3. Example of call flow.

However, one can easily extend the concept of call flow and allow the state machine to assume any topology, to invoke any arbitrary function (action) at each node, and assume any arbitrarily complex condition on the arcs. Furthermore, one can allow any arbitrarily complex data structures (session state) to be writable and readable by the actions associated to the nodes. In this new extended form, the finite state control dialogue manager (we will refer to it as the *functional model*) has enough expressive power to represent sophisticated directed dialogue and mixed-initiative interactions. A full functional model of dialogue management can also allow for recursion, i.e. full dialogues specified in a functional fashion can be, themselves, used as actions and associated to nodes of a higher level dialogue, enabling thus hierarchical description of applications, and promoting modularity and reuse. An example of a control graph that handles over-specified utterances is shown in Figure 4 (explained later in this chapter). More detailed descriptions of functional models of dialogue management can be found in Pieraccini et al. (1997, 2001).

There are common misconceptions about the effective expressiveness and computational power of the finite state dialogue model. In fact limited capabilities with respect to more sophisticated abstractions are often wrongly attributed to finite state models of dialogue control. These misconceptions derive from the confusion that often exists between the functional model described above and the simplistic call flow model which is completely described by a state machine with prompts on the nodes and choices on the arcs. In its simpler form the call flow model is indeed, computationally, a *finite state model* of dialogue: i.e. the state of the dialogue is univocally determined by the node of the call flow. By contrast, the functional model allows arbitrary functions at each node to manipulate arbitrary memory structures that can be shared across nodes. Thus

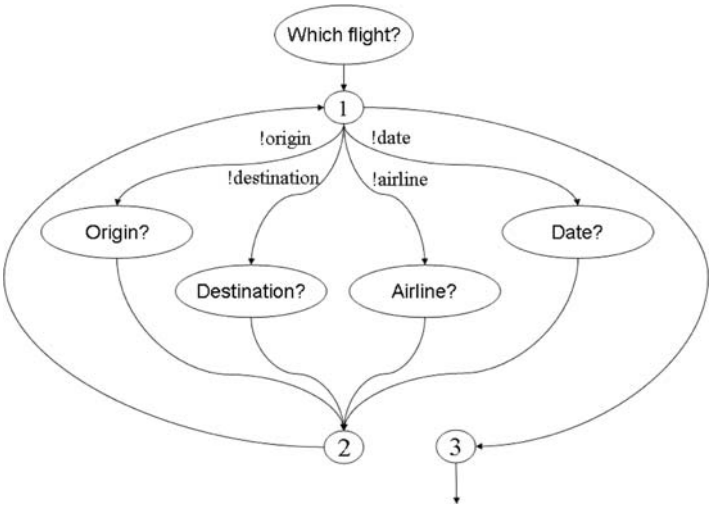


Figure 4. Graph representing a functional dialogue controller able to handle over-specified utterances. The conditions on the arcs exiting a node are verified in a left-to-right fashion. Arcs without conditions are to be considered as having an *else* condition.

the extended functional model is not, computationally, a finite state model of dialogue; it just makes use of a *finite state representation* – i.e. nodes and arcs – for the dialogue control mechanism. In fact each *node* of the finite state machine describing the dialogue control does not represent univocally the *state* of the dialogue because we need also to take into consideration the *state* of all the memory structures associated with the controller (e.g. the session state).

A functional dialogue manager is equivalent to a procedural program with a fixed structure based on nested conditional or *case* statements. The nodes are equivalent to function calls, while the conditions are equivalent to the conditional statements, and a whole dialogue is analogous to the definition of a function. However, a functional dialogue manager specification is much easier to author and debug than a set of nested conditional or *case* statements.[23]

6.1 Handling Mixed-Initiative in Functional Models

A clear limitation of functional models is that they often require a complete topological definition of the task that may be rather complex for certain types of applications. For instance, the implementation of mixed-initiative interactions

[23] As a proof of this, we leave to the reader the exercise of rewriting the controller in Figure 4 as a series of nested if/else-if/else statements.

may result in a control graph with a large, unmanageable number of arcs. One way to reduce the cost of designing and developing mixed-initiative dialogue applications within the functional model paradigm consists in providing the controller engine with a behaviour that corresponds to complex topologies, without the need for the developer to specify those in term of nodes and arcs. For example, in Pieraccini et al. (2001), the concept of state transition was extended to include special GOTO and GOSUB arcs to easily implement topic changes and digressions at any node of the dialogue. Powerful engines for functional dialogue models can also allow for effective authoring of *global* transitions that apply to whole sets of nodes.

6.2 Fixed Topology Models

One can implement functional dialogue managers that allow the developer to specify the control graph topology (Carpenter et al., 2002). On the other hand one could restrict the control graph to assume a fixed topology and allow developers to specify only a limited number of parameters.

The Form Interpretation Algorithm (FIA), the basis for the VoiceXML standard, is an example of a functional model of dialogue management with a fixed topology. The topology of the FIA controller is in fact the one shown in the example of Figure 4. The FIA topology is particularly suited for handling overspecified requests, allowing filling forms with multiple fields in any order. For instance, if after the initial question *Which flight?* the user specifies the destination and the airline, the arc *!origin* (i.e. *NOT origin*, meaning that the origin slot has not been filled) is traversed and the node *origin?* is executed next. As a result the user is asked to provide the origin of the flight. Then, the *date?* node is executed next since the condition *!date* proves to be true (i.e. a specific date is not available yet). After the user has provided all the required pieces of information (origin, destination, airline, and date) the subdialogue exits through node 3.

Another example of functional model with a fixed topology controller is the MIT dialogue management system (Seneff and Polifroni, 2000). In this case the control is defined by a sequence of functions that are activated when the associated conditions fire. Each function can modify a session state (i.e. a *frame*-like, or attribute-value memory structure) by adding additional information, including a flag which instructs the controller on what to do next. Possible flags are: CONTINUE, causing the execution of the next rule in the sequence, RETURN, causing the controller to return to the initial rule, or STOP the execution. Again, as in the VoiceXML case, developing a dialogue does not require a topological description of the control graph, which is fixed and has the functional form described by Figure 5, but the specification of the functions associated to the

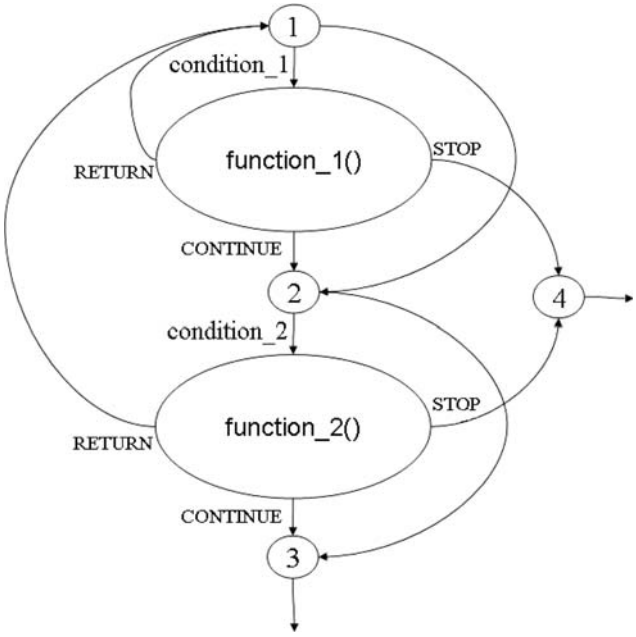


Figure 5. Functional control graph representing a rule based system.

nodes, and the conditions. The following is an example of a set of rules that implement the same subdialogue as the one in Figure 4.

$$!origin \rightarrow prompt\_origin()$$
$$!destination \rightarrow prompt\_destination()$$
$$!airline \rightarrow prompt\_airline()$$
$$!date \rightarrow prompt\_date()$$

7. Inference-Based Dialogue Managers

We have shown in the previous section how several types of dialogue manager can be reduced to a unique underlying model: the functional finite-state dialogue controller. The difference between them is whether developers are allowed to change the topology of the controller, and in the way they can author an application (e.g. by specifying a graph or a set of rules). However, there are classes of applications for which a specification through a finite state controller may appear impractical. As we discussed earlier, transactional applications with a well defined goal (e.g. giving information to the user, selling or buying) can often be effectively implemented with a finite state controller. On the contrary, some applications of the problem solving type (Allen et al., 2000)

with a high complexity require a higher degree of planning, for which the finite state controller can be inappropriate. Although we start seeing commercial technical support applications, which belong to the problem solving category, being successfully adopted by certain industries, most of the more complex problem solving applications have not yet found a channel to the market of spoken dialogue systems. This is probably because the research prototypes have not yet demonstrated the level of usability needed for commercial use. For instance, the deployment of some sophisticated research systems would require highly specialised development teams that may be prohibitively expensive in a commercial setting. Moreover the performance of the systems for the most complex problem solving tasks is not yet at the level required for commercial exploitation.

In spite of its difficulty, the research community has been actively pushing the technology towards the solution of the dialogue management problem for complex systems, especially under the auspices of the DARPA Communicator program. Successful prototypes have been demonstrated and tested based on sophisticated dialogue managers that deviate from the finite-state controller model, and include some degrees of inference. A distinguishing feature of the inference based systems is that they refrain from attempting at a more or less explicit description of the relationship between states and actions, as in the finite state controllers, but rather resort to engines that draw decisions on the next action to perform based on a general strategy and on a formal description of the domain, typically in terms of goals and subgoals. Thus, in order to develop an application, rather than describing the VUI, one starts from a formal description of the domain model in such a way to allow the inference engine to drive the system to a cooperative solution.

In Stallard (2001) the dialogue control model is described by a tree representing the goal/sub-goal structure, with the leaves of the tree being the actions. Actions, which include pre-conditions for their execution, are associated to individual goals. Internal nodes represent conditional controls on the execution of the underlying nodes. A dialogue manager based on task ontology and a hierarchy of nodes is described in Pellom et al. (2000). The dialogue manager described in Wei and Rudnicky (2000) constructs a dynamic structure, called *agenda*, which is a list of subgoals, where each subgoal corresponds to the collection of some piece of information. A task is completed when all the items in the agenda are completed. The agenda is created, dynamically, by traversing a tree (i.e. the *product tree*) that describes, at any point in time, the current task to be completed. The product tree has to be created dynamically since the nature of the task may be dynamic as well (e.g. the number of legs of a flight is determined during the interaction and not known beforehand). In the form based dialogue manager described in Papineni et al. (1999) the inference mechanism is driven by a numeric function computed on a set of

partially completed forms (i.e. sets of task-relevant slots), based on how close to the goal (i.e. the retrieval of information from the database) each individual hypothesis is.[24]

Another line of research is based on statistical learning of the dialogue strategy using mathematical models derived from statistical machine learning, such as Markov Decision Process (Levin et al., 2000) or Bayesian network frameworks (Meng et al., 2003). It is still too early to be able to understand whether automated design of dialogue can allow building usable systems with a quality comparable to that of those designed by VUI expert designers.

It is not yet clear whether any of the sophisticated inference dialogue managers developed in research could be effectively used for mass production of commercial systems. One of the problem is that their behaviour is quite complex, and it may be difficult to predict all possible situations that could arise during the interaction. Thus VUI completeness may be hard to achieve. Research prototypes, so far, have been built by researchers with an intimate knowledge of the quirks of the dialogue manager itself, and often by those who built it. Thus, in order to succeed in the commercial arena, inference engines have to produce systems with usability and robustness comparable or superior to that of an equivalent directed dialogue for the same task, or implement applications that are so complex that they cannot be approached with directed dialogue, still with usability as their main goal. VUI completeness is an essential requirement which should be seriously taken into proper consideration for the more sophisticated dialogue manager models.

8. Current Industrial Trends

Reusable components (Huerta et al., 2005) and pre-packaged applications are the main trends of the industry of spoken dialogue systems today. Componentization and reuse effectively allow reducing deployment costs and risks and, at the same time, simplifying the design and development of more sophisticated applications. Thus the commercial world is approaching the creation of more complex applications through more and more sophisticated building blocks which allow reuse and interplay.

Additionally, the need for language flexibility and robustness has motivated the use of Natural Language Understanding (NLU) technology. This requirement has allowed NLU technology to move from just call routing (Gorin et al., 1997) to a more sophisticated use, like for instance understanding and categorising symptoms in technical support applications.[25]

[24] A commercial version of this dialogue manager was implemented by IBM and used in a financial application (T. Rowe Price).

[25] http://www.speechcycle.com.

9. Conclusions

The way applications are authored, what capabilities the systems have, and the overall usability that is eventually perceived by users reflect the different goals that research and industry have in the field of spoken dialogue systems. Whereas usability and cost effectiveness are the primary goals of the commercial community, research has traditionally aimed at naturalness of interactions and freedom of expression. However, often the latter does not necessarily lead to the former. The actual form assumed by dialogue managers in both communities is the consequence of those different goals. In fact, in order to achieve high usability, commercial deployments aim at having completely definable interfaces (control and VUI completeness), using efficient languages and architectures (expressiveness and simple-things-should-be-easy) while keeping the ability to achieve complex levels of interaction (complex-things-should-be-possible). At the same time, the focus of research is towards abstracting, validating and achieving complex levels of natural interaction. While at first glance both sets of goals might seem in conflict, we believe that an evolution towards more complex levels of interaction, while using an effective development framework and implementing a "controllable" (VUI complete) interface is possible.

We have shown that most commercial dialogue management abstractions fall into the functional finite-state controller mechanism, as well as some of the dialogue managers developed in research. The difference is in the constraints applied to the topology of the controller and in the type of authoring (graphs vs rules). We have also shown that there is a second category of dialogue managers, inference based, which is devoted to handling more complex interactions, such as problem solving applications. VUI completeness and economy of development are required for them to become viable and reach the level of usability needed to succeed in the commercial arena.

We believe that the authoring of applications should be aligned with the model used at design time, and possibly to the runtime environment. In this way efficiency can be achieved at all levels: design, development, and deployment. The framework should allow for the encapsulation of dialogue mechanisms into templates, components, and subroutines that abstract behaviours. Beyond allowing for a reduction of development costs, this is also the first step towards the implementation of more complex interaction mechanisms. Finally, the framework should have strict "directed" and thus controllable default behaviour, but at the same time should allow for more complex interactions to be triggered if and when these dialogue mechanisms would benefit the interaction (e.g. expert and power users).

An important consideration that needs to be made when talking about the use and usability of commercial dialogue systems is that their success has to

take into account the willingness of the user to cooperate. Even the best designed application based on the most advanced architecture fails when callers refuse to use it. The problem is that the general public, the population of users of commercial dialogue systems, is at best annoyed when they are faced with a computer rather than a live agent. This phenomenon can be attested by the *cheat-sheets*[26] published on the Web that suggest words and sequences of touch tone keys for callers of commercial customer care applications to get a human operator right away. We can make an analogy with many other automated systems that are massively used today, and that provoked a similar reaction when they were introduced first a few years ago, such as ATMs (or cash machines) and answering machines. ATMs and answering machines are ubiquitous today and nobody would ever think of them as *replacements* for bank tellers and receptionists, but useful tools that improve our way of life. Similarly dialogue systems are to be considered as tools that, if used properly, can provide faster and better service for the most common situations and problems.

So, what is the main difference between research and commercial dialogue systems? We can certainly say that while some research was originally inspired by the dream of moving towards human-like interfaces characterised by fully unconstrained interactions (the dream of a HAL 9000-like machine from the celebrated 1968 *2001, A Space Odyssey* movie), commercial dialogue systems have to have more practical goals such as robustness, usability, testability, and ease of design and maintenance. Because of that, the commercial community took the approach of very controlled interfaces with constrained input and limited initiative on the part of the user. In other words they assumed that compliant users will *learn* how to use non-natural and highly limited interfaces, when compared with a human–human analogy, in order to get their task done. While the goal of research is certainly more ambitious, it has not had yet the opportunity to bring to life truly natural language mixed-initiative dialogue systems as a viable alternative to the constrained commercial directed dialogues. This is due, in part, to the lack of exposure that research systems have to a high volume usage that helps drive their improvement, the lack of research funds and the interest of the funding agencies, and also in part to the choice of research applications that often can be easily outperformed by analogous directed dialogue commercial applications. Should research concentrate on applications that are so complex that they *could not* be approached with equal or higher effectiveness by commercial systems, that would help establish a research goal way beyond the current technology.

[26]http://gethuman.com.

Finally we believe that a consolidation of the goal priorities (i.e. usability and naturalness of interaction) between research and the commercial world will foster further maturation of the technology. For this to happen, though, the *dialogue* needs to start.

References

Allen, J. F., Ferguson, G., and Stent, A. (2000). Dialogue Systems: From Theory to Practice in TRAINS-96. In Dale, R., Moisl, H., and Somers, H., editors, *Handbook of Natural Language Processing*, pages 347–376. Marcel Dekker, New York.

Barnard, E., Halberstadt, A., C., K., and Phillips, M. (1999). A Consistent Approach to Designing Spoken-Dialog Systems. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 363–366, Keystone.

Bohus, D. and Rudnicky, A. (2007). Sorry, I Didn't Catch That! An Investigation of Non-Understanding Errors and Recovery Strategies. In Dybkjær, L. and Minker, W., editors, *Recent Trends in Discourse and Dialogue*. Springer. (This volume).

Carpenter, B., Caskey, S., Dayanidhi, K., Drouin, C., and Pieraccini, R. (2002). A Portable, Server-Side Dialog Framework for VoiceXML. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 2705–2708, Denver.

Chu-Carroll, J. and Carpenter, B. (1999). Vector-Based Natural Language Call Routing. *Computational Linguistics*, 25(3):361–388.

Goel, V., Kuo, H.-K., Deligne, S., and Wu, S. (2005). Language Model Estimation for Optimizing End-to-End Performance of a Natural Language Call Routing System. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 565–568, Philadelphia.

Gorin, A. L., Riccardi, G., and Wright, J. H. (1997). How May I Help You? *Speech Communication*, 23:113–127.

Grice, H. P. (1975). Logic and Conversation. Syntax and Semantics. In Cole, P. and Morgan, J. L., editors, *Speech Acts*, volume 3, pages 41–58. Academic, New York.

Huerta, J., Akolkar, R., Faruquie, T., Kankar, P., Rajput, N., Raman, T., Udupa, R., and Verma, A. (2005). Reusable Dialog Component Framework for Rapid Voice Application Development. In *Proceedings of International SIGSOFT Symposium on Component-Based Software Engineering (CBSE)*, pages 306–321, St. Louis.

Levin, E., Pieraccini, R., and Eckert, W. (2000). A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23.

McTear, M. (2004). *Spoken Language Technology: Toward the Conversational User Interface*. Springer, London.

Meng, H. M., Wai, C., and Pieraccini, R. (2003). The Use of Belief Networks for Mixed-Initiative Dialog Modeling. *IEEE Transactions on Speech and Audio Processing*, 1(6):757–773.

Oviatt, S. L. (1995). Predicting Spoken Disfluencies during Human-Computer Interaction. *Computer Speech and Language*, 9:19–35.

Papineni, K., Roukos, S., and Ward, R. (1999). Free-Flow Dialog Management Using Forms. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 1411–1414, Budapest.

Pellom, B., Ward, W., and Pradhan, S. (2000). The CU Communicator: An Architecture for Dialog Systems. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 723–726, Beijing.

Pieraccini, R., Carpenter, B., Woudenberg, E., Caskey, S., Springer, S., Bloom, J., and Phillips, M. (2005). Multimodal Spoken Dialogue with Wireless Devices. In Minker, W., Bühler, D., and Dybkjær, L., editors, *Spoken Multimodal Human-Computer Dialogue in Mobile Environments*, pages 169–184. Springer.

Pieraccini, R., Caskey, S., Dayanidhi, K., Carpenter, B., and Phillips, M. (2001). ETUDE, a Recursive Dialog Manager with Embedded User Interface Patterns. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 244–247, Madonna di Campiglio, Trento.

Pieraccini, R., Levin, E., and Eckert, W. (1997). AMICA: The AT&T Mixed Initiative Conversational Architecture. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 1875–1878, Rhodes.

Sagawa, H., Mitamura, T., and Nyberg, E. (2004). Correction Grammars for Error Handling in a Speech Dialog System. In *Proceedings of Annual Meeting of the Human Language Technology Conference/North American chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 61–64, Boston.

Seneff, S., Lau, R., and Polifroni, J. (1999). Organization, Communication, and Control in the Galaxy-II Conversational System. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 1271–1274, Budapest.

Seneff, S. and Polifroni, J. (2000). Dialogue Management in the MERCURY Flight Reservation System. In *Proceedings of ANLP/NAACL*, pages 11–16, Seattle.

Stallard, D. (2001). Dialogue Management in the Talk'n Travel System. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 235–239, Madonna di Campiglio, Trento, Italy.

Wei, X. and Rudnicky, A. (2000). Task-Based Dialog Management using an Agenda. In *Proceedings of ANLP/NAACL*, pages 42–47, Seattle.

Williams, J. and Witt, S. (2004). A Comparison of Dialog Strategies for Call Routing. *International Journal of Speech Technology*, 7(1):9–24.

Xu, W. and Rudnicky, A. (2000). Language Modeling for Dialog System. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, volume 1, pages 118–121, Beijing.

Chapter 2

DESIGNING SPEECH-CONTROLLED MEDIA FILE SELECTION FOR AUTOMOTIVE SYSTEMS

Yu-Fang H. Wang and Stefan W. Hamerich
Harman/Becker Automotive Systems GmbH
Speech Dialog Systems
Dialog Research and Tools, Ulm, Germany
hwang@harmanbecker.com, shamerich@harmanbecker.com

Abstract We present the prototype of an in-car speech-controllable MP3 player. In addition to basic commands such as "next", "stop", or "repeat", one main feature of the system is the selection of titles, artists, albums, or genres by directly uttering them. We first describe the characteristics of automotive speech dialogue systems in general, their technology and design. We then motivate the need to design-implement-evaluate iterations as a proof-of-concept method and describe the design process of the implemented MP3 player dialogue. Finally, we discuss real-world challenges and future extensions for this application. The findings and considerations of the chapter straightforwardly extend to other embedded systems and general audio media.

Keywords: Automotive dialogue design; embedded speech recognition; ID3 tags; in-car speech dialogue systems; MP3; speech-based music selection; voice control

1. Introduction

The Speech Dialog Systems division[1] of Harman/Becker Automotive Systems (HBAS) is a leading manufacturer of speech control systems for the automotive market. Ever since joining Harman/Becker, the automotive division of Harman International Industries, HBAS also gained a strong foothold in the audio market. This provides the context for its work with voice-controlled audio systems.

The first product using HBAS' technology was the Linguatronic system available in Mercedes cars since 1996 (Heisterkamp, 2001). This first system

[1]Former Temic SDS GmbH – Speech Dialog Systems.

L. Dybkjær and W. Minker (eds.), Recent Trends in Discourse and Dialogue, 25–43.

enabled the hands-free usage of the built-in car phone and was already completely speaker-independent. The vocabulary comprised about 30 words and allowed continuous recognition of e.g. digit strings. Since then, these systems have become more and more complex. Voice control of the audio system in the car has become fairly common, and recent speech dialogue systems in infotainment systems even allow for navigation destination input via voice.

Nevertheless, a direct selection of audio tracks by voice is not yet offered in mobile or embedded products. Current audio systems with speech input only allow general commands such as "next title" or "previous CD". With compressed formats such as MP3, digital music has become omnipresent. The number of titles that can be stored on devices increases rapidly and this "classical" speech interface quickly becomes unwieldy. By contrast, direct selection of audio media by speaking title, artist, or album name offers an attractive and intuitive way to navigate through a mobile music collection.

In this chapter, we describe a prototypical speech dialogue system for selecting audio media in a car. The chapter will focus on dialogue but will also briefly discuss the challenges for such a system.

2. Related Work

Speech enabled media selection in embedded systems is not available as a product yet, although several concept studies have been made (see, e.g., McGlaun et al., 2001; Pieraccini et al., 2003).

On the other hand, projects are currently under way to allow speech-based media selection on server-based systems (e.g. refer to Baumann and Klüter, 2002; Schulz et al., 2004). Here, the main challenge for speech recognisers comes not so much from the extreme limitation of computing power and memory but rather from the additional distortion through channel coding and bandwidth limitation.

3. Speech Dialogue Systems for Cars

In-car speech dialogue systems have been available since 1996. These systems are embedded solutions, provided either as a separate hardware box (as shown in Figure 1) or integrated into infotainment systems running under a variety of real-time operating systems.

Cars are exposed to different climatic conditions, so their electronic components have to be robust and long-living. Moreover, electronic spare parts should be available for a car's lifetime. To meet these high-quality requirements, costly flash memory needs to be deployed; current speech control systems have to work on platforms that offer a standard memory size of as little as

Figure 1. Hardware speech dialogue module.

128 KB.[2] To run speech dialogue systems on such limited environments, special algorithms and tools are needed. HBAS' dialogue description language named GDML (Generic Dialogue Modelling Language) has been tailored to be used on embedded devices. Furthermore, dialogue descriptions and speech grammars are compiled to minimise resource consumption. For more details refer to Hamerich and Hanrieder (2004).

The integration of such systems in cars is one of the central issues. Basically, there are several bus systems in a car. The entertainment bus is used to control the built-in devices, e.g. loudspeakers, tuner, CD player, and speech control. It is independent of the motor control bus. For entertainment bus systems, several variants exist, among them D2B (Domestic Digital Data Bus), CAN (Controller Area Network), and MOST (Media Oriented Systems Transport).

3.1 Speech Recognition

Speech recognition performance in automobiles is in general subject to the same conditions as for standard application domains (cf. Jelinek, 1976). However, in-car ASR adds particular challenges since high recognition accuracy is required under aggravating conditions, namely noisy environment and very low computational costs. Additionally, speaker-independent recognition[3] and a close to real-time system response are taken for granted.

[2]More memory will be provided in the future since technology will be cheaper and certain applications such as media selection are simply memory-intensive. Nevertheless, the memory size for embedded devices is not comparable with that of a standard PC.

[3]This means the system can be used by new users without a preceding training session.

For optimal signal-to-noise ratio, the microphone should be positioned as close to the driver's mouth as possible. However, a close-talking microphone in a car is not a realistic option due to its lack of comfort. Regarding the obvious fact that each driver's seat position varies, it is not trivial to find the optimal microphone position. A further means to optimise recognition is the use of microphone arrays. For example, in the current Mercedes-Benz E-class, four microphones are installed in the rear-view mirror. About the peculiarities of microphone arrays in the automotive environment, see Nordholm et al. (2001). Even with the best microphone position, background noise, such as driving noise or wind, and echo from the audio source, such as radio or navigation prompts, need to be reduced, using echo cancellation and noise reduction algorithms, respectively (Hänsler and Schmidt, 2004).

With regard to the small amount of memory, acoustic models need to be extremely compact. Modelling approaches such as discriminative training get the most from only a few model parameters (Valtchev, 1995; Willett, 2004).

The StarRec recognition engine[4] features an advanced interface which allows to set timeout values and confidence thresholds when calling the recogniser. Generally, the best recognition result is transferred to the parser, in this way handing over the semantic representation to the dialogue manager. These semantic representations are language-independent, allowing for simultaneous dialogue development in several languages.

3.2 Automotive versus Server-Based Systems

Certain differences of automotive dialogues from server-based dialogues (such as telephony applications) must be taken into account:

As already mentioned, one difference is the available *memory and computing resources*. While there are practically no limitations for server-based systems, there are severe memory restrictions for in-car dialogue systems. Limitations on the target system often imply limitations on dialogue design – a small memory can mean a loss of certain dialogue features and thus convenience. For example, there might not be enough space for an extensive tutorial or demo mode.

An automotive dialogue system has to monitor *multiple modalities*. Apart from keeping track of the current dialogue state, it has to take into account both the user's haptic actions and the currently displayed information to compute the current system state. For instance, the radio can be turned on by speech or haptically.

The domain in telephony applications is often well known, e.g. in telephone banking, the callers can check their current balance. Automatic reservation

[4]A product from Harman/Becker.

systems will book a flight given the departure and arrival times, etc. As fallback, in all of these systems the caller will be connected to a human operator. However, applications in a car are less obvious to be used for two reasons: first, they do not offer a comparable fallback mode. Second, on-board devices become less self-explanatory as their number and complexity grows. This urges the need for our systems to provide a sophisticated help functionality.

Another challenge particular to automotive systems is the *limited attention* that the driver can pay to the system. The dialogues must be designed to minimise distraction from the road. Thus, the dialogues are typically much less complex than server-based dialogues. Sometimes they are reduced to very simple command & control dialogues.

An aspect that is realizable with automotive systems is the *personalisation* of certain user's data. A system can easily connect to a mobile phone or an MP3 player to read out the address book or the MP3 songs. This is particularly relevant for this chapter where songs from an MP3 collection are to be selected. With respect to these data, the system is always up to date.

3.3 Infotainment Systems and Deployment

Infotainment systems integrate several information and communication services, like telephone, tuner, CD player, CD changer, MP3 player, TV, and navigation.[5] Apart from the standard haptic use, these devices can be controlled by speech, some systems additionally allow for controlling the air condition or the seat heating by voice. In this way, the speech interface is an integral part of a multimodal HMI, allowing for a convenient and safe control of the attached devices in a car.

The most novel feature available in a car is speech-controlled navigation. A destination is entered by spelling the first few letters of a city name. In addition, the biggest, say 1,000, cities can be spoken as full words where the number of cities depends on the available memory. Spelling mode is necessary since for the speech recogniser there are too many possible recognition alternatives. For instance, there are more than 68,000 different city names for Germany. For embedded speech recognition systems, high accuracy word recognition of this vocabulary size is still a challenge but will be available in products soon.

The speech dialogue systems described above have to be ordered ex works. Meanwhile, many car manufacturers offer speech control systems as standard original equipment, e.g. the Linguatronic system for Mercedes-Benz cars. Figure 2 shows the current infotainment system of the Mercedes E-Class.

All available devices are accessible via buttons, while the display provides the current information. A dialogue is activated by pressing the *Push-to-talk*

[5]Refer to http://www.harmaninfotainment.com for further details.

Figure 2. Infotainment system (COMAND APS) in the Mercedes Benz E-class.

(PTT) button, located on the steering wheel for comfortable reach (not visible in the figure). As of today, keyword activation, which would render the PTT button superfluous, is not available.

Currently, applications based on the described technology are available with several car manufacturers, among them Audi, BMW, Lancia, Maybach, Mercedes-Benz, Porsche, Rolls-Royce, and also in the aftermarket. So far, all of these systems are still closed applications whose primary task is to control on-board devices. Even for navigation destination input, the domain and the vocabulary are known in advance. That is, there is no need to access data dynamically.

4. Automotive Dialogue Design

To assess the expected advantages in a car, let us recite Cameron (2000) for the conditions under which people would use speech:

- They are offered no alternative.
- It corresponds to the privacy of their surroundings and the task at hand.
- Their hands or eyes are busy on another task.
- It is quicker than any alternative.

These items correspond nicely to the situation in the car:

Safety: voice control contributes to safety because the driver can direct his eyes to the traffic and leave his hands on the steering wheel instead of being distracted by fumbling around with buttons.

Naturalness: speech provides a natural and quick access especially when the user is lost with an ever more complex haptic interface.

Shortcuts: while in haptic interfaces, the user has to move through menu levels. Speech provides an easy way to directly say what you want. For example, by saying: "dial number" a telephone number can be easily entered without understanding the menu structure of the complete infotainment system.

Since we depend on the quality of our products, proven usability of our systems is of vital importance. Therefore, the primary task of in-car dialogue design is to assure the fulfilment of the prerequisites for the above-mentioned conditions, which means to bridge the gap between the user's intuitive expectations of how the system works and the technical conditions of the system. It also means to cope with the innate, disadvantageous characteristics of speech: First, speech recognition is not perfect, and second, speech is not persistent.

In our understanding, speech does not compete with haptics, it offers an alternative communication channel to the already existing haptic interface. While simple operations like volume control might still preferably be done by turning the respective button, a more complex operation like destination input in a navigation system is more straightforwardly accomplished by speech. Since infotainment systems for cars become more and more complex, speech interfaces offer a natural interface to control such complex functions. On the other hand, speech is less appropriate to present information than a display.

For the design of an application, appropriate concepts are drawn from several sources, research findings, experience, and involvement of users in questionnaires or experiments.

First, there are established design principles, having emerged from research in general voice operated systems (refer, e.g. to Larsen, 2003; Suhm, 2003), as well as classical user interface design (see, e.g. Shneiderman, 2003).

One important principle is *consistency*: similar dialogue parts are modelled as analogously as possible to simplify the user's understanding of the system.

Next is *confirmation* of user input, e.g. there is acoustic or visual feedback on a user command. While it is superfluous when the command works as expected, the user feels abandoned when the system reaction deviates from the user's expectations. However, to decide on how much feedback should be implemented needs careful consideration and depends on the respective dialogue context.

The user can get *help* on commands that are possible in the current dialogue context.

Both system and user should be able to recover from errors by sophisticated *error handling* strategies.

Shortcuts can be used to allow for more than one way to have a command accomplished. They will be preferred by experienced users, while novices will rather fall into a guided dialogue mode.

Last, the principle *what you see is what you speak* ensures that all commands on a display or on device buttons are speakable.

Every new application has its particular needs. Established principles might not apply or may need to be modified, new features have to be adopted. To find out the requirements for a new application, we follow the design-implement-evaluate paradigm as described in Bernsen et al. (1998).

The initial design draft is the result of a user questionnaire, a Wizard of Oz (WOZ) experiment (Fraser and Gilbert, 1991), and expertise, where the current technological conditions of the target system have to be kept at the back of the designer's mind. It serves as starting point for further design evaluations to finally have a stable and user-approved version after a number of cycles.

The method finally chosen will depend on the quality of the dialogue issues and the number of open questions. We have used the WOZ technique for conducting usability tests (Petrik et al., 2005) under driving simulation conditions (Mattes, 2003). This has proven to be an indispensable tool, especially for data collection and validation of dialogue concepts before implementation.

5. Overall System Description

Since portable media players get more and more common, users wish to control these devices by speech as well. Such systems can be available nearly everywhere, carried by the user or integrated in an automotive audio system. In all cases, a huge collection of audio files can be stored on such devices. This could, for example, be a mobile MP3 player, a USB memory stick, a memory card, or the hard disk inside a car audio system or head unit. Several such media can be available in the system. Generally, it is required that certain meta information such as title, artist, and album is available. In case of MP3 files, these could be the ID3 tags that such files typically contain. If files do not come with meta information, e.g. titles from a CD, such information can be looked up in a database such as the Gracenote CDDB.

Our goal is to be able to select subsets from the entire music collection by voice. For example, it should be possible to play songs from a certain artist or all titles of a particular album or music of a certain genre. For this purpose, the system keeps a database of all available media on all connected devices with corresponding meta information. The information in the database is used to dynamically configure the speech recogniser.

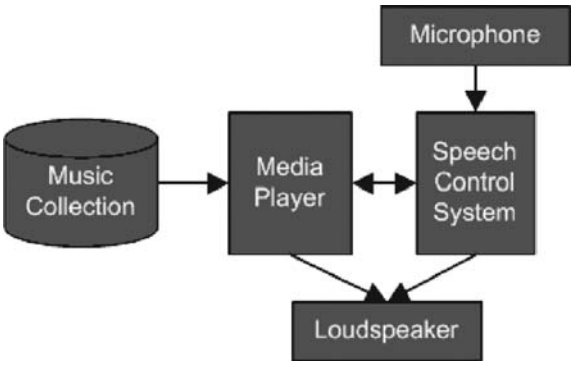


Figure 3. Schematic architecture of the system.

To provide the user with a comfortable speech interface, the main features of the system are:

- The intuitive dialogue aiming at minimising the steps necessary to accomplish a selection
- Enabling the user to select a subset of the collection even if it is several gigabytes large
- A special "more like this" feature which allows the user to select songs similar to the one currently playing
- The dynamic enrolment[6] of MP3 tags into the recogniser grammar
- The possibility to attach external USB memory devices (including an iPod)

A schematic overview of the system architecture is shown in Figure 3. The system runs on a PC simulation environment as well as on a 200 MHz SH4 embedded platform.

6. MP3 Dialogue Design

MP3 players have become very popular and are available both in software (e.g., Windows Media Player, iTunes) as well as in hardware. The iPod, for example, features an innovative user interface which allows the user to quickly navigate through the music collection. Similar interfaces for speech are not yet available. It is not clear what an intuitive speech interface looks like, nor what users expect from it.

[6]Enrolment means the addition of words as textual representations to the vocabulary at run-time.

In order to find answers to these questions, the MP3 dialogue design was investigated in two iterations. In the first round, a questionnaire was set up and evaluated to collect user requirements and expectations. Established dialogue design principles were applied to top off the dialogue design. Additionally, an alternative search mode was introduced to enable the user to handle large lists. Taking the concepts of the initial implementation as starting point, the second iteration was conducted within the framework of a master's thesis.

6.1 First Iteration

The following subsection describes the dialogue design process of the first prototype.

6.1.1 Questionnaire. In order to find out which MP3 functionalities are considered important for a speech-controlled MP3 application, the first round started with a user questionnaire which was handed out to employees of Harman/Becker in several countries. The majority of them were frequent MP3 users. The questionnaire covered a number of different features which can be grouped into several categories:

1. Selection of track, album, artist, genre or composer by speaking its name[7]
2. While a track is playing: read out information about the track, such as track number, album, or artist name on demand[8]
3. Read out a list of tracks, albums, artists, genres, or composers to the user on demand (as alternative to visual display)

As result, subjects regarded function category 1 as a required feature, while category 2 was considered nice to have. Number 3 was rejected by the majority. People said they would feel annoyed by long lists of names read out, and would rather prefer the visual display to get the respective information. Consequently, only the features of categories 1 and 2 were integrated into the first prototype, while type 3 was rejected.

Moreover, especially the frequent users required handling of user-defined playlists which will be integrated in a more advanced version of the application.

There was also a general consensus favouring simple, short dialogues. Users want to enjoy the music rather than being entangled in lengthy dialogues. This

[7]Users were also asked whether they would like to select a track by humming disregarding technical feasibility. The overwhelming majority rated this feature as nice to have but also as not necessary. Few users pointed out that it was difficult enough to sing a melody at all.

[8]This item accounts for multimodality to provide car-drivers with a second information source while driving (the respective information is also shown on display).

is somewhat contrary to, e.g., navigation destination input where people expect that separate dialogue steps are required to enter an address.

6.1.2 Dialogue. Based on the above mentioned findings, the MP3 player dialogue had the following prominent dialogue features:

- It let the user select tracks by album, track, artist, genre, or composer name.
- It provided conventional navigation commands while a track was played: "next", "previous", "skip", "shuffle", "stop" plus the respective album commands. Of course, it was also possible in this dialogue state (not only on top-level) to speak the selection and readout commands of category 1 and category 2 described above, such that the user could hear another track or change the current album.
- Of the category 2 features mentioned above, only the information of the concepts "name of current track" and "name of current artist" were provided.
- The system could also play songs of a similar genre like the current music files ("play me more like this"). There was no sophisticated mechanism to fulfil this request, instead, just a track with identical genre information would be selected. In reality, genre selection poses a non-trivial problem (see Section 7).

As is often the case, dialogue design had been a tightrope walk between satisfying the needs of a *novice* and an *expert*, respectively: while a novice needs guidance, an expert feels bored by repetitive actions of the dialogue system. The system did justice to both requests by offering a guided dialogue mode for the novice and shortcut commands for the expert where the decision for either mode was taken by the user's initiative. Moreover, having in mind that people mainly want a system to quickly accomplish a command rather than one that is overly verbose, the system was designed to support the users when necessary but not to bother them with too many questions or numerous dialogue turns. During the dialogue, we designed the system to offer little explicit support.

Help was provided, but in an unintrusive way:

- Visual help: when the system waited for user input, a help screen displaying currently speakable commands was shown (Figure 4). It was active by default but could be turned off (and on again) by voice-command.

Figure 4. Main display of the MP3 application.

- Dialogue help (timeout/nomatch): the system offered possible commands after a timeout period or if the system did not understand the user utterance.
- Explicit dialogue help: the user could always ask for help, which was given in a context sensitive form.

The default mode for the system was the play mode. So, after a command, such as "(play) album Live in Paris", the system would start playing the album immediately. The dialogue flow, as described so far, did not yet account for the possibly huge size of an MP3 collection. However, large lists imply a further use case besides just playing, namely the possibility to search for certain pieces of music. The system therefore offered a second command mode, the *browse mode*. It allowed to search through collections by subcategories, assuming a natural categorisation hierarchy: *genre* → *artist* → *album* → *track*. For example, the command "browse the genre Jazz"[9] showed all Jazz artists, and saying "browse the artist Diana Krall" would display the artist's albums. The command "play all" would play the current selection. At the bottom of the hierarchy, all tracks of an album were shown. The additional functionality has meanwhile be evaluated with regard to the naturalness of the subcategories as well as to the transparency of both command modes. Results regarding these issues are presented in the following subsection where final results of the second iteration are sketched.

[9]The keyword "browse" was required here.

To start the MP3 player application, the user could say "MP3 player" or just "music".[10] The system then prompted the user to speak, at the same time offering a help screen that displayed all possible selection criteria, as shown in Figure 4.

After having chosen a selection criterion (e.g. album or artist), the system provided the appropriate item list. In order to optimise speech recognition, the active vocabulary was adapted dynamically to the current selection. For example, the track names of a current selection, in that case the track names of Diana Krall's album "Live in Paris" (this album being the current selection in the example), were given preference to the track names of the whole, possibly huge MP3 collection.

6.1.3 Sample runs. To hear MP3 encoded music files, users could choose between the filters track, album, artist, genre, or composer which are regarded as the most common use cases. After the selection, the system prompted the user to speak the respective item. The help screen, shown in Figure 4, was active by default, in this way suggesting a menu-driven dialogue entry for a novice:

Sample A

User[1]:	Music Player
System[1]:	MP3 Player
Display[1]:	[see Figure 4]
User[2]:	Select an album
System[2]:	Which album?
Display[2]:	[shows available albums for choice]
User[3]:	Live in Paris
System[3]:	Now playing the album Live in Paris
System[3]:	[starts playing 1st track of album]
Display[3]:	[see Figure 5]

More experienced users could use shortcuts, ignoring the visual help:

Sample B

... (see dialogue step [1] above)	
User[2]:	Play me the album "Live in Paris"
System[2]:	Now playing the album Live in Paris
System[2]:	[starts playing 1st track of album]
Display[2]:	[see Figure 5]

[10] The MP3 application is part of a larger demo environment where several applications are available, but only one music application.

Figure 5. Display showing track information.

6.2 Second Iteration: Sketch of the Final Results

As mentioned earlier, the second iteration was realized as part of a Master's thesis (Schulz, 2006). It investigated how users would like to use speech in complex application domains such as media selection. It comprised the steps questionnaire, a WOZ experiment under driving simulation conditions, and a final evaluation. The prototype has been implemented completely in German.

With the dialogue concepts presented in the previous subsection as starting point, the relevant main findings are:

6.2.1 Play/browse hierarchy. In the first prototype, the browse mode as alternative mode for the play mode had been implemented. This turned out to be intransparent since both modes coexist in parallel, but only the play mode is active by default.

As a solution, the selection categories have to be treated differently: In the genre/artist case, people preferred the appropriate items of the subcategory to be displayed for further selection. In the album/track case, subjects found it natural when the system automatically started playing. As an additional result, the album category was the subjects' main selection criterion.

6.2.2 Explicit "play" command. In the WOZ test, the subjects had to explicitly say "play" after having successfully selected the required music items. The feature was clearly rejected. Instead, subjects' natural expectation was that the music would start playing immediately without any further commands, a circumstance especially true for the selection criteria track and album.

6.2.3 Creation of playlists. A few tasks required the subjects to create a playlist and add more items to it during driving. Subjects clearly disfavoured the option to create and edit playlists by voice.

6.2.4 Ambiguous input. Names are often ambiguous. For example, album and artist names are often identical such as in utterances like "Play Diana Krall". Subjects expected a clarification prompt to help them select the wanted item.

6.2.5 Display. The displayed information was perceived as highly distractive in a driving situation. Instead, users would prefer more voice feedback.

Taking these results into account, a further prototype is currently being developed.

7. Some Notes on MP3 Tags

One of the main challenges is making good use of the information contained in the MP3 tags. While there are tags for artist names, titles, album names, etc. the information is much less structured than one might expect. This leads to a number of challenges. A solution to these problems is the use of a database of meta information such as the CDDB by Gracenote. Embedded versions of this database are available and can be used to significantly improve the quality of the MP3 tags.

7.1 Recognition and Synthesis

One of the biggest challenges for both recognition and synthesis is finding the possible pronunciations for titles and artists. The current system focuses on the US market but multilingual issues occur even here. There is a large Spanish-speaking community in the USA and even French titles are not uncommon, e.g. titles by Celine Dion. Slang words are also very common in music titles and person names; even the human user is not always certain how to pronounce them. All these effects can occur within one track name, for example: "Femme like U". Another problem is creative orthography such as "M!ss Understood" or simply erroneous entries.

HBAS is working with Gracenote to provide a solution to these problems. Gracenote will be providing phonetic transcriptions for the problem cases, greatly enhancing the recognition accuracy as well as the speech output quality. If there is no database available, however, rule-based automatic transcription and exception dictionaries will be the fallback solution. See Figure 6 for a schematic overview of this process.

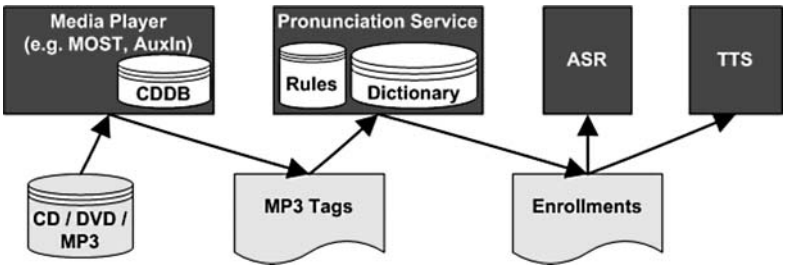


Figure 6. Schematic overview of the enrolment procedure of ID3 tags.

Partial matching is another issue. Many titles have longer official names than are usually spoken, such as "The Shoop Shoop Song (It's In His Kiss)". In such cases it is rare that the full title is spoken. Sometimes the title contains additional information that is not really part of the spoken name such as "All Cried Out (Unplugged)", or "Ka-Ching! (Red Disc)". The information can be used to distinguish different versions of the same title (sometimes even on the same album).

Here again, it is helpful to work with a partner who provides high quality meta information to ensure that names and titles are transcribed in a consistent way. This in turn makes it easier to parse the data and search through them.

7.2 Classification of Genre

The current system allows to play songs similar to the one currently playing. This is called the "play me more like this" feature. Although interesting approaches for automatic classification from the young field of music retrieval exist (Tzanetakis and Cook, 2002; Habich et al., 2005; Neumayer et al., 2005), they do not yet provide reliable results. Therefore, we use, so far, simple string matching which is based entirely on the genre tag contents of the songs.

Given a good genre classification, this can be very useful. Nevertheless, classification by string comparison is problematic for a number of reasons:

- Many users set up the genre descriptions of their MP3 files themselves, using the fact that the respective ID3 tag "genre" allows arbitrary contents.
- An attempt to establish a standardisation of genres existed for ID3v1, but was given up since it was found to be inconsistent and obsolete (ID3-Homepage, 2005). Furthermore, 79 genres are neither easy to remember nor fine-grained enough to support the "more like this" feature well.
- One and the same album or track can belong to different genres.

The reason for these problems lies in the fact that genre names are completely unrelated even for genres that are obviously similar, e.g. “Rock” and “Hard Rock”: the taxonomic nature of genres cannot be exploited since genre names are just strings.

Using a database of titles with a consistent and fine-grained genre classification will help to solve this problem.

8. Conclusion and Future Work

In this chapter an MP3 player for embedding into automotive systems was described, featuring the selection of titles and other selection criteria by speech. We described the embedded environment for such a system and the technology behind its implementation. To provide the appropriate background, general concepts of automotive dialogue design were sketched. Moreover, the dialogue design of the application was covered. Furthermore, we illustrated the challenges of speech-controllable ID3 tags.

To meet real-world requirements, several future extensions are planned. With regard to the possible commands that the system can understand, it will be extended by simple handling of playlists (“play my favourite playlist”), following the results of user questionnaires. Additionally, multilingual recognition and prompting are to be improved. Moreover, we want to go to the limits of the speech recogniser by allowing quasi unrestricted utterances such as “Play Live in Paris”. By contrast to example B above, no keyword indicates the filter criterion “album”, which clearly makes the recognition task more complex.

The initial design of the system presented here was based on user questionnaires, general design guidelines, designer’s expertise and intuition. It was therefore a first prototype that served as starting point for further design evaluations. A second prototype was then developed and usability tested. Based on these findings, a further improved implementation is under development.

Acknowledgements We thank the co-authors of our original SIGdial paper Marcus Hennecke and Volker Schubert for helpful discussions. Our application would not have been possible without the work of Peng Huang. Special thanks for his support during the implementation of our prototype to Andreas Löw. Last but not least we would like to thank Gerhard Hanrieder for supporting our work on this chapter.

References

Baumann, S. and Klüter, A. (2002). Super-convenience for Non-musicians: Querying MP3 and the Semantic Web. In *Proceedings of International Conference on Music Information Retrieval (ISMIR)*, pages 297–298, Paris.

Bernsen, N. O., Dybkjær, H., and Dybkjær, L. (1998). *Designing Interactive Speech Systems: From First Ideas to User Testing*. Springer, London.

Cameron, H. (2000). Speech at the Interface. In *Proceedings of COST249 Workshop on Voice Operated Telecom Services*, pages 1–7, Ghent.

Fraser, N. and Gilbert, N. (1991). Simulating Speech Systems. *Computer Speech and Language*, 5(1):81–99.

Habich, D., Lehner, W., Hinneburg, A., Kitzmantel, P., and Kimpl, M. (2005). Eyes4Ears - More than a Classical Music Retrieval System. In *Proceedings of the 5th Open Workshop of Musicnetwork – Integration of Music in Multimedia Applications (MUSICNETWORK)*, Vienna.

Hamerich, S. W. and Hanrieder, G. (2004). Modelling Generic Dialog Applications for Embedded Systems. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 237–240, Jeju Island.

Hänsler, E. and Schmidt, G. (2004). *Acoustic Echo and Noise Control*. Wiley, New York.

Heisterkamp, P. (2001). Linguatronic – Product-level Speech System for Mercedes-Benz Cars. In *Proceedings of International Conference on Human Language Technology Research (HLT)*, pages 1–2, San Diego.

ID3-Homepage (2005). http://www.id3.org/.

Jelinek, F. (1976). Continuous Speech Recognition by Statistical Methods. *Proceedings of the IEEE*, 64(4):532–556.

Larsen, L. B. (2003). *On the Usability of Spoken Dialogue Systems*. Ph.D. thesis, Faculty of Engineering and Science, Aalborg University, Aalborg.

Mattes, S. (2003). The Lane Change Task as a Tool for Driver Distraction Evaluation. In *Proceedings of Conference of the International Society for Occupational Ergonomics and Safety Conference (ISOES)*, pages 57–60, Munich.

McGlaun, G., Althoff, F., Rühl, H.-W., Alger, M., and Lang, M. (2001). A Generic Operation Concept for an Ergonomic Speech MMI under Fixed Constraints in the Automotive Environment. In *Proceedings of International Conference on Human-Computer Interaction (HCI)*, pages 288–290, New Orleans.

Neumayer, R., Lidy, T., and Rauber, A. (2005). Content-Based Organization of Digital Audio Collections. In *Proceedings of the 5th Open Workshop of Musicnetwork – Integration of Music in Multimedia Applications (MUSICNETWORK)*, Vienna.

Nordholm, S., Claesson, I., and Grbić, N. (2001). Optimal and Adaptive Microphone Arrays for Speech Input in Automobiles. In Brandstein, M. and Ward, D., editors, *Microphone Arrays: Signal Processing Techniques and Applications*, pages 307–330. Springer.

Petrik, S., Wang, Y.-F. H., and Hamerich, S. W. (2005). Hierarchical Dialogue Structure Representation for Wizard of Oz Experiments in Speech Dialogue Systems. In *Proceedings of International Conference on Speech and Computer (SPECOM)*, pages 207–210, Patras.

Pieraccini, R., Dayanidhi, K., Bloom, J., Dahan, J.-G., Phillips, M., Goodman, B. R., and Prasad, K. V. (2003). A Multimodal Conversational Interface for a Concept Vehicle. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 2233–2236, Geneva.

Schulz, C. H., Rubinstein, D., Diamantakos, D., Kaisser, M., Schehl, J., Romanelli, M., Kleinbauer, T., Klüter, A., Klakow, D., Becker, T., and Alexandersson, J. (2004). A Spoken Language Front-end for a Multilingual Music Data Base. In *Proceedings of Berliner XML-Tage*, pages 276–290, Berlin.

Schulz, S. (2006). Sprachgesteuerte Navigation in Komplexen Strukturen am Beispiel eines MP3-Players [in German]. Master's thesis, Faculty of Computer Science, TU Dresden, Dresden.

Shneiderman, B. (2003). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, Reading.

Suhm, B. (2003). Towards Best Practices for Speech User Interface Design. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 2217–2220, Geneva.

Tzanetakis, G. and Cook, P. (2002). Musical Genre Classification of Audio Signals. *IEEE Transactions on Speech and Audio Processing*, 10(5): 293–302.

Valtchev, V. (1995). *Discriminative Methods in HMM-based Speech Recognition*. PhD thesis, Engineering Department, Cambridge University, Cambridge.

Willett, D. (2004). Error-weighted Discriminative Training for HMM Parameter Estimation. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 1661–1664, Jeju Island.

Chapter 3

A VIRTUAL HUMAN DIALOGUE MODEL FOR NON-TEAM INTERACTION

David Traum, William Swartout, Jonathan Gratch and Stacy Marsella
University of Southern California
Marina del Rey, CA, USA
traum@ict.usc.edu, swartout@ict.usc.edu, gratch@ict.usc.edu, marsella@ict.usc.edu

Abstract We describe the dialogue model for the virtual humans developed at the Institute for Creative Technologies at the University of Southern California. The dialogue model contains a rich set of information state and dialogue moves to allow a wide range of behaviour in multimodal, multiparty interaction. We extend this model to enable non-team negotiation, using ideas from social science literature on negotiation and implemented strategies and dialogue moves for this area. We present a virtual human doctor who uses this model to engage in multimodal negotiation dialogue with people from other organisations. The doctor is part of the SASO-ST system, used for training for non-team interactions.

Keywords: Dialogue; negotiation; virtual humans; embodied conversational agents

1. Introduction

Virtual Humans (Rickel and Johnson, 1999b) are autonomous agents who can play the role of people in simulations or games. These agents generally have some or all of the following properties:

- Humanoid body (either a physical robot, or animated body in a virtual environment)
- Cognitive state, including beliefs, desires or goals, intentions, and perhaps other attitudes
- Embeddedness in the real or a virtual world

L. Dybkjær and W. Minker (eds.), Recent Trends in Discourse and Dialogue, 45–67.

- Interactivity with the world (or a virtual world), other virtual humans, and real people, including perception of events and communication, and ability to manipulate the world and/or communicate with outers
- Believable human-like behaviour, including affective reasoning and behaviour

Virtual humans can play an important role in helping train skills of interacting with others who have different beliefs, goals, and styles of behaviour. By building virtual humans that are not just humanoid in appearance and external behaviour, but which also have internal models (including beliefs, goals, plans, and emotions) and ability to reason over these models and formulate appropriate strategies and behaviours on the basis of the models and perceptual input, virtual humans can behave appropriately for a range of social relationships. These kinds of agents have also been referred to by similar terms, including animated agents (Rickel and Johnson, 1999a) or embodied conversational agents (Cassell et al., 2000).

With respect to the dialogue capability, virtual humans have a number of similarities with both task-oriented dialogue systems and chatterbots. Like task-oriented dialogue systems, they generally have knowledge of tasks, and models of the steps involved in the task and how to talk about them. However, generally task-oriented dialogue systems strive to solve the problem as efficiently as possible, minimizing the opportunity for misunderstanding, even if this leads to unnatural and un-human-like dialogue. On the other hand, virtual humans strive for human-like dialogue so as to train communication behaviours that might transfer to real human interaction. Moreover, for training, efficiency in task performance and brevity is not necessarily an advantage – the longer the interaction the more opportunity for learning. Like chatterbots, virtual humans have a focus on believable conversation, but their purpose is not to convince someone that they are actually human, but merely serve as competent role-players to allow people to have a useful interactive experience.

Our virtual humans have been developed incrementally over a number of years, with developments being made in several aspects (Rickel and Johnson, 1999a; Hill, 2000; Rickel et al., 2002; Traum and Rickel, 2002; Traum et al., 2003; Gratch and Marsella, 2004). These virtual humans are embedded in a dynamic virtual world, in which events can happen, agents can perform actions, and humans and virtual humans can speak to each other and communicate using verbal and non-verbal means. The virtual humans are extensions of the Steve agent (Rickel and Johnson, 1999a), and include sophisticated models of emotion reasoning (Gratch and Marsella, 2004), dialogue reasoning (Traum and Rickel, 2002) and a model of team negotiation (Traum et al., 2003). Agents use a rich model of dialogue closely linked with a task model and emotional

appraisals and coping strategies for both interpretation of utterances as well as for decisions about when the agent should speak and what to say.

In previous work (Rickel et al., 2002; Traum et al., 2003), we described a negotiation model that could allow virtual humans to engage as teammates. To negotiate and collaborate with humans and artificial agents, virtual humans must understand not only the task under discussion but also the underlying motivations, beliefs and even emotions of other agents. The virtual human models build on the causal representations developed for decision-theoretic planning and augment them with methods that explicitly model commitments to beliefs and intentions. Plan representations provide a concise representation of the causal relationship between events and states, key for assessing the relevance of events to an agent's goals and for assessing causal attributions. Plan representations also lie at the heart of many reasoning techniques (e.g., planning, explanation, natural language processing) and facilitate their integration. The decision-theoretic concepts of utility and probability are key for modelling non-determinism and for assessing the value of alternative negotiation choices. Explicit representations of intentions and beliefs are critical for negotiation and for assessing blame when negotiations fail (Mao and Gratch, 2004).

This model assumed that teammates shared common end goals, participated in a social institution with roles that the participants played, and had strong trust in the other teammates' abilities and veracity. It did not address how virtual humans might interact in the case where these factors were lacking, and how to begin to form them through interaction.

In this chapter, we extend the dialogue model to allow for non-team negotiation. The extended model allows for the case in which relationships may need to be developed during the interaction, and in which the virtual human's behaviour may be very different depending on the nature and strength of the relationships. We also present Dr Perez, an implemented virtual human who uses this model to negotiate in a prototype training application.

In the next section, we describe the information state dialogue model for virtual humans. This includes both aspects of information state and dialogue moves. In Section 3, we describe how this model is used in understanding and producing communicative behaviour. In Section 4, we discuss non-team negotiation. After a brief survey of literature in the area, we describe our domain testbed and then our first synthesis of this work in terms of strategies for virtual humans, and then extensions to the dialogue model to make use of these strategies. In Section 5, we show two example interactions with this agent, showing how the dynamic trust model is developed during the interaction and how this can affect the agent's choice of utterance. We conclude with some brief remarks about evaluation and future directions.

2. Dialogue Model

Our virtual human dialogue model uses the *Information state approach* (Larsson and Traum, 2000; Traum and Larsson, 2003). In this approach, dialogue is modelled using the following aspects:

- An Information State – including representations of the information used to model dialogue context, distinguishing one (point in a) dialogue from another
- A set of dialogue moves, which represent contributions to dialogue and packages of change to the information state
- A set of rules (or other decision procedures) for modelling the dynamics of dialogue, including the following types of rules:
  - Recognition rules – that interpret raw communication input (e.g., speech, text, gestures) as dialogue moves
  - Update rules – that govern the change in information state based on observation of dialogue acts
  - Selection rules – that choose a set of dialogue acts to perform, given a configuration of the information state
  - Realization rules – that produce communicative output behaviour that will perform the set of dialogue moves

  Rules have a condition part (that specifies constraints on the information state that must be satisfied in order for the rule to fire) and an effect part (that specifies how the information state changes when the rule applies)
- An algorithm that specifies the order and priority of rule application

There are several toolkits that allow one to specify an information state, dialogue moves, rules, and an algorithm, in order to create an information state dialogue system. These include TrindiKit (Larsson et al., 1999), Dipper (Bos et al., 2003) and Midiki (Midiki Users Manual, 2005). Rather than using one of these toolkits, our dialogue manager is implemented in SOAR (Laird et al., 1987). Like these information state toolkits, SOAR has an information state, consisting of objects with links to values and other objects. In this sense it is very much like the information state of Godis (Cooper and Larsson, 1999) and EDIS (Matheson et al., 2000) which are based primarily on AVM-like record structures. SOAR also is a rule-based language. SOAR's main algorithm is to apply all rules simultaneously, and order of application is achieved by referring to dynamic aspects of the information state in the condition parts of the rule. For example, if rule 1 has a condition that requires the presence of a particular

value in the information state and that value is only set by rule 2, then rule 2 will fire before rule 1. While the main body of dialogue processing is achieved by application of rules in SOAR, there are also other computational mechanisms that can be used, e.g., general programs in TCL, and an input/output interface that can send and receive information from external system modules written in any language.

There are two main differences in our virtual human dialogue model that distinguish it from most other information state based dialogue managers. First, the information state and sets of dialogue moves are divided into a number of *layers*, each covering a different aspect of communication (Traum and Rickel, 2002). We believe the scope and breadth of these layers exceeds any other implemented dialogue system in terms of the range of phenomena modelled, allowing our virtual humans to engage in multiparty dialogue, multiple, temporally overlapping conversations, and both team and non-team negotiation. Second, many other parts of the virtual human model, including task reasoning, planning, emotion reasoning, and goal-directed behaviour are also represented in the same information state approach within SOAR as the dialogue model, allowing very rich interaction between these components. Dialogue rules may make use of these aspects of the information state in all phases of processing, from recognition of dialogue moves to generating behaviour.

In the rest of this section, we give an overview of the aspects of information state and dialogue moves that are most important for dialogue processing. In the next section we overview the arrangement of dialogue processing rules.

2.1 Information State Aspects

The top level of the dialogue information state includes a number of aspects including **Ontology, Lexicon, Participants, Social State, Speech Event History, Conversation(s)**, and **Social Planning**. The ontology contains mostly static information about subcategorizations, including selection restrictions of roles for events, and group membership. The lexicon maps words from English and the external recognisers to the internal task and dialogue ontology. The participants list keeps track of all participants (both real and virtual) in the simulation, including information about distance and accessibility for contact, and hypotheses about current gaze and attention of the participants. Social state information includes both the roles and relationships that participants hold to tasks and each other, as well as the obligations and social commitments to propositions that participants hold toward each other.

Multiple conversations can be active at a time, and each one has its own internal structure. Conversation structure includes

- A list of participants in the conversation (who are assumed to understand the grounded contributions), divided into active participants who

perform speaker and/or addressee roles in utterances of the conversation, and overhearers (who do not)

- Modality of the conversation (face to face, radio, etc.)
- The turn-holder (a participant, or none)
- The initiative-holder (a participant or none)
- The purpose of the conversation (e.g. to negotiate a task), if any
- A history of utterances that are part of the conversation
- A history of concept mention
- A structure of questions under discussion
- A grounding structure, consisting of a bounded stack of common ground units (Traum, 1994)

The social planning structure contains information useful for planning and recognising future dialogue actions. The main aspects are:

- A set of *potential obligations*, including actual discourse obligations (Traum and Allen, 1994), as well as those that would be established if an open grounding unit were to be grounded and those that would be established based on conditional rules if the antecedent is planned
- A set of expectations of what is likely to be said next, following from what has been said (e.g., reactions to a suggestion, or discussion of a next step in a plan after the current topic of discussion
- An agenda of partially ordered dialogue goals

The goals on the agenda can come from domain goals in the task model (including various types of communication, such as getting another agent to do something, agreeing on a solution, getting permission, or seeking knowledge), the emotion model, or other aspects of the agent's reasoning process. The agenda is used both for generating new initiatives (see Section 3.1), and as a further source of expectations for use in interpreting utterances that do not refer to the context of what has been recently said or observed, in a manner similar to the account of plan and question accommodation in Larsson (2002).

In addition to these aspects of the dialogue information state, dialogue processing also makes use of a number of information state elements from other modules, including a causal history of past events, the current world state, and plans. Also used are assessments of utility of possible actions and emotional appraisals of potential actions.

2.2 Dialogue Moves

The dialogue model includes multiple layers of interaction, each with associated parts of the information state and dialogue moves. These layers are described in more detail in Traum and Rickel (2002). Figure 1 shows the set of acts in each layer used in the current implementation.

The forward and backwards acts together are classed as *core speech acts*, while the other classes are grouped together as other *dialogue acts*. Core speech acts are most directly connected to the social state part of the information state, adding and relieving obligations, social commitments, and affecting social relations. These acts also have functions related to influencing the topics under discussion in the conversations that they are a part of.

Core speech acts have a content which is either a state, an action description or a question about one of these. Each of the states and actions in the task model is annotated with semantic information that can be used to describe and recognise description of those states in natural language (and our speech-act based agent communication language). Speech recognition and natural language interpretation produces similar contents from spoken utterances. Dialogue processing then compares the NL representation to the relevant task model representations, and, if a sufficiently close match can be found with a task model state or action, that is seen as the referent.

Unlike many accounts of the effects of these speech acts (e.g. Cohen and Perrault, 1979; Allen, 1983; Cohen and Levesque, 1990; Fipa, 1997), there are no direct effects on the beliefs, desires or intentions of the conversational participants. This allows for the possibility that participants are insincere in their utterances. Following Traum and Allen (1994), the direct effects involve social commitments, and one may then infer from these commitments the beliefs or intentions commonly associated with these utterance types, given additional assumptions.

forward acts	assert, info-req, order, request, thank, greeting, closing, express, check, suggest, promise, offer, apology, encourage, accuse, intro-topic, avoid
backward acts	accept, reject, address, answer, divert, counterpropose, hold, clarify-parameter, redirect, confirm
conversation	start-conversation, end-conversation, confirm-start, deny-start, pre-close
grounding	initiate, continue, repair, acknowledge, request-repair, cancel
turn-taking	keep-turn, hold-turn, release-turn, assign-turn
initiative	take-initiative

Figure 1. Types of dialogue moves.

Assertions will have the effect of establishing a commitment by the speaker that the state holds, or that action happened, is happening, will happen, or should happen, depending on the tense and aspect of the utterance. **Info-requests** have a question as their contents. Questions are (possibly partial) propositions together with a designated *q-slot* indicating the part of the proposition asked about. Info-requests have as their effect an obligation to address the question. **Requests** have an action as content, and the effect is an obligation to address the request, e.g., to consider and give feedback on the request. **Orders**, which can only be performed by a superior to a subordinate in the social structure, have as their effect an obligation to perform the action that is its content. **Suggestions** do not impose obligations, but do focus the topic on the action.

3. Dialogue Processing

Language processing occurs in two distinct and interleavable "cycles", one for understanding language and updating the information state, and a second for producing language. This separation of input and output processing cycles allows the agent to have an arbitrary interleaving of contributions by itself and others rather than enforcing a rigid turn-alternation. Each communicative contribution is simultaneously interpreted at each layer, and may correspond to a number of acts at different layers. The interpretation cycle includes stages for speech recognition, semantic parsing, contextual processing (including reference resolution, intention recognition and dialogue act interpretation), and finally updating the information state.

Generation usually starts from an intention to perform one or a small set of acts, however any realized utterance will also correspond to a number of acts, some of which (e.g., turn-taking) may be as much a result of the timing of the performance with respect to other events as to the planned behaviour. Generation proceeds from one of two sources: *reactions* to utterances of others and events in the virtual world, and *initiatives* which proceed from the agents own goals and agenda.

There are different sorts of reactions which are prompted by different aspects of the dialogue information state, including returning greetings, grounding understood material and repairing material that was not understood, addressing obligations, and reacting to proposals. Some are predicated on the agent having the turn (or at least another agent not having the turn), while others (e.g., repairing errors, reacting to danger) are not, and can produce interruptions. There is also a partial order priority scheme so that, e.g., addressing an obligation takes priority over merely acknowledging that a question has been asked.

3.1 Initiative Model

The initiative model consists of three main components, which handle three central problems:

- What to talk about
- When to talk about it
- How to talk about

The first component is modelled by the agenda, mentioned in Section 2.1. Whenever the agent forms communicative goals, these are added to the agenda. Goals on the agenda may be ordered with respect to each other, and special track is kept of all possible next items and the current agenda item that the agent is focused on. For each item, a record is also kept of how many times and in which ways this item has been talked about.

There are multiple ways to trigger the agent to take the initiative. Policies can be set to monitor certain conditions of the dialogue, other mental state, or the environment. These initiative policies can also be individually turned on or off according to a threshold for initiative level, and a current initiative level that is part of the agent's personality profile. Currently the following policies are used:

- When a threshold for too much silence has been exceeded: this is used to insure that a conversation will not stagnate, even if a user does not know what to say.
- When a threshold for too many cumulative errors in understanding has passed: this is used to take control when the user is having problems understanding or making him or herself understood.
- When a threshold for too many consecutive irrelevant utterances has been exceeded: this is used to ensure that the conversation does not drift off the topic of the conversation. In some cases a user may be trying to speak relevantly, but goes beyond the comprehension capability of the agent, either by using unknown vocabulary, constructions, or implicit connections that the agent's inferential power is unable to connect. In all of these cases having the agent take the initiative often makes the conversation more fluent.
- At the directive of a human controller or director agent: this is used to trigger initiative "manually" or for some other reason outside of the reasoning of the agent itself.

When an initiative trigger is reached, the agent will choose an item at the top of the agenda (usually the current item, if there is one), and proceed to take

the initiative. There is still the issue of how to address the agenda item. This depends in part on what kind of item it is and other factors of the domain and context for the conversation. In general, though, there is a cascade of several different modes and a count of how many times an agent will use that mode. For instance, in the MRE domain (Traum et al., 2004), when the Sergeant wants to propose an item that he would like the Lieutenant (his superior, but also his trainee) to do, he sequentially uses the following modes to bring up the desired action:

1 Hint: mention an end goal of the action (according to the task model) that is currently unfulfilled, or a pre-condition of the action that has already been met, thus enabling the action.

2 Suggest: directly suggest the action itself as a possibility.

3 Request: specifically request permission to perform the action.

4 Perform: perform the action without authorization (unless specifically prohibited).

Together, these factors of the initiative model allow the agents to engage in mixed-initiative dialogue, with the level of initiative that the agent takes being a factor both of customizable parameters as well as dynamic conditions of the dialogue.

4. Non-Team Negotiation

The model presented in the previous sections was designed mainly for team interaction, where it is assumed that the teammates have the same general goals, although they may disagree and negotiate about the best ways to achieve those goals. For more general situations, we must generalize the model of negotiation to include neutral and adversarial conditions.

4.1 Orientations Toward Negotiation

One of the central ways to characterize negotiation under adversarial conditions is with respect to the tension between competition and cooperation. Negotiators may have different goals, perceive themselves in conflict over those goals but may also perceive the need to cooperate to some degree to achieve their goals. In this view, one can characterize the state of a negotiation process from the perspective of the competitive/cooperative orientation of the parties to the negotiation and the strategies they employ in light of those orientations. Specifically, one oft-made distinction is between integrative and distributive (Walton and Mckersie, 1965) situations. If a negotiation is a win–lose game where there is a fixed value to be distributed, then it is called distributive.

There will be a winner and a loser. In contrast, an integrative situation is one where both sides can potentially win, a win–win situation where negotiation could add value and be of benefit to both sides. These basic distinctions presume some commitment to engage in negotiation. However, an individual may simply believe that there is no possible benefit or even need to negotiate. This individual may have an orientation to simply avoid the negotiation or deny the need for it, what is termed avoidance (e.g. Sillars et al., 1982). We thus start with three basic orientations toward a negotiation: avoidance, distributive, and integrative. Whenever an agent seriously considers a negotiation situation it will choose one of these three orientations.

Negotiators may perceive a situation as one to be avoided, or as a distributive or integrative situation regardless of whether this reflects the true situation. Changing the perceptions of other agents is often one of the main tasks in a successful negotiation. Based on current perceptions, people tend to use a range of dialogue tactics consistent with their orientations (Putnam and Jones, 1982; Sillars et al., 1982). Avoidance tactics include shifting the focus of conversation and delays. Distributive tactics can include various defensive moves such as stating prior commitments that bind the negotiator or arguments that support the negotiator's position. Distributive tactics can also be more offensive, such as threats, criticisms and insults. Integrative tactics are more cooperative with negotiators actually attempting to see issues from the other's perspective. Tactics can be arguments that support the other's position, acceptances of offers, offers of support, etc. Note at a finer grain of analysis, the tactics employed have both instrumental and affective components. For example, distributive tactics, besides trying to gain competitive advantage, tend to be associated with angry or intimidating behaviour whereas the integrative tactics try to promote a positive affective climate (Putnam and Jones, 1982).

Negotiators will often shift orientations during the course of a negotiation. Several factors have been identified as being critical to moving towards an integrative orientation, including acts of reciprocity, establishing trust and reinforcing shared goals (e.g. Wilson and Putnam, 1990).

4.2 Domain Testbed: Support Operations

Whether it is Kosovo, East Timor, or Iraq, one lesson that has emerged from attempts at "peacemaking" is that negotiation skills are needed across all levels of civilian and government organisations involved. To have a lasting positive effect, interactions between military and locals must be carried out in a way that generates goodwill and trust. We have selected this general class of operations as a testbed for our work on negotiation.

More specifically, we are developing a training scenario in which a local military commander (who has a rank of captain) must negotiate with a medical

Figure 2. VR clinic and virtual human doctor.

relief organisation. A virtual human plays the role of a doctor running a clinic. A human trainee plays the role of the captain, and is supposed to negotiate with the doctor to get him to move the clinic, which could be damaged by a planned military operation. Ideally, the captain will convince the doctor without resorting to force or threats and without revealing information about the planned operation. Figure 2 shows the trainee's view of the doctor in his office inside the clinic. The success of the negotiation will depend on the trainee's ability to follow good negotiating techniques, when confronted with different types of behaviour from the virtual doctor.

4.3 Negotiation Strategies for Virtual Humans

One of our first steps toward implementing a virtual doctor character was to analyze how people act in that role. To this end, we have conducted a series of role-play sessions, in which one person plays the role of the captain while another plays the role of doctor. Each is given a short set of instructions with different background information, goals, and resources for the negotiation, but given freedom as to how to conduct the negotiation and react to their partner. In these dialogues we can see examples of each of the orientations described in the previous section. For example, in the first one, the doctor displays an avoidance orientation, and is able to divert the topic of the conversation from the move to the military's role in upcoming operations for over 10 turns (only the first few are shown here). In the second one, we see a doctor illustrating the

distributive orientation, contesting the basic facts and goals rather than working together on common issues. In the third one, we see an example of integrative orientation, the doctor having accepted the danger of the current location and willing to meet the captain's goals if his own are also addressed.

(1) C: It's a temporary move, once the battle is over, you will be moved back.
D: Why don't you cancel your battle? Why don't you not kill these people.
C: We're not the ones deciding the battle.
D: You're the ones here. You're telling me this.

(2) C: We need to move as soon as possible. There are insurgents in the area. This is very unsafe, you're putting yourself and your patients in danger.
D: Why? I don't want to move. I have all these patients here. They won't move, if I move who would who could save them?
C: Sir, everyone is in danger! If we stay here there's ...
D: I'm not in danger.

(3) C: Insurgents will not hesitate to harm civilians if that's their path that they need to take. They won't hesitate to harm doctors, a doctor or even injured patients if they feel that's the the means to their end.
D: Well.
C: This is why you need to come to us.
D: I think we can make a deal. You can give me medical supply, and then we can go with you. I need supplies as soon as possible. As you can see, we are running out of supplies.

We have developed *strategies* for each of these orientations. Our virtual humans can use the strategies to adjust their behaviour toward the orientations described above. A strategy consists of several aspects including: **entry conditions**, which indicate when adoption is appropriate; **exit conditions**, which indicate when the strategy should be dropped (often in favour of more appropriate strategies); **associated moves**, which can be performed as tactics to implement the strategy; and **influences** of the strategy on behaviour and reasoning. These aspects result from the underlying emotion and dialogue models of the virtual humans.

The EMA (**EM**otion and **A**daptation) model of emotion (Gratch and Marsella, 2004) describes how coping strategies arise as cognitive and physical responses to important events, based on the appraisal (Scherer et al., 2001) of perceptions related to goals and beliefs. Appraisal characterizes events in terms of variables that guide the selection of an appropriate response (e.g., is this desirable? can it be avoided?), but the event need not be physical. Negotiation

strategies can thus be seen as types of coping strategies in which the event in question is the negotiation itself, and moves are the types of dialogue actions an agent will perform as part of a negotiation.

The avoidance orientation arises from an appraisal that the negotiation is undesirable but avoidable. The main motivation is to try to escape from the negotiation. When this appraisal is active, the agent chooses an **avoidance** strategy. Exit conditions will be the negation of either of the entry conditions — when the agent believes either that the negotiation has some utility or that it is not avoidable, the agent will abandon the avoidance strategy. The avoidance strategy involves attempts to change the topic of a conversation or get out of it entirely. When applying the avoidance strategy an agent will refrain from commenting on the object of negotiation, even to refute claims.

When in distributive mode, the agent will attempt to "win" rather than "lose" the negotiation. This can be associated with several strategies, depending on the type of decisions to be made and the range of possible alternatives. An *attack* strategy is appropriate when the appraisal is that a negotiation is not avoidable and the proposal is undesirable. Other strategies are also appropriate for a distributive orientation, including defence against a threat rather than attack, or making unreasonable demands in the hope the other party will drop the negotiation. We defer this for future work. One should drop an attack strategy when either the negotiation becomes desirable, or it becomes more profitable to avoid (or defend) than attack. The attack strategy involves pointing out the reasons why a proposal is flawed, or ad hominem attacks on the negotiator.

An integrative orientation leads to attempts to satisfy the goals of each of the participants. The **negotiate** strategy is appropriate when an agent thinks there is a possible value to the negotiation — e.g., there is a higher expected utility from the expected outcomes than would be the case without the negotiation. This strategy is dropped either when the perceived utility of continuing to negotiate drops below a threshold, or when the negotiation has been completed. Moves in the negotiation strategy involve problem solving and bargaining, much in the manner of the team negotiation in Traum et al. (2003).

The success of a negotiation is also mediated by factors that influence the perceived trust between parties, including a belief in shared goals, credibility and interdependence. The doctor is unlikely to be swayed by an offer of aid if he does not believe the captain can and will fulfil his commitments. Trust issues are pervasive throughout the negotiation, since there is usually not much point in negotiating with someone you expect to lie, be ill-disposed toward you, or not keep their side of a bargain.

4.4 Extensions to the Dialogue Model

Several extensions to the dialogue model were needed to handle possibly adversarial negotiation and the types of phenomena occurring in this domain. The most basic is sensitivity to the dialogue strategy, which involves overriding some basic reaction rules in some cases. For example, when applying the avoidance strategy one must not directly address a proposal that is on the topic of avoidance. Sometimes these utterances are not even grounded as a way of avoiding talking about an unpleasant topic. In this section we will examine two other extensions: updates to the initiative model and a new type of backward dialogue act for intentionally flouting the Gricean maxim of cooperativity (Grice, 1975).

4.4.1 Modelling trust. According to the dialogue model in Matheson et al. (2000), the direct effect of an assertion is the introduction of a commitment, whether or not either party believes in the assertion. While this is sufficient for reasoning about the claims and responsibility for information, we need to go further and potentially change beliefs and intentions based on communicated information. Trust is used to decide whether to adopt a new belief based on the commitments of another.

Similar to Marsella et al. (2004) and Cassell and Bickmore (2003), trust is modelled as function of underlying variables that are easily derived from our task and dialogue representations. *Solidarity* is a measure of the extent to which parties have shared goals. It is derived from a running tally of how many times the trainee makes assertions or demands that are congruent with the agent's goals. *Credibility* is a measure of the extent to which a party makes believable claims. It is derived from a running tally of how many times the trainee makes assertions that are consistent with the agent's beliefs. Finally, *familiarity* is a measure of the extent to which a party obeys norms of politeness. Currently, an overall measure of trust is derived as a linear combination of these three factors.

4.4.2 Extended initiative model. For properly modelling negotiation strategies the initiative model must be changed from that of team collaboration. The basic mechanism remains in place, however the current strategy is also made part of the agenda. Some agenda items are tied to particular strategies. For the avoidance strategy, initiative will concern the agent's own goals that are unrelated to the topic of negotiation. The irrelevance threshold is also disabled, as this strategy "succeeds" when the agent is able to shift the topic somewhere else.

For the attack strategy, each problem that the agent foresees with the action that is the topic of negotiation is added to the agenda. These problems include

- Pre-conditions of the action that are not met
- Other, better plans (according to the utility calculations in the task model)
- Undesirable side effects of the action

Another general agenda item for this domain in particular, is a desire for Doctor Perez to get back to his patients and end the conversation. This generally takes lower priority than more specific items within a strategy, but can come out if no other agenda items are available. Following the model in Section 3.1, there are several levels of utterances to support this goal, ranging from pre-closing reminders that he is busy, to finally ending the conversation.

4.4.3 Avoidance moves. Another important extension to the dialogue model is the addition of an *avoidance* dialogue act. According to Traum and Allen (1994), when presented with a request or question, one has an obligation to address this utterance, though not necessarily accept it and perform the desired act. There are many cases in which one might not want to perform the act. Rejection is an option which addresses the speech act, but may have negative consequences of its own. First, it commits the speaker to a negative position, which may not be desired. Second, it serves as a face threat (Brown and Levinson, 1978). A third option is to try to avoid an explicit commitment. One way to do this is by deferring a resolution to the future (e.g., "let me get back to you about that"), which does also commit one, and also may not be satisfactory to the interlocutor. Another type of action is to attempt to change the topic altogether. Depending on how explicit the request is, this may or may not go against an obligation to address. Thus, mentioning a topic indirectly is more polite than a direct request, because it does not force someone to not meet an obligation when avoiding.

We can see examples of avoidance from both participants in the dialogue in Figure 4. In turns 8, 10, and 12 the doctor is avoiding the move-clinic topic, following an explicit strategy to talk about something else. In turn 11, the captain is avoiding giving a direct answer to the doctor's question about whether the captain will attack. The captain is in a tough situation here: if he says "yes", he will hurt both solidarity (by showing conflicting goals) as well as possibly endanger his mission by revealing future plans. On the other hand, if he says "no", he will be lying which will hurt credibility if he is found out. By avoiding the question, he does not fall into either of these traps, though there is a danger of not fulfilling the discourse obligations. The explanation ("I can not disclose more information") does partially address this issue, and leaves the door open for future trust-building, such as in turn 13.

5. Example Interactions

Consider the dialogue in Figure 3. This is just one of many possible interactions, depending on the choices of the human captain, as well as several aspects (some probabilistic) influencing the choice of moves and strategy transitions of the virtual human doctor. Shown along with the dialogue are strategy transitions and the major influences on the trust components: familiarity, solidarity and credibility.

Here the captain acts as he might with a team member — after starting the conversation, launching directly into his purpose, and answering a question straightforwardly. While this would have worked with a subordinate teammember, it has disastrous effects on the neutral doctor, bringing his trust level almost down to zero and failing to accomplish both objectives.

In this dialogue, nothing was done by the captain to try to establish a better relationship with the doctor, or address the issue of differing objectives and

1	C	Hello Doctor Perez.
2	D	Hello.
3.1	C	I have orders to move this clinic to another location.
4	D	You want to move the clinic?
5	C	Yes
-SOLIDARITY'		
Strategy shift: Avoid negotiating about move		
6.1	D	Look at these people!
6.2	D	We need to help them.
7	C	It is not safe here, we can't protect you.
-CREDIBILITY'		
8.1	D	Protect me? Protect me from what?
8.2	D	Are you going to attack?
9	C	Yes
-SOLIDARITY'		
10.1	D	We need proper supplies here
Strategy shift: Disrupt negotiation about move		
10.2	D	To move the clinic is not possible, we have many patients in critical condition.
11	C	It is very important to move now!
-SOLIDARITY'		
12.1	D	You are the threat, I need protection from you!
Strategy shift: Unsuccessfully conclude the negotiation		
12.2	D	I would have to refuse this decision.
12.3	D	I must go now.

Figure 3. Unsuccessful negotiation dialogue between C, a captain (human trainee) and D, a doctor (virtual human) showing positive and negative effects on trust.

beliefs. The first exchange after the greetings (utterances 2–5) lowers solidarity by showing different objectives, setting up more of an antagonistic than cooperative interaction. The doctor tries to avoid the topic, focusing instead on his patients, rather than the captain's stated goal. The captain tries to argue for his proposed course of action, but only makes things worse with utterance 7. First, he says something the doctor doesn't believe (that the clinic is in danger), lowering his credibility. The doctor is able to reason though that perhaps the captain knows of a reason why it will be unsafe, and challenges by asking if he is going to cause the danger. In 9, the captain answers sincerely, which is a mistake on two fronts. First, he reveals more about his mission than he should to an outsider, possibly endangering its success if word gets out to his enemies. Second, he shows even further divergence from the doctor's goals — attacking rather than helping the patients. After one more brief attempt to change the topic and get help for his own goals, the doctor gives up on the captain in (10.2), and tries to get out of the negotiation. The captain has failed in his objective and prospects are not good for future relations.

For really learning about negotiation it is very helpful to know not just what the other party did, but why. In real negotiations it is usually not possible to get "inside the head" of the negotiating partner, and even subsequent questions can sometimes damage the nature of the interaction itself. In this respect, virtual humans present a real opportunity to improve on training. As shown in Figure 3, we have implemented a trace facility that provides an annotated transcript of the dialogue, showing not just what the virtual human thought was said, but how it influenced his trust, beliefs, and strategy choice. This tool can be used in an "after action review" (AAR) to look in detail at the specific effects the trainee's negotiation tactics had. Here we can see the reasons for decreases in credibility and solidarity as effects of the commitments the captain makes in relation to desires and beliefs of the doctor.

Figure 4 shows a more successful interaction, where the captain tries to build bonds as well as accomplish his task. While the captain's behaviour in this dialogue is not perfect either (the captain might have been better served spending more time up front establishing familiarity and solidarity and perhaps addressing the doctor's concerns first), it is a big improvement over the dialogue in Figure 3. Here the greetings in turn 3 add some familiarity, and the evasion in turn 11 does not do as much damage as the blanket statement of acting against the doctor's interest in the previous dialogue. Things are still not going very well, though, until the captain establishes some common goals with turn 13. With slightly higher trust, the doctor does not break off negotiation at this point, but rather raises a series of objections. By addressing each of the doctor's concerns: safety of patients, lack of supplies, lack of transport, and neutrality, the captain is able to bring him around to the point where the move is not an absolute negative, but is worthy of consideration, as part of a team

1 C Hello Doctor Perez.
2 D Hello.
3 C Very nice to meet you! [+*FAMILIARITY*]
4 D What do you want?
5 C I have orders to assist you in moving this clinic to a safer location.
6 D You want to move the clinic?
7 C Yes [-*SOLIDARITY*]

Strategy shift: Avoid negotiating about move

8.1 D This conflict is madness, it is killing people!
8.2 D We need proper supplies here!
9 C It is not safe here we can't protect you. [-*CREDIBILITY*]
10.1 D Protect me? Protect me from what?
10.2 D Are you going to attack?
11 C I can not disclose more information but it will be a problem to stay here.
12 D You are the problem your bombs are killing these people!
13 C Sir, we are trying to help these people too. [-*CONTROL* +*SOLIDARITY*]

Strategy shift: Disrupt negotiation about move

14.1 D The patients are here,
14.2 D to move the clinic is not possible we have many patients in critical condition.
15 C We can move you to a safer location.
16 D I can't move all these patients.
17 C We can organise the transportation and assist you with the moving. [+*SOLIDARITY*]
18 D I can't move all these patients.
19 C We can have locals move you. [+*SOLIDARITY*]
20 D There are no supplies there.
21 C We can provide you with the supplies you need to help your patients. [+*SOLIDARITY*]

Strategy shift: Willingly negotiate about move

22.1 D I think I understand what you say.
22.2 D Well perhaps we can reach an agreement,
22.3 D but before we can think about moving, we need you to bring antibiotics and plasma to stabilize the patients. [+*INTERDEPENDENCE*]
23 C We can do that! [+*SOLIDARITY*]
24.1 D Well,...
24.2 D Very well captain contact my assistant to make further arrangements.
25 C I'll see to it personally. [+*SOLIDARITY*]
26.1 D I understand your position.
26.2 D My patients need my attention now.
27 C Thank you Doctor!
28.1 D Well,....
28.2 D I must go now
29 C Goodbye.
30 D Good bye.

Figure 4. More successful negotiation.

plan. Finally, the two participants reach an agreement including giving needed supplies as part of the conditions of moving the clinic.

We can see several distinct phases of the dialogue in Figure 4, relating to different negotiation strategies. The initial segment (turns 1–7) includes initial greetings and establishing the topic for the conversation — the captain wants

Figure 5. More relaxed and open doctor.

to move the clinic. In turns 8–12, the doctor engages in an *avoidance* strategy, trying to avoid this topic by bringing up other issues, such as his need for supplies, and the general problems of conflict. In turns $14-20$, the doctor has adopted an *attack* strategy, and points out problems with the proposed move. In turns 22–25, the doctor adopts a more open negotiation strategy, and an actual bargain is struck. Finally, turns 26–30 show a closing phase in which the doctor disengages from the conversation, while the captain tries to establish good relations for future interaction. Application of these strategies influences not just the choice of dialogue move, but the whole body posture of the doctor and use of gestures and expressions as well. For example, when the doctor is feeling more distant and less trusting, he adopts a closed posture (Figure 2). When he is more trusting and open to negotiation, the posture becomes more relaxed (Figure 5).

6. Preliminary Evaluation and Future Directions

As part of the development of the system, we have so far tested the system with over 50 different people acting as a trainee, most with several dialogues. As of this writing we have not fully analyzed this data, however we can draw some general conclusions. Usually people are able to have a coherent dialogue with the Doctor, although some problems arise when concepts are brought up that are beyond his vocabulary. An advantage of this domain is that when the doctor is following an avoidance or attack strategy, it is natural for him to take the initiative and complain about his own concerns rather than being directly

responsive to the human's utterance, so some non-understandings do not lead to problems. Most people who talk to the doctor do not convince him in the first session, however, a little bit of explaining of proper negotiating techniques (e.g., build trust before arguing) generally leads to a successful negotiation in a follow-up round.

Future work involves extension of the models to include additional negotiation strategies, emotion-based styles of interaction within the strategies, and application to other scenarios, some involving cultural differences in behaviour and interpretation, as well as translated and multilateral dialogue.

Acknowledgements We would like to thank the entire SASO-ST team for producing the system in which this Dialogue and Negotiation model has been embedded. The work described in this chapter was supported by the Department of the Army under contract number DAAD 19-99-D-0046. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the Department of the Army.

References

Allen, J. F. (1983). Recognizing Intentions from Natural Language Utterances. In Brady, M. and Berwick, R. C., editors, *Computational Models of Discourse*, pages 107–166. MIT Press, Cambridge.

Bos, J., Klein, E., Lemon, O., and Oka, T. (2003). Dipper: Description and Formalisation of an Information-State Update Dialogue System Architecture. In *Proceedings of SIGdial Workshop on Discourse and Dialogue*, pages 115–124, Sapporo.

Brown, P. and Levinson, S. (1978). Universals in Language Usage: Politeness Phenomena. In Goody, E. N., editor, *Questions and Politeness: Strategies in Social Interaction*, pages 56–311. Cambridge University Press, Cambridge.

Cassell, J. and Bickmore, T. (2003). A Relational Agent: A Model and Implementation of Building User Trust. In *Proceedings of ACM CHI Conference*, pages 396–403, New York.

Cassell, J., Sullivan, J., Prevost, S., and Churchill, E. (2000). *Embodied Conversational Agents*. MIT Press, Cambridge.

Cohen, P. R. and Levesque, H. J. (1990). Rational Interaction as the Basis for Communication. In Cohen, P. R., Morgan, J., and Pollack, M. E., editors, *Intentions in Communication*. MIT Press, Cambridge.

Cohen, P. R. and Perrault, C. R. (1979). Elements of a Plan-Based Theory of Speech Acts. *Cognitive Science*, 3:177–212.

Cooper, R. and Larsson, S. (1999). Dialogue Moves and Information States. In *Proceedings of International Workshop on Computational Semantics (IWCS)*, pages 398–400, Tilburg.

Fipa (1997). Fipa 97 Specification Part 2: Agent Communication Language. http://drogo.cselt.stet.it/fipa/spec/fipa97/f8a21.zip.

Gratch, J. and Marsella, S. (2004). A Domain-Independent Framework for Modeling Emotion. *Journal of Cognitive Systems Research*, 5:269–306.

Grice, H. P. (1975). Logic and Conversation. Syntax and Semantics. In Cole, P. and Morgan, J. L., editors, *Speech Acts*, volume 3, pages 41–58. Academic, New York.

Hill, R. (2000). Perceptual Attention in Virtual Humans: Towards Realistic and Believable Gaze Behaviors. In *Proceedings of AAAI Fall Symposium on Simulating Human Agents*, pages 46–52, North Falmouth.

Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). SOAR: An Architecture for General Intelligence. *Artificial Intelligence*, 33:1–64.

Larsson, S. (2002). *Issue-Based Dialogue Management*. PhD thesis, Göteborg University, Sweden.

Larsson, S., Bohlin, P., Bos, J., and Traum, D. (1999). *Trindikit Manual*.

Larsson, S. and Traum, D. (2000). Information State and Dialogue Management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering,Special Issue on Spoken Language Dialogue System Engineering*, 6:323–340.

Mao, W. and Gratch, J. (2004). Social Judgment in Multiagent Interactions. In *Proceedings of 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 210–217, Columbia University, New York.

Marsella, S., Pynadath, D., and Read, S. (2004). Psychsim: Agent-Based Modeling of Social Interactions and Influence. In *Proceedings of International Conference on Cognitive Modelling*, pages 243–248, Pittsburgh.

Matheson, C., Poesio, M., and Traum, D. (2000). Modelling Grounding and Discourse Obligations using Update Rules. In *Proceedings of Conference of the North American chapter of the Association for Computational Linguistics (NAACL)*, pages 1–8, Seattle.

Midiki Users Manual (2005). Mitre Corporation. https://sourceforge.net/docman/display_doc.php?docid=25561&group_id=123104.

Putnam, L. L. and Jones, T. S. (1982). Reciprocity in Negotiations: An Analysis of Bargaining Interaction. *Communications Monograph*, 49:171–191.

Rickel, J. and Johnson, W. L. (1999a). Animated Agents for Procedural Training in Virtual Reality: Perception, Cognition, and Motor Control. *Applied Artificial Intelligence*, 13:343–382.

Rickel, J. and Johnson, W. L. (1999b). Virtual Humans for Team Training in Virtual Reality. In *Proceedings of 9th International Conference on Artificial Intelligence in Education*, pages 578–585, Le Mans.

Rickel, J., Marsella, S., Gratch, J., Hill, R., Traum, D., and Swartout, W. (2002). Toward a New Generation of Virtual Humans for Interactive Experiences. *IEEE Intelligent Systems*, 17:32–38.

Scherer, K. R., Schorr, A., and Jonstone, T. (2001). *Appraisal Processes in Emotion*. Series in Affective Science. Oxford University Press, Oxford.

Sillars, A. L., Coletti, S. F., Parry, D., and Rogers, M. A. (1982). Coding Verbal Conflict Tactics: Nonverbal and Perceptual Correlates of the Avoidance-Distributive-Integrative Distinction. *Human Communication Research*, 9(1):83–95.

Traum, D. R. (1994). *A Computational Theory of Grounding in Natural Language Conversation*. PhD thesis, Department of Computer Science, University of Rochester, USA. Also available as TR 545, Department of Computer Science, University of Rochester, New York.

Traum, D. R. and Allen, J. F. (1994). Discourse Obligations in Dialogue Processing. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–8, Las Cruces, New Mexico.

Traum, D. R. and Larsson, S. (2003). The Information State Approach to Dialogue Management. In van Kuppevelt, J. and Smith, R., editors, *Current and New Directions in Discourse and Dialogue*, pages 325–353. Kluwer Academic.

Traum, D. R. and Rickel, J. (2002). Embodied Agents for Multi-Party Dialogue in Immersive Virtual Worlds. In *Proceedings of 1st International Joint conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 766–773, Bologna.

Traum, D. R., Rickel, J., Marsella, S., and Gratch, J. (2003). Negotiation over Tasks in Hybrid Human-Agent Teams for Simulation-Based Training. In *Proceedings of Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 441–448, Melbourne.

Traum, D. R., Robinson, S., and Stephan, J. (2004). Evaluation of Multi-Party Virtual Reality Dialogue Interaction. In *Proceedings of International Conference on Language Resources and Evaluation (LREC)*, pages 1699–1702, Lisbon.

Walton, R. E. and Mckersie, R. B. (1965). *A Behavioral Theory of Labor Negotiations: An Analysis of a Social Interaction System*. McGraw-Hill, New York.

Wilson, S. R. and Putnam, L. L. (1990). Interaction Goals in Negotiation. *Communication Yearbook*, 13:374–406.

Chapter 4

EVALUATING INTERACTIONS WITH SPOKEN DIALOGUE TELEPHONE SERVICES

Sebastian Möller
Deutsche Telekom Laboratories
Berlin University of Technology, Germany
sebastian.moeller@telekom.de

Abstract The quality experienced during the interaction with telephone-based spoken dialogue services results from a perception and judgement process. As a consequence, quality has to be measured in a subjective way, with the help of human test persons. To complement subjective quality judgements, parameters can be logged which quantify the flow of the interaction, the behaviour of the user and the system, and the performance of individual system modules during the interaction. Although such parameters are not directly linked to the quality perceived by the user, they provide useful information for system development, optimisation, and maintenance. This chapter presents standardised methods for both measurement approaches. Firstly, a brief overview of subjective evaluation experiments is provided, following Recommendation P.851 issued by the International Telecommunication Union. Secondly, a collection of parameters is presented which has proven to be useful for system design. An initial evaluation study in is described which shows that the parameters correlate only weakly with subjective judgements; thus, both types of evaluation provide complementary types of information. Linear regression models may be used to predict subjective judgements from interaction parameters, but their prediction accuracy is still limited.

Keywords: Spoken dialogue system; subjective evaluation; interaction parameters; quality prediction

1. Introduction

Speech technology devices, such as automatic speech recognition (ASR), speaker verification, speech synthesis, or spoken dialogue systems (SDSs), provide an increasing number of automatic voice-enabled services in wireline

L. Dybkjær and W. Minker (eds.), Recent Trends in Discourse and Dialogue, 69–100.

and mobile telephone networks. In contrast to simple interactive voice response systems with tone input, spoken dialogue systems offer the full range of speech interaction capabilities, including the recognition of user speech, the assignment of meaning to the recognised words, the decision on how to continue the dialogue, the formulation of a linguistic response, and the generation of spoken output to the user. In this way, a more-or-less "natural" spoken interaction between user and system is enabled.

In order to quantify the interaction quality with SDSs, the perceptions of the human user have to be taken into account. In fact, quality has been defined in ITU-T Rec. P.851 (2003) as the

> result of appraisal of the perceived composition of the service with respect to its desired composition.

This definition of quality, which is based on the concepts and terminology introduced by Jekosch (2000, 2005), highlights two very important characteristics:

1. Quality involves a *perception* and a *judgement process*.

2. Quality is a *relative* concept; it results from a comparison between the perceived and the desired or expected.

These characteristics have consequences for the measurement of quality. Firstly, quality can only be quantified with the help of perceiving and judging humans, e.g. with test persons. Secondly, the reference to which the perceived features are compared should reflect the one of the application situation. Thus, in order to obtain valid quality measurements, the conditions under which an interaction experiment is carried out in the laboratory should match the ones of later system usage as closely as possible.

So far, quality has been addressed from the system user's point of view. However, the system designer has an interest in providing high-quality services too, e.g. for reaching high acceptance rates and/or maximum profit. System designers need to have indications of the factors of the system and its usage environment which potentially carry an influence on perceived quality. If relationships between system characteristics and perceived quality can be established, SDSs can be set up and optimised quite efficiently. The criterion for system optimisation should be the best possible quality judgement of the user, and not necessarily the highest performance of individual system components.

In order to compare services and systems, a common set of evaluation methods and criteria has to be defined. For this aim, the Telecommunication Standardisation Sector of the International Telecommunication Union (ITU-T) set up a new Recommendation describing subjective evaluation methods for telephone services based on spoken dialogue systems (ITU-T Rec. P.851, 2003).

This Recommendation addresses quality from a user's point of view. Recently, it has been complemented by a set of interaction parameters, addressing system performance from a system developer's and service operator's point of view. The parameters are summarised in ITU-T Suppl. 24 to P-Series Rec. (2005). They help to quantify the flow of the interaction, the behaviour of the user and the system, and the performance of the speech technology devices involved in the interaction.

The present chapter provides an overview of subjective evaluation methods and of interaction parameters which have been recommended by the ITU-T. It is based on theoretical work which is described in Möller (2005b). Section 2 summarises the main aspects which have to be taken into account when collecting users' quality judgements in a subjective experiment. Exemplary judgements have been collected in a case study with an SDS for controlling domestic devices; these judgements will be analysed in Section 3, revealing the perceptual dimensions underlying the users' judgements. Section 4 presents a brief characterisation of interaction parameters, with respect to the interaction aspect they address and the measurement method which is required to determine the parameter. The parameters are categorised and listed in Section 5. Section 6 presents an initial evaluation of the set of parameters, showing their correlation to subjective quality judgements and their contribution to predicting quality, using linear regression models. Finally, Section 7 summarises the main findings and identifies future work to obtain a reduced set of parameters to be recommended by the ITU-T.

2. Subjective Evaluation of Dialogue Services

In this section, we report on subjective evaluation methods which aim at quantifying the quality of dialogue services by means of experiments with human test participants ("test subjects"). Such experiments are a specific way to assess the quality of dialogue services, or of software products in general. They are particularly suited to provide quantitative data, but may not reveal all issues which are relevant for quality and usability. Thus, they should be supplemented by other types of evaluation and usability engineering principles, e.g. the ones described in Shneiderman (2003), Nielsen (1994), and Nielsen and Mack (1994).

Quantitative measurements of quality may be obtained best from laboratory experiments carried out under controlled conditions, where test users interact with the system to be evaluated. In case the system is still under development and thus not fully available for the evaluation, parts of the system may be simulated in a so-called Wizard-of-Oz paradigm (Fraser, 1997). This allows decisions on individual system components to be taken early in the design process, and avoids costly reimplementation. Wizard-of-Oz simulations may also be used to extrapolate quality for component performances which are beyond the

current state of the art. The interactions are usually logged, and interaction parameters can be extracted from the log files (see Section 4).

The results of subjective evaluation experiments reflect the characteristics of the test user group. Characteristics like age and gender, physical status, speaking rate, vocal effort, native language, dialect, or accent may have a strong influence on the speech produced by the user, and subsequently on speech recognition and understanding performance. As a consequence, quality judgements obtained from a user group differing in the acoustic and language characteristics might not reflect the quality which can be expected for the target user group. User groups are however variable and ill-defined. A service which is open to the general public will sooner or later be confronted with a large range of different users. Testing with specified users outside the target user group will therefore provide a measure of system robustness with respect to the user characteristics.

In addition to these characteristics, judgements are influenced by the user's experience and expertise with the system, and with the task and/or domain the service is designed for. Investigations show that user experience affects a large range of speech and dialogue characteristics, e.g. the number of tasks solved in a single dialogue, the interaction length (Delogu et al., 1993), the number of in-vocabulary utterances, and the task completion rate (Kamm et al., 1997). System familiarity may also lead to a reduced number of user inputs and help messages, and to a reduced transaction time (Lamel et al., 2002).

Most dialogue systems which are available on the market enable task-oriented interactions. Because of the lack of a real motivation, laboratory tests often make use of experimental tasks which the participants have to carry out. The experimental task provides an explicit goal, but this goal should not be confused with a goal which a user would like to reach in a real-life situation. Because of this discrepancy, valid user judgements on system helpfulness and acceptability cannot easily be obtained in a laboratory test set-up.

Examples of experimental tasks are included in ITU-T Rec. P.851 (2003). They are frequently presented in a graphical way, in order to avoid a direct priming of the user's language. A comparison between written and graphical scenarios showed that the massive priming effect of written scenarios can be nearly completely avoided by a graphical representation, but that the diversity of linguistic items (total number of words, number of out-of-vocabulary words) is similar in both cases (Dybkjær et al., 1995; Bernsen et al., 1998). Thus, language diversity still has to be assured by collecting utterances from a sufficiently high number of different users, e.g. in a field test situation.

In order to quantify perceived quality, test persons are asked to fill in questionnaires before the actual experiment, after each interaction with the system, and at the end of the whole experiment. Such questionnaires are designed to collect as many different quality aspects as possible from the user.

Questionnaires given before the experiment usually contain questions related to the user's background, namely some personal information (age, gender, profession, etc.), some task-related information (frequency of the task, usual approach to resolve the task, alternative interfaces, etc.), as well as system-related information (experience with tone-based systems or SDSs). Questionnaires relating to an individual interaction may extract diagnostic information on the behaviour of the system in that particular interaction, for example related to

- The *information obtained from the system*, e.g. availability, accuracy, completeness, consistency, reliability, clarity, or truth
- The system's *speech input and output capability*, e.g. perceived system understanding, frequency of system errors, perceived system reasoning, listening effort required to understand the system's messages, perceived intelligibility, or perceived comprehensibility
- The system's *interaction behaviour*, e.g. transparency of the interaction, congruence with the user's expectations, flexibility of the interaction, perceived reliability of system processing, distribution of initiative, interaction control capability, confirmation and correction capabilities, recovery from interaction problems, naturalness of the interaction, length of the dialogue, perceived system speed, or smoothness of the dialogue
- The perceived *system personality*, e.g. friendliness or politeness
- The *impression* the interaction leaves with the user, e.g. perceived naturalness of the user's own behaviour, pleasantness, cognitive demand put on the user, stress, or fluster
- The perceived *task fulfilment*, e.g. in terms of task success and reliability of the task results

Finally, questionnaires given after the experiment may use similar questions, but the judgements then relate to the overall experience made with the service so far. In this case, very analytic questions should be avoided. Example questionnaires are given in ITU-T Rec. P.851 (2003).

3. Multidimensional Analysis of Subjective Quality Judgements

The questionnaires described above usually contain a large number of questions related to a diversity of quality aspects. Although the evaluator can select individual questions according to the aim of the evaluation (e.g. a comparative assessment of the performance of individual system modules, or a diagnostic

assessment to identify system weaknesses), it is interesting to analyse which dimensions of perceived quality can be distinguished at all with such a questionnaire.

For this aim, a multidimensional analysis has been carried out in the frame of the EC-funded IST project INSPIRE (INfotainment management with SPeech Interaction via REmote microphones and telephone interfaces). In this project, a prototype of a spoken dialogue system for controlling domestic devices (lamps, blinds, video recorder, answering machine, etc.) has been set up. The prototype has been evaluated in a controlled laboratory experiment at IKA, Ruhr-Universität Bochum. Because the speech recogniser was not available when the experiment was carried out, it had to be replaced by a human transcriber, making this a partly Wizard-of-Oz-based experiment.

During this experiment, 24 test users interacted with the system in a realistic home environment, following 3 scenario-guided interactions, each comprising several tasks. After each interaction, users were asked to fill in a questionnaire with 37 statements which has been designed following the methodology of ITU-T Rec. P.851 (2003). The statements were presented either on continuous rating scales which were labelled with describing attributes (with overflow areas in order to avoid saturation of user ratings at the extremities of the scale), or on 5-point Likert scales, see Figure 1. Details on the experimental set-up and on the individual statements are given in Möller et al. (2006).

The ratings have been transformed into numbers and then analysed using a principal component analysis with Varimax rotation and Kaiser normalisation (pairwise exclusion of missing values). This analysis reveals eight components, resulting in 72.6% of the variance covered by the cumulated factors. The components were interpreted on the basis of statements which have loadings higher than or equal to +/−0.6 as follows:

1. *Acceptability*: Highest loadings are observed for the question whether the service would be used again, or whether a different interface would be preferred for carrying out the given tasks.

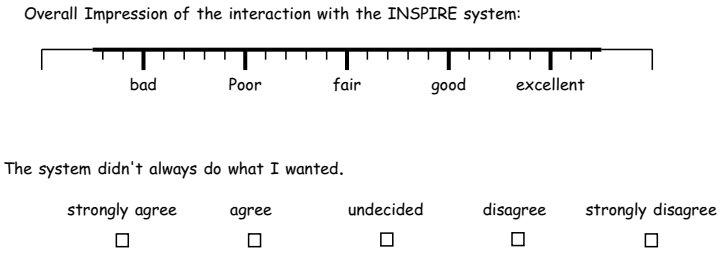


Figure 1. Examples of rating scales used in the experiment. Upper panel: Continuous rating scale according to Bodden and Jekosch (1996); lower panel: Likert-type category rating scale.

2. *Cognitive demand*: Highest loadings were found for the required concentration and the stress imposed on the user.

3. *Task efficiency*: This dimension shows high loadings on task success, on the clarity of the provided information, and on the transparency of the system behaviour.

4. *System errors*: This dimension reflects the frequency of system errors and the reliability of the system.

5. *Ease of use*: This dimension includes the ease of listening and the ease of learning how to use the system.

6. *System cooperativity*: This dimension observes whether the system behaves in a cooperative way or not.

7. This dimension shows two high loadings which cannot be interpreted in combination: The *naturalness of the system voice*, and the *symmetry of the dialogue*.

8. *Speed of the interaction*: This dimension loads on the speed and the length of the interaction.

The extracted dimensions may be somehow specific for the tested system and for the user group. In addition, the sample size of this study was quite limited. Thus, the dimensions extracted from our database should be seen as an initial result only. In order to come to dimensions which somehow generalise across systems and user groups, the results have been compared to dimensions extracted from other experiments (Möller, 2005a), namely from the evaluation of automated credit-card payment services (Love et al., 1994; Jack et al., 1992), from the SASSI study addressing different dialogue services (Hone and Graham, 2000), from the ELSNET Olympics (den Os and Bloothooft, 1998), and from a telephone-based restaurant information system (Möller, 2005b). The result of this comparison in depicted in Figure 2.

The classification shows five perceptual dimensions which are extracted in most of the described experiments:

1. *Acceptability*: This dimension covers task success, future use, reliability, the perceived control capability, as well as the enjoyment or frustration experienced when using the system. It is most closely related to overall quality and user satisfaction

2. *Communication efficiency*: This dimension often perceived as the speed of the interaction

3. *Cognitive effort*: This dimension comprises stress, fluster, cognitive demand, confidence in the system, and perceived annoyance

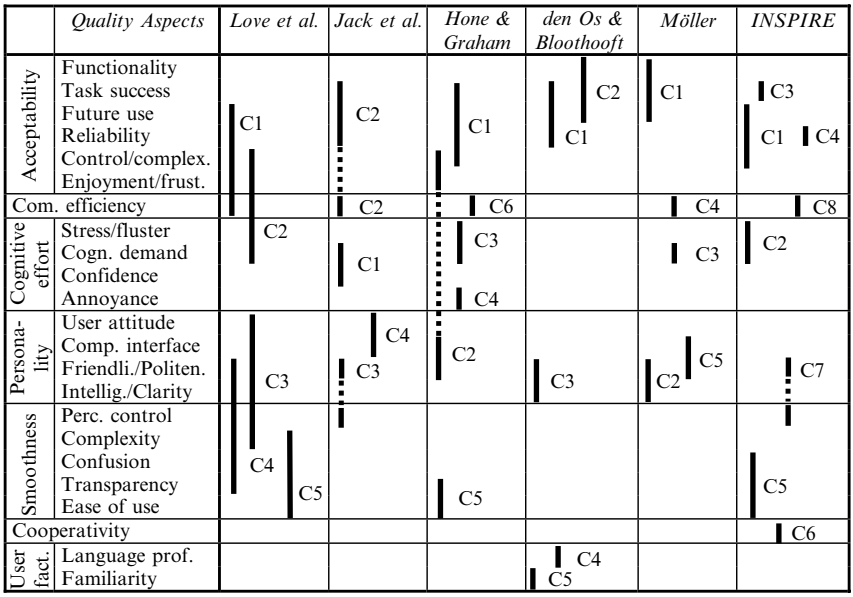


Figure 2. Comparison of perceptual quality dimensions extracted from different experiments. Cx indicates the corresponding factors extracted from the factor analysis. See text for literature references.

4. *Personality*: This dimension includes the perceived politeness and friendliness, intelligibility and clarity of system speech, the general user attitude towards the system, as well as the preference for other interfaces performing the same task

5. *Smoothness*: This dimension is characterised by the perceived control over the interaction, the complexity and confusion caused by the dialogue, the transparency of the interaction, and the ease of use

Apart from these five common dimensions, "cooperativity" has been extracted as a separate dimension in the INSPIRE experiment, and the ELSNET Olympics also revealed the user-related dimensions "language proficiency" and "familiarity" with dialogue systems. In fact, the ELSNET Olympics were carried out with conference participants as a specific user group, evaluating systems mostly not in their mother tongue. This finding emphasises that perceptual dimensions may be specific to individual systems and user groups.

4. Characteristics of Interaction Parameters

Interaction parameters can be extracted when real or test users interact with the telephone service under consideration. The extraction is performed on the basis of log files, be it instrumentally or with the help of a transcribing and

annotating expert. Parameters which relate to the surface form of the utterances exchanged between user and system, like the duration of the interaction or the number of turns, can usually be measured fully instrumentally. On the other hand, human transcription and annotation is necessary when not only the surface form (speech signals) is addressed, but also the contents and meaning of system or user utterances (e.g. to determine a word or concept accuracy). Both (instrumental and expert-based) ways of collecting interaction parameters should be combined in order to obtain as much information as possible.

Because interaction parameters are based on data which has been collected in an interaction between user and system, they are influenced by the characteristics of the system, of the user, and of the interaction between both. These influences cannot be separated, because the user's behaviour is strongly influenced by that of the system (e.g. the questions asked by the system), and vice versa (e.g. the vocabulary and speaking style of the user influences the system's recognition and understanding accuracy). Consequently, interaction parameters strongly reflect the characteristics of the user group they have been collected with.

Interaction parameters are either determined in a laboratory test setting under controlled conditions, or in a field test. In the latter case, it may not be possible to extract all parameters, because not all necessary information can be gathered. For example, if the success of a task-oriented interaction (e.g. collection of a train timetable) is to be determined, then it is necessary to know about the exact aims of the user. Such information can only be collected in a laboratory setting, e.g. in the way it is described in Section 2.

Interaction parameters can be calculated at word level, at sentence or utterance level, or at the level of a full interaction or dialogue. In case of word or utterance level parameters, average values are often calculated for each dialogue. The parameters collected with a specific group of users may be analysed with respect to the impact of the system (version), the user group, and the experimental setting (scenarios, test environment, etc.), using standard statistical methods. A characterisation of these influences can be found in Möller (2005b).

5. Review of Interaction Parameters

Based on a broad literature survey, parameters were identified which have been used in different assessment and evaluation experiments during the last 15 years. The respective literature includes Billi et al. (1996), Boros et al. (1996), Carletta (1996), Cookson (1988), Danieli and Gerbino (1995), Fraser (1997), Gerbino et al. (1993), Glass et al. (2000), Goodine et al. (1992), Hirschman and Pao (1993), Kamm et al. (1998), Polifroni et al. (1992), Price et al. (1992), SanSegundo et al. (2001), Simpson and Fraser (1993), Skowronek (2002),

Strik et al. (2000), Strik et al. (2001), van Leeuwen and Steeneken (1997), Walker et al. (1997), Walker et al. (1998) and Zue et al. (2000).

The parameters can broadly be classified as follows:

- Dialogue- and communication-related parameters
- Meta-communication-related parameters
- Cooperativity-related parameters
- Task-related parameters
- Speech-input-related parameters

These categories will be briefly discussed in the following sections. The respective parameters are listed in an appendix to this chapter, together with a definition, the interaction level addressed by the parameter (word, utterance or dialogue), as well as the measurement method (instrumental or expert annotation).

5.1 Dialogue- and Communication-Related Parameters

Parameters which refer to the overall dialogue and to the communication of information give a very rough indication of how the interaction takes place. They do not specify the communicative function of each individual utterance in detail. These parameters are listed in Tables 4.A.1 and 4.A.2 of the Appendix, and include duration-related parameters (overall dialogue duration, duration of system and user turns, system and user response delay), and word- and turn-related parameters (average number of system and user turns, average number of words per system and per user turn, number of system and user questions).

Two parameters which have been proposed by Glass et al. (2000) are worth noting: The query density gives an indication of how efficiently a user can provide new information to a system, and the concept efficiency describes how efficiently the system can absorb this information from the user. These parameters also refer to the system's language understanding capability, but they have been included in this section because they result from the system's interaction capabilities as a whole, and not purely from the language understanding capabilities.

All parameters in this category are of global character and refer to the dialogue as a whole, although they are partly calculated at utterance level. Global parameters are sometimes problematic, because the individual differences in cognitive skill may be large in relation to the system-originated differences, and because test participants might learn strategies for task solution which have a significant impact on global parameters.

5.2 Meta-Communication-Related Parameters

Meta-communication, i.e. the communication about communication, is particularly important for the spoken interaction with systems which have limited recognition, understanding and reasoning capabilities. In this case, correction and clarification utterances or even subdialogues are needed to recover from misunderstandings.

The parameters belonging to this group quantify the number of system and user utterances which are part of meta-communication. Most of the parameters are calculated as the absolute number of utterances in a dialogue which relate to a specific interaction problem, and are then averaged over a set of dialogues. They include the number of help requests from the user, of time-out prompts from the system, of user utterances rejected by the system in the case that no semantic content could be extracted (ASR rejections), of diagnostic system error messages, of barge-in attempts from the user, and of user attempts to cancel a previous action.

The ability of the system (and of the user) to recover from interaction problems can be described in two ways: Either explicitly by the correction rate, i.e. the percentage of all (system or user) turns which are primarily concerned with rectifying an interaction problem, or implicitly with the implicit recovery parameter, which quantifies the capacity of the system to regain utterances which have partially failed to be recognised or understood.

In contrast to the global measures, most meta-communication-related parameters describe the function of system and user utterances in the communication process. Thus, most parameters have to be determined with the help of an annotating expert. The parameters are listed in Tables 4.A.3 to 4.A.5 of the Appendix.

5.3 Cooperativity-Related Parameters

Cooperativity has been identified as a key aspect for a successful interaction with a spoken dialogue system (Bernsen et al., 1998). Unfortunately, it is difficult to quantify whether a system behaves cooperatively or not. Several of the dialogue- and meta-communication-related parameters somehow relate to system cooperativity, but they do not attempt to quantify this aspect.

Direct measures of cooperativity are the contextual appropriateness parameters introduced by Simpson and Fraser (1993). Each system utterance has to be judged by a number of experts as to whether it violates one or more of Grice's maxims for cooperativity (see Grice, 1975). These principles have been stated more precisely by Bernsen et al. (1998) with respect to spoken dialogue systems.

The utterances are classified into the categories of appropriate (not violating Grice's maxims), inappropriate (violating one or more maxims), incomprehensible (the content of the utterance cannot be discerned in the dialogue context), or total failure (no linguistic response from the system). It has to be noted that the classification is not always straightforward, and that interpretation principles may be necessary.

5.4 Task-Related Parameters

Current state-of-the-art telephone services enable task-oriented interactions between system and user, and task success is a key issue for the usefulness of a service. Task success may be defined on a per-dialogue basis (thus taking into account all subtasks addressed in a dialogue), or on a per-subtask basis. It may best be determined in a laboratory situation where explicit tasks are given to the test participants (see Möller, 2005b). However, realistic measures of task success have to take into account potential deviations from the scenario by the user, either because he/she did not pay attention to the instructions given in the scenario, because of his/her inattentiveness to the system utterances, or because the task was irresolvable and had to be modified in the course of the dialogue.

Modification of the experimental task is considered in most definitions of task success which are reported in the literature. Success may be reached by simply providing the right answer to the constraints set in the instructions, by constraint relaxation by the system or by the user (or both), or by spotting that no solution exists for the defined task. Task failure may be tentatively attributed to the system's or to the user's behaviour, the latter however being influenced by the one of the system.

A different approach to determine task success is the κ coefficient (Carletta, 1996; Walker et al., 1997). It normalises for task complexity, and thus enables a better comparison between different tasks and applications. The calculation of κ assumes a speech-understanding approach which is based on attributes (concepts, slots) for which allowed values have to be assigned in the course of the dialogue, resulting in attribute-value-pairs (AVPs). A set of all available attributes together with the values assigned by the task (a so-called attribute-value matrix, AVM) completely describes a task which can be carried out with the help of the system. In order to determine the κ coefficient, a confusion matrix $M(i, j)$ is set up for the attributes in the key (scenario definition) and in the reported solution (log file of the dialogue). Then, the agreement between key and solution $P(A)$ and the chance agreement $P(E)$ can be calculated from this matrix (see Table 4.A.7). $M(i, j)$ can be calculated for individual dialogues, or for a set of dialogues which belong to a specific system or system configuration.

The κ coefficient relies on the availability of a simple task coding scheme, namely in terms of an AVM. However, some tasks cannot be characterised as easily. In that case, more elaborated approaches to task success are needed, approaches which usually depend on the type of task under consideration. Dialogue success measures for non-task-oriented SDSs are still under development.

5.5 Speech Input-Related Parameters

The speech input capability of a spoken dialogue system is determined by its capability to recognise words and utterances, and to extract the meaning from the recognised string. The speech recognition task can be categorised into isolated word recognition, keyword spotting, or continuous speech recognition. Speech understanding is often performed on the basis of attribute-value pairs, see the previous section. The parameters described in the following paragraph address both speech recognition and speech understanding.

Continuous speech recognisers generally provide a word string hypothesis which has to be aligned with a reference transcription produced by an annotating expert. On the basis of the alignment, the number of correctly determined words c_w, of substitutions s_w, of insertions i_w, and of deletions d_w is counted. These counts can be related to the total number of words in the reference n_w, resulting in two alternative measures of recognition performance, the word error rate WER and the word accuracy WA, see Table 4.A.8.

Complementary performance measures can be defined at sentence level, in terms of sentence accuracy, SA, or sentence error rate, SER, see Table 4.A.8. In general, sentence accuracy is lower than word accuracy, because a single misrecognised word in a sentence impacts the SA parameter. It may however become higher than the word accuracy, especially when many single-word sentences are correctly recognised.

The fact that SER and SA penalise a whole utterance when a single misrecognised word occurs has been pointed out by Strik et al. (2000, 2001). The problem can be circumvented by calculating the average number of errors per sentence, NES, or the word error per sentence, WES (see Table 4.A.9). When utterances are not separated into sentences, all sentence-related metrics can also be calculated at utterance level instead of at sentence level.

Isolated word recognisers provide an output hypothesis for each input word or utterance. Input and output words can be directly compared, and similar performance measures as in the continuous recognition case can be defined, omitting the insertions. Instead of the insertions, the number of "false alarms" in a time period can be counted, see vanLeeuwen and Steeneken (1997). WA

and WER can also be determined for keywords only, when the recogniser operates in a keyword-spotting mode.

For speech understanding assessment, two common approaches have to be distinguished. The first one is based on the classification of system answers to user questions into categories of correctly answered, partially correctly answered, incorrectly answered, or failed answers. The individual answer categories can be combined into measures which have been used in the US DARPA program, see Tables 4.A.9 and 4.A.10. The second way is to classify the system's parsing capabilities, either in terms of correctly parsed utterances, or of correctly identified AVPs. On the basis of the identified AVPs, global measures such as the concept accuracy CA, the concept error rate CER, or the understanding accuracy UA can be calculated. These parameters are listed in Table 4.A.11.

5.6 Further Parameters

When separating the quality of an SDS-based service into quality aspects, in the way which is indicated in Möller (2005b), it can be observed that several aspects of quality are not addressed by interaction parameters. No parameters directly relate to usability, user satisfaction, acceptability, or speech output quality. So far, only very few approaches have been made which address the quality of speech output (be it concatenated or synthesised) in a parametric way. Instrumental measures related to speech intelligibility are defined, e.g., in IEC Standard 60268-16 (1988), but they have not been designed for a telephone environment. Concatenation cost measures have been proposed which can be calculated from the input text and the speech database of a concatenative synthesis system (Chu and Peng, 2001). Although they sometimes show high correlations to mean opinion scores obtained in subjective experiments, such measures are very specific to the speech synthesiser and its concatenation corpus.

6. Initial Evaluation of Interaction Parameters

In order to determine the relationship between subjective user judgements and interaction parameters, the interactions of the INSPIRE experiment have been logged, transcribed and annotated using a specifically designed annotation interface (Skowronek, 2002; Möller, 2005b). From the annotation, 64 parameters could be extracted for each interaction which are mainly identical to the ones listed in Section 5. Thus, the user judgements analyzed in Section 3 are complemented by interaction parameters, reflecting the same set of interactions with a prototypical system. Details on the experiment can be found again in Möller et al. (2006).

6.1 Correlation between Interaction Parameters and User Judgements

From this database, correlations between interaction parameters and subjective judgements have been calculated. Because several interaction parameters and user judgements do not follow a Gaussian distribution, Spearman rank-order correlations ρ have been chosen. The results were disappointing at first sight: The highest coefficients were around 0.6.

Interestingly, quality-related information seems to be captured mostly in the speech-recognition- and speech-understanding-related parameters. This is astonishing, because the (simulated) recognition accuracy of the INSPIRE system was nearly perfect (mean WA = 97.2%). The recognition-related parameters were shown to have correlations of up to 0.6 with interaction control, up to 0.52 with interaction pleasantness, up to 0.47 with the difficulty of operation, up to 0.43 with system helpfulness, up to 0.42 with dialogue smoothness, and up to 0.40 with error recovery. The correlation between speech-recognition- and speech-understanding-related parameters is only moderate, justifying measuring both types of parameters to obtain a maximum of information. Perceived system understanding correlates only moderately with the measured understanding accuracy, UA (ρ = 0.41).

With respect to efficiency, humans do not seem to be adequate measurement instruments either. The correlation between the perceived length of a dialogue and DD (communication efficiency) is very low, as well as the correlation between annotated and perceived task success (task efficiency). The subjective judgement on overall quality seems to be mainly dominated by the characteristics of the system turns (STD: ρ = 0.40), by the understanding accuracy (UA: ρ = 0.39; UCT: ρ = 0.36), and by the recognition accuracy (ρ between 0.39 and 0.42). Still, this correlation is not high enough to be able to predict overall system quality on the basis of individual interaction parameters.

6.2 Quality Prediction Models

More sophisticated models have been developed to predict system usability and acceptability from a combination of parameters. The most popular approach is the PARADISE framework developed by Walker et al. (1997, 1998). The model aims at predicting "user satisfaction", which is calculated as an arithmetic mean over several user judgements on different quality aspects, as a linear combination of several interaction parameters. In its original version, Walker et al. used 8–9 interaction parameters as input to the model, including a subjective judgement on task success. The weighting coefficients of the linear prediction function are determined by the help of a multivariate linear regression analysis, using a database of user judgments and interaction parameters which have been collected under controlled (laboratory) conditions.

From the INSPIRE database, several PARADISE-style models have been calculated, using different user judgements as the prediction target (judgement on "overall quality", "user satisfaction", or the arithmetic mean over all 37 judgements), and several sets of interaction parameters as the input variables (full set of 64 parameters or restricted set of 5 parameters similar to Walker et al. (1997)). In particular, 2 types of parameters have been used for describing task success: Either an expert-derived weighted task success index TSe (which is calculated from the TS labels of Table 4.A.7, assigning a value of one for each sub-task which has been successfully achieved by the user, and a value of zero for all failures), or a user judgement of task success TSu (as it was the case in the experiments reported in Walker et al. (1997, 1998). The regression algorithm used a stepwise (forward–backward) inclusion of parameters (for 64 parameters) or a forced inclusion of all parameters (for 5 parameters only), did not include a constant term, and replaced missing values by their respective means.

The results are shown in Table 1. Indicated is the amount of variance in the subjective judgements which can be covered by the respective model (R^2_{corr}) and the number of input parameters selected by the regression algorithm. For the large set of input parameters, R^2_{corr} reaches 0.46 in the best case, which is comparable to the prediction accuracy reported by Walker et al. (1997, 1998). However, when using only the restricted set of parameters as an input to the regression analysis, the prediction accuracy is much lower. The user-derived judgement of task success leads in all cases to better prediction results; it is particularly important when only few input parameters are available. All in all, the prediction accuracy does not depend on the number of input parameters, but on their informative value.

Table 1. Regression models.

Input parameters		*Target variable*	*Prediction result*	
# Parameters	*Task success*		R^2_{corr}	*# Parameters*
64	TSe	Overall quality	0.247	2
64	TSe	User satisfaction	0.409	4
64	TSe	Mean of all judgments	0.420	4
64	TSu	Overall quality	0.409	3
64	TSu	User satisfaction	0.409	4
64	TSu	Mean of all judgments	0.459	3
5	TSe	Overall quality	0.091	5
5	TSe	User satisfaction	0.022	5
5	TSe	Mean of all judgements	0.133	5
5	TSu	Overall quality	0.310	5
5	TSu	User satisfaction	0.086	5
5	TSu	Mean of all judgements	0.305	5

7. Conclusions

An overview has been presented of subjective evaluation methods and interaction parameters for spoken dialogue systems. The former provide direct measurements of quality, the latter quantitative descriptions of the interaction between the user and the system. Interaction parameters can be used in the design, implementation, optimisation and operation phase of SDS-based services. They provide important information to the system developer, but no direct measures of quality, as it would be perceived by the user of the respective service.

Reliable quality judgements may be obtained for example in laboratory experiments carried out under controlled conditions. Using questionnaires with a number of rating scales, different aspects of interaction quality, system usability, user satisfaction and acceptability may be obtained. A multidimensional analysis of user judgements obtained with an exemplary system revealed five perceptual dimensions underlying the users' judgements which were somehow stable across systems and users: Acceptability, communication efficiency, cognitive effort, system personality and dialogue smoothness.

Definitions have been provided for five categories of interaction parameters, based on a broad literature survey. This set has been evaluated in a pilot experiment, showing that the correlation between individual interaction parameters and subjective user judgements is indeed relatively low: Highest correlations were in the area of 0.6, and for overall quality not higher than 0.42. Thus, subjective quality judgements and interaction parameters provide complementary information to the evaluator.

Despite the low correlation, a combination of parameters can be used to predict overall quality or user satisfaction. Exemplary models have been calculated for the INSPIRE data, using a linear regression model as it was defined by the PARADISE framework. The models captured about 45% of the variance in the subjective data, provided that the right – informative – parameters are selected as input to the model. Still, this value is too low to replace subjective quality judgements by interaction parameters when the quality of SDS-based services is to be measured.

Both methods – subjective evaluation and parametric description of interactions – have been recommended by the ITU-T for telephone-based spoken dialogue services (ITU-T Rec. P.851, 2003; ITU-T Suppl. 24 to P-Series Rec., 2005). However, the set of interaction parameters described here is still very large, carrying in mind that most parameters rely on expert annotation. As soon as further evaluation data obtained with the described methods becomes available, the full set of interaction parameters should be reduced to parameters which have proven to be relevant for quality. Such a restricted set will form the basis for a new Recommendation P.PST which

will be developed by ITU-T SG12 in the next 1–2 years. It may also be used as input to quality prediction models; a Recommendation for such models is foreseen in the near future. Contributions to both Recommendation projects are invited by the ITU-T, see roadmap on http://www.itu.int/ITU-T/studygroups/com12/q12roadmap/index.html.

The evaluation methods addressed here are limited in several ways. Although the INSPIRE prototype used in the case study provides visual output as well, the evaluation metrics were limited to the speech modality. Evaluation metrics for multimodal dialogue systems are still under study (see Dybkjaer et al., 2004 for a discussion). In addition, it is assumed that the dialogue is task-oriented. For non-task-oriented dialogues, notions like "communication efficiency" or "task efficiency" will have a different signification. First steps for the evaluation of such systems have been presented recently (Bernsen and Dybkjær, 2005), but they are not yet conclusive.

Acknowledgements The present work has been performed at IKA, Ruhr-Universität Bochum, in the context of the EC-funded IST-project INSPIRE (IST-2001-32746), see http://www.knowledge-speech.gr/inspire-project. Partners of INSPIRE were: Knowledge S.A., Patras, and WCL, University of Patras, both Greece; IKA, Ruhr-Universität Bochum, and ABS Jena, both Germany; TNO Human Factors, Soesterberg and Philips Electronics Nederland B.V., Eindhoven, both The Netherlands; and EPFL, Lausanne, Switzerland. The author would like to thank Rosa Pegam for acting as a Wizard and for reviewing the manuscript; Noha El Mehelmi and Jörn Opretzka for annotating the dialogues; as well as all other INSPIRE partners for their support in the experiments and for fruitful discussions.

References

Bernsen, N. O., Dybkjær, H., and Dybkjær, L. (1998). *Designing Interactive Speech Systems: From First Ideas to User Testing*. Springer.

Bernsen, N. O. and Dybkjær, L. (2005). User Evaluation of Conversational Agent H. C. Andersen. In *Proceedings of 9th European Conference on Speech Communication and Technology (INTERSPEECH)*, pages 2473–2476, Lisbon.

Billi, R., Castagneri, G., and Danieli, M. (1996). Field Trial Evaluations of Two Different Information Inquiry Systems. In *Proceedings of Third IEEE Workshop on Interactive Voice Technology for Telecommunications Applications (IVTTA)*, pages 129–134, Basking Ridge.

Bodden, M. and Jekosch, U. (1996). *Entwicklung und Durchführung von Tests mit Versuchspersonen zur Verifizierung von Modellen zur Berechnung der Sprachübertragungsqualität*. Project report, Institut für Kommunikationsakustik, Ruhr-Universität, Bochum.

Boros, M., Eckert, W., Gallwitz, F., Görz, G., Hanrieder, G., and Niemann, H. (1996). Towards Understanding Spontaneous Speech: Word Accuracy vs. Concept Accuracy. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 1009–1012, Philadelphia.

Carletta, J. (1996). Assessing Agreement on Classification Tasks: The Kappa Statistics. *Computational Linguistics*, 22(2):249–254.

Chu, M. and Peng, H. (2001). An Objective Measure for Estimating MOS of Synthesized Speech. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 2087–2090, Aalborg.

Cookson, S. (1988). Final Evaluation of VODIS – Voice Operated Database Enquiry System. *Proceedings of SPEECH'88, 7th FASE Symposium*, 4:1311–1320.

Danieli, M. and Gerbino, E. (1995). Metrics for Evaluating Dialogue Strategies in a Spoken Language System. In *Empirical Methods in Discourse Interpretation and Generation. Papers from the 1995 AAAI Symposium, US-Stanford CA*, pages 34–39. AAAI Press, Menlo Park.

Delogu, C., di Carlo, A., Sementina, C., and Stecconi, S. (1993). A Methodology for Evaluating Human-Machine Spoken Language Interaction. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 1427–1430, Berlin.

den Os, E. and Bloothooft, G. (1998). Evaluating Various Spoken Dialogue Systems with a Single Questionnaire: Analysis of the ELSNET Olympics. In *Proceedings of International Conference on Language Resources and Evaluation (LREC)*, volume 1, pages 51–54, Granada.

Dybkjær, L., Bernsen, N. O., and Dybkjær, H. (1995). Scenario Design for Spoken Language Dialogue Systems Development. In *Proceedings of ESCA Workshop on Spoken Dialogue Systems*, pages 93–96, Vigsø.

Dybkjær, L., Bernsen, N. O., and Minker, W. (2004). Evaluation and Usability of Multimodal Spoken Language Dialogue Systems. *Speech Communication*, 43:33–54.

Fraser, N. (1997). Assessment of Interactive Systems. In Gibbon, D., Moore, R., and Winski, R., editors, *Handbook on Standards and Resources for Spoken Language Systems*, pages 564–615. Mouton de Gruyter, Berlin.

Gerbino, E., Baggia, P., Ciaramella, A., and Rullent, C. (1993). Test and Evaluation of a Spoken Dialogue System. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 135–138, Minneapolis.

Glass, J., Polifroni, J., Seneff, S., and Zue, V. (2000). Data Collection and Performance Evaluation of Spoken Dialogue Systems: The MIT Experience. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, volume 4, pages 1–4, Beijing.

Goodine, D., Hirschman, L., Polifroni, J., Seneff, S., and Zue, V. (1992). Evaluating Interactive Spoken Language Systems. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, volume 1, pages 201–204, Banff.

Grice, H. P. (1975). Logic and Conversation. Syntax and Semantics. In Cole, P. and Morgan, J. L., editors, *Speech Acts*, volume 3, pages 41–58. Academic, New York.

Hirschman, L. and Pao, C. (1993). The Cost of Errors in a Spoken Language System. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, volume 2, pages 1419–1422, Berlin.

Hone, K. S. and Graham, R. (2000). Towards a Tool for the Subjective Assessment of Speech System Interfaces (SASSI). *Natural Language Engineering*, 3(3-4):287–303.

IEC Standard 60268-16 (1998). *Sound System Equipment – Part 16: Objective Rating of Speech Intelligibility by Speech Transmission Index*. European Committee for Electrotechnical Standardization, Brussels.

ITU-T Rec. P.851 (2003). *Subjective Quality Evaluation of Telephone Services Based on Spoken Dialogue Systems*. International Telecommunication Union, Geneva.

ITU-T Suppl. 24 to P-Series Rec. (2005). *Parameters Describing the Interaction with Spoken Dialogue Systems*. International Telecommunication Union, Geneva.

Jack, M. A., Foster, J. C., and Stentiford, F. W. M. (1992). Intelligent Dialogues in Automated Telephone Services. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, volume 1, pages 715–718, Banff.

Jekosch, U. (2000). *Sprache Hören und Beurteilen: Ein Ansatz zur Grundlegung der Sprachqualitätsbeurteilung*. Habilitation thesis (unpublished), Universität/Gesamthochschule, Essen.

Jekosch, U. (2005). *Voice and Speech Quality Perception. Assessment and Evaluation*. Springer, Berlin.

Kamm, C., Narayanan, S., Dutton, D., and Ritenour, R. (1997). Evaluating Spoken Dialogue Systems for Telecommunication Services. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, volume 4, pages 2203–2206, Rhodes.

Kamm, C. A., Litman, D. J., and Walker, M. A. (1998). From Novice to Expert: The Effect of Tutorials on User Expertise with Spoken Dialogue Systems. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, volume 4, pages 1211–1214, Sydney.

Lamel, L., Bennacef, S., Gauvain, J. L., Dartigues, H., and Temem, J. N. (2002). User Evaluation of the MASK Kiosk. *Speech Communication*, 38(1):131–139.

Love, S., Dutton, R. T., Foster, J. C., Jack, M. A., and Stentiford, F. W. M. (1994). Identifying Salient Usability Attributes for Automated Telephone Services. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, volume 3, pages 1307–1310, Yokohama.

Möller, S. (2005a). Perceptual Quality Dimensions of Spoken Dialogue Systems: A Review and New Experimental Results. In *Proceedings of 4th European Congress on Acoustics (Forum Acusticum Budapest 2005)*, pages 2681–2686, Budapest.

Möller, S. (2005b). *Quality of Telephone-Based Spoken Dialogue Systems*. Springer, New York.

Möller, S., Smeele, P., Boland, H., and Krebber, J. (2006). Evaluating Spoken Dialogue Systems According to De-facto Standards. *Computer Speech and Language*, 21:26–53.

Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann, San Francisco.

Nielsen, J. and Mack, R. L., editors (1994). *Usability Inspection Methods*. Wiley, New York.

NIST (2001). *Speech Recognition Scoring Toolkit*. National Institute of Standards and Technology, http://www.nist.gov/speech/tools, Gaithersburg.

Polifroni, J., Hirschman, L., Seneff, S., and Zue, V. (1992). Experiments in Evaluating Interactive Spoken Language Systems. In *Proceedings of DARPA Speech and Natural Language Workshop*, pages 28–33, Harriman.

Price, P. J., Hirschman, L., Shriberg, E., and Wade, E. (1992). Subject-Based Evaluation Measures for Interactive Spoken Language Systems. In *Proceedings of DARPA Speech and Natural Language Workshop*, pages 34–39, Harriman.

San-Segundo, R., Montero, J. M., Colás, J., Gutiérrez, J., Ramos, J. M., and Pardo, J. M. (2001). Methodology for Dialogue Design in Telephone-Based Spoken Dialogue Systems: A Spanish Train Information System. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, volume 3, pages 2165–2168, Aalborg.

Shneiderman, B. (2003). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, Reading.

Simpson, A. and Fraser, N. M. (1993). Black Box and Glass Box Evaluation of the SUNDIAL System. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, volume 2, pages 1423–1426, Berlin.

Skowronek, J. (2002). *Entwicklung von Modellierungsansätzen zur Vorhersage der Dienstequalität bei der Interaktion mit einem natürlichsprachlichen Dialogsystem*. Diploma thesis (unpublished), Institut für Kommunikationsakustik, Ruhr-Universität, Bochum.

Strik, H., Cucchiarini, C., and Kessens, J. M. (2000). Comparing the Recognition Performance of CSRs: In Search of an Adequate Metric and Statistical

Significance Test. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, volume 4, pages 740–743, Beijing.

Strik, H., Cucchiarini, C., and Kessens, J. M. (2001). Comparing the Performance of two CSRs: How to Determine the Significance Level of the Differences. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, volume 3, pages 2091–2094, Aalborg.

van Leeuwen, D. and Steeneken, H. (1997). Assessment of Recognition Systems. In Gibbon, D., Moore, R., and Winski, R., editors, *Handbook on Standards and Resources for Spoken Language Systems*, pages 381–407. Mouton de Gruyter, Berlin.

Walker, M. A., Litman, D. J., Kamm, C. A., and Abella, A. (1997). PARADISE: A Framework for Evaluating Spoken Dialogue Agents. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL-EACL)*, pages 271–280, Morgan Kaufmann, Madrid.

Walker, M. A., Litman, D. J., Kamm, C. A., and Abella, A. (1998). Evaluating Spoken Dialogue Agents with PARADISE: Two Case Studies. *Computer Speech and Language*, 12(3):317–347.

Zue, V., Seneff, S., Glass, J. R., Polifroni, J., Pao, C., Hazen, T. J., and Hetherington, L. (2000). JUPITER: A Telephone-Based Conversational Interface for Weather Information. *IEEE Transactions on Speech and Audio Processing*, 8(1):85–96.

Appendix: Definition of Interaction Parameters

Table 4.A.1. Dialogue- and communication-related parameters (1). Interaction level: word, utt. (utterance), dial. (dialogue); measurement method: instr. (instrumental), exp. (expert-based).

Abbr.	*Name*	*Definition*	*Int. level*	*Meas. meth.*
DD	Dialogue duration	Overall duration of a dialogue in ms (see, e.g. Fraser, 1997)	dial.	instr.
STD	System turn duration	Average duration of a system turn, from the system starting speaking to the system stopping speaking, in ms. A turn is an utterance, i.e. a stretch of speech spoken by one party in the dialogue (Fraser, 1997)	utt.	instr.
UTD	User turn duration	Average duration of a user turn, from the user starting speaking to the user stopping speaking, in ms (Fraser, 1997)	utt.	instr.
SRD	System response delay	Average delay of a system response, from the user stopping speaking to the system starting speaking, in ms (Fraser, 1997)	utt.	instr.
URD	User response delay	Average delay of a user response, from the system stopping speaking to the user starting speaking, in ms (Fraser, 1997)	utt.	instr.
# Turns	Number of turns	Overall number of turns uttered in a dialogue (Walker et al., 1998)	dial.	instr./ exp.
# System turns	Number of system turns	Overall number of system turns uttered in a dialogue (Walker et al., 1998)	dial.	instr./ exp.
# User turns	Number of user turns	Overall number of user turns uttered in a dialogue (Walker et al., 1998)	dial.	instr./ exp.
$WPST$	Words per system turn	Average number of words per system turn in a dialogue (Cookson, 1988)	utt.	instr./ exp.
$WPUT$	Words per user turn	Average number of words per user turn in a dialogue (Cookson, 1988)	utt.	instr./ exp.
# System questions	Number of system questions	Overall number of questions from the system per dialogue.	dial.	exp.
# User questions	Number of user questions	Overall number of questions from the user per dialogue (Polifroni et al., 1992; Goodine et al., 1992)	dial.	exp.

Table 4.A.2. Dialogue- and communication-related parameters (2).

Abbr.	*Name*	*Definition*	*Int. level*	*Meas. meth.*
QD	Query density	Average number of new concepts (slots) introduced per user query. Being n_d the number of dialogues, $n_q(i)$ the total number of user queries in the ith dialogue, and $n_u(i)$ the number of unique concepts correctly "understood" by the system in the ith dialogue, then $$QD = \frac{1}{n_d}\sum_{i=1}^{n_d}\frac{n_u(i)}{n_q(i)}$$ A concept is not counted to $n_u(i)$ if the system has already "understood" it from a previous utterance (Glass et al., 2000)	set of dial.	exp.
CE	Concept efficiency	Average number of turns necessary for each concept to be "understood" by the system. Being n_d the number of dialogues, $n_u(i)$ the number of unique concepts correctly "understood" by the system in the ith dialogue, and $n_c(i)$ the total number of concepts in the ith dialogue, then $$CE = \frac{1}{n_d}\sum_{i=1}^{n_d}\frac{n_u(i)}{n_c(i)}$$ A concept is counted whenever it was uttered by the user and was not already "understood" by the system (Glass et al., 2000)	set of dial.	exp.

Table 4.A.3. Meta-communication-related parameters (1).

Abbr.	*Name*	*Definition*	*Int. level*	*Meas. meth.*
# Help requests	Number of help requests	Overall number of user help requests in a dialogue. A user help request is labelled by the annotation expert if the user explicitly asks for help. This request may be formulated as a question (e.g. "What are the available options?") or as a statement ("Give me the available options!") (Walker et al., 1998)	utt.	exp.

Table 4.A.4. Meta-communication-related parameters (2).

Abbr.	*Name*	*Definition*	*Int. level*	*Meas. meth.*
# System help	Number of diagnostic system help messages	Overall number of help messages generated by the system in a dialogue. A help message is a system utterance which informs the user about available options at a certain point in the dialogue	utt.	instr./ exp.
# Time-out	Number of time-out prompts	Overall number of time-out prompts, due to no response from the user, in a dialogue (Walker et al., 1998)	utt.	instr.
# ASR rejection	Number of ASR rejections	Overall number of ASR rejections in a dialogue. An ASR rejection is defined as a system prompt indicating that the system was unable to "hear" or to "understand" the user, i.e. that the system was unable to extract any meaning from a user utterance (Walker et al., 1998)	utt.	instr.
# System error	Number of diagnostic system error messages	Overall number of diagnostic error messages from the system in a dialogue. A diagnostic error message is defined as a system utterance in which the system indicates that it is unable to perform a certain task or to provide a certain information (Price et al., 1992)	utt.	instr./ exp.
# Barge-in	Number of user barge-in attempts	Overall number of user barge-in attempts in a dialogue. A user barge-in attempt is counted when the user intentionally addresses the system while the system is still speaking. In this definition, user utterances which are not intended to influence the course of the dialogue (laughing, expressions of anger or politeness) are not counted as barge-ins (Walker et al., 1998)	utt.	exp.
# Cancel	Number of user cancel attempts	Overall number of user cancel attempts in a dialogue. A user turn is classified as a cancel attempt if the user tries to restart the dialogue from the beginning, or if he/she explicitly wants to step one or several levels backwards in the dialogue hierarchy (Kamm et al., 1998; San-Segundo et al., 2001)	utt.	exp.

Table 4.A.5. Meta-communication-related interaction parameters (3).

Abbr.	*Name*	*Definition*	*Int. level*	*Meas. meth.*
SCT, SCR	Number of system correction turns, system correction rate	Overall number (SCT) or percentage (SCR) of all system turns in a dialogue which are primarily concerned with rectifying a "trouble", thus not contributing new propositional content and interrupting the dialogue flow. A "trouble" may be caused by speech recognition or understanding errors, or by illogical, contradictory, or undefined user utterances. In case that the user does not give an answer to a system question, the corresponding system answer is labelled as a system correction turn, except when the user asks for information or for an action which is not supported by the current system functionality (Simpson and Fraser, 1993; Gerbino et al., 1993)	utt.	exp.
UCT, UCR	Number of user correction turns, user correction rate	Overall number (UCT) or percentage (UCR) of all user turns in a dialogue which are primarily concerned with rectifying a "trouble", thus not contributing new propositional content and interrupting the dialogue flow (see SCT, SCR) (Simpson and Fraser, 1993; Gerbino et al., 1993)	utt.	exp.
IR	Implicit recovery	Capacity of the system to recover from user utterances for which the speech recognition or understanding process partly failed. Determined by labelling the partially parsed utterances (see definition of PA:PA) as to whether the system response was "appropriate" or not: $IR = \frac{\text{\# utt. with appropriate syst. answer}}{PA{:}PA}$ For the definition of "appropriateness" see Grice (1975) and Bernsen et al. (1998) (Danieli and Gerbino, 1995)	utt.	exp.

Table 4.A.6. Cooperativity-related parameters.

Abbr.	*Name*	*Definition*	*Int. level*	*Meas. meth.*
CA	Contextual appropriateness	Overall number of system utterances which are judged to be appropriate in their immediate dialogue context. Determined by labelling utterances according to whether they violate one or more of Grice's maxims for cooperativity: – CA:AP: Appropriate, not violating Grice's maxims, not unexpectedly conspicuous or marked in some way. – CA:IA: Inappropriate, violating one or more of Grice's maxims. – CA:TF: Total failure, no linguistic response. – CA:IC: Incomprehensible, content cannot be discerned by the annotation expert. For more details see Simpson and Fraser (1993) and Gerbino et al. (1993); the classification is similar to the one adopted in Hirschman and Pao (1993).	utt.	exp.

Table 4.A.7. Task-related parameters.

Abbr.	*Name*	*Definition*	*Int. level*	*Meas. meth.*
TS	Task success	Label of task success according to whether the user has reached his/her goal by the end of a dialogue, provided that this goal could be reached with the help of the system. The labels indicate whether the goal was reached or not, and the assumed source of problems: – S: Succeeded (tasks for which solutions exist). – SCs: Succeeded with constraint relaxation by the system. – SCu: Succeeded with constraint relaxation by the user. – $SCuCs$: Succeeded with constraint relaxation both from the system and from the user. – SN: Succeeded in spotting that no solution exists. – Fs: Failed because of the system's behaviour, due to system inadequacies. – Fu: Failed because of the user's behaviour, due to non-cooperative user behaviour. See also Fraser (1997), Simpson and Fraser (1993) and Danieli and Gerbino (1995)	dial.	exp.
κ	Kappa coefficient	Percentage of task completion according to the kappa statistics. Determined on the basis of the correctness of the result AVM reached at the end of a dialogue with respect to the scenario (key) AVM. A confusion matrix $M(i,j)$ is set up for the attributes in the result and in the key, with T the number of counts in M, and t_i the sum of counts in column i of M. Then $$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$ with $P(A)$ being the proportion of times that the AVM of the actual dialogue and the key agree, $P(A) = \sum_{i=1}^{n} \frac{M(i,i)}{T}$. $P(E)$ can be estimated from the proportion of times that they are expected to agree by chance, $P(E) = \sum_{i=1}^{n} (\frac{t_i}{T})^2$ (Carletta, 1996; Walker et al., 1997)	dial. or set of dial.	exp.

Table 4.A.8. Speech-input-related parameters (1).

Abbr.	*Name*	*Definition*	*Int. level*	*Meas. meth.*
WER, WA	Word error rate, word accuracy	Percentage of words which have been (in-) correctly recognised, based on the orthographic form of the hypothesized and the (transcribed) reference utterance, and an alignment carried out with the help of the "sclite" algorithm, see (NIST, 2001). Designating n_w the overall number of words from all user utterances of a dialogue, and s_w, d_w and i_w the number of substituted, deleted and inserted words, respectively, then the word error rate and word accuracy can be determined as follows: $$WER = \frac{s_w + i_w + d_w}{n_w}$$ $$WA = 1 - \frac{s_w + i_w + d_w}{n_w} = 1 - WER$$ See also Simpson and Fraser (1993); details on how these parameters can be calculated in case of isolated word recognition are given in van Leeuwen and Steeneken (1997)	word	instr./ exp.
SER, SA	Sentence error rate, sentence accuracy	Percentage of entire sentences which have been (in-) correctly identified. Denoting n_s the total number of sentences, and s_s, i_s and d_s the number of substituted, inserted and deleted sentences, respectively, then: $$SER = \frac{s_s + i_s + d_s}{n_s}$$ $$SA = 1 - \frac{s_s + i_s + d_s}{n_s} = 1 - SER$$ (Simpson and Fraser, 1993)	utt.	instr./ exp.

Table 4.A.9. Speech-input-related parameters (2).

Abbr.	*Name*	*Definition*	*Int. level*	*Meas. meth.*
NES	Number of errors per sentence	Average number of recognition errors in a sentence. Being $s_w(k)$, $i_w(k)$ and $d_w(k)$ the number of substituted, inserted and deleted words in sentence k, and n_s the number of sentences, then $$NES(k) = s_w(k) + i_w(k) + d_w(k)$$ The average NES can be calculated as follows: $$NES = \frac{\sum_{k=1}^{n_s} NES(k)}{n_s} = \frac{WER \cdot n_w}{n_s}$$ (Strik et al., 2001)	utt.	instr./ exp.
WES	Word error per sentence	Related to NES, but normalised to the number of words in sentence k, $w(k)$: $$WES(k) = \frac{NES(k)}{w(k)}$$ The average WES can be calculated as follows: $$WES = \frac{\sum_{k=1}^{n_s} WES(k)}{n_s}$$ (Strik et al., 2001)	word	instr./ exp.
$AN{:}CO$, $AN{:}IC$, $AN{:}PA$, $AN{:}FA$, $\%AN{:}CO$, $\%AN{:}IC$, $\%AN{:}PA$, $\%AN{:}FA$	Number or percentage of user questions with correct/ incorrect/ partially correct/ failed system answers	Overall number or percentage of questions from the user which are – correctly ($AN{:}CO$) – incorrectly ($AN{:}IC$) – partially correctly ($AN{:}PA$) – not at all ($AN{:}FA$) answered by the system, per dialogue, see Polifroni et al. (1992), Goodine et al. (1992) and Hirschman and Pao (1993)	utt.	exp.

Table 4.A.10. Speech-input-related parameters (3).

Abbr.	*Name*	*Definition*	*Int. level*	*Meas. meth.*
$DARPA_s$, $DARPA_{me}$	DARPA score, DARPA modified error	Measures according to the DARPA speech understanding initiative, modified by Skowronek (2002) to account for partially correct answers: $DARPA_s = \frac{AN{:}CO - AN{:}IC}{\text{\# user questions}}$ $DARPA_{me} = \frac{AN{:}FA + 2 \cdot (AN{:}IC + AN{:}PA)}{\text{\# user questions}}$ (Polifroni et al., 1992; Goodine et al., 1992; Skowronek, 2002)	utt.	exp.
$PA{:}CO$, $PA{:}PA$, $PA{:}FA$, $\%PA{:}CO$, $\%PA{:}PA$, $\%PA{:}FA$	Number of correctly/ partially correctly/ incorrectly parsed user utterances	Evaluation of the number of concepts (attribute-value pairs, AVPs) in an utterance which have been extracted by the system: – $PA{:}CO$: All concepts of a user utterance have been correctly extracted by the system. – $PA{:}PA$: Not all but at least one concept of a user utterance has been correctly extracted by the system. – $PA{:}FA$: No concepts of a user utterance have been correctly extracted by the system. Expressed as the overall number or percentage of user utterances in a dialogue which have been parsed correctly/ partially correctly/incorrectly (Danieli and Gerbino, 1995)	utt.	exp.

Table 4.A.11. Speech-input-related parameters (4).

Abbr.	*Name*	*Definition*	*Int. level*	*Meas. meth.*
CA, CER	Concept accuracy, concept error rate	Percentage of correctly understood semantic units, per dialogue. Concepts are defined as attribute-value pairs (AVPs), with n_{AVP} the total number of AVPs, and s_{AVP}, i_{AVP} and d_{AVP} the number of substituted, inserted and deleted AVPs. The concept accuracy and the concept error rate can then be determined as follows: $CA = 1 - \frac{s_{AVP} + i_{AVP} + d_{AVP}}{n_{AVP}}$ $CER = \frac{s_{AVP} + i_{AVP} + d_{AVP}}{n_{AVP}}$ (Gerbino et al., 1993; Simpson and Fraser, 1993; Boros et al., 1996; Billi et al., 1996)	utt.	exp.
UA	Understanding accuracy	Percentage of utterances in which all semantic units (AVPs) have been correctly extracted: $UA = \frac{PA : CO}{\text{\# user turns}}$ (Zue et al., 2000)	utt.	exp.

Chapter 5

HANDLING MISCOMMUNICATION: WHY BOTHER?

Michael McTear
University of Ulster, Northern Ireland
mf.mctear@ulster.ac.uk

Abstract Miscommunication occurs frequently in any communicative interaction. Despite a long history of studies of miscommunication across a range of different disciplines, the focus in spoken dialogue technology has been almost exclusively on miscommunication caused by speech recognition errors. This chapter reviews approaches to miscommunication in several different disciplines and examines the issues involved in assessing the need to deal with miscommunication that arises in interactions with spoken dialogue systems and what methods can be used in different interactional contexts to repair miscommunication.

Keywords: Miscommunication; Error handling; Grounding.

1. Introduction

Miscommunication is a frequent occurrence in any communicative interaction, whether this communication is at a global level between different nations, ethnic or religious groups, or at an individual level, for example, between manager and subordinate worker, doctor and patient, male and female, or older and younger person. The analysis of the causes of miscommunication can be traced back to the writings of the classical Greeks and there is an immense literature dealing with miscommunication in all its various forms and guises. At the same time there are many views on how miscommunication should be handled in the different forms and contexts in which it occurs.

This chapter provides an overview of miscommunication in the context of spoken natural language interaction between humans and computers — an area known generally as spoken dialogue technology (McTear, 2004). Most work in this area has focussed on miscommunication caused by inaccurate speech recognition. In a recent study, for example, it was found that the majority of

L. Dybkjær and W. Minker (eds.), Recent Trends in Discourse and Dialogue, 101–122.

errors occurred at the speech recognition level (Bohus and Rudnicky, 2007). Various ways of detecting and handling speech recognition errors will be reviewed below. However, it is interesting that a fairly large number of the errors analysed in this study were caused by other factors such as user's utterances that were outside the system's domain or that were outside the system's semantic grammar. Indeed, as speech recognition accuracy improves, we can anticipate that miscommunication at other levels of interaction will become more prominent and that it will be necessary to consider ways of dealing with these different types of miscommunication.

This chapter is organised as follows. The next section examines some common definitions of miscommunication with a view to determining the types of miscommunication that need to be addressed in spoken dialogue systems. This is followed by a review of approaches to miscommunication across a number of disciplines. To date, many of the issues raised in these approaches have not been addressed in spoken dialogue technology and thus it is important to assess their relevance. The next section briefly reviews current work on error handling in spoken dialogue systems, looking at how errors can be detected by monitoring the dialogue for cues that an error has occurred; how potential errors can be predicted based on evidence from the dialogue so far; and what strategies can be used to recover from errors. Based on these overviews of approaches from other disciplines and current work in spoken dialogue systems, two important questions will be discussed:

1. Why bother with errors — what are the costs and benefits?

2. If we decide to deal with errors, how should we go about it?

It will be seen that some of the decisions relating to these questions will depend crucially on various aspects of the content of the interaction — for example, whether the main focus of the interaction is on referential communication (i.e. to support some transaction), or whether the focus is more on interpersonal aspects of the communication. The significance of these distinctions for the more novel types of dialogue system that are currently being created will be assessed.

2. Defining Miscommunication

There are many different definitions of miscommunication. The following is a typical dictionary definition (Dictionary, 2004):

> Lack of clear or adequate communication.

However, this definition is not particularly helpful as it fails to identify the source of the problem and does not specify what is meant by 'clear' or 'adequate' communication.

In many approaches miscommunication is viewed from the perspective of the recipient (or hearer) of the communication. For example, McRoy (1998) distinguishes between misunderstanding, non-understanding, and misinterpretation. In this definition misunderstanding is where one participant obtains an interpretation that she believes is complete and correct, but which is, however, not the one that the other speaker intended her to obtain. A non-understanding is where a participant either fails to obtain any interpretation at all, or obtains more than one interpretation, with no way to choose among them. Finally, with a misinterpretation the most likely interpretation of a participant's utterance suggests that their beliefs about the world are unexpectedly out of alignment with those of the other dialogue participant. In terms of spoken dialogue systems, non-understandings are generally easier to detect and repair, misunderstandings may be detected and resolved given appropriate verification strategies, but misinterpretations may only become apparent at a later point in the dialogue, if at all.

This notion of misalignment is taken a stage further by Traum and Dillenbourg (1996) who define miscommunication as

> one particular case of a lack of alignment of agents' mental state, specifically one in which they diverge on the occurrence or results of communication..., miscommunication can be viewed as instances of action failure (when the speaker fails to produce the intended effect), misperception (when the hearer cannot recognise what the speaker intended to communicate), or both.

In this definition the focus is on the communicative dyad and the source of the miscommunication can be the speaker, the hearer, or both speaker and hearer. This definition is close to the concept of 'grounding', to be discussed later, in which successful communication is achieved through mutual grounding of information between the dialogue participants, without attempting to assign blame to any of the participants for the miscommunication.

In graphical user interfaces error handling is viewed in terms of addressing 'ill-formed input', where the user has made an illegal or erroneous choice of input either as a result of a mistake or due to a lack of knowledge of what is permissible in the current context. If the system detects the error it addresses it with some form of warning or help. While ill-formed input is also possible in spoken dialogue systems, for example, a mispronunciation or choice of an incorrect word, the blame for errors is usually attributed to the system, generally in terms of inadequate or inaccurate speech recognition. Errors could also occur at other levels of the linguistic hierarchy, i.e. at the syntactic, semantic, pragmatic, and discourse levels, however few spoken dialogue systems address errors at these levels. Table 1 provides some examples of miscommunication at different levels of the linguistic hierarchy.

Generally error handling deals with errors that involve referential aspects of communication, i.e. the content of what is being communicated. Increasingly,

Table 1. Linguistic levels of miscommunication.

Level	*Example*
Speech recognition	'four candles' or 'fork handles'
Lexical	'plugs' electrical or bath plugs?
Syntactic	'drink the beer in the fridge'
	the beer which is in the fridge or do drinking in fridge?
Semantic	'every student has to write an essay'
	different or same essay for each student?
Discourse	Referential ambiguity: 'put it there'
Pragmatic: Intentions	'is that door open' question or command?

Table 2. Miscommunication across a range of disciplines.

Discipline	*Examples of areas of miscommunication*
Sociolinguistics and Ethnography	cross-cultural, intergenerational, cross-gender
Communication Science	global and organisational
Social Psychology	gender, professional e.g. doctor–patient
Conversation Analysis	organisation of repair in conversation
Natural Language Processing	referential ambiguity, misconceptions

however, there has been an interest in exploring affective aspects of communication. Here there is less emphasis on the content and more on the emotional overtones being conveyed. Indeed, in many cases of miscommunication between humans it is the affective aspect that gives rise to the miscommunication rather than the content of the message itself. This more advanced aspect of spoken language dialogue will be considered in more detail later.

3. Approaches to Miscommunication in Other Disciplines

Miscommunication has been addressed in a wide range of disciplines, including: sociolinguistics, ethnography, communication science, media studies, social psychology, conversation analysis, and natural language processing. Moreover, the domain of enquiry has ranged across a wide variety of interactional contexts, including global miscommunication between nations, ethnic groups and religions; organisational miscommunication within businesses and in contexts of employment; and various other areas such as cross-cultural, male–female, intergenerational, and professional miscommunication, such as doctor–patient and teacher–student. Table 2 lists a range of disciplines along with the areas of miscommunication that they have mainly been concerned with.

Most of this literature has been ignored or considered irrelevant to spoken dialogue research. In this section a selection of different approaches and issues will be briefly reviewed and assessed in terms of their relevance to the treatment of miscommunication in spoken dialogue systems.

3.1 Global and Organisational Miscommunication

There is a long tradition of research on global and organisational communication that can be traced back to the writings of the ancient Greeks, such as Socrates, and that has addressed the breakdown of communication between groups. At the broadest level this includes nations, ethnic groups, and religious groups, and within organisations groups such as management and workforce as well as individuals within the organisations. In this tradition miscommunication is viewed as being a result of conflicting experiences, lifestyles, desires, and values. Much of this work on miscommunication involves studies of 'effective communication', where 'dialogue' is recommended as a way to resolve differences. Note that in this work dialogue is viewed as a tool to enable groups and individuals to communicate more effectively, simply as a result of engaging in dialogue. In spoken dialogue technology, on the other hand, the process of dialogue is viewed as being problematic and error prone. For this reason this tradition of research has little relevance to spoken dialogue research.

3.2 Cross-Cultural and InterGender Miscommunication

A more relevant research tradition is the micro-analysis of communication processes between individuals from different cultural and linguistic backgrounds. In this work the aim is to explore how communication breaks down as a result of differences in cultural norms for language use, interaction, and the use of conversational inferences. An example of this type of analysis is the work of Gumperz (1978). For example, in one study it was shown how the use of different prosodic conventions can give the impression of rudeness, even though this might be unintentional, as in the use of stress on the word 'please' in the phrase 'exact change please' by West Indian bus drivers in London. Other bus drivers saying exactly the same words without this particular prosodic feature did not seem to give offence in the same way. Thus miscommunication in the sense of the communication of the affective aspects of a message is closely related to subtle differences in what can be conveyed by a particular linguistic feature across different cultural groups that share the same language. This finding has particular relevance for the deployment of call centre agents as well as for the use of automated systems using particular accents and prosodic patterns.

There is an extensive literature on language and gender in which differences in the use of language are used to explain miscommunication between males and females. Earlier studies focused on differences at more traditional linguistic levels, such as phonology, morphology, syntax and lexis (Coates, 1998), while more recently the emphasis has shifted to differences in conversational style (Tannen, 2001). Findings include differences in the use of back channel cues to acknowledge a preceding utterance, where women tend to use these cues to indicate attention and men use them to indicate agreement. Miscommunication arises where men expect that the use of a minimal back channel response signals agreement rather than simply attention. Similarly, it has been reported that women tend to use questions more to help maintain a conversation while men use questions to request information (Maltz and Borker, 1982). Findings such as these are potentially relevant for advanced spoken dialogue systems where there are gender differences between the dialogue participants.

More generally, there has been a long tradition of the study of miscommunication in different social contexts. For example, Coupland et al. (1991) is a collection of papers that investigates miscommunication in terms of gender, generational, and cultural differences, in clinical and organisational contexts, and in person–machine communication. Similarly, Clark (1996) has conducted numerous detailed studies of how humans communicate with one another, while Nass and Brave (2004) report a range of studies of how people interact with spoken dialogue systems, looking at issues such as the gender of the system voice, accent, race, ethnicity, and personality. Some of the findings of these studies are relevant to the handling of miscommunication in spoken dialogue systems, particularly more advanced systems in which there is more focus on affective aspects of communication. Some examples will be presented later.

3.3 Conversation Analysis

There is a long tradition of research known as Conversation Analysis (CA) that involves micro-analysis of transcripts of conversation and that is potentially relevant to spoken dialogue researchers. CA developed out of ethnomethodology, which is a branch of sociology that examines how people interpret and act within a social world. Applied to conversation the methods of CA involve searching for recurring patterns across many records of naturally occurring conversations with the aim of discovering the systematic properties of the sequential organisation of talk and how utterances are designed to manage such sequences. One example of this work is the study of how conversational repair is organised (Schegloff et al., 1977). Four types of repair were distinguished, depending on who initiated the repair and who carried it out. The speaker whose utterance contains a repairable element is described as

'self' while the recipient of the utterance is referred to as 'other'. This gives rise to a four-way distinction:

- Self-initiated self-repair (in own turn, turn 1)
- Self-initiated self-repair (in turn transition)
- Other-initiated self-repair (in turn 2)
- Other-initiated other-repair (in turn 2)

Some interesting findings arose from the analysis of conversational repairs using these distinctions. On the one hand, the opportunities for repair are sequentially ordered in that opportunities for self initiation of repair occur before those for other initiation of repair, as indicated above. Similarly repair by self occurs before repair by other. These findings led to a more general conclusion that repairs are organised in terms of a system of 'preference' in which self-initiation of repair is preferred to other-initiation, while self-repair is preferred to other-repair. In this analysis the term 'preference' is used to describe the feature of markedness. In other words, items that are preferred are unmarked, while those that are dispreferred are marked. With reference to repair this means that dialogue acts such as corrections of the other person's utterance are normally delayed and marked in some way — for example, with indicators of uncertainty or hesitation. In this way the speaker of the utterance containing a potential error is given the opportunity to self-correct, and corrections by the recipient of the utterance are modulated so that they appear less direct.

Generally the CA tradition has been ignored in spoken dialogue research, except for occasional references to commonly used terms such as turn-taking and adjacency pairs. One problem is that the CA approach is to avoid pre-mature formalisation, so that descriptions of phenomena such as turn-taking, adjacency pairs, and repair are not easily formulated as rules that can be implemented as algorithms. Indeed, as Button et al. (1995, chapter 7) argue strongly, the patterns and processes that are described in CA represent normative principles rather than formal rules. Ultimately the point being made here is that it is conceptually impossible to develop computers that can talk (and think) in the ways that humans do.

Notwithstanding these objections, there are aspects of the findings of CA that are relevant to the engineering of spoken dialogue interfaces, such as enabling the system to behave in a way that simulates the processes of human conversation. So, for example, in relation to miscommunication, it might be assumed that, if the system detects a potential error, one way to rectify it would be to immediately correct the error. In this way the system would assume responsibility for the dialogue. However, in terms of conversational style this would be a dispreferred strategy and a more human-like system might use

either a more mitigated form of correction or a request for clarification, which is essentially an invitation to self-correct. Whether this would be a suitable strategy for all types of spoken dialogue interaction would then be an empirical issue requiring further investigation.

3.4 Natural Language Interfaces

A considerable amount of research was carried out in the 1980s concerned with miscommunication involving natural language interfaces. Natural language interfaces at that time used natural language text for input and output, and were mainly systems for retrieving information from databases using natural language or, in some cases, interfaces for communicating with expert systems. Given that these interfaces did not use speech, the problems associated with speech recognition did not need to be addressed and attention could be focused on other problems such as various types of ill-formed input at other levels of linguistic analysis. In the simplest case ill-formed input might be a misspelling; however, more complex cases involved issues such as reference identification failures, false assumptions, misconceptions, and discrepancies between the beliefs, goals and intentions of the dialogue participants. A collection of a number of papers illustrating this research can be found in Reilly (1987). The following are some examples.

3.4.1 False assumptions. False assumptions can occur when a dialogue participant incorrectly applies a default rule of inference. For example, in the following dialogue the default inference is that associate professors have tenure (in universities in the United States), but the recipient of the question blocks the misapplication of this default rule so that the first speaker does not make a false assumption (Joshi et al., 1987):

> Q: Is Sam an associate professor?
> R: Yes, but he doesn't have tenure.

3.4.2 Misconceptions. Misconceptions occur when a dialogue participant has a belief that is incorrect. In the following example, the first speaker believes that destroyers can have a mast height above 190. However, within the database destroyers have a mast height between 85 and 90. The system corrects the misconception and indicates an alternative formulation of the question (McCoy, 1987):

> Q: Give me the HULL-NO of all DESTROYERS whose MAST-HEIGHT is above 190.
> R: There are no DESTROYERS in the database having MAST-HEIGHT above 190. All DESTROYERS have a MAST-HEIGHT between 85 and 90.
> Were you thinking of an AIRCRAFT-CARRIER?

3.4.3 Pragmatic overshoot. A similar type of misconception has been described by Carberry (1987) as 'pragmatic overshoot' which arises when the speaker's view of the world may differ from that of the listener, as in the following example:

Q: What apartments are for sale?

This question is pragmatically inappropriate in a context in which apartments are not sold, only apartment buildings. Correction of this error requires a context model of the speaker's underlying task-related plan as inferred from the preceding dialogue.

Much of this work is potentially relevant to spoken dialogue technology, yet currently there has been little reference to the issues addressed in this tradition and the techniques used. In general the motivation for much of the research in this tradition was the development of co-operative interfaces that could detect errors, misconceptions, and other forms of problematic input that would potentially lead to miscommunication. The techniques applied were based on artificial intelligence research on topics such as inference, plan recognition, and reasoning. One reason why these issues have not been addressed is the need to focus in current spoken dialogue systems on the more immediate problem of the detection and handling of speech recognition errors. It might be expected that, as spoken dialogue systems become more advanced and incorporate more human-like capabilities, there will be a need to address issues such as co-operativity and misconceptions. The work of the TRIPS project at the University of Rochester, which derives from AI-based work on dialogue and planning in the 1980s, is a good example of more advanced spoken dialogue systems that incorporate AI-based components for planning and context representation to resolve issues such as referential ambiguity and ill-formed plans (Allen et al., 2001). However, it is interesting to note that problem solving and reasoning in the TRIPS system have to be constrained in the interests of real-time performance. Some issues that would need to be investigated in respect of AI-based approaches to advanced natural language interfaces are the degree of computational complexity involved in the reasoning process, the scalability of the solutions, and their generalisability to other domains.

4. Current Approaches to Miscommunication in Spoken Dialogue Systems

As mentioned earlier, current spoken dialogue systems deal predominantly with miscommunication at the level of speech recognition errors. This section reviews current approaches to the detection and handling of speech recognition errors in spoken dialogue systems, followed by a brief look at multimodal dialogue systems that include other modalities as well as speech.

The following is a typical example of the speech recognition errors that might occur in a spoken language dialogue between a human user and an automated system (the system's interpretation of the user's input is in brackets and upper case):

1 S: What city are you leaving from?
2 U: Urbana Champaign [OKAY IN THAT SAME PAY]
3 S: Sorry I'm not sure I understood what you said. What city are you leaving from?
4 U: Urbana Champaign [FOR MINUTE SINCE HEY]
5 S: I'm still having trouble understanding you. To go on, I need you to answer the following question. What city are you leaving from?
6 U: Chicago [CHICAGO]
7 S: Traveling from Chicago. Where would you like to go?
8 U: Huntsville [SEOUL]
9 S: Traveling to Seoul. What day did you need to travel?
10 U: No no I'm traveling to Birmingham [THE TRAVELING TO BERLIN P M]
11 S: Traveling in the afternoon. Okay what day would you be departing chicago?
12 U: Uh the tenth of august [AT THE TENTH OF AUGUST]
13 S: A flight tomorrow I have a Korean Airlines flight departing Chicago at 1:40PM, arrives Seoul at 5PM the next day. Do you want to take that?
14 U: My destination is Birmingham [FLIGHT DESTINATION MR WEEK-END]
15 S: Traveling on Saturday, August 12th — I have a flight departing Chicago at 1:40PM arrives Seoul at

(Source: Dan Bohus)

As can be seen from this example, the system (S) is unable to correctly recognise the user's (U) spoken input for the departure and arrival cities and also makes errors with the time and date.

There are several basic strategies that the system might adopt when faced with a potential error. On the one hand, it can simply accept the utterance and continue with the dialogue, as happens in this example at lines 8–9. However, with this strategy there is the danger of false acceptance, where the system makes an erroneous interpretation of what the user actually said. Such errors are difficult to rectify and in this example the system persists with 'Seoul' as the destination city despite the user's attempt to correct this in line 10.

A second strategy is to reject the utterance and ask the user to repeat or rephrase. An example occurs in lines 1–5 where the system is unable to derive a suitable interpretation of the user's input. In this case the user gives up on the original source city.

Finally, the system can accept the utterance but attempt to verify it, either explicitly or implicitly. An explicit verification would take a form such as 'did you say Chicago?' where the user has to answer either 'yes' or 'no' to verify the system's interpretation. In this example the system adopts an 'implicit

verification' strategy, incorporating its interpretation of the user's input in its next question (see lines 7, 9, 11). This strategy is less costly in terms of transaction time and user frustration compared with the 'explicit confirmation' strategy but has the disadvantage of making error recovery more difficult, as this example illustrates throughout.

Error handling is an active research topic within spoken dialogue technology (see, e.g. a recent set of articles in Carlson et al., 2005; see also Bohus and Rudnicky, 2007; Skantze, 2005). A number of stages can be distinguished:

- Error prevention — this occurs at the design stage and is concerned with the design of appropriate prompts, recognition grammars, dialogue flows, and verification strategies (Cohen et al., 2004).
- Error detection — in this approach the dialogue is monitored for cues that some error has occurred.
- Error prediction — this involves predicting potential problems based on evidence from the dialogue so far.
- Error recovery — this stage is concerned with strategies for putting the dialogue back on track.

See also Turunen and Hakulinen (2001) for a more fine-grained set of stages.

4.1 Error Detection

Error detection, which involves looking for cues that an error has occurred, can be divided into early error detection and late error detection.

With early error detection the system detects that there is something wrong in the user's current utterance and takes immediate steps to correct this potential error. This type of error detection is usually based on acoustic confidence scores of the user's input and an error is suspected if the confidence score falls below a predetermined threshold (Komatani and Kawahara, 2000; Hazen et al., 2000). However, this measure is not entirely reliable as there is no one-to-one correspondence between low confidence scores and errors, nor between high confidence scores and correct recognition (Bouwman et al., 1999). Other approaches include:

- The use of secondary properties of the decoding process, such as language models backoff patterns and information in the word-hypothesis lattice (Wessel et al., 1998; Evermann and Woodland, 2000)
- Comparison of prosodic cues in correctly and incorrectly recognised utterances to predict speech recognition performance (Litman et al., 2000)

- The use of combinations of cues such as parsing confidence, degree of context shift, and salience to reliably predict errors (Walker et al., 2000b)
- The use of concept confidence scores derived from speech recognition confidence scores using a discourse model of what has been said in the dialogue and what entities have been referred to (Skantze, 2005)

4.2 Late Error Detection

Late error detection has been investigated within the context of the user's response to the system's attempt to verify the user's previous utterance. More specifically, a number of cues have been identified that indicate whether the user is satisfied with the system's verification (positive or 'go on' cues) or whether the user rejects the verification (negative or 'go back' cues). Examples of positive cues are short turns, unmarked word order, and confirmations, while negative cues include longer turns, marked word order, disconfirmations and corrections. Krahmer et al. (2001) investigated the predictive capacity of these cues, finding that the best results (almost 97% accuracy) were obtained when all the cues were used in combination. For detection of errors at later points in the dialogue, see Skantze (2007).

4.3 Error Prediction

Error prediction is concerned with predicting potential problems in a dialogue based on features monitored in the dialogue so far. Litman et al. (1999) applied machine learning techniques to develop a classifier that could predict problematic dialogues, using features such as confidence scores, dialogue efficiency and quality. The features were collected over the complete dialogue so that it was not possible for the rules learned by the classifier to influence the course of the ongoing dialogue. However, in another study Walker et al. (2000a) showed how information gathered early in the dialogue could be used to modify the current dialogue strategy, with accuracy rates ranging from 72% to 87% depending on how much of the dialogue had been seen so far.

4.4 Error Recovery

Error recovery can involve a number of different strategies, including asking the user to repeat or rephrase a problematic utterance or using different verification techniques based on the acoustic confidence scores (Sturm et al., 1999). Krahmer et al. (2001) proposed the use of information available in the detection of the error, such as the negative cues in late error detection, to enable the system to construct a more useful follow-up question. Another approach involves the system automatically adapting its dialogue strategies on encountering problems within the dialogue. For example, Litman and Pan (2002) developed a

system that reverted to a more conservative dialogue initiative strategy if the dialogue model indicated that misrecognitions had exceeded a given threshold.

Bohus and Rudnicky (2005) examined the performance of ten different error recovery strategies following non-understanding by the system of the user's utterance, finding that the three most successful strategies were:

- MoveOn, where the system advances the task by moving on to a different question
- Help, where a help message is provided that explains the current state of the dialogue and indicates what the user can say
- TerseYouCanSay, where the system states briefly what the user can say at the current point in the dialogue

These findings are interesting as they suggest that it may not always be necessary to invest considerable effort in error detection and repair, as moving on to a different plan or task might remedy the problem. Similar findings were reported in a Wizard-of-Oz study of error handling strategies (Skantze, 2005).

4.5 Errors in Multimodal Dialogue

Multimodal dialogue systems that offer more than one input mode can provide error handling capabilities in which speech recognition error rates can be improved by using information from the current context, including input from other modes. One example of this from the QuickSet system involves a user interacting with a dynamic map interface and saying 'zoom out' while making a checkmark on the map. The words 'zoom out' were ranked fourth on the speech recogniser's n-best list, but the information from the checkmark gesture enabled the system to override the other phrases as less appropriate in the current context and to rank the phrase 'zoom out' as first on the combined multimodal n-best list (Oviatt, 1999). Similar findings have been reported in other studies comparing the recognition rates of a unimodal speech-based interface with a multimodal interface (Oviatt, 1999; Oviatt, 2000).

Using multimodal information in this way is not without its problems, however, especially with respect to how the information from the different modes is to be interpreted and integrated (Oviatt, 2000). In some cases the information is closely synchronised temporally, as in speech and lip movements. In other cases there is a less direct coupling and so each item of information has to be interpreted semantically to determine whether they are related. Different methods have been used for modality integration, including the unification of typed feature structures from the outputs of the different modalities (Johnston et al., 1997, 2002). Little work has been reported as yet on how to resolve the issue of conflicting information from different modalities.

5. Handling Miscommunication: Why Bother and What to Do

Having reviewed current approaches to miscommunication in spoken dialogue systems as well as related research in miscommunication in various other disciplines, it is appropriate to return to the questions raised earlier in terms of considerations and recommendations for future work:

1. Why bother with errors — what are the costs and benefits?

2. If we decide to deal with errors, how should we go about it?

This section examines some recent research on the issues involved in handling miscommunication and describes some of the approaches adopted. It is argued that decisions as to whether to deal with errors and how to deal with them are complex and predominantly context dependent.

5.1 Dealing with Errors: Why Bother?

Decisions regarding the need to handle miscommunication revolve around the assessment of costs and benefits. Not dealing with potential miscommunication increases the risk that the system might have misinterpreted the user's utterance. In cases where this misinterpretation is not detected by the user — for example, following an attempt by the system to verify what it believes the user said — the error may go undetected. Errors that are detected later in the dialogue are generally more difficult to rectify. Furthermore, it is often the case that errors can have cumulative effects, with one undetected error leading to more errors. Hence there are significant risks in not addressing potential miscommunication. Indeed, as Bohus and Rudnicky (2007) have shown, non-understanding errors have a negative effect on task success and this effect increases rapidly as the errors become more frequent.

Against this, however, there are the costs involved in error handling. Methods such as explicit verification of everything the user says increase transaction time and can lead to user dissatisfaction. As well as this, many of the methods for detecting and handling errors, particularly those described in the section on natural language interfaces, are complex and computationally expensive. For these reasons, researchers are beginning to examine error handling in terms of a costs/benefits analysis.

These issues have been explored in the Conversational Architectures Project in which conversation is modelled as a process of decision making under uncertainty (Paek and Horvitz, 1999; Paek and Horvitz, 2000). In this work the system reasons about the sources of miscommunication to determine the level at which the miscommunication has occurred. For example, the miscommunication could be at the channel level, as in the listener not attending to the speaker's utterance, or at the attention level, which involves the semantic content of

the communication. In order to determine whether to resolve the miscommunication and which actions to take, the system applies a costs/benefits analysis. For example, the costs of repairing a problem at the channel level (such as listener inattention) would have low costs compared with the costs involved in continuing the dialogue without ensuring mutual attention.

5.2 Dealing with Errors: What to Do?

A more radical approach to error handling is to accept that miscommunication is inevitable so that, rather than attempting to detect errors, which often involves some of the complex mechanisms described earlier, miscommunication should be handled in terms of the theory of grounding. In the traditional view of communication involving a sender, receiver and transmission channel, miscommunication is viewed as 'noise' that can be detected and corrected. An alternative view, which has been referred to as 'radical intersubjectivity', sees each participant in a dialogue as having a unique perspective (Reddy, 1979; Sperber and Wilson, 2004; Grice, 1975). Within this view people interpret the messages of others in terms of their own perspective. Communication is possible because there is sufficient common ground between people's perspectives, but at the same time miscommunication is inevitable because of these differences in perspective. Given this view of communication, dialogue is seen not as a transmission of information between a speaker and a listener but as a collaborative activity in which the aim is to achieve mutual understanding. This process is referred to as 'grounding' (Clark, 1996; Traum, 1999).

The Conversational Architectures Project was concerned with the development of a computational model of grounding in which models of uncertainty and decision theory were used to identify the most useful actions to take when resolving miscommunication. A different approach was taken in the Queens Communicator system, where decisions on what actions to take involved the use of information within the system's dialogue model (McTear et al., 2005). One of the decisions to be taken by the system was the type of confirmation strategy to be used — for example, explicit or implicit. This decision was based on factors such as the state of the information to be confirmed and its degree of confirmedness. Information could be 'new for the system', 'inferred by the system', 'repeated by the user', 'modified by the user', 'negated by the user', and so on, while the degree of confirmedness varied according to whether the user had just repeated the information, modified it, or negated it (see Skantze, 2005 for a similar approach). Using a set of rules the system took this information and determined its strategy in relation to how it would confirm the item of information. The following is an example:

> 1 User: I'd like to book a four-star hotel in Belfast from the 15th of December to the 20th of December.
> 2 System: OK, that's a four star hotel in Belfast from 15th December to 20th

December.
3 User: Could you make that a three-star?
4 System: OK, got it. So that's a three star. Is that correct?

The information provided in utterance 1 is 'new for the system' so the strategy applied is implicit confirmation. However one of the values is 'modified by the user' in utterance 3, thus affecting the degree of confirmedness, resulting in a more constraining confirmation type (repair_confirm) in utterance 4.

Some interesting research in social psychology has examined the issue of how to address miscommunication (Nass and Brave, 2004). In this work three different strategies are identified:

- System takes responsibility
- Blame user
- Find a scapegoat

The strategy of 'system takes responsibility' is often used in spoken dialogue systems — for example, where the system is unable to interpret the user's input ('Sorry, I don't understand'). Findings from studies reported by Nass and Brave (2004) suggest that systems using this strategy were perceived as modest and likeable, but unfortunately also as unintelligent and incompetent.

The strategy of 'blame user' is also used in some dialogue systems where it is obvious where the problem lies (for example, 'You must speak more clearly'). This strategy was perceived as unpleasant but also as competent and reliable, presumably because the system seemed to be in control of the interaction and was able to detect sources of difficulty.

Finally, the strategy 'find a scapegoat' ('There seems to be a problem with the telephone line'), where the blame is deflected away from both the system and the user, resulted in the system being perceived as more likeable.

In their discussions regarding the relative advantages and disadvantages of the use of these strategies in human–machine dialogue Nass and Brave (2004) indicate that one of the major risks of engaging in communication repair is that it can foreground communicative inadequacy. On this basis they recommend the following strategies:

- Take responsibility when unavoidable
- Ignore errors whenever possible
- Shift the blame

Studies such as these are particularly relevant for spoken dialogue research as user perceptions are a major factor in the acceptability of a system by users, influencing whether users will be willing to engage with the system or not.

5.3 Context of Interaction

The type of interaction and its context have a bearing on decisions regarding the estimation of the costs and benefits of handling miscommunication. Most current spoken dialogue systems support transactional communication such as the retrieval of information or the provision of services. The accuracy of the information that is conveyed in these transactions is critical for the success of the transaction — such as a user's PIN, cash transfers, flight details, or hotel reservations — and thus it is essential that the information is confirmed (grounded) for the transaction to be concluded successfully. The costs of confirming information would thus outweigh the costs of increased transaction times. The same would apply to an even greater extent for safety critical interactions involving information that is high-risk, as in flight control, military operations, and medical applications. In these cases the grounding of the referential content of the information is essential.

Different criteria apply in the case of systems where the focus is on interpersonal communication, as in conversational companions for the elderly or talking toys for children. Here referential accuracy is less important compared with affective aspects of communication, such as the use of feedback to indicate agreement or solidarity, or the ability to convey and detect emotions. Embodied conversational agents and other applications, such as car navigation systems, that convey a persona are examples where, in addition to the accuracy of the referential content, interpersonal aspects of the communication are an important contributor to the acceptability of the system by users. For example: while a female voice is perceived as conveying a more sensitive and emotionally responsive personality and would be better suited in an application involving complaint handling, a male voice is more suited to contexts requiring the expression of authority — such as stating a policy of 'no refunds' — since females are generally evaluated negatively when they assume a position of dominance (Nass and Brave, 2004). Indeed, in a commercial context, BMW in Germany had to recall its car navigation system that had a female voice as most BMW drivers, who are male, did not like taking directions from females (Nass and Brave, 2004).

Finally there are systems where miscommunication could be incorporated as an integral part of the human–computer dialogue, as in games where the system tries to intentionally mislead the player or engage in a battle of wits in which the challenge for the player is to try to communicate with a less than compliant partner. A similar approach could be used to enliven instructional materials for language learning and other types of edutainment.

6. Conclusions

Miscommunication within the discipline of spoken dialogue technology has been viewed essentially in terms of error handling with particular application to the detection and repair of speech recognition errors. A number of sophisticated techniques from artificial intelligence have been applied to error handling in spoken dialogue with computers, either in terms of detecting and correcting the errors or in terms of grounding, where communication is seen as a collaborative activity.

Miscommunication occurs in a variety of contexts and has been investigated extensively by a wide range of disciplines. Discussions of miscommunication can be traced back to the earliest writings in civilisation. As spoken dialogue systems develop and extend their range of application beyond task-based transactions to interpersonal interactions, other aspects of miscommunication, such as the misinterpretation of emotion and intent, will need to be taken into account and it is here that the contributions of various disciplines, such as social psychology, conversation analysis, and sociolinguistics will become more relevant.

References

Allen, J. F., Ferguson, G., and Stent, A. (2001). An Architecture for More Realistic Conversational Systems. In *Proceedings of Intelligent User Interfaces (IUI)*, pages 1–8, Santa Fe.

Bohus, D. and Rudnicky, A. (2005). A Principled Approach for Rejection Threshold Optimization in Spoken Dialog Systems. In *Proceedings of 9th European Conference on Speech Communication and Technology (INTERSPEECH)*, pages 2781–2784, Lisbon.

Bohus, D. and Rudnicky, A. (2007). Sorry, I Didn't Catch That! An Investigation of Non-Understanding Errors and Recovery Strategies. In Dybkjær, L. and Minker, W., editors, *Recent Trends in Discourse and Dialogue*. Springer. (This volume).

Bouwman, A., Sturm, J., and Boves, L. (1999). Incorporating Confidence Measures in the Dutch Train Timetable Information System Developed in the ARISE Project. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 493–496, Phoenix.

Button, G., Coulter, J., Lee, J. R. E., and Sharrock, W. (1995). *Computers, Mind and Conduct*. Polity Press, Oxford.

Carberry, S. (1987). Inferred Knowledge and Understanding Pragmatically Ill-formed Queries. In Reilly, R., editor, *Communication Failure in Dialogue and Discourse: Detection and Repair Processes*, pages 187–200. North Holland, Amsterdam.

Carlson, R., Hirschberg, J., and Swerts, M. (2005). Special Issues on Error Handling in Spoken Dialogue Systems. *Speech Communication*, 45: 207–359.

Clark, H. H. (1996). *Using Language*. Cambridge University Press, Cambridge.

Coates, J. (1998). *Language and Gender*. Blackwell, Oxford.

Cohen, M., Giangola, J., and Balogh, J. (2004). *Voice User Interface Design*. Addison-Wesley, Boston.

Coupland, N., Giles, H., and Wiemann, J. (1991). *Miscommunication and Problematic Talk*. Sage Publications, London.

Dictionary (2004). *The American Heritage Dictionary of the English Language*. Houghton Mifflin Company, Boston.

Evermann, G. and Woodland, P. C. (2000). Large Vocabulary Decoding and Confidence Estimation Using Word Posterior Probabilities. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2366–2369, Istanbul.

Grice, H. P. (1975). Logic and Conversation. Syntax and Semantics. In Cole, P. and Morgan, J. L., editors, *Speech Acts*, volume 3, pages 41–58. Academic, New York.

Gumperz, J. (1978). The Conversational Analysis of Interethnic Communication. In Ross, E. L., editor, *Interethnic Communication*, pages 13–31. University of Georgia Press, Athens.

Hazen, T., Burianek, T., Polifroni, J., and Seneff, S. (2000). Integrating Recognition Confidence Scoring with Language Understanding and Dialogue Modeling. In *Proceedings of 6th International Conference on Spoken Language Processing (ICSLP)*, pages 1042–1045, Beijing.

Johnston, M., Bangalole, S., Vasireddy, G., Stent, A., Eblen, P., Walker, M., Whittaker, S., and Maloor, P. (2002). MATCH: An Architecture for Multimodal Dialogue Systems. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 376–383, Philadelphia.

Johnston, M., Cohen, P., McGee, D., Oviatt, S., Pittman, J., and Smith, I. (1997). Unification-based Multimodal Integration. In *Proceedings of the International Conference on Computational Linguistics and the 35th Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 281–288, Madrid.

Joshi, A., Webber, B., and Weischedel, R. (1987). Some Aspects of Default Reasoning in Interactive Discourse. In Reilly, R., editor, *Communication Failure in Dialogue and Discourse: Detection and Repair Processes*, pages 213–219. North Holland, Amsterdam.

Komatani, K. and Kawahara, T. (2000). Flexible Mixed-Initiative Dialogue Management Using Concept-Level Confidence Measures of Speech

Recognizer Output. In *Proceedings of 18th International Conference on Computational Linguistics (COLING)*, pages 467–473, Saarbrücken.

Krahmer, E., Swerts, M., Theune, M., and Weegels, M. (2001). Error Detection in Spoken Human-Machine Interaction. *International Journal of Speech Technology*, 4(1):19–29.

Litman, D. J., Hirschberg, J., and Swerts, M. (2000). Predicting Automatic Speech Recognition Performance using Prosodic Cues. In *Proceedings of 1st Meeting of the North American chapter of the Association for Computational Linguistics (NAACL)*, pages 218–225, Seattle.

Litman, D. J. and Pan, S. (2002). Designing and Evaluating an Adaptive Spoken Dialogue System. *User Modeling and User-Adapted Interaction*, 12:111–137.

Litman, D. J., Walker, M. A., and Kearns, M. S. (1999). Automatic Detection of Poor Speech Recognition at the Dialogue Level. In *Proceedings 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 218–225, University of Maryland, Maryland.

Maltz, D. and Borker, R. (1982). A Cultural Approach to Male-Female Miscommunication. In Gumperz, J., editor, *Language and Social Identity*, pages 196–216. Cambridge University Press, Cambridge.

McCoy, K. (1987). Generating Responses to Property Misconceptions using Perspective. In Reilly, R., editor, *Communication Failure in Dialogue and Discourse: Detection and Repair Processes*, pages 149–160. North Holland, Amsterdam.

McRoy, S. (1998). Preface-Detecting, Repairing and Preventing Human-machine Miscommunication. *International Journal of Human-Computer Studies*, 48:547–552.

McTear, M. (2004). *Spoken Dialogue Technology: Toward the Conversational User Interface*. Springer, London.

McTear, M., O'Neill, I., Hanna, P., and Liu, X. (2005). Handling Errors and Determining Confirmation Strategies - An Object-Based Approach. *Speech Communication*, 45(3):249–269.

Nass, C. and Brave, S. (2004). *Wired for Speech: How Voice Activates and Advances the Human-Computer Relationship*. MIT Press, Cambridge.

Oviatt, S. (1999). Mutual Disambiguation of Recognition Errors in a Multimodal Architecture. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, pages 576–583, ACM Press, New York.

Oviatt, S. (2000). Taming Recognition Errors with a Multimodal Interface. *Communications of the ACM*, 43:45–51.

Paek, T. and Horvitz, E. (1999). An Uncertainty, Utility, and Misunderstanding. In *AAAI Fall Symposium on Psychological Models of Communication*, pages 85–92, North Falmouth.

Paek, T. and Horvitz, E. (2000). Conversation as Action under Uncertainty. In *Proceedings of 6th Conference on Uncertainty in Artificial Intelligence*, pages 455–464, Morgan Kaufmann Publishers, San Francisco.

Reddy, M. (1979). The Conduit Metaphor: a Case of Frame Conflict in one Language about another Language. In Ortony, A., editor, *Metaphor and Thought*, pages 284–324. Cambridge University Press, Cambridge.

Reilly, R. (1987). *Communication Failure in Dialogue and Discourse: Detection and Repair Processes*. North Holland, Amsterdam.

Schegloff, E., Jefferson, G., and Sacks, H. (1977). The Preference for Self-Correction in the Organisation of Repair in Conversation. *Language*, 53:361–382.

Skantze, G. (2005). Exploring Human Error Recovery Strategies: Implications for Spoken Dialogue Systems. *Speech Communication*, 45:325–341.

Skantze, G. (2007). Galatea: A Discourse Modeller Supporting Concept-Level Error Handling in Spoken Dialogue Systems. In Dybkjær, L. and Minker, W., editors, *Recent Trends in Discourse and Dialogue*. Springer. (This volume).

Sperber, D. and Wilson, D. (2004). *Relevance: Communication and Cognition*. Blackwell, Oxford.

Sturm, J., den Os, E., and Boves, L. (1999). Dialogue Management in the Dutch ARISE Train Timetable Information System. In *Proceedings of 6th International Conference on Speech Communication and Technolology (EUROSPEECH)*, pages 1419–1422, Budapest.

Tannen, D. (2001). *You just don't Understand: Women and Men in Conversation*. Quill.

Traum, D. (1999). Computational Models of Grounding in Collaborative Systems. In *Working Notes of AAAI Fall Symposium on Psychological Models of Communication*, pages 124–131, North Falmouth.

Traum, D. and Dillenbourg, P. (1996). Miscommunication in Multi-Modal Collaboration. In *Working notes of the AAAI Workshop on Detecting, Repairing, And Preventing Human-Machine Miscommunication*, pages 37–46, Portland.

Turunen, M. and Hakulinen, J. (2001). Agent-Based Error Handling in Spoken Dialogue Systems. In *Proceedings of 7th International Conference on Speech Communication and Technology (EUROSPEECH)*, pages 2189–2192, Aalborg.

Walker, M. A., Langkilde, I., Wright, J., Gorin, A., and Litman, D. J. (2000a). Learning to Predict Problematic Situations in a Spoken Dialogue System: Experiments with How May I Help You? In *Proceedings of 1st Meeting of the North American chapter of the Association for Computational Linguistics (NAACL)*, pages 210–217, Seattle.

Walker, M. A., Wright, J., and Langkilde, I. (2000b). Using Natural Language Processing and Discourse Features to Identify Understanding Errors in a Spoken Dialogue System. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1111–1118, Stanford University, California.

Wessel, F., Macherey, K., and Schluter, R. (1998). Using Word Probabilities as Confidence Measures. In *Proceedings of International Conference on Acoustic, Speech, and Signal Processing (ICASSP)*, pages 225–228, Seattle.

Chapter 6

SORRY, I DIDN'T CATCH THAT!

An Investigation of Non-Understanding Errors and Recovery Strategies

Dan Bohus and Alexander I. Rudnicky
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
dbohus@cs.cmu.edu, air@cs.cmu.edu

Abstract We present results from an extensive empirical analysis of non-understanding errors and ten non-understanding recovery strategies, based on a corpus of dialogues collected with a spoken dialogue system that handles conference room reservations. More specifically, the issues we investigate are: what are the main sources of non-understandings? What is the impact of these errors on global performance? How do various strategies for recovery from non-understandings compare to each other? What are the relationships between these strategies and subsequent user response types, and which response types are more likely to lead to successful recovery? Can dialogue performance be improved by using a smarter policy for engaging the non-understanding recovery strategies? If so, can we learn such a policy from data? Whenever available, we compare and contrast our results with other studies in the literature. Finally, we summarise the lessons learned and present our plans for future work inspired by this analysis.

Keywords: Spoken dialogue systems; Error handling; Error sources; Non-understandings; Error recovery strategies; Error recovery policies.

1. Introduction

One of the most important challenges facing spoken language interfaces today is their brittleness when faced with understanding errors. The problem is present across all domains and interaction types, and arises primarily from the inherent unreliability of the speech recognition process. The recognition difficulties are further exacerbated by the conditions under which these systems typically operate: spontaneous speech, large vocabularies and user populations,

L. Dybkjær and W. Minker (eds.), Recent Trends in Discourse and Dialogue, 123–154.

and large variability in input line quality. In these settings, average word error rates of 20–30% (and up to 50% for non-native speakers) are quite common. Unless mediated by better error awareness and robust recovery mechanisms, these errors exert a strong negative influence on the overall performance of spoken dialogue systems (Sanders et al., 2002; Walker et al., 1998), and severely limit the naturalness of the interaction and the complexity of the tasks that can be addressed.

Left unchecked, speech recognition errors can lead to two types of *understanding errors* in a spoken dialogue system: *misunderstandings* and *non-understandings*. In a *misunderstanding*, the system obtains an incorrect interpretation of the user's turn. In contrast, in a *non-understanding*, the system fails to obtain any interpretation of the input.

In this paper, we focus our attention on non-understandings. If for misunderstandings detection is a key problem (San-Segundo et al., 2000; Litman et al., 2000; Carpenter et al., 2001), and the set of recovery strategies is limited and fairly well understood, e.g. explicit and implicit confirmation (Krahmer et al., 1999), for non-understandings the situation is almost the opposite. By definition, systems know when a non-understanding has happened. However, a mechanism for diagnosing the source of the non-understanding is largely missing. Moreover, the number of potential recovery strategies is significantly larger (see Table 1) and the relative trade-offs between them are less well understood. This further increases the difficulty of selecting the right recovery strategy at runtime. Most systems use a small number of non-understanding recovery strategies in conjunction with uninformed, simple heuristic rules for engaging them. For instance, a system might apologise and repeat its question on the first non-understanding, provide more help on the second non-understanding, and transfer the user to a human operator if a third consecutive non-understanding occurred.

As a first step towards better error handling for non-understandings, we have conducted an empirical study of these errors and of ten recovery strategies. More specifically, the questions under investigation were:

- What are the main sources of non-understandings (and what are their relative frequencies)?
- How large is the impact of non-understandings on global dialogue performance?
- How do various strategies for recovering from non-understandings compare to each other?
- What are the relationships between each strategy and subsequent user behaviours, and which behaviours are more likely to lead to successful recovery?

Table 1. Ten strategies for recovering from non-understandings.

	S: Would you like a small room or a large one? U: [Non-understanding]
1	**AskRepeat(AREP)** [system asks the user to repeat the utterance] S: Can you please repeat that?
2	**AskRephrase(ARPH)** [system asks the user to rephrase the utterance] S: Can you please try to rephrase that?
3	**Reprompt(RP)** [system repeats the previous prompt] S: Would you like a small room or a large one?
4	**DetailedReprompt(DRP)** [system repeats a more verbose version of the previous prompt] S: I'm sorry, I don't think I understood you correctly. Right now I'm trying to find out if you would prefer a small room or a large one?
5	**Notify(NTFY)** [system notifies the user that a non-understanding has occured] S: Sorry, I'm not sure I understood what you said ...
6	**Yield(YLD)** [system remains silent, and thus implicitly notifies the user that a non-understanding has occurred] S: ...
7	**MoveOn(MOVE)** [system advances the task by moving on to a different question] S: Sorry, I didn't catch that. One choice would be Wean Hall 7220. This room can accommodate 20 people and has a whiteboard and a projector. Would you like a reservation for this room?
8	**YouCanSay(YCS)** [system tells the user what he or she can say at this point in the dialogue] S: Sorry, I didn't catch that. Right now I'm trying to find out if you would prefer a small room or a large one. You can say "I want a small room" or "I want a large room". If the size of the room doesn't matter to you, just say "I don't care".
9	**TerseYouCanSay(TYCS)** [a terser version of the YouCanSay strategy] S: Sorry, I didn't catch that. You can say "I want a small room" or "I want a large room". If the size of the room doesn't matter to you, just say "I don't care".
10	**FullHelp(HELP)** [system provides a longer help message that includes an explanation of the current state of the dialogue, as well as what the user can say at this point in the dialogue] S: I'm sorry, I don't think I understood you correctly. So far I have found 5 conference rooms available matching your constraints. Right now I'm trying to find out if you would prefer a small room or a large one. You can say "I want a small room" or "I want a large room". If the size of the room doesn't matter to you, just say "I don't care".

- Can global dialogue performance be improved by using a smarter policy for engaging the non-understanding recovery strategies?
- If yes, can we learn a better policy from data?

We begin by describing the data collection experiment which provided the corpus of dialogues used in this investigation. Then, in Sections 2 through 8, we address in turn each of the questions raised above. Whenever possible, we compare our findings to other results previously reported in the literature, in an effort to shed more light on the generalisability of these results across different domains. Finally, in Section 9 we summarise the lessons we learned from this investigation and the ideas it inspired for future work.

2. Experiment and Corpus

2.1 Data Collection Experiment

2.1.1 System. The data was collected through a user study in which 46 participants, mostly undergraduate and staff personnel on campus interacted with RoomLine (RoomLine, 2003), a spoken dialogue system for making conference room reservations. RoomLine is a phone-based mixed-initiative system which has access to live information about the schedules and characteristics (e.g. size, location, A/V equipment) of 13 conference rooms in two buildings on campus. To make a room reservation, the system finds the list of available rooms that satisfy an initial set of user-specified constraints, and engages in a follow-up negotiation dialogue to present this information to the user and identify which room best matches their needs. Sample conversations with the system are available online (RoomLine, 2003).

The system uses two parallel SPHINX-II recognition engines, configured with telephone-based acoustic models and a trigram statistical language model (the dictionary size is 1049). The resulting top hypothesis from each engine is parsed using the Phoenix robust parser (Ward and Isaar, 1994). Subsequently, semantic confidence scores are computed for each hypothesis. The winning hypothesis is forwarded to the RavenClaw-based dialogue manager (Bohus and Rudnicky, 2003). For output, the system uses a template-based language generation module and the Theta synthesiser (Theta, 2004).

The system was equipped with ten different *strategies* for recovering from non-understandings, described and illustrated in Table 1. By *strategy* we denote a simple, single-turn action that the system can take to attempt recovery. A number of these strategies, such as asking the user to repeat or rephrase, reissuing the system prompt or providing various levels of help are often encountered in spoken dialogue systems. Two strategies that we would like to draw the reader's attention upon are *Yield* and *MoveOn*. In the *Yield* strategy, the system remains silent, as if it did not hear the user's response, and hence implicitly signals a communication problem. In the *MoveOn* strategy, the system ignores the problem altogether and tries to advance the task by moving on to a different question. Note that this is possible only at certain points in the dialogue, where an alternative dialogue plan for achieving the same goals is

available. For instance, in the case illustrated in Table 1, the *MoveOn* strategy gives up on trying to find whether the user wants a small or a large room, and starts suggesting rooms one by one. In other cases, the system would try to advance the dialogue by using a more directed question, for instance asking "For which day do you need the room?" instead of "How can I help you?"

2.1.2 Experimental design. The user study was designed as a between-groups experiment, with two conditions: *control* and *wizard*.

Participants in the *control* condition interacted with a version of the RoomLine system which used an uninformed (random) policy to engage the non-understanding recovery strategies: each time a non-understanding happened, the system randomly chose one of the ten available strategies.

Participants in the *wizard* condition interacted with a modified Wizard-of-Oz version of the same system. In this version, each time a non-understanding happened a human wizard decided which one of the ten recovery strategies should be used. In all other aspects, this system was identical with the system used in the *control* condition. The wizard had live access to the user's speech. Several other system state variables were presented to the wizard via a graphical user interface (e.g. recognition result, confidence score, semantic parse). When a non-understanding occurred, the wizard selected which strategy should be used through the GUI, and the decision was communicated back to the system. The wizard had to make this decision during a relatively short time interval (1–2 seconds) in order to maintain the illusion that the users were interacting with an autonomous system. A single wizard, the first author of this paper, was employed throughout the whole experiment. The wizard had very good knowledge of the system's functionality and of the domain.

The experimental design described above satisfies two needs. On one hand, we wanted to be able to comparatively evaluate the ten recovery strategies, when engaged in an uninformed fashion. This analysis can be performed based on data collected in the *control* condition, where the system randomly chooses which strategy to use. The results are discussed in detail in Sections 5 and 6. At the same time, we wanted to verify whether or not a better policy for engaging the ten strategies (implemented in this case by the human wizard) can significantly improve performance. The results of this comparative analysis are presented in Section 7.

At this point we would like to briefly comment on the decision to give the wizard full access to the live user speech. This puts the wizard in an apparently privileged position when compared to a system that would have to make the same recovery decisions (e.g. the system does not accurately know what the user says, especially during non-understandings). However, recall that our goal is only to show that a better recovery policy exists, and ***not*** to prove that this particular policy can be learned or implemented by the system. Without access

to the user's speech, the decision making task might have been too difficult for the wizard, especially given the response-time constraints. In this case, a negative result, i.e. the lack of detectable differences in the performance of the two policies, would not be very informative. On the other hand, a negative result obtained when the wizard had full access to the user's speech would cast more serious doubts with respect to the existence of a better non-understanding recovery policy.

2.1.3 Participants. Forty-six subjects, mostly undergraduate students and staff personnel on campus, participated in the data collection experiment. The participants had only marginal prior experience with spoken language interfaces (some of them had previously interacted with phone-based customer-service interactive systems). We randomly assigned the participants into two groups corresponding to the *control* and *wizard* conditions. At the same time, a balance was maintained between groups in terms of the participants' gender and whether or not their first language was North-American English.

2.1.4 Tasks and experimental procedure. Each participant attempted a maximum of 10 scenario-based interactions with the system, within a set time period of 40 minutes. The same 10 scenarios were presented in the same order to all participants. The scenarios were designed to cover all the important aspects of the system's functionality and had different degrees of difficulty. To avoid language entrainment, the scenarios were presented graphically. Descriptions of the 10 scenarios as well as a concrete example of the graphical representation are available online (Bohus, 2005). In order to motivate the users, we compensated them according to the number of scenarios they completed successfully.

At the end of the experiment, the participants filled in a SASSI questionnaire (Hone and Graham, 2000) containing 35 questions grouped in 6 factors: response accuracy, likeability, cognitive demand, annoyance, habitability, and speed. Additionally, participants were asked to describe what they liked most, what they liked least and what would be the first thing they would change in the system.

2.2 Corpus Statistics and Annotations

The corpus of dialogues collected in this experiment (including both the *control* and *wizard* conditions) contains 449 sessions and 8,278 user turns. Table 2 shows a number of additional descriptive statistics. Since pronounced differences exist on a large number of metrics between native and non-native users, we also present the breakdown of the figures in these two populations.

The user speech data was orthographically transcribed by a human annotator, and subsequently checked by a second annotator. The transcriptions

Table 2. Overall corpus statistics.

	Total	Native	Non-native
# Subjects	46	34	12
# Sessions	449	338	111
# Turns	8278	5783	2495
Word error rate	25.6%	19.6%	39.5%
Concept error rate	35.7%	26.3%	57.6%
% Non-understandings	17.0%	13.4%	25.2%
% Misunderstandings	13.5%	9.8%	22.5%
Task success rate	75.1%	85.2%	44.1%

include annotations for various human and non-human noises in the audio signal. Based on these transcriptions, a number of additional annotations were created. At the turn level, we manually labelled:

- *Concept transfer* and *misunderstandings*: each user turn was annotated with the number of concepts that were correctly and incorrectly transferred from the user to the system; each turn with at least one incorrectly transferred concept was automatically labelled as a *misunderstanding*.
- *Transcript grammaticality*: each user turn was manually annotated as either *in-grammar*, *out-of-grammar*, *out-of-application-scope* or *out-of-domain* (for a discussion, see Section 3).
- *User responses to non-understandings*: the user response types following non-understandings were labelled using a tagging scheme first introduced by Shin et al. (2002).
- *Corrections*: turns in which the user was attempting to correct a system understanding error were flagged as corrections, as in (Swerts et al., 2000).

At the session level, we labelled task completion.

3. Sources of Understanding Errors

We now turn our attention to the first question: ***what are the main sources of non-understandings, and what are their relative frequencies?***

While the main focus of this chapter is on non-understandings, the analysis we present in this section covers sources of understanding errors in general, i.e. both misunderstandings and non-understandings. To avoid potential biases introduced by the wizard's recovery policy, the analysis was conducted using only data from the *control* condition, where the recovery strategies were engaged in an uninformed fashion.

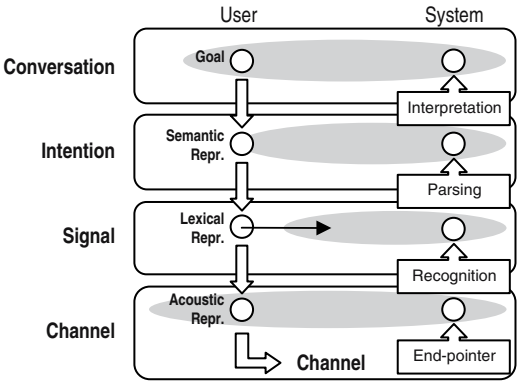


Figure 1. Grounding in communication.

We used Clark's grounding model for human–human communication (Clark, 1996) as the starting point for our error source analysis. The model is illustrated in Figure 1. In this model, participants in a conversation coordinate on four different levels to achieve mutual understanding: channel, signal, intention and conversation. While Clark's original model addressed issues in human–human communication, the same layered coordination scheme has been adopted to model grounding in human–machine interaction (Paek and Horvitz, 2000; Paek, 2003). In this context, we can also map the flow of information from the user to the system on the four grounding levels. At the conversation level, the user has a high-level goal, which subsequently acquires a corresponding semantic, lexical and eventually an acoustic representation in the lower levels. The acoustic signal then passes through a noisy channel, and arrives at the system side. Here, a series of chained components (speech recognition, language understanding, and discourse interpretation) are used to progressively reconstruct the user's higher level goal from the incoming acoustic signal.

Understanding errors typically occur due to mismatches at different levels between the expressed form of the user's intent and the system's modelling abilities (the system's modelling abilities are represented as grey ovals in Figure 1). For example, at the highest level, the user might not be aware of certain system limitations and might try to formulate a goal which the system cannot handle. In this case it will be impossible for the system to correctly reconstruct the user's goal, and we will have an understanding error. Similarly, at the signal level, mismatches between a user's pronunciation style and the system's acoustic models can lead to speech recognition errors, and ultimately to understanding errors. This view of understanding errors highlights two complementary approaches that can be used to mitigate the mismatches. One is to create models which can provide better coverage, while still maintaining good

performance (enlarge the grey ovals in Figure 1). The other is to steer the user's responses into the space covered by the system's models.

Based on the level at which the mismatch occurs, we identify the following sources of errors:

- *Out-of-application* [conversation level]: The user's utterance falls outside the application's functionality. These errors can be further divided into *out-of-domain* utterances (e.g. the user asks the room-reservation system about the weather), and *out-of-application-scope* utterances, i.e. utterances which express in-domain goals which the system is however not able to handle (e.g. the user asks if a conference room has windows).

- *Out-of-grammar* [intention level]: The user's utterance is within the domain and scope of the application, but outside of the system's semantic grammar (e.g. the user says "erase reservation", which is not in the system's grammar; the system could have handled the request had the user said "cancel reservation" or "delete reservation", which are in the system's grammar).

- *ASR error* [signal level]: The user's utterance is within the application's domain, scope and grammar, but is not recognised correctly due to acoustic or statistical language modelling mismatches (e.g. the user says "Thursday morning" but this is misrecognised as "Friday morning").

- *End-pointer error* [channel level]: The end-pointer is not able to correctly segment the incoming audio signal (e.g. it truncates the utterance or sends an empty utterance into the input line).

Figure 2 illustrates the breakdown of non-understandings and misunderstandings by error source. The majority of errors originate at the signal (i.e. speech recognition) level. At the same time, a large number of non-understandings, and a smaller but still significant number of misunderstandings are caused by *out-of-application* and *out-of-grammar* utterances.

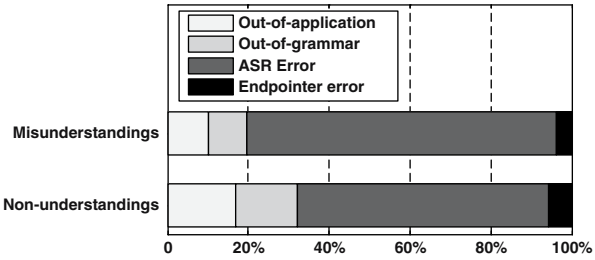


Figure 2. Breakdown of non-understandings and misunderstandings by error source.

The out-of-application errors encountered in our data consist almost entirely of *out-of-application-scope* utterances. These utterances are in-domain, but they refer to inexistent application functionality (the lack of *out-of-domain* utterances is most likely due to the scenario-driven nature of the interactions). A closer inspection of these errors revealed that they subsume about an equal number of requests for inexistent task-level functionality (e.g. "I need a room for Monday or Tuesday" — the system does not handle "or" requests), and requests for inexistent meta functionality, such as "go back!" or various types of corrections (e.g. "You got the wrong day!", "Change the date!", "The time is wrong").

Together with the *out-of-grammar* utterances, the *out-of-application* utterances reflect one facet of an existing mismatch between user and system at the intention and conversation levels. A second interesting facet, revealed through an analysis of the transcripts, is that there are certain aspects of system functionality which are *never* (or *very rarely*) addressed by the users. For instance, although the users were told during the briefing that they can say "Help" to the system at any time, this function was invoked in only 7 of 226 sessions. Other types of help commands like "where are we?", "what can you do?", "what can I say?", "interaction tips", although available at all times were not discovered by the users and therefore were never used. We found similar examples with respect to task-level functionality, for commands like "tell me all the rooms", "I want a smaller / larger room", "I don't care" (about room size), "how big is this room", "tell me about this room", etc. This reflects the fact that, apart from *out-of-grammar* errors, users are also not aware of the full functionality of the application.

The fairly large number of *out-of-application* and *out-of-grammar* utterances suggests that the number of non-understandings can potentially be reduced by better informing the users about the application capabilities and boundaries and by steering them into this space. How exactly this shaping can be performed remains an open research issue (Tomko, 2004). We will return to this issue in our discussion from Section 9.

The majority of non-understandings — 62% (and even more so for misunderstandings — 77%) originate at the speech recognition level. Here, a large number of contributing factors can be identified, but more precise blame assignment is harder to perform. For instance, non-native accents have a significant impact on ASR performance: average WER is 20.7% for natives, versus 42.3% for non-natives. Ambient noises also have a pronounced effect on recognition performance: average WER for noisy utterances is 32.8% $>$ 25.1% for noise-free utterances. Other factors, such as speaking rate, user frustration, and hyper-articulation, have been shown to correlate with recognition accuracy (Choularton, 2005).

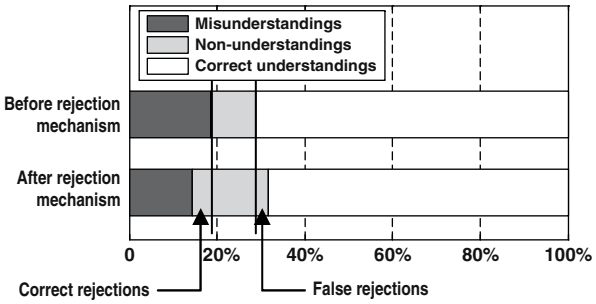


Figure 3. Misunderstandings and non-understandings before and after rejections.

Rejections. The discussion so far has focused on *genuine non-understandings*, i.e. situations in which the system was not able to extract any meaningful information from the user's turn. However, our dialogue manager also uses a rejection mechanism to guard against potential misunderstandings: if the system has obtained an interpretation of the user's input, but the confidence score is below a preset threshold, then the utterance will be rejected by the dialogue manager. These rejected utterances will also appear as non-understandings at the dialogue management level. Figure 3 illustrates the ratios of non-understandings and misunderstandings, as computed before and after the rejection mechanism. After rejections, the total ratio of non-understandings grows by 7.1% absolute from 10.1% to 17.2%. About 40% of the rejections (2.9% of the total number of turns, and 17% of the total number of non-understandings) are false-rejections, i.e. utterances correctly understood but falsely rejected because of a low confidence score. The relatively high false rejection rate contributes significantly to the total number of non-understandings, on par with other sources of errors. The false-rejection rate can be lowered by building better confidence annotators, or by tuning the rejection threshold to the domain. In Bohus and Rudnicky (2005), we describe a data-driven method for optimising the rejection process in light of domain and dialogue-state-specific trade-offs.

4. Impact of Non-Understandings on Dialogue Performance

We now turn our attention to the second question: ***what is the impact of non-understandings on global dialogue performance?*** Again, we only used the data from the *control* condition in our analysis.

To address this question, we constructed a logistic regression model (Myers et al., 2001) which relates the frequency of non-understandings in a dialogue

to the probability of task success. The same approach can be used for studying the impact on other global performance metrics.

$$P(TS = 1) = \frac{1}{1 + e^{-(\alpha + \beta \cdot FNON)}} \tag{6.1}$$

The independent variable is the frequency of non-understandings in a session ($FNON$), and the dependent variable is the binary task success indicator (TS). Each data-point corresponds to an entire dialogue session.

We fitted a model using 205 dialogue sessions. Sessions with less than three turns and sessions with differences between perceived and objective task completion were eliminated. The fitted model increased the average data log-likelihood from the majority baseline of -0.5200 to -0.4306 ($p < 10^{-4}$) in a likelihood-ratio test, indicating that there is indeed an effect of the frequency of non-understandings on task success. Figure 4 illustrates the expected probability of task success, as predicted by the model. The plot shows that when the frequency of non-understandings is between 0% and 10%, the impact on task success is relatively minor. However, as the frequency of non-understandings exceeds 10%, the expected probability of task success starts to drop faster: an increase of the frequency of non-understandings from 10% to 30% reduces the expected chance of success from 90% to 52%.

Apart from non-understandings, misunderstandings represent a second important contributor to breakdowns in interaction. To assess the relative costs of these two types of errors with respect to task success, we extended the model described above to include the frequency of misunderstandings as a second independent variable ($FMIS$). As expected, the new model predicts task success even better: the average log-likelihood of the data was further increased to -0.2795 ($p < 10^{-4}$). The estimated regression coefficients, together with their associated standard errors and p-values are illustrated in Table 3. The resulting

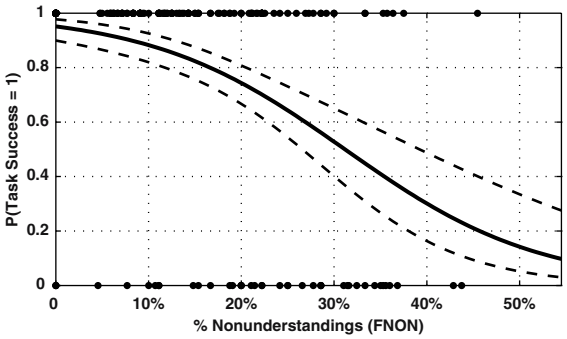


Figure 4. Expected probability of task success (and confidence bounds) at different frequencies of non-understandings.

Table 3. Regression coefficients for a task success model using the frequency of non-understandings and misunderstandings as the independent variables.

	Coefficients	S.E.	p-value
Constant	5.28	0.70	<0.0001
$FNON$	−7.41	2.09	0.0004
$FMIS$	−16.62	2.74	<0.0001

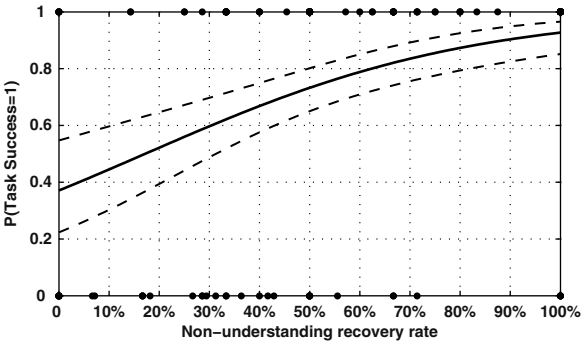


Figure 5. Expected probability of task success (and confidence bounds) at different non-understanding recovery rates.

average cost for misunderstandings (-16.62) is 2.24 times higher than the average cost for non-understandings (-7.41). The result confirms that the rule-of-thumb that "misunderstandings cost twice as much as non-understandings" holds in our domain. While the relative costs of these errors can vary across different domains, and even across different dialogue states within the same system, the proposed regression approach can be used to establish these costs in a principled manner (see also Bohus and Rudnicky, 2005).

Finally, we analysed the impact of *recovery rate* on task success. We say that a strategy has successfully recovered from a non-understanding if the following user turn is correctly understood by the system (i.e. it is not a non-understanding and it is not a misunderstanding). The *average non-understanding recovery rate* is then defined as the ratio of successful recoveries, with respect to the total number of attempts to recover. Again, a significant effect on task success was detected ($p < 10^{-4}$). The dependence is illustrated in Figure 5. As this figure shows, the impact of the recovery rate on performance is greatest when the recovery rate is below 60–70%, and becomes less significant as we pass that limit.

While it is to be expected that non-understandings and the associated recovery rate have an effect on global performance, the analyses that we have performed quantify this effect and provide useful information for focusing

future efforts. In our domain, we expect that further improvements in the non-understanding recovery rate are likely to translate into significant increases in task success, especially for the non-native user population, where 26.3% of the turns are non-understandings and the recovery rate is only 39.3%. In Section 7, we will see that our experiments with different non-understanding recovery policies indeed confirm this conjecture.

5. Performance of Non-Understanding Recovery Strategies

We now turn our attention to the third question: ***how do the ten strategies compare with each other in terms of recovery performance?***

We computed the non-understanding recovery rate (as defined in the previous section) for each of the ten recovery strategies. The analysis is again performed only using the data collected in the *control* condition of our experiment. In this condition, the recovery strategies were engaged in an uninformed (random) fashion, and therefore they were on an equal footing. Figure 6 illustrates the resulting performance of each strategy, and the 95% confidence intervals for these estimates.

An overall analysis of variance for binary response variables (logistic ANOVA) revealed that there are statistically significant differences between the mean recovery rates of the 10 strategies ($p = 0.000035$). Next, we used logistic ANOVAs to perform pairwise strategy comparisons, in an effort to establish which are the best performers. Since performance in general (and hence recovery performance also) varies considerably between native and non-native speakers, we added a binary factor in the ANOVA model to capture whether the user was a native speaker or not. Adding this factor to the ANOVA models considerably improved our power to detect statistically significant differences.

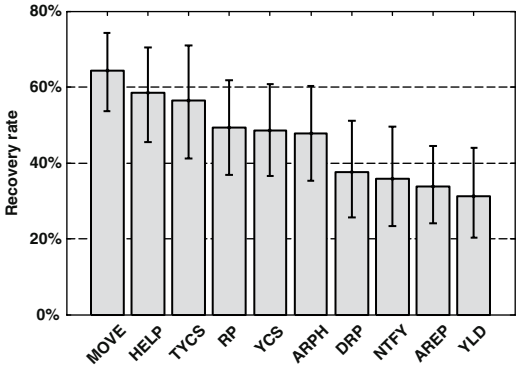


Figure 6. Individual strategy recovery rate.

Table 4. Comparison of non-understanding recovery rates; the cells show the ratio of the non-understanding recovery rate between the strategy in the corresponding row and column; the shading indicates the false-discovery-rate level (from lightest to darkest $\mathrm{FDR} = 0.15$, $\mathrm{FDR} = 0.10$, $\mathrm{FDR} = 0.05$).

			MOVE	HELP	TYCS	RP	YCS	ARPH	DRP	NTFY	AREP	YLD
MoveOn	MOVE	64.4%	–	1.10	1.14	1.31	1.33	1.35	1.71	1.80	1.91	2.06
FullHelp	HELP	58.5%	–	–	1.03	1.19	1.20	1.22	1.55	1.64	1.73	1.87
TerseYouCanSay	TYCS	56.5%	–	–	–	1.15	1.16	1.18	1.50	1.58	1.68	1.81
Reprompt	RP	49.2%	–	–	–	–	1.01	1.03	1.31	1.38	1.46	1.58
YouCanSay	YCS	48.6%	–	–	–	–	–	1.02.	1.29	1.36	1.44	1.55
AskRephrase	ARPH	48.6%	–	–	–	–	–	–	1.27	1.34	1.42	1.53
DetailedReprompt	DRP	37.7%	–	–	–	–	–	–	–	1.06	1.12	1.21
Notify	NTFY	35.7%	–	–	–	–	–	–	–	–	1.06	1.14
AskRepeat	AREP	33.7%	–	–	–	–	–	–	–	–	–	1.08
Yield	YLD	31.2%	–	–	–	–	–	–	–	–	–	–

The results of the pairwise strategy comparisons are illustrated in Table 4. The first column contains the individual recovery rates for each strategy. The other cells contain the ratio of the recovery rates between the strategies in the corresponding row and column. For instance, the last cell in the first row indicates that the *MoveOn* strategy was 2.06 times more successful than the *Yield* strategy (e.g. 64.4% versus 31.2%).

Since we performed 45 comparisons (each strategy was compared with each other strategy), we had to account for multiple comparisons when analysing the statistical significance of these results. To accomplish this, we used the false-discovery-rate method (Benjamini and Hochberg, 1995). The method allows us to compute the expected rate of false significant differences among the detected significant differences. The false-discovery-rate (FDR) for each result is illustrated by the shade of grey. For instance, we expect that 5% of the 10 cells with $\mathrm{FDR} = 0.05$ are actually not significant differences. While significant differences could not be established for every strategy pair, the detected differences allow us to identify a partial ordering.

The *MoveOn*, *Help* and *TerseYouCanSay* strategies occupy the top three positions, with no statistically significant differences detectable between them. In retrospect, this result is not surprising. A number of studies (Swerts et al., 2000; Rotaru and Litman, 2005) have shown that once an error has occurred, the likelihood of having an error in the next turn is significantly increased (our data also confirms this result). As we go deeper into a spiral of errors, patience runs out, frustration is likely to increase, and the acoustic and language mismatches are likely to become more pronounced. Moreover, the fact that there was a non-understanding in the first place indicates that the system is in a difficult position in terms of decoding the current user intention. When the system abandons the current question and attempts to solve the problem by using a different dialogue plan, these effects are likely to be attenuated, and chances

of correct understanding become higher. Similarly, when the system provides help including sample responses for the current question, the users might find better ways (from a system's perspective) to express their goals, or they might find out about other available options for continuing the dialogue from this point.

The high performance of the *MoveOn* strategy is consistent with prior evidence from a Wizard-of-Oz study of error handling strategies (Skantze, 2003). Skantze's study has revealed that, unlike most spoken dialogue systems, human wizards often did ***not*** signal the non-understandings to the user when they occurred. Instead, they asked different task-related questions to advance the dialogue. This strategy generally led to a speedier recovery. In the RoomLine system, the *MoveOn* strategy implements this idea in practice, and the observed performance confirms the prior evidence from Skantze's study. Although not surprising, we do find this result very interesting, as it points towards a road less travelled in spoken dialogue system design: when non-understandings happen, instead of trying to repair the current problem, use an alternative dialogue plan to advance the task.

The next three strategies — *Reprompt*, *YouCanSay* and *AskRephrase*, form a second tier, all having a statistically better recovery rate than the last four strategies. Finally, no significant differences could be detected in terms of recovery rate between the last four strategies: *DetailedReprompt*, *Notify*, *AskRepeat* and *Yield*.

6. User Responses to Non-Understanding Recovery Strategies

We now move on to the fourth question: ***what are the relationships between each strategy and subsequent user behaviours, and which behaviours are more likely to lead to successful recovery?*** Like before, the analysis is based on data from the *control* condition, where the strategies were engaged in an uninformed fashion.

To perform this analysis, we annotated each user turn that followed a non-understanding according to a tagging scheme for error segments introduced by Shin et al. (2002), and subsequently used by others (Choularton and Dale, 2004; Raux et al., 2005). Like Choularton and Dale (2004), we used an abbreviated version of the original scheme, containing five labels: *repeat* — when the user repeats the previous utterance identically, *rephrase* — when the user rephrases the same semantic content in a different lexical manner, *change* — when the user changes the semantic concepts with respect to the previous utterance, *contradict* — when the user contradicts the system, often as a barge-in and *other* — subsumes response types which do not fall in any of the previous categories (e.g. hang-ups and timeouts).

Figure 7 shows the overall distribution of user response types in our dataset. As a reference, we also show the user response type distributions found by Shin in an analysis of the Communicator corpus, and Choularton and Dale in an analysis of a deployed system for ordering pizza. Note however that a direct comparison between these experiments is not valid since we only considered the user responses which followed a non-understanding (as opposed to throughout any error segment). The distribution of user response types we observed is nonetheless similar to previous studies. When faced with non-understandings, users tend to rephrase (∼45%) more than repeat (∼20%). A notable difference in the distribution appears between the change and contradict user response types. The fact that we only considered turns following non-understandings potentially explains the absence of contradicts (which happen mostly when a system misunderstands), while the large number of change responses is introduced by the *MoveOn* strategy — see Figure 8 and also additional plots available on-line (Bohus, 2005). While in Shin's study of the Communicator data a lot of change responses occurred as users were changing their travel plans to go around weaknesses in the system, this is not the case in this data. Participants in our study were compensated according to the number of scenarios they managed to complete successfully, and the change responses

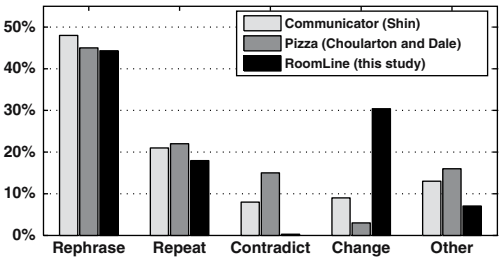


Figure 7. Distribution of user response types.

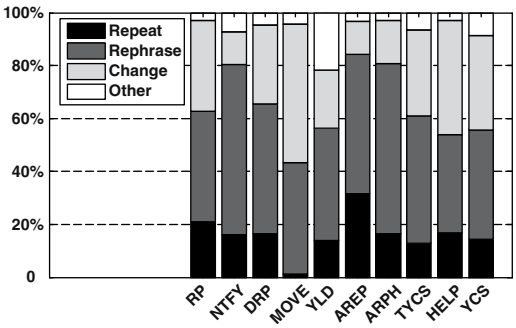


Figure 8. Distribution of user response types by non-understanding recovery strategy.

represent valid contributions to the dialogue, within the confines of the given scenarios.

Next, we analysed the impact of strategies on user response types. The results are presented in Figure 8. The *AskRepeat* strategy leads to the largest number of repeat responses (31%); the *MoveOn* strategy leads to the largest number of change responses (52%); the *AskRephrase* and *Notify* strategies lead to the largest number of rephrase responses (64%). While there is clearly an effect of strategy on user response types, the numbers shown above are not extremely large. Under the assumption that certain types of user responses are more desirable in certain circumstances, these results raise the question of whether the user response types can be controlled even more, for instance by using a more aggressive prompting style (e.g. "Could you repeat what you just said?" instead of "Can you please repeat that?").

Finally, we analysed which type of user responses are more likely to lead to recovery. Figure 9 shows the recovery rate for each user response type. The best recovery performance is attained on change responses (63%). Together with the large number of change responses on the *MoveOn* and help strategies, this result corroborates the high performance of these strategies, and the discussion from Section 5. Somewhat surprisingly, we were not able to establish a statistically significant difference between the recovery rates of user repeat and rephrase responses. In this respect, our results conflict with prior studies which have shown that user rephrases are better recognised and more likely to lead to recovery (Goldberg et al., 2003). Furthermore, the same analysis performed on the sessions collected in the *wizard* condition (recall that in this case a human wizard decided which strategy should be engaged to recover) shows that in that case repeat responses were actually significantly better recognised than rephrase responses. Briefly, we believe this last result is explained by the fact that the wizard made intensive use of the *AskRepeat* strategy, *when this strategy was appropriate*; this in turn boosted the overall number as well as

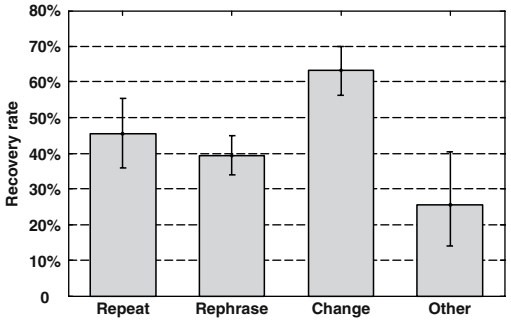


Figure 9. Recovery rate for different user response types.

recovery performance of repeat responses. This does not imply that the system would be able to accomplish the same performance; it merely shows that the relationship between strategies, user responses and how these responses can be interpreted by the system depends on the policy used to engage the strategies.

Given these observations, we conclude this section on a cautionary note: while informative, results regarding the performance of various strategies and user responses do not necessarily generalise across domains and systems. The success of various types of user responses can be strongly influenced by a number of factors such as the nature of the task, the user population, as well as the policy used to engage the strategies. We believe that the solution for successful recovery lies in endowing spoken dialogue systems with the capacity to dynamically adjust their error handling behaviours to the specific characteristics of the domains in which they operate.

7. The Effect of Recovery Policy on Performance: Wizard versus Uninformed

So far we have concentrated our attention on the function and performance of individual recovery strategies. In the two remaining sections we will shift our focus to the *non-understanding recovery policy*. The *recovery policy* describes which strategy should be used in each situation.

Our goal is to endow spoken dialogue systems with the ability to automatically learn good recovery policies from their own experience. We start with the conjecture that the performance of various recovery strategies can be improved by engaging them at the right time, i.e. by using a good recovery policy. For example, asking the user to repeat is not a good course of action if the non-understanding was the result of an out-of-vocabulary word. In contrast, if the non-understanding was caused by a transient noise (e.g. a door slam), asking the user to repeat is probably more likely to succeed. This hypothesis can be stated as: ***a good non-understanding recovery policy can improve the (local) recovery performance***. However, in the end we are interested in improving not only local, but also global dialogue performance. A second hypothesis we therefore need to investigate is: ***a good non-understanding recovery policy can improve global dialogue performance***.

The validity of these hypotheses is not as obvious as it might seem. The performance of the error recovery process is a product of both the set of available strategies and the policy used to engage them. If the set of strategies does not provide good coverage for the types of problems we encounter, a good policy will fail to significantly increase performance. Should this be the case, our efforts would probably be better focused on developing more (and different) recovery strategies, rather than trying to learn a better policy. Finally, even if local recovery performance can be improved by using a smarter recovery

policy, will these local improvements be sufficient to improve global performance, i.e. task success?

To find an answer for the questions raised above, we compared the performance of the wizard's recovery policy against the performance of the uninformed policy. Recall that the wizard had access to more information than a system would have at runtime, and therefore the detection of a performance gap between the policies does not prove that the wizard's policy is also attainable for a system; it only proves that a better policy exists (see discussion in Section 2.1). We start by describing the dialogue performance metrics we used in the comparison in Section 7.1, and we present the results of the comparison in Section 7.2. Finally, in Section 7.3 we analyse the effect of the wizard policy on the performance of the individual non-understanding recovery strategies.

7.1 Performance Metrics

To evaluate global dialogue performance we used two metrics: *task success* and *user satisfaction*. Task success was defined as a binary variable for each of the 10 scenarios performed by a user. User satisfaction was expressed on a 1–7 Likert scale, and was elicited through a post-experiment questionnaire. The user satisfaction score corresponds therefore to the overall experience the user had with the system.

Apart from global dialogue performance, we also wanted to assess the impact of the wizard policy on local non-understanding recovery performance. To our knowledge no traditional, well-established metrics exist in the community for performing this type of evaluation. We therefore constructed a number of metrics which we describe below. Each of these metrics evaluates various characteristics of the user response following the system's attempt to recover from a non-understanding.

The first metric, which we have already introduced in Section 4, was *recovery rate*. To compute this metric, we simply look at whether the next user turn following a system attempt to recover is correctly understood or not. If the next turn is correctly understood (i.e. it is not a misunderstanding and it is not a non-understanding), then we say that the system has successfully recovered. *Average recovery rate* is then simply defined as the number of successful recoveries with respect to the total number of attempts to recover. The underlying variable in this metric is binary — the next turn is either correctly understood or not. The metric therefore does not take into account the magnitude or costs of potential errors. Nevertheless, this metric provides a first order estimate of recovery performance and (because of low variance) is especially useful when we have only a small number of samples to evaluate from.

A second metric we considered was *recovery word error rate*. Instead of looking at whether the next turn is correctly understood or not, we compute and average the word error rate for the user turns following non-understanding recovery attempts. This metric captures in more detail the magnitude of the speech recognition errors in the user responses. However, in a spoken dialogue system we are interested in the correctness of concepts acquired by the system rather than the correctness of the recognition process per se.

The third metric we used, *recovery concept utility*, operates at the concept level. This metric takes into account the number of concepts that are correctly and incorrectly acquired by the system, as well as their relative utilities. The metric is computed as follows:

$$CU = Util_{CC} \cdot CC + Util_{IC} \cdot IC \tag{6.2}$$

where CC is the number of concepts that are correctly acquired by the system from the user's response, and IC is the number of concepts that are incorrectly acquired from that turn. $Util_{CC}$ and $Util_{IC}$ are weighting factors for the correctly and incorrectly acquired concepts and are obtained through a logistic regression model which relates the average number of correctly and incorrectly acquired concepts per turn to overall task success. A model constructed with in-domain data showed that $Util_{CC} = +7.81$, and $Util_{IC} = -7.19$. For the interested reader, the methodology for deriving these costs is described in more detail in (Bohus and Rudnicky, 2005). Because it takes the domain-specific costs for correct and incorrect concepts into account, we consider this metric more appropriate than the traditional concept error rate.

Finally, the last metric we considered was *recovery efficiency*. This metric goes one step further than the recovery concept utility, and also normalises for the amount of time spent by the system during the recovery strategy. The motivation behind this metric is that some recovery strategies use shorter prompts than others, and therefore might succeed (or fail) faster. To normalise for the amount of time spent during recovery, we compute the number of concepts (correct and incorrect) we would expect the system to acquire on average during that time interval. We then subtract these numbers from the number of correct and incorrect concepts we did actually acquire in the next user turn. The formula for this metric is:

$$RE = Util_{CC} \cdot (CC - t \cdot rcc) + Util_{IC} \cdot (IC - t \cdot ric) \tag{6.3}$$

where t is the time elapsed between the original non-understanding and the next user turn, and rcc (and ric) are the average rates (per second) of acquiring correct (and incorrect) concepts during non-understanding recovery segments. In other words, during the amount of time t the system spent in its attempt to recover, we would expect to obtain on average $t \cdot rcc$ correct concepts and $t \cdot ric$ incorrect concepts. We subtract these from the actual number of correct

(CC) and incorrect (IC) concepts we obtained in the user response, and then we take the corresponding utilities into account.

7.2 Results

The results of the comparison are shown in Table 5 and illustrated in Figure 10 (a)–(f). Since performance varies considerably between the native and non-native users, we present the breakdown of the differences in these two populations. In Table 5, the second column shows the overall performance (both groups together); the third column shows the overall differences between the *wizard* and the *control* conditions, while columns 4 and 5 show the differences

Table 5. Performance comparison between the wizard and the uninformed recovery policy (bold entries mark differences that are significant at $p < 0.05$).

Metric	Overall	Wizard vs Uninformed	Wizard vs Uninformed (natives)	Wizard vs Uninformed (non-natives)
Task Success (%)	(a) 75.1	$78.5 \approx 71.7$	$85.2 \approx 85.2$	$\mathbf{57.4 > 31.6}$
User Satisfaction (1–7)	(b) 3.93	$3.87 \approx 4.00$	$4.29 \approx 4.47$	$2.67 \approx 2.67$
Recovery Rate (%)	(c) 48.7	$50.1 \approx 46.5$	$61.0 \approx 56.4$	$37.9 \approx 34.4$
Recovery WER (%)	(d) 38.9	$\mathbf{35.4 < 44.5}$	$\mathbf{26.6 < 35.7}$	$\mathbf{46.4 < 55.7}$
Recovery concept utility	(e) 2.80	$3.01 \approx 2.58$	$4.13 \approx 4.12$	$\mathbf{1.62 > 0.63}$
Recovery efficiency	(f) 0.41	$\mathbf{0.81 > 0.00}$	$1.74 \approx 1.50$	$\mathbf{-0.34 > -1.90}$

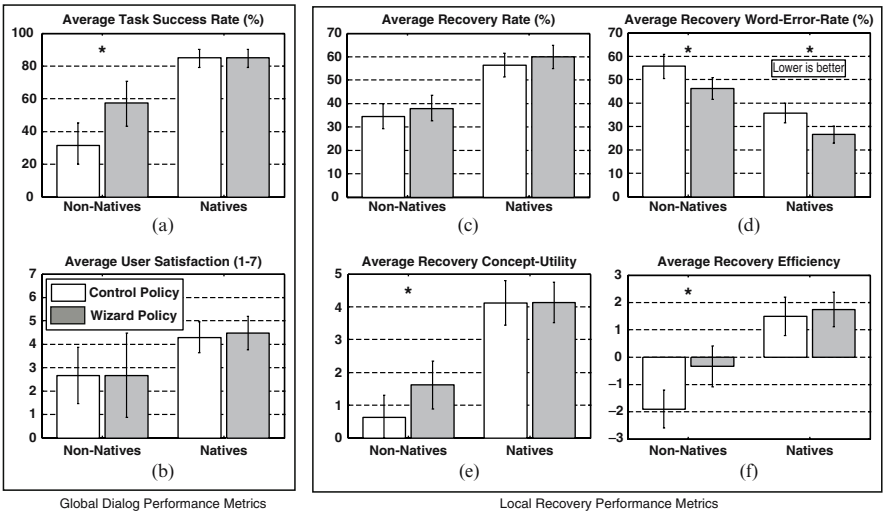


Figure 10. Performance comparison between the uninformed (white bars) and wizard (grey bars) recovery policy (* marks statistically significant differences at $p < 0.05$).

between conditions within the native and non-native populations. To test for statistical significance we used t-tests when comparing proportions (e.g. task success or recovery rate), and non-parametric Mann–Whitney U-tests for the other continuous-valued metrics (their values are not normally distributed).

As Figure 10 and Table 5 illustrate, an overall pattern emerges. The wizard policy does indeed lead to statistically significant performance improvements on a number of metrics, but the improvements appear mostly within the non-native population, i.e. in the group of users that had more difficulties using the system.

For instance, while no task success improvement can be detected for native users, there is a large task success improvement for non-native users (see Figure 10a). The average task success rate grows from 31.6% in the *control* condition to 57.4% in the *wizard* condition. This increase bridges half of the original performance gap between native and non-native users in the *control* condition. Despite this increase in task success rate, no statistically significant differences can be detected with respect to user satisfaction (Figure 10b); the small number of samples we have (one per user) and the large variance of this metric lead to wide confidence bounds on the mean estimates and preclude a reliable comparison. Nevertheless, the same trend of larger, statistically significant improvements for the non-native users is observed again on the local recovery performance metrics (Figure 10c–f). Statistically significant improvements can be detected in the non-native population for three of these metrics: recovery word error rate, recovery concept utility, and recovery efficiency.

We believe the explanation for the observed result lies in the simple fact that it is easier to improve performance when performance is low (in our case, for the non-native users). This result also confirms our conjecture from Section 4: improvements in non-understanding recovery performance do indeed translate into significant increases in task success for the non-native population.

7.3 Effect of Policy on Individual Recovery Strategy Performance

Next, we analysed the effect of the policy on the performance of individual recovery strategies. Our original hypothesis was that, if the strategies are engaged "at the right time", their performance would improve.

Figure 11 shows the number of times each non-understanding recovery strategy was engaged by the wizard. Figure 12 shows the *recovery rate* for each of the 10 strategies, under the two different conditions. We were able to establish a statistically significant difference ($p = 0.0023$, or $p = 0.023$ Bonferroni corrected for multiple comparisons (Savin, 1980)) only for the *AskRepeat* strategy. *AskRepeat* is however the strategy most often engaged by the wizard. While this strategy ranked 9th when engaged in an uninformed fashion, its

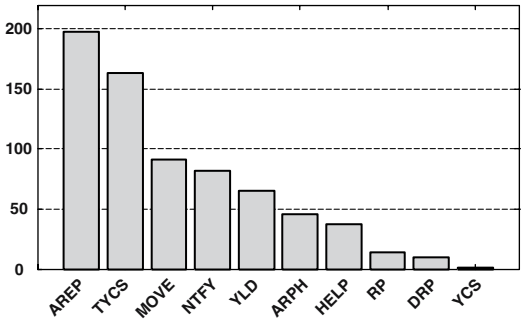


Figure 11. Number of times each strategy was engaged by the wizard.

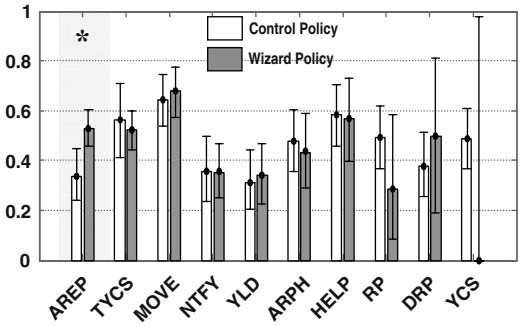


Figure 12. Impact of policy on individual strategy performance.

performance improved considerably from 33.7% to 53% under the wizard policy and is on par with the other top-performing strategies such as giving help (*TerseYouCanSay* and *Help*) or advancing the task by asking a different question (*MoveOn*). The same improvement in the *AskRepeat* strategy was also detected on the other three recovery performance metrics.

This result shows that strategy performance can indeed be improved by the use of a better recovery policy. At the same time, the lack of detectable differences in the other strategies is somewhat disappointing. In retrospect, this result might be explained by the fact that the decision task the wizard had to perform was quite difficult, even with access to the full audio signal. To maintain the illusion that users were interacting with an autonomous system, the wizard had to choose one of ten recovery strategies in a very short time interval: 1–2 seconds. This selection task is easier for some of the strategies than for others. Furthermore, a number of strategies, such as *YouCanSay*, *Reprompt*, and *DetailedReprompt*, were very rarely engaged by the wizard and as a result the confidence intervals on their performance estimates are very wide, and preclude accurate comparisons.

8. Towards Learning a Recovery Policy

In the previous section we have established that significant improvements in performance can be achieved by using a better policy with the current set of strategies. In this section we present a set of preliminary results on the problem of ***learning such a policy from data***.

We take a simple, decision-theoretic approach. First, we learn to predict the likelihood of success for each non-understanding recovery strategy from features available at runtime. Then, to implement a policy, we compute the expected utility for each strategy (taking into account the probabilities and costs for success and failure), and select the strategy with the maximum expected utility.

In Section 8.1, we describe the construction of predictors for the likelihood of success of each strategy. Next, in Section 8.2 we discuss two recovery policies based on these predictors.

8.1 Predicting the Likelihood of Success for Non-Understanding Recovery Strategies

We use logistic regression models to develop runtime predictors for the likelihood of success of each non-understanding recovery strategy.

8.1.1 Data. As training data, we use the turns in which a non-understanding occurred and the strategy we are interested in was engaged. The training target value is the success or failure of the strategy in that particular case. Success is defined as "the next user turn is correctly understood by the system".

Note that for learning we only use data collected in the *control* condition, where the non-understanding recovery strategies were engaged in an uninformed fashion. The wizard policy might introduce a potential bias in the distribution of features, which can negatively affect learning and generalisation. For instance, if the wizard never used the *AskRepeat* strategy when the number of words in the original non-understood utterance was very large, we will never encounter that set of circumstances, or see how the *AskRepeat* strategy behaves under those conditions. In this case, the distribution of the *number_of_words* feature might be skewed towards small values, and that might negatively affect the learning process.

Given the relatively large number of different recovery strategies in the system (10), the number of available instances to learn from is fairly small — at about 60–70 invocations per strategy. The small number of samples further complicates an already difficult learning problem, since we face a relatively high risk of over-fitting the training data.

8.1.2 Features. We identified a large number of features available at runtime which could carry information about the likelihood of success for various non-understanding recovery strategies. The features are collected from different levels of processing in the spoken dialogue system. For instance, from the speech recognition level we collected various features characterising the current non-understood utterance: the number of words, the signal and noise levels, the number of and proportion of words tagged as unconfident by the speech recogniser. Similarly, from the language understanding level we collected various features reflecting the quality of the parse. From the dialogue management level we used information about the dialogue state, as well as the history of the dialogue up to that point (e.g. how many previous consecutive non-understandings we encountered, what was the average confidence score so far, etc.)

As a first measure to guard against over-fitting, we transformed continuous features into binary features by using a preset threshold. Furthermore, we eliminated features that had small class-conditional counts. The remaining set of features which was used in training is available as an online appendix (Bohus, 2005).

8.1.3 Models. Since we are interested in predicting the expected likelihood of success for each strategy (rather than a binary success or failure), we decided to use stepwise logistic regression models. These models are simple, easy to build and incorporate a mechanism for feature selection. Moreover, as opposed to a number of other discriminative classifiers, logistic regression provides good class posterior probability scores (e.g. estimates for the likelihood of success).

In stepwise logistic regression, features (variables) are added to the model one by one, as long as they increase the likelihood of the data. A feature is accepted in the model if it produces a data likelihood increase that is statistically significant with a p-value below a preset $P - accept$. At the same time, in each step features already in the model are tested for exclusion. A feature is rejected if the resulting model is not significantly worse, as determined by a preset $P - reject$. In our case, we set $P - accept = 0.05$ and $P - reject = 0.30$. Finally, as a second preventive measure against over-fitting, we evaluated the model after each regression step using a leave-one-out procedure and stopped adding features as soon as the average data likelihood in the leave-one-out evaluation decreased.

8.1.4 Results. We fitted ten stepwise logistic regression models, one for each strategy. The results are illustrated in Table 6. For five of the ten strategies we can build models which perform better than a majority baseline, on both a soft (average log-likelihood) and hard (binary classification) error

Table 6. Performance of success predictors for the 10 non-understanding recovery strategies.

Strategy	Avg. data log-likelihood			Hard error rate (%)		
	Majority baseline	Leave-one out perf.	Avg. LL increase	Majority baseline	Leave-one out perf.	Relative reduction in error rate
DetailedReprompt	−0.6626	−0.6330	0.0296	37.7%	34.4%	8.7%
Reprompt	−0.6930	−0.6645	0.0286	49.2%	32.8%	33.3%
TerseYouCanSay	−0.6846	−0.6576	0.0270	43.5%	32.6%	25.0%
MoveOn	−0.6508	−0.6357	0.0151	35.6%	30.0%	15.6%
YouCanSay	−0.6927	−0.6729	0.0199	48.6%	34.3%	29.4%
AskRepeat	−0.6391	−0.6389	0.0002	33.7%	33.7%	0.0%
Help	−0.6788	−0.6776	0.0012	41.5%	46.1%	−11.1%
AskRephrase	−0.6921	–	–	47.8%	–	–
Notify	−0.6518	–	–	35.7%	–	–
Yield	−0.6211	–	–	31.2%	–	–

metric. For the last three models in Table 6 no features ever entered the regression. In this case the constructed predictors simply predict a probability of success equal to the majority baseline in the training data. In general, the performance of the individual predictors is not very good, but this is not surprising given the small number of training instances, the reduced number of features used, and difficulty of the prediction task (we are trying to predict in advance whether or not the next turn is correctly understood, without any information from that turn.)

8.2 Policies for Recovery

If we can predict for the likelihood of success of each non-understanding recovery strategy, a recovery policy is easy to construct: we simply choose the action with the maximum expected utility:

$$\Pi = \arg\max\{P_{SUCC}(S) \cdot U_{SUCC}(S) + P_{FAIL}(S) \cdot U_{FAIL}(S)\} \quad (6.4)$$

where $P_{SUCC}(S)$ is the estimated probability of success for strategy S, $P_{FAIL}(S) = 1 - P_{SUCC}(S)$ is the probability of failure, and $U_{SUCC}(S)$ and $U_{FAIL}(S)$ are the utilities of success and failure for strategy S.

We defined two policies. The first policy (*max-recovery-rate*) aims to maximise the recovery rate by choosing the strategy with the maximum likelihood of success. This is equivalent to using the values $U_{SUCC} = 1$ and $U_{FAIL} = 0$ as the utilities for success and failure. The second policy (*max-recovery-efficiency*) aims to maximise the recovery efficiency, as defined in Section 7.1. In this case $U_{SUCC}(S)$ is the average recovery efficiency of strategy S when

S was successful (i.e. the next turn is correctly understood), while $U_{FAIL}(S)$ is the average recovery efficiency of strategy S when S failed.

To obtain a preliminary estimate for the performance of these policies, we looked at what happened in the data from the *wizard* condition, when the wizard happened to make the same decisions as our learned policies would have made. Since the *MoveOn* strategy was not available at all points in the dialogue, we eliminated it from the learned policies in this analysis (this is a simple way to avoid the policy deciding to engage the *MoveOn* strategy when it is not available). The results show that, within the subset of instances where the wizard made the same decision as the *max-recovery-rate* policy, the recovery rate performance was 69.8%. At the same time, the wizard's overall recovery rate (throughout the whole wizard dataset) was significantly lower — 50.1% (see Table 5); also, the overall recovery rate with the uninformed policy from the *control* group was 46.5% (see Table 5).

Similarly, on the instances where the wizard agreed with the *max-recovery-efficiency* policy, the recovery efficiency performance was 2.02, significantly larger than the overall wizard recovery efficiency (0.81), and the uninformed policy recovery efficiency (0.00).

While we view these results as promising, we would like to point out a potential problem in this type of evaluation. Given that both the wizard and the learned policy strive to maximise performance, the distribution of the subset of non-understandings where they agree might not be representative for the true distribution of non-understandings — these might be the cases where it is easier to tell which strategy should be used to recover. Ultimately, a new user study where the system runs with the learned policy is required in order to robustly evaluate its performance.

9. Conclusion

The work described in this chapter is part of a larger research program (Bohus, 2004) aimed at endowing spoken dialogue systems with better error handling capabilities. In an effort to shed more light on non-understandings, we performed an empirical analysis of these errors and 10 associated recovery strategies, based on a corpus of dialogues collected with a mixed-initiative spoken dialogue system for making conference room reservations.

An error source analysis has confirmed that a large number of non-understandings (and misunderstandings) can be blamed on speech recognition errors. Nevertheless, a significant number of non-understandings ($\sim$30%) stem from requests for inexistent application functionality, user corrections, and out-of-grammar expressions. At the same time, users are not always aware of the full functionality provided by the system. We believe these language-domain errors can be addressed by better steering the users into the application's space.

In this vein, we plan to explore more carefully the design and use of *you-can-say* help messages. Currently these messages inform users about possible ways to answer the current question. In the future, we plan to investigate the possibility of providing information about other options available at this point in the dialogue. This raises an interesting design issue, since the number of different options available to the user can be fairly large in a mixed-initiative spoken dialogue system. A *targeted-help* approach (Gorrell et al., 2002) may provide a potential solution to this problem. A second path we intend to explore is issuing *you-can-say* messages preemptively (e.g. without waiting for a non-understanding to happen) if we can detect that the user belongs to a "problematic" population such as non-native speakers, first time users, etc.

We have also confirmed that non-understandings exert a negative impact on task success, and we have quantified this impact. Models discussed in Section 4 revealed that the effect of non-understandings on task success is marginal when the frequency of non-understandings is below 10–15%, but increases fast after that. Similarly, we found that the effect of the non-understanding recovery rate on performance is greatest when the recovery rate is below 60–70%, and smaller once we are above that limit. While the specific numbers might differ across applications and domains, we expect that the nature of the relationship remains similar. The type of analysis we presented in Section 4 can provide useful information for focusing future development efforts. In our domain, it indicates that improvements in the non-understanding recovery rate are likely to lead to significant increases in task success, especially for non-native users (where the non-understanding rate is relatively high, and the non-understanding recovery rate is relatively low).

Next, we compared the individual performance of the ten different recovery strategies. The results show that, when engaged without any prior knowledge, the best performing strategies in our domain are: (1) advancing the conversation by ignoring the non-understanding and trying an alternative dialogue plan (*MoveOn*), and (2) providing help messages containing sample responses for the current system question. The high performance of the *MoveOn* strategy corroborates prior evidence from a Wizard-of-Oz study (Skantze, 2003) which showed that human operators often do not signal non-understandings, but rather try to advance the task by asking different questions. In the future, we plan to explore in more detail the potential uses of this strategy, as well as its pitfalls. Specific issues we plan to investigate include identifying more situations in which this strategy is applicable, studying the extent to which this strategy can be decoupled from the system's task, and developing more appropriate metrics for assessing its performance.

In the final part of this chapter we shifted our attention to the recovery policy. We showed that a more informed policy for engaging the non-understanding recovery strategies (implemented by a human wizard) led to

significant improvements in task success, as well as a number of other local performance metrics. The improvements occurred mostly within the non-native population, i.e. the group of users who had more difficulties in using the system.

Finally, we reported a set of preliminary results with respect to learning a recovery policy from data. We used features available at runtime from various levels in the dialogue system to build predictors for the likelihood of success of each non-understanding recovery strategy. For five of the strategies, the learned predictors perform better than a majority baseline. Based on these predictors, we constructed two policies: one aims to maximise the recovery rate, the other aims to maximise the recovery efficiency. Preliminary estimates indicate that these policies are expected to outperform both the wizard and the uninformed policy. While these experiments were conducted with very few training instances and an empirical validation of the learned policy is still necessary, we find these results encouraging as they indicate the feasibility of using a learning approach for deriving a non-understanding recovery policy from data.

Acknowledgements This work, part of the CALO project, was supported by DARPA grant NBCH-D-03-0010. The content of the information in this publication does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred. The authors would like to thank Antoine Raux, Valerie Ventura, Mihai Rotaru, the members of the "Dialogs on Dialogs" group, as well as the anonymous reviewers for helpful suggestions and feedback. Last but not least, we would like to thank the workshop organisers and program committee for the opportunity to combine our two accepted submissions into a single longer paper format.

References

Benjamini, Y. and Hochberg, Y. (1995). Controlling the False Discovery Rate: a Practical and Powerful Approach to Multiple Testing. *Journal of Royal Statistics Society*, B(57):289–300.

Bohus, D. (2004). Error Awareness and Recovery in Task-Oriented Spoken Dialog Systems. Technical report, Carnegie Mellon University, Pittsburgh.

Bohus, D. (2005). Investigation of Non-Understandings Errors and Recovery Strategies. http://www.cs.cmu.edu/dbohus/nonu_investigation/.

Bohus, D. and Rudnicky, A. (2003). RavenClaw: Dialogue Management Using Hierarchical Task Decomposition and an Expectation Agenda. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 597–600, Geneva.

Bohus, D. and Rudnicky, A. (2005). A Principled Approach for Rejection Threshold Optimization in Spoken Dialog Systems. In *Proceedings of 9th*

European Conference on Speech Communication and Technology (INTERSPEECH), pages 2781–2784, Lisbon.

Carpenter, P., Jin, C., Wilson, D., Zhang, R., Bohus, D., and Rudnicky, A. (2001). Is this Conversation on Track? In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 2121–2124, Aalborg.

Choularton, S. (2005). Investigating the Acoustic Sources of Speech Recognition Errors. Macquarie University, Sydney.

Choularton, S. and Dale, R. (2004). User Responses to Speech Recognition Errors: Consistency in Behaviour across Domains. In *Proceedings of 10th Australian International Conference on Speech, Science and Technology (SST)*, pages 457–462, Sydney.

Clark, H. H. (1996). *Using Language*. Cambridge University Press, Cambridge.

Goldberg, J., Ostendorf, M., and Kirchhoff, K. (2003). The Impact of Response Wording in Error Correction Subdialogs. In *Proceedings of ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems*, pages 101–106, Chateaux d'Oex.

Gorrell, G., Lewin, I., and Rayner, M. (2002). Adding Intelligent Help to Mixed Initiative Spoken Dialogue Systems. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 2065–2068, Denver.

Hone, K. and Graham, R. (2000). Towards a Tool for Subjective Assessment of Speech System Interfaces (SASSI). *Natural Language Engineering*, 6:287–303.

Krahmer, E., Swerts, M., Theune, M., and Weegels, M. (1999). Error Detection in Human-Machine Interaction. In Levelt, W., editor, *Speaking: From Intention to Articulation*. MIT Press, Cambridge.

Litman, D., Hirschberg, J., and Swerts, M. (2000). Predicting Automatic Speech Recognition Performance Using Prosodic Cues. In *Proceedings of Conference of the North American chapter of the Association for Computational Linguistics (NAACL)*, pages 218–225, Seattle.

Myers, R., Montgomery, D., and Vining, G. (2001). *Generalized Linear Models: with Applications in Engineering and the Sciences*. Wiley, New York.

Paek, T. (2003). Toward a Taxonomy of Understanding Errors. In *Proceedings of ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems*, pages 53–58, Chateaux D'Oex.

Paek, T. and Horvitz, E. (2000). Conversation as Action Under Uncertainty. In *Proceedings of 16th Conference on Uncertainty in Artificial Intelligence*, pages 455–464, Stanford.

Raux, A., Langner, B., Bohus, D., Black, A., and Eskenazi, M. (2005). Let's Go Public! Taking a Spoken Dialog System to the Real World. In *Proceedings*

of 9th European Conference on Speech Communication and Technology (INTERSPEECH), pages 885–888, Lisbon.

RoomLine (2003). A Spoken Dialog System for Conference Room Reservations. http://www.cs.cmu.edu/dbohus/RoomLine.

Rotaru, M. and Litman, D. (2005). Interactions between Speech Recognition Problems and User Emotions. In *Proceedings of 9th European Conference on Speech Communication and Technology (INTERSPEECH)*, pages 2481–2484, Lisbon.

San-Segundo, R., Pellom, B., and Ward, W. (2000). Confidence Measure for Dialogue Management in the CU Communicator System. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1237–1240, Istanbul.

Sanders, G., Le, A., and Garofolo, J. (2002). Effects of Word Error Rate in the DARPA Communicator Data during 2000 and 2001. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 277–280, Denver.

Savin, N. (1980). The Bonferroni and the Scheffe Multiple Comparison Procedures. In *Review of Economic Studies*, volume 47, pages 255–273. Blackwell Publishing.

Shin, J., Narayanan, S., Gerber, L., Kazemzadeh, A., and Byrd, D. (2002). Analysis of User Behavior Under Error Conditions in Spoken Dialogs. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, volume 3, pages 2069–2072, Denver.

Skantze, G. (2003). Exploring Human Error Handling Strategies: Implications for Spoken Dialogue Systems. In *Proceedings of ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems*, pages 71–76, Chateaux d'Oex.

Swerts, M., Litman, D., and Hirschberg, J. (2000). Corrections in Spoken Dialogue Systems. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 615–618, Beijing.

Theta (2004). Theta: A Small Footprint Text-to-Speech Sythesizer. Technical report, Cepstral LLC, Pittsburgh.

Tomko, S. (2004). Improving User Interaction with Spoken Dialog Systems via Shaping. Technical report, Carnegie Mellon University, Pittsburgh.

Walker, M., Litman, D., Kamm, C., and Abella, A. (1998). Evaluating Spoken Dialogue Systems with PARADISE: Two Case Studies. *Computer Speech and Language*, 12(3):317–347.

Ward, W. and Isaar, S. (1994). Recent Improvements in the CMU Spoken Language Understanding System. In *Proceedings of ARPA Workshop on Human Language Technology*, pages 213–216, Plainsboro.

Chapter 7

GALATEA: A DISCOURSE MODELLER SUPPORTING CONCEPT-LEVEL ERROR HANDLING IN SPOKEN DIALOGUE SYSTEMS

Gabriel Skantze
Department of Speech, Music and Hearing
KTH, Stockholm, Sweden
gabriel@speech.kth.se

Abstract In this chapter, a discourse modeller for conversational spoken dialogue systems, called GALATEA, is presented. Apart from handling the resolution of ellipses and anaphora, it tracks the "grounding status" of concepts that are mentioned during the discourse, i.e., information about who said what when. This grounding information also contains concept confidence scores that are derived from the speech recogniser word confidence scores. The discourse model may then be used for concept-level error handling, i.e., grounding of concepts, fragmentary clarification requests, and detection of erroneous concepts in the model at later stages in the dialogue. An evaluation of GALATEA, used in a complete spoken dialogue system with naive users, is also presented.

Keywords: Spoken dialogue systems; Grounding; Clarification; Error handling; Discourse modelling.

1. Introduction

A common source of errors in spoken dialogue systems is the speech recogniser (ASR), and the handling of such errors is a crucial issue in the design of spoken dialogue systems. The most common way of handling such errors has been to use utterance confidence scores for selecting implicit or explicit verification of full utterances. This is often feasible in dialogues where utterances are relatively short and predictable. However, in dialogue systems that are designed to allow relatively free, conversational speech, with longer, more unpredictable utterances, such utterance-level error handling is too simplistic.

L. Dybkjær and W. Minker (eds.), Recent Trends in Discourse and Dialogue, 155–189.

In such systems, it is common to use statistical n-gram language models in the ASR. These tend to be better at covering conversational language and degrade more gracefully when the user's utterances are poorly covered by the grammar (Knight et al., 2001), but they may also give rise to speech recognition results which are often only partly correct. When making semantic interpretations of such results, some semantic concepts will be correct and some not. This calls for error handling on the concept level, i.e., individual concepts in utterances should be assigned confidence scores and be considered for error handling strategies, such as grounding, clarification and error detection.

In this chapter, a discourse modeller for conversational spoken language, called GALATEA, is presented. It is especially designed to support concept-level error handling. GALATEA is not a complete dialogue manager, but rather a processing step in the interpretation process, where utterances are interpreted in context. Apart from handling the resolution of ellipses and anaphora, it tracks the grounding status of concepts that are mentioned during the discourse, i.e., information about who said what when. This grounding information also contains concept confidence scores that are derived from the speech recogniser word confidence scores. GALATEA builds a discourse model – a model of what has been said during the discourse and which entities are referred to. The discourse model may then be consulted by an action manager that selects error handling strategies, such as grounding, clarification and late error detection, on the concept level.

GALATEA is developed within the HIGGINS project (Edlund et al., 2004). HIGGINS is a test bed for investigating error handling techniques in spoken dialogue systems. The initial domain chosen for HIGGINS is that of pedestrian city navigation and guiding, which is similar to the now classic MapTask domain (Anderson et al., 1991), as well as to a number of guide systems, such as REAL (Baus et al., 2002). A user gives the system a destination and the system guides the user by giving verbal instructions. The system does not have access to the users' positions. Instead, it has to rely on their descriptions of their surroundings. Since the user is moving, the system continually has to update its model of the user's position and provide new, possibly amended instructions until the destination is reached. For simulation, a 3D model of a virtual city is used (see Figure 1). The domain is complex enough to generate a variety of error types. The surroundings the user and system talk about contain complex landmarks and relations that are challenging to interpret and represent semantically. For such semantic representations, deep semantic structures are needed.

Table 1 shows a made-up scenario with recognition results, which displays the current capabilities of the system.[1] The different error handling techniques illustrated in the example will be described later on.

[1]The system is currently built for Swedish, but all examples in this chapter have been translated into English.

Figure 1. The 3D simulation that is used for user tests.

Table 1. A made-up example scenario. The rightmost column shows ASR results in which word confidence scores are represented with different shades of grey, where dark means high confidence.

Turn	*Transcription*	*ASR result*
S.1	*Where do you want to go?*	
U.2	To an ATM.	TO AN **ATM**
S.3	*Ok, an ATM. Can you describe where you are now?*	
U.4	I have a large concrete building on my left and a tree in front of me.	I HAVE **LARGE** HELLO CONCRETE **BUILDING** ON MY **LEFT** AND **TREE** ON RIGHT
S.5	*What colour is the concrete building?*	
U.6	Red.	**RED**
S.7	*Ok, can you see a wooden building in front of you?*	
U.8	No, but I have one on my right.	NO I **HAVE** ONE ON RIGHT
S.9	*On your right?*	
U.10	Yes.	**YES**
S.11	*Ok, I think I know where you are. Walk a little bit forward and take a left after the red building.*	

2. Error Handling in Dialogue Systems

In this section, issues important for error handling in spoken dialogue systems will be discussed.

2.1 Miscommunication

Miscommunication is a general term that denotes all kinds of problems that may occur in dialogue. A common distinction is made between misunderstanding and non-understanding (e.g. Hirst et al., 1994). Misunderstanding means that the addressee obtains an interpretation that is not in line with the speaker's intentions. If the addressee fails to obtain any interpretation at all or obtains more than one interpretation with no way to choose among them, a non-understanding has occurred. One important difference between non-understandings and misunderstandings is that non-understandings are recognised immediately by the addressee, while misunderstandings may not be identified until a later stage in the dialogue. Both of these forms of miscommunication may concern complete utterances or parts of utterances, i.e., partial misunderstanding and partial non-understanding.

One may also classify problems depending on at which "action level" they occur. Clark (1996) and Allwood et al. (1992) make a distinction between four levels of action that take place when a speaker is trying to say something to an addressee. According to Clark, the levels are (from higher to lower):

- Acceptance: proposal and consideration
- Understanding: signalling and recognition
- Perception: presentation and identification
- Contact: execution and attention

For successful communication to take place, communication must succeed at all levels on the "action ladder", from contact to acceptance: the addressee must attend to the listener, hear what is said, understand it and accept the proposal. The order of the levels is important; in order to succeed on one level, all the levels below it must be completed.

2.2 Early Error Detection

Given a speech recognition result, a system must determine if it should accept the utterance and hypothesise that this is what the user said, or reject it. This decision is often based on some sort of confidence score, which is compared to a certain threshold. This is the process of early error detection. The confidence scores are usually based on the probabilities from the acoustic and language models, and the structure of the n-best list (Evermann and Woodland, 2000).

To improve early error detection, machine-learning has been used to classify utterances as correct or incorrect, based on features from the recognition result, acoustic features, and dialogue history (e.g. Gabsdil and Lemon, 2004).

For concept-level error handling, confidence scores should be calculated for the individual words in the speech recognition result, just like in Table 1, and transferred into concept confidence scores during semantic interpretation (Gabsdil and Bos, 2003). Skantze and Edlund (2004a) investigate the use of machine learning for early error detection on the word-level, using confidence scores as well as features from the utterance and discourse context.

2.3 Grounding

Speakers in dialogue can never take for granted that they have correctly understood what the interlocutor is saying. To deal with this uncertainty, speakers constantly provide positive and negative "evidence of understanding" (or "feedback" as in Allwood et al. (1992)) to each other. This is the process of *grounding* (Clark, 1996). If positive evidence is given on one of the action levels discussed above, all the levels below it are presumed to have succeeded. If negative evidence is signalled on one of the levels, all the levels above it are also presumed to have failed, while the ones below it are presumed to have succeeded.

2.3.1 Display of understanding. According to Clark, every contribution requires positive evidence, if it is to be regarded as common ground. Clark (1996) discusses different kinds of positive evidence:

- Assertion of understanding
- Presupposition of understanding
- Display of understanding
- Exemplification of understanding

Assertion of understanding, such as "mhm", "okay", "I understand", is commonly used as evidence of understanding. However, it can only give evidence at the utterance level, and it does not ground anything at the concept level. Presupposition of understanding means that the addressee continues with a relevant next contribution, which may also give evidence at the utterance level. The distinction between "display" and "exemplification" is not so clear, but both include cases where the addressee displays or exemplifies what she has constructed the speaker to mean in the next turn, including verbatim repetitions or paraphrases. We will use the term "display" here for the cases where parts of what is said is repeated. By displaying understanding – in this sense – speakers may verify that their understanding is correct, which may also be done on

the concept level. Take, for example, the turn S.5 in Table 1. Apart from requesting the colour of the building, the system also gives some evidence of understanding. The understanding of the concepts CONCRETE and BUILDING are displayed, but not the concept LARGE. If the understanding was incorrect, i. e., a misunderstanding, the user now has the opportunity to correct the system.

2.3.2 Clarification. If the addressee does not understand, or is not sufficiently confident in parts of her understanding, she may choose to clarify those parts, by posing a request. S.9 in Table 1 is an example, where the system lacks confidence in the concept RIGHT. Such requests are often formulated as yes/no questions. If the addressee is missing some part of the utterance, the clarification request may instead be formulated as a wh-question. The system could, for example, have said:

(1) What do you have on your right?

Purver et al. (2004) explore the different forms that clarification requests may take, by studying the British National Corpus. An interesting finding is that 45% of the clarification requests were elliptical or fragmental – just like S.9 in Table 1. For concept-level error handling, these are especially interesting, since they focus on problematic concepts and thereby make the dialogue more efficient.

It is important to note that while clarification requests may signal non-understanding, they may also give positive evidence of understanding by display, as is illustrated in (1). In this example, the system displays that it has understood that the user has something on her right, but at the same time gives negative evidence on its understanding of what it is.

The most explored techniques for grounding in spoken dialogue systems are "explicit" and "implicit" verification, illustrated below:

(2)
	U.1	I have a large concrete building on my left
	S.2e	Do you have a large concrete building on your left?
	S.2i	(You have) a large concrete building on your left.
	S.2ii	What colour is the large concrete building that you have on your left?

S.2e exemplifies explicit verification, which may be described as a clarification request. S.2i exemplifies implicit verification, where the system displays its understanding. These techniques may often be experienced as tedious and unnatural, since they operate at the utterance level (including all concepts in the utterance) and are realised as separate communicative acts.

Implicit confirmation may also be integrated into the next act, as in S.2ii, but as Gabsdil (2003) notes, this is done at the utterance-level. As the example shows, this may sound unnatural and tedious, compared to just including the most important concepts, as in S.5.

An important difference between clarification and display of understanding is that the concepts are not considered as grounded after clarification requests if they are not verified by the user, whereas a display of understanding does not require such verification.

A dialogue system also needs to consider the case of complete non-understanding. Typically, the system says something like "Sorry, I didn't understand", thereby encouraging the user to repeat. Skantze (2005) shows in an experiment that this is not what humans typically do when faced with complete non-understanding. Instead, they tend to ask new task-related questions without signalling non-understanding.

2.3.3 Computational models of grounding. Traum (1994) presents a computational model of grounding where a recursive transition network is used to model the stages of the grounding process, including acknowledgements, repairs, and request for repairs. Larsson (2002) describes a model for grounding utterances, or issues, where positive and negative evidence is given at all action levels, using the information state approach. While these models handle the process of grounding information, the units that are considered for grounding are utterances or issues. The examples provided do not show how individual concepts can be grounded. On these accounts, there are special grounding or feedback utterances that are treated differently compared to other utterances.

The approach taken here is instead to model the way all utterances may display understanding and how they contribute to the grounding process. On this account, clarification requests and communicative acts that display understanding are not special types of utterances. Instead, a general model for handling assertions, requests and grounding is used. A problem with many models of grounding is that negative answers to grounding and clarification moves are not used as constraining information. For example, in Larsson (2002), a "backup" copy of the dialogue state is kept to restore the state if the proposed interpretation is rejected. Consider example (2) above. If the user would answer "no", the fact that the user does not have a large concrete building on her left is valuable information for constraining possible user positions. In HIGGINS, clarification requests are treated as other requests, including the support for negations. The way they are realised – in full form or as ellipses – will affect how they display understanding.

Concept-level clarification requests have been studied to a greater extent than concept-level display. Rieser (2004) and Schlangen (2004) describe

implementations of systems that are capable of posing fragmentary clarification requests based on concept confidence scores on all action levels. However, the models do not handle the user's reactions to those requests.

2.4 Late Error Detection

Clarification requests correspond to what Schegloff (1992) calls second-turn repair, i.e., the repair is done in the second turn counting from the problematic utterance. Third-turn repair occurs if an interlocutor gives positive evidence of understanding and the first speaker realises that she has been misunderstood and initiates a repair. By displaying evidence of understanding, the system may detect errors by letting the user initiate a third-turn repair. Late error detection is the task of detecting that the user has initiated such a repair, i.e., to detect a previous misunderstanding. In Krahmer et al. (2001), memory-based learning is used for detecting negative and positive cues in the third turn for late error detection. Bohus and Rudnicky (2005) call this the "belief updating problem", i.e., how an initial belief in an interpretation of an utterance (first turn), a system response to that utterance (second turn), and a user reaction to that response (third turn), should be combined to form an updated belief in the interpretation of the first turn. In their approach, binary logistic regression is used to learn this classification, based on features such as initial confidence score, prosody, barge-in, etc.

Late error detection may also be performed if an interlocutor realises that her model of the world contains contradictory information. An example can be seen in Table 1. After turn U.4, the system believes that the user has a tree on her right, since there was an undetected misrecognition. But after turn U.10, the system realises that there is no place the user could be. The only fact that has a relatively low confidence and has not been grounded is that the user has a tree on her right. The system may now assume that this was a misunderstanding, and update its model.

2.5 Choosing Error Handling Strategies

The previous discussion shows that there are several ways to handle uncertainty and errors in dialogue. A speaker may display understanding, request clarification on what is not understood, or presuppose understanding and defer the detection of errors to a later stage in the dialogue. As Allen et al. (1996) point out, sometimes it may be better to "choose a specific interpretation and run the risk of making a mistake as opposed to generating a clarification subdialogue". The choice of strategy should depend on the result of the early error detection, i.e., how confident the system is in its understanding, but also on the consequence that a misunderstanding would have; the cost of a potential misunderstanding should be compared to the cost of making the grounding move.

As an example, take utterance S.3 in Table 1. The system has a fairly high confidence in the user's position, but still chooses to give positive evidence of understanding, instead of deferring the detection. The reason is that if the system would misrecognise the user's goal, the system would not detect the error until the user had already reached the goal.

The possibility to defer error handling based on consequence of misunderstanding has not been explored to a great extent; confidence scores are most often only considered once and not stored for late error detection. To be able to defer error detection, as well as choosing from different error handling strategies on the concept level, the system should keep a model of when concepts have been grounded, by whom and how confident the system is in this.

3. The Higgins Spoken Dialogue System

In this section, the architecture, semantic structures and modules of the HIGGINS spoken dialogue system will be described.

3.1 Architecture

The HIGGINS spoken dialogue system is a distributed architecture with modules communicating over sockets. Each module has well-defined interfaces, and can be implemented in any language, running on any platform. The interfaces are described using XML schema. Figure 2 shows the most important modules and messages in HIGGINS, in the present configuration. From the

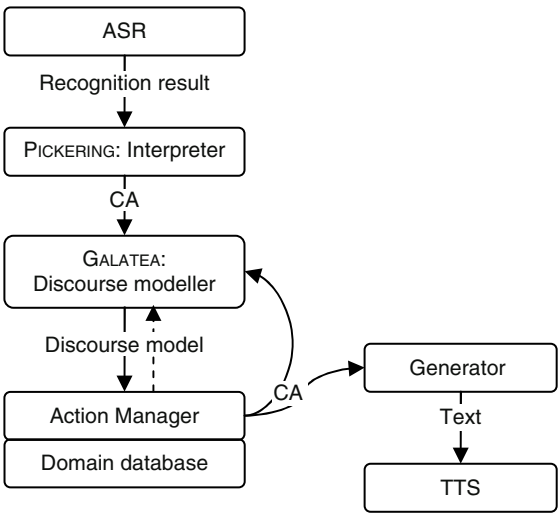


Figure 2. The most important modules and messages in the HIGGINS architecture. CA stands for communicative act.

ASR, the top hypothesis with word confidence scores is sent to an interpreter, called PICKERING. PICKERING recognises and creates context-free semantic representations of communicative acts (CAs).

In HIGGINS, dialogue management is not implemented as a single module. Instead, this processing is divided into a discourse modeller (GALATEA) and an action manager. This division is similar to the approaches taken in Allen et al. (2001) and Pfleger et al. (2003). The communicative acts are sent from PICKERING to GALATEA, which does a context aware interpretation and builds a discourse model. The discourse model is then sent to the action manager, which consults the discourse model and the domain database to make decisions and send communicative system acts. These acts are sent back to GALATEA, as well as to a generator. Thus, GALATEA models communicative acts both from the user and the system; ellipses, anaphora and grounding status are handled and modelled in the same way for all communicative acts. The action manager may also make changes to the discourse model, for example, if an error is detected, and send it back to GALATEA.

From the generator, the textual representation of the system's communicative act, enriched with prosodic markup, is sent to a speech synthesiser (TTS).

GALATEA is fairly generic – a set of rules and semantic mappings (represented in XML) are written for the specific application. The action manager, on the other hand, is highly domain dependent. However, much of the work that a typical dialogue manager has to do, such as ellipsis and anaphora resolution, is already resolved by GALATEA.

3.2 Semantic Representations

Semantic descriptions are consistently represented as rooted unordered trees of semantic concepts. Nodes in the tree represent objects, relations, properties and attribute-value pairs. Such structures are very flexible and can be used to represent deep semantic structures, such as nested feature structures, as well as simple forms, depending on the requirements of the domain. By using tree matching, similar to Kilpelainen (1992), a pattern tree can be used to search for instances in a given target tree. Thus, larger semantic structures can form databases which may be searched. It is also possible to include variables in a pattern tree for specifying constraints and extracting matching nodes, as well as using special pattern nodes for negation, etc.

The semantic tree structures in HIGGINS, including the database, are represented in XML, using a schema that is specific for the domain. Figure 3 shows an example: an abstract representation of a wooden building. Figure 4 shows how the same structure can be visualised graphically as a tree structure using XSLT and XHTML. The database in the HIGGINS navigation domain is a large XML structure containing all landmarks and their properties, as well as a set of

```
<object id="$id4">
  <properties>
    <type>
      <value>building</value>
    </type>
    <material>
      <value>wood</value>
    </material>
  </properties>
</object>
```

Figure 3. An abstract semantic representation of a wooden building in XML.

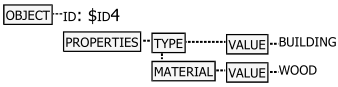


Figure 4. The same structure as in Figure 3, visualised graphically.

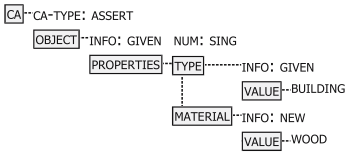


Figure 5. The semantic representation of the utterance "the building is made of wood".

possible user positions and how they relate to the landmarks. All objects in the database have IDs. The XML in Figure 3 could be used as a pattern to search the database. Values starting with a dollar sign – `id4` in the example – are interpreted as variables. The result of this search would be a list of all possible bindings of variable `id4`, i.e., a list of the IDs of all the wooden buildings in the database.

The semantic representations may be enhanced with "meta-information", such as confidence scores, communicative acts, and if information is new or given. Figure 5 shows the representation of the utterance "the building is made of wood". The structure tells us that this is a communicative act (CA) of the type ASSERT, that the object is singular (SING), and that the object and type are GIVEN information but the material NEW. This meta-information is needed for representing utterances, but is not contained in the database. By removing meta-information, the structure can be transformed to a database search pattern, like the one in Figure 4, in order to find possible referents to the object denoted in the utterance.

These structures make it fairly straightforward to represent the semantics of verbs, relations, etc. They are just represented as single nodes or tree

fragments. Tree structures may also be unified. A semantic template is used to specify how the nodes may be structured, to guide the unification. The template also makes it possible to unify structures starting at different levels in the tree.

3.3 Pickering: The Semantic Interpreter

The interpreter developed within HIGGINS is called PICKERING and is implemented in Oz (http://www.mozart-oz.org/). The grammar used to parse recognition results is based on context-free grammars, with some modifications. Grammar rules are enhanced with semantic rules for generating the kind of semantic trees described above. PICKERING can automatically make exceptions from the syntax given in the grammar by handling insertions and non-agreement inside phrases and by combining non-continuous phrases. While deviations from the grammar are allowed by PICKERING, they are taken into account when choosing the best interpretation. Since PICKERING has access to the semantic results, it can automatically filter out semantically equivalent solutions by using tree comparison. For a more detailed description of PICKERING, see Skantze and Edlund (2004b).

To make error handling on the concept level possible, PICKERING also automatically transfers word confidence scores into the semantic trees. The semantic template used for unification can be marked with slots for confidence scores. The confidence scores for the words that are involved in creating a node with such a slot are then averaged to compute a confidence score for the node. An example semantic result with concept confidence scores is shown in Figure 6.

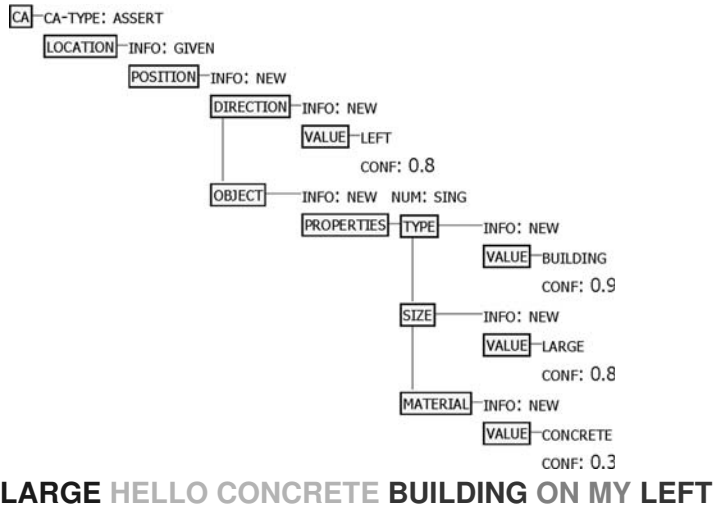


Figure 6. The semantic result from Pickering, interpreting the first part of U.4 in Table 1, with concept confidence scores.

Insertions, such as "hello" in the example, should ideally lower the confidence score for the concepts involved in the phrase, but they are not considered in the current implementation.

3.4 The Action Manager

The discourse model can be compared with the information state used in TrindiKit (Larsson and Traum, 2000). However, the discourse model only contains information about what has been said, not the system's plans or agenda – this is modelled by the action manager. The action manager makes decisions based on a fairly simple decision algorithm, similar to a decision tree, which is traversed each time the discourse model gets updated. For example, after U.2 in Table 1, the system first checks if there are any concepts that should be grounded, and chooses to display understanding of "an ATM". It then checks the discourse model for the user's goal, and finds that it is known. The next thing to check is the user's position, and since there is no information on that in the discourse model, the action manager poses an open request on the user's position (S.3).

The notion of "issues" is central in the "issue-based approach" to dialogue management proposed by Larsson (2002). In this approach, the system keeps track of which issues are raised and when they are resolved or rejected. In the domain considered here, we could say that an issue has been raised for example when the system requests the user's position. However, we have not found the explicit representation of such issues necessary for managing this domain using the approach presented in this chapter. Actually, it would be quite problematic to model issues in this domain, since it may often be hard to determine when issues are resolved or rejected. Consider the following example, where the system needs more information about the user's position:

(3) S.1 Can you see a tree in front of you?
U.2 I have a red building on my left.

In this example, the user does not directly answer the question. However, the system may now find out that it has enough information to continue with route directions. Whether the "issue" raised by the first question is resolved or not does not matter.

4. Galatea: The Discourse Modeller

The discourse modeller developed within HIGGINS and introduced here is called GALATEA and is also implemented in Oz. The discourse model consists of two lists: a **ca-list** and an **entity list**. The CA-list is a list of past communicative acts in chronological order, with the most recent act first.

The entity list is a list of entities mentioned in the discourse, with the most recently mentioned entity first. The discourse model is represented in XML.

The discourse modeller has three main tasks:

- Resolve ellipses by transforming them into full propositions, based on the CA-list.
- Resolve anaphora by extracting entities from the CAs and integrating them into the entity list.
- When new CAs are added to the model, grounding status is added to nodes in the semantic representation, i.e., information about who added the concept to the model, in which turn, and how confident the system is in the concept. This information is also transferred to the entity list.

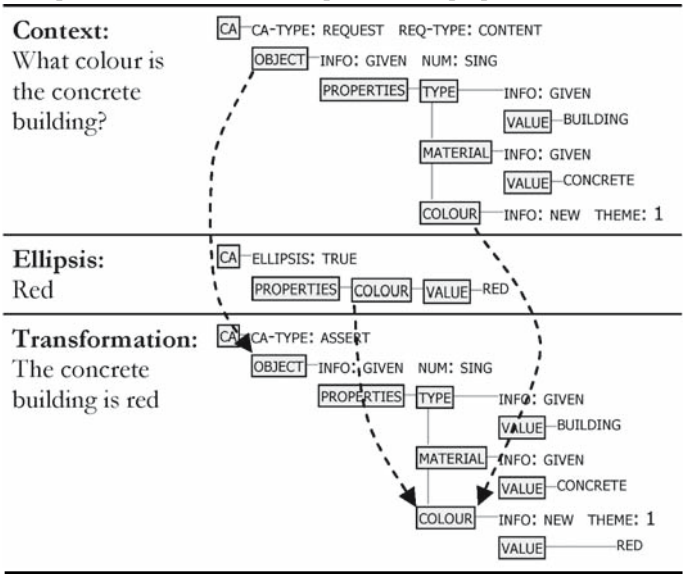
4.1 Ellipsis Resolution

GALATEA resolves ellipses by transforming them into full propositions. To do this, domain-dependent transformation rules are used that transform communicative acts based on previous acts, similar to Carbonell (1983). Each rule has semantic preconditions for the current elliptical CA and the previous CAs, and a transformation description. The preconditions are formulated as semantic pattern trees that are matched against the target CAs. Each rule is applied in order, if the matching and transformation is successful, the algorithm restarts with the transformed CA, until no more transformations can be done. Thus, a cascade of rules may be applied. The rules are written in XML, but will not be explained in more detail here.

Table 2 exemplifies a transformation based on a rule that handles all answers to wh-requests, which are called content-requests here. The preconditions for this rule is that the new CA is an ellipsis, and that there is a content-request in the CA-list with a requested node marked with THEME:1. The transformation description states that the ellipsis should be replaced by a new CA of type ASSERT, where the top node in the request is copied and the THEME node is unified with the first node that can be unified in the ellipsis – in this case the COLOUR node. If the unification fails, the rule is not applied.

Transformation rules may also be used to interpret erroneous ASR results in context, where PICKERING may have identified some fragment from a full proposition. It is, of course, also possible to transform sentential ellipses such as U.8 in Table 1, as well as non-elliptical CAs that are dependent on the context for their interpretation. Each rule has a fairly generic purpose. Currently, about 10 different transformation rules are used for the navigation domain.

Table 2. Example transformation of an ellipsis into full proposition.



4.2 Anaphora Resolution

GALATEA has no access to the domain database. Thus, it cannot map entities in the discourse to real objects in the world. Instead, it keeps a list of entities that are mentioned, e.g., "a large building", in the discourse and assigns variable IDs to them. The action manager may then use the entities in the discourse model as patterns and make a database search to find possible referents, i.e., bindings to the entity id variables (as described on page 165).

When semantic structures are created in PICKERING, they are marked with given/new status, based on definiteness and sentence structure. Some parts may be given and some new, for example when asserting new information about a given object. See Figure 5 for an example. After a CA has been transformed and added to the CA-list, entities are extracted and added to the entity list. Each time an entity marked as new is added to the entity list, it is placed on top. If an entity marked as given is added, i.e., an anaphor, the entity list is searched from top to bottom for an antecedent. The nodes marked as given in the entity to be added are used as a search pattern and the potential antecedents as target, and a pattern match is performed. If an antecedent is found, it is moved to the first position in the entity list and unified with the added entity. If no antecedent is found, the added entity is treated as new and simply placed first. The entity list represents unified asserted information about entities – not the structure of the utterances they were extracted from. Therefore, information about theme and

given/new status is removed before integrating an entity. Also, nodes that only denote requested concepts in a question are removed.

Since assertions about entities are unified in the entity list, it is possible to refer to an entity using a description that has not been used before to refer to that entity. For example, there is a reference in utterance S.11, in Table 1, to "the red building". There is no entity directly referred to in this way before, but the entity list will contain one after U.6.

Before entities are added to the entity list, some early error detection is also done on the concept level – concepts with low confidence are filtered out. These concepts may be added later in the dialogue if they are clarified (which will be described later on in this chapter). This means that the entity list will contain unified information about entities in which the system has relatively high confidence. Thus, the entity list could also be viewed as the system's model of the common ground.

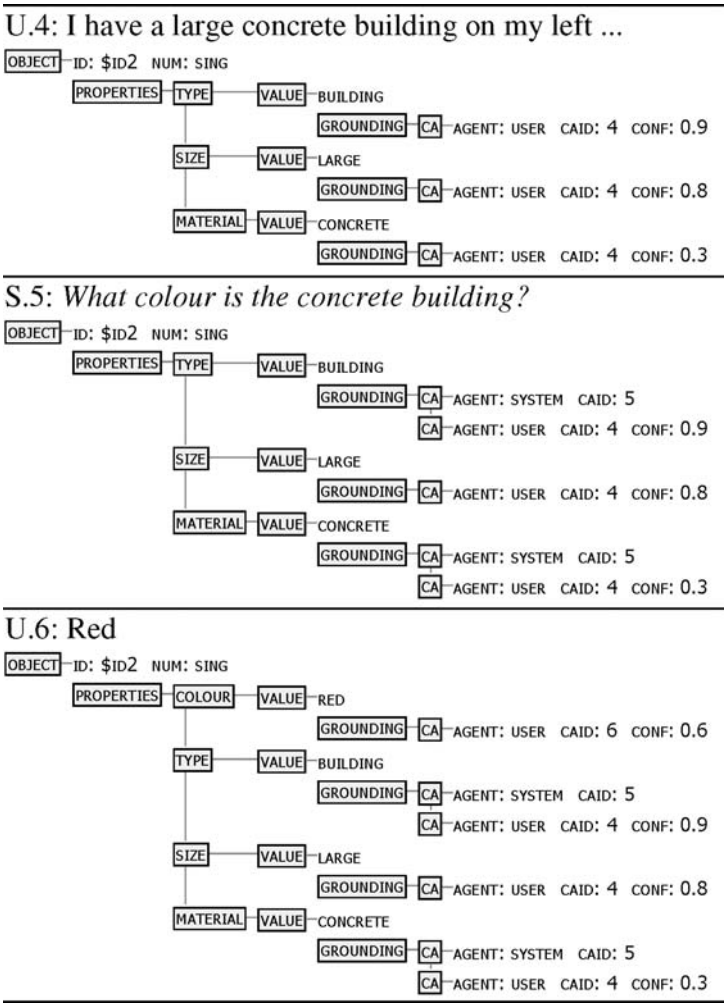
The entity list may also be used by the action manager to select an appropriate referring expression for an entity, such as S.5 and S.11 in Table 1. If the entity is on top of the list, a simple pronoun may be used (unless the entity needs more grounding, which will be described later on in this chapter). If there are other entities above it, the system may use a more elaborate definite noun phrase.

4.3 Grounding Status

Handling anaphora and ellipses can be regarded as necessary basic capabilities of a discourse modeller. The approaches to these issues presented here are indeed quite simplistic and there are much more elaborate accounts (see, e.g. Schlangen 2003 for ellipsis resolution). However, the approach taken here is to model how ellipses and anaphora affect the grounding process. To do this, GALATEA tracks the grounding status of concepts. The grounding status is information about who added the concept to the model, in which turn and how confident the system is in the concept. Since the same concept may be mentioned several times, the grounding status is represented as a list of grounding data. This way, the system may model grounding information over time. This information may then be consulted for various error handling strategies, which is described in the next section. The grounding status can be compared with the "contextual functions" used in Heisterkamp and McGlashan (1996), and the "discourse pegs" used in McTear et al. (2005), that are used to keep track of the system's belief in what has been said.

The grounding information is added before resolving ellipses and anaphora. This ensures that only concepts that were part of the original utterance are grounded, not those that are added in the ellipsis resolution. In the semantic template used for unification, places where grounding information should be

Table 3. How the grounding status for the entity that represents the concrete building gets updated. Note that the COLOUR node is not included until U.6, since it is only requested in S.5.



added are marked. An example of how grounding is updated is provided in Table 3. As can be seen in the table, each GROUNDING tag contains a list of CA-tags, which represent communicative acts in which the concept was grounded. Each such tag contains information on the speaker (AGENT), the turn (CAID) and, if applicable, the concept confidence score (CONF), taken from the parse result, as exemplified in Figure 6.

5. Error Handling in Higgins

In this section, the techniques used in GALATEA and HIGGINS, as described above, will be related to the error handling issues discussed in Section 2.

5.1 Early Error Detection

Early error detection is done in several steps in HIGGINS. First, PICKERING ignores words that appear to be insertions inside phrases (like "hello" in 6), and assigns confidence scores to concepts. Second, as entities are extracted and added to the entity list in GALATEA, concepts with low confidence scores are filtered out, and possibly clarified.

As described above, the system's confidence in the concepts is then dynamically modelled with grounding status. Currently, a very simple distinction is made between high and low grounding status. If a concept is mentioned by the system, it has a high grounding status. If it is only mentioned by the user, the highest confidence score is compared against a threshold to determine if the grounding status is high or low.

5.2 Display of Understanding

The entity list may be used by the action manager to display positive evidence of understanding, by searching for concepts with low grounding status. This may be done as a separate communicative act, as in S.3 in Table 1, or integrated in another act, as in S.5. Since GALATEA also models the system's actions, those concepts will then have a high grounding status.

Every time the system refers to a given entity, appropriate integrated display of understanding is automatically done. To construct a referring expression to an entity that is on the entity list, the action manager simply makes a copy of the entity and removes all concepts with high grounding status. This ensures that the concepts with low grounding status will get a high grounding status. An example is shown in Table 3. When the system needs to ask a question on the colour of the building, it copies the entity and removes the concept LARGE, since it has a high grounding status, based on the confidence score. The TYPE concept (BUILDING) is not removed, since it is needed for a valid referring expression – otherwise the system would say "what colour is the concrete".

5.3 Fragmentary Clarification

The latest CA in the discourse model may be searched for concepts, or trees of concepts, with low grounding status. As mentioned previously, these concepts are not transferred to the entity list. The action manager may then embed such a fragment in a CA of type request and send it to GALATEA and the generator. The generator will make a surface realisation of a fragmentary clarification

request (with prosodic markup) and send it to the TTS. When GALATEA receives this elliptical CA, it will be transformed into a full yes/no request. This way, subsequent reactions to this request will be interpreted correctly, while only the concepts that are actually realised in the ellipsis will get an updated grounding status. Table 4 shows how a clarification dialogue with elliptical communicative acts is interpreted by GALATEA.

Negations and confirmations are represented with POLARITY nodes that are attached to concepts. This makes it easy to represent and integrate "yes" and "no" answers, as well as adverbial negations. Figure 7 shows the resulting entity in the entity list after the dialogue in Table 4. As can be seen, the negative answer is kept in the model. This is useful when constraining possible user locations, since the POLARITY nodes are taken into account when doing tree pattern matching. In Figure 7, the concept RED has been grounded and negated after the clarification. Like all requests, clarification requests do not need to be answered. If the clarification request would not have been answered, there would be no information about the concept RED for this entity. This is also true if the user would have answered just "green", in which case the entity would have the concept GREEN, but no information on the concept RED.

A concept may have several POLARITY nodes with different polarities and the POLARITY nodes may also have grounding status, as can be seen in the example. This makes it possible for one participant to confirm something while

Table 4. How a clarification dialogue is interpreted by GALATEA.

Turn	*Utterance*		*After ellipsis resolution*
U.1	I have a red building on my left		[same]
S.2	red?	+ U.1 =	is the building red?
U.3	no	+ S.2 =	the building is not red
	green	+ S.2 =	the building is green

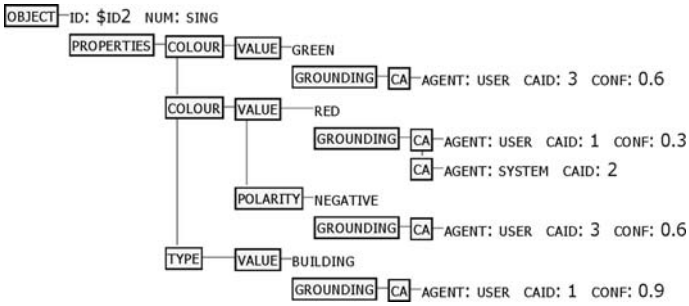


Figure 7. The resulting entity in the entity list after the clarification dialogue.

another participant negates it. Thus, POLARITY nodes can be used to model the "acceptance level" mentioned on page 158. This also means that POLARITY nodes themselves may have a low grounding status, for example if the "no" in Table 4 would get a low confidence score, and need further grounding.

5.4 Late Error Detection

Due to the misrecognition in U.4 (in Table 1), the discourse model will contain an error. However, after turn U.10, the system discovers that there is no place where the user can be. It may now use the grounding status in the discourse model to look for errors. The only concept with a low grounding status is shown in Figure 8. This concept may be removed, and the system will still have the information that the user can see a tree somewhere.

It is also possible to remove information that is associated with a specific turn, by looking at the CAID attribute in the grounding status. The most obvious case is when the system has just grounded some concepts and the user signals a problem. The concepts associated with the system's previous turn may then be removed. For example, if the user had signalled a problem after S.5 in Table 3, the system could remove the concepts CONCRETE and BUILDING, or add negative POLARITY nodes to them. The model would still contain the fact that the user has something large on her left.

Since the model also contains information about what the user has grounded, it is possible to detect cases where the user misunderstands the system. For example, the user never displays any understanding of the concepts WOOD and BUILDING in U.8 in Table 1, which can be detected in the discourse model.

5.5 Choosing Strategy

Given these different error handling strategies the discourse model allows, the question is how to choose strategy. Generally, low confidence scores lead to clarification requests, mid confidences scores lead to display of understanding, and high confidence scores lead to no grounding actions. Since the choice of strategy is done in the action manager, it is possible to make a task-related choice, i. e., to have different confidence thresholds for different tasks, depending on the consequence of misunderstanding, as discussed on page 162. For example, when the user asserts the goal, as in S.1 in Table 1, the system has a

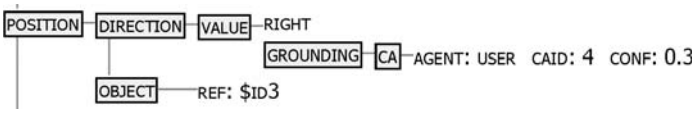


Figure 8. A concept with low grounding status is detected.

much higher threshold for not displaying understanding. When the user asserts her position in U.4, the system instead chooses to defer the handling of the low confidence score in the concept RIGHT.

6. Evaluation

To evaluate the performance of the HIGGINS system in general and GALATEA in particular, the complete system was tested with naive subjects using the system in a laboratory setting with given scenarios. The analysis of the results should be viewed as a proof of concept, to confirm that the system can interact with naive users and perform reasonably well. No comparative evaluation, involving other systems or different settings, is made. It is not possible to evaluate all aspects of the techniques discussed in this chapter; the analysis of the results will focus on robustness, ellipsis resolution and fragmentary clarification.

6.1 Setting

6.1.1 Subjects. Sixteen subjects participated in the evaluation, all native speakers of Swedish. They were 7 women and 9 men, ranging in age from 24 to 63 years (38 on average). Four of the subjects had some experience of speech technology, but no experience of dialogue system design.

6.1.2 Procedure. Each subject was given the task of finding the way to a given goal in a virtual city, by talking to a computer to get route directions. They were told that they needed to tell the computer where they wanted to go, that the computer had no knowledge of their current position, but that the computer had a very detailed map of the city, and it was able to locate them if they described their surroundings. The subjects were not given any information about what kind of expressions the system could handle, or the level of detail of the system's knowledge. Four subsequent scenarios (goals) were given to each subject, resulting in a total of 64 dialogues. During the interaction, the experiment conductor was sitting in another room, overlooking the interaction through a window, and did not answer any questions from the subject. The subject was sitting in front of a computer screen, where the virtual city was displayed, and used a mouse to walk around. The room was sound-proof and the subject was wearing a headset. For each scenario, there was a time limit of 10 minutes to reach the goal. After each scenario, the subjects filled out a questionnaire about their experience of the interaction. However, the results from this survey will not be used in the current analysis.

6.1.3 Data, ASR and TTS. The system was trained and configured mainly based on two different sets of data. First, data from an experiment in a similar domain in which subjects gave route directions to each other

(Skantze, 2005). Second, a data collection where subjects were walking around in the 3D virtual city, describing what they saw (Edlund et al., 2004). In addition to this, data were collected from four pilot sessions.

The collected data was used to write rules for PICKERING and GALATEA and to program the action manager, as well as to train the ASR. An off-the-shelf ASR was used with tri-gram class-based language models, trained on the 1,500 utterances that had been collected. The resulting vocabulary size was approximately 600 words.

For TTS, a diphone Swedish male MBROLA voice was used (Dutoit et al., 1996). The most challenging task for the TTS in this experiment was to produce fragmentary clarification requests, since the interpretation of them is dependent on prosody to a large extent. Since system utterances were dynamically generated, it was not possible to handcraft the pitch curve for each utterance. A commonly described tonal characteristic for questions is overall higher pitch (Hirst and Di Cristo, 1998). Experiments on the prosody of synthesised fragmentary grounding utterances (Edlund et al., 2005) have also shown that a mid or late higher F_0 peak differentiates clarification requests from display of understanding, although there are differences in the interpretation of the request depending on the position of the peak. Since the only available parameters for the TTS were pitch range, speaking rate and pitch base, the pitch range was simply increased for fragmentary clarification requests, as an approximation to these findings.

6.2 Results

6.2.1 Annotation. The dialogues were transcribed and annotated by one annotator. The segmentation of units for assigning features on the "utterance" level is not straightforward. One segmentation was based on the ASR endpoint detector, i.e., each ASR result was assigned a set of features. Another set of features was assigned to each CA, as segmented by the annotator. There was a pretty large discrepancy between these two methods for segmenting "utterances"; some CAs were not detected at all by the ASR, some were split over several ASR results, some ASR results contained several CAs. There were a total of 1,894 ASR results and 2,007 CAs, 1,565 of the ASR results contained only one unsplit CA.

Both ASR results and CAs were annotated based on how well they were understood by the system. Full "understanding" in this context means accurate speech recognition, semantic interpretation, and discourse modelling (ellipsis and anaphora resolution). A common measure for the performance of semantic interpretation is "concept error rate". This measure is easier to use when the result of the interpretation process is a simple feature-value list that represents the semantic meaning of the utterance. In HIGGINS, however, the result is

Table 5. The definitions of the understanding levels used in the annotation of ASR results and CAs.

Und.	*Definition*
FULLUND	All concepts, relevant to the domain and task, are fully understood by the system, including speech recognition, semantic interpretation, and discourse modelling. Note that the full propositional meaning must be understood, i. e., fragmentary utterances have to be correctly transformed into full propositions, and anaphoric expressions correctly resolved. This includes cases where the system asks for clarification of some of the concepts in which it lacks confidence.
PARTUND	Some (but not all) concepts are fully understood, according to the definition above. This includes cases where just a fragment of the utterance is interpreted, but the action manager uses this to pose a relevant question.
MISUND	Some (or all) concepts in the interpretation of the utterance are incorrect. This includes cases where the system asks for clarification on the incorrect concepts.
NONUND	No concepts are understood. This includes cases where PICKERING might correctly interpret a fragment of the utterance (or a complete fragmentary utterance), but GALATEA fails to resolve the ellipsis, and the action manager fails to use the fragment to pose a relevant question.
NONHEAR	The CA is not detected at all. (Not applicable to ASR results.)

an updated discourse model with tree-structured concepts, including identified referents and enriched fragments. It is not easy to handcraft a target discourse model to compare with for each utterance. Neither is it straightforward to measure the degree to which these structures are similar: the individual concepts should be correct, but they should also be correctly structured. To make the analysis more straightforward and the results easier to understand, five different levels of understanding were defined. These levels are described in Table 5.

The CAs were also annotated based on the form and function of the utterance. However, the scheme for this will not be described in detail here. An excerpt from the collected data with the different understanding levels for each CA is shown in Table 6.

6.2.2 General results. Of the 64 tasks, 50 were completed within the time limit of 10 minutes. The tasks that succeeded took 4.3 minutes on average.

On average, there were 3.9 words per CA. This relatively low figure can be explained by the frequent use of short acknowledgements (like "yes" and "ok") during the direction giving phases of the dialogues, as well as the high number of fragmentary utterances. Of all words spoken by the subjects, 1.6% were out-of-vocabulary, not counting truncated words.

The ASR results had an average word error rate (WER) of 23.6%. Table 7 shows the distribution of understanding levels for ASR results and CAs, as well

Table 6. A dialogue excerpt from the evaluation data, translated from Swedish. The third column shows ASR results with the same representation as in Table 1. The fourth column shows the understanding level that was assigned to the CA by the annotator.

Turn	*Transcription*	*ASR result*	CA *und.*
U.1	I– I want to see a bus stop.	YES I WANT TO GO TO A BUS STOP	FULLUND
S.2	*Ok, where are you?*		
U.3	Eh, I stand in front of a wooden house with three storeys.	NOW IS I STAND IN FRONT OF A WOODEN HOUSE WITH THREE STOREYS	FULLUND
S.4	*Three?*		
U.5	Yes.	YES	FULLUND
S.6	*OK, describe something more.*		
U.7	I see a sign that reads ...	I SEE A SIGN THAT READS	PARTUND (with U.7b)
S.8	*OK ...*		
U.7b	Må– Måbär street.	[undetected]	(part of U.7)
S.9	*Please wait ...*		
S.10	*Can you see a brick building on your left?*		
U.12	No,	NOW ON MY RIGHT	NONUND
U.13	on my right.		PARTUND
S.14	*What do you see on your right?*		
U.15	A brick building with three storeys.	A BRICK BUILDING WITH THREE STOREYS	FULLUND
S.16	*OK, a brick building with three storeys, hm ...*		
U.17	That is correct.	[undetected]	NONHEAR
S.18	*I think I know where you are. You should see an orange building. Position yourself between it and the three storey building made of wood.*		
U.19	OK.	OK	FULLUND

as for the ASR results containing just one CA, and for the subset of these that had a WER of 0%.

The rightmost column in Table 7 reflects the performance of PICKERING and GALATEA, i.e., how well the rules written to handle the training data generalised to new data. Considering the limited "training data", 92.9% FULLUND should be considered as promising performance.

The third column (CAs) shows that 8.0% of the CAs were not detected by the system at all. This is partly explained by the fact that there were a lot of

Table 7. The number of instances and distribution of understanding for ASR results and CAs. The fourth column shows ASR results which contain one unsplit CA. The fifth column shows the subset of these that had a WER of 0%. The understanding levels are defined in Table 5.

	ASR results	CA*s*	*ASR res.* = CA	*ASR res.* = CA 0% WER
Instances	1894	2007	1565	1033
FULLUND	65.4%	66.2%	73.8%	92.9%
PARTUND	9.2%	5.8%	5.0%	1.6%
MISUND	10.1%	8.3%	9.5%	1.5%
NONUND	15.3%	11.8%	11.7%	3.9%
NONHEAR		8.0%		

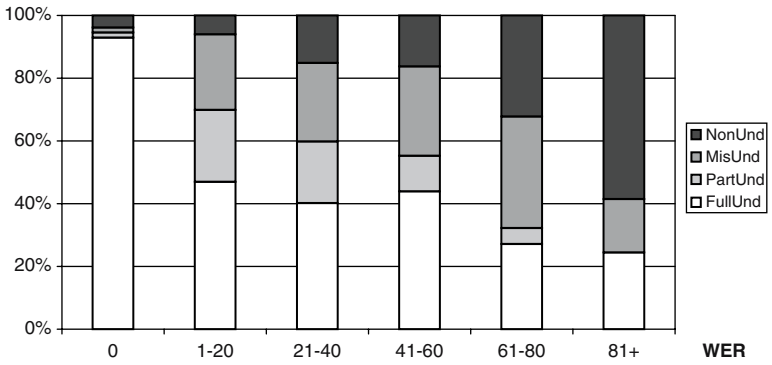


Figure 9. The distribution of understanding depending on the WER of the ASR result (rounded up).

feedback utterances from the user (such as "mhm") after the system's display of understanding, which did not have enough intensity to trigger the endpoint detection. Another explanation is that the ASR was not allowed to deliver any results while the system was talking. It was never shut-off, but if a speech endpoint was detected and a system utterance was playing, no result was delivered. This was done to prevent some turn-taking problems that cannot yet be handled. An example of this is U.7b in Table 6, where the utterance ends while S.9 is spoken.

6.2.3 Robustness and early error detection. To study the robustness of PICKERING and GALATEA, the ASR results were divided in WER intervals. The distribution of understanding for these spans is shown in Figure 9.

Table 8. The distribution of understanding for task-related complete recognised CAs of different forms, where WER = 0%.

	Assertion	*Fragment*	*"Ok"*	*"Yes"/"No"*
FULLUND	90.1%	88.6%	100.0%	98.2%
PARTUND	1.6%	8.4%	0.0%	0.0%
MISUND	2.6%	1.8%	0.0%	0.0%
NONUND	5.8%	1.2%	0.0%	1.8%

The figure shows that the introduction of ASR errors immediately decreases the proportion of FULLUND by about 40%. Roughly half of this performance drop consists of some deleted concepts (PARTUND) and half by inserted concepts (MISUND). Interestingly, as the WER increases, the performance degrades gracefully up to a WER as high as 60%. Even with a WER above this, the proportion of misunderstandings seems to be stable, indicating an acceptable early error detection performance.

6.2.4 Ellipsis resolution. To find out how well GALATEA manages to resolve ellipses, correctly recognised task-related complete CAs were grouped based on their form. Table 8 shows the distribution of understanding for some relevant forms. In this context, the form "fragment" includes adjectives, nouns, nominal phrases, propositional phrases, etc.

As Table 8 shows, fragmentary utterances – where ellipsis resolution is needed for full understanding – were almost as successful as assertions. For fragments, there was a larger proportion of partial understandings. These are utterances like U.13 in Table 6, which can be tricky to resolve correctly, but may be used by the system to ask a request like S.14, resulting in a partial understanding. Acknowledgements and yes/no-utterances also need ellipsis resolution. But, as the table shows, this is an easier task.

6.2.5 Fragmentary clarification. There was a total of 94 fragmentary clarification requests in the data. Of these, 68.1% concerned concepts that were correctly understood by the system, and 31.9% concerned concepts that were incorrect, i.e., the request followed a misunderstanding. The function and type of the user CA following the request were used to group the user reactions to the requests based on six different types. Table 9 shows the distribution of these types.

Considering the large proportion of reactions that either ignore the request or signal non-understanding, fragmentary clarification requests seem to be hard for the users to understand. This is not very surprising, considering the fact that this form of clarification is not very common in dialogue with computers,

Table 9. Immediate user reaction to fragmentary clarification requests. The second column shows the distribution for cases where the clarified concepts were correctly understood by the system, and the third column cases where the clarified concepts were incorrect (i.e., after a misunderstanding).

Reaction	*Correct*	*Incorrect*
"Yes"-answer	59.4%	0.0%
"No"-answer	3.1%	40.0%
A correction or elaboration, in the form of a fragment or assertion, as an answer to the request.	10.9%	20.0%
An utterance that relates to the request, but does not answer it.	4.7%	6.7%
A signal of non-understanding (such as "what?").	9.4%	6.7%
The request is ignored.	12.5%	26.7%

and that no elaborated model for the prosody of these utterances was used. Fragmentary clarification requests seem to be even harder to understand after misunderstandings. This is of course due to the fact that such requests may not make sense to the user in some situations. For example, the utterance "red?" after "I want to go to a bus stop" may seem inadequate.

However, it is also possible that when users ignore a clarification request after a correct recognition, it should be interpreted as a "silent consent". Purver (2004) found that clarification requests in human-human dialogue are very often not answered (in 17–39% of the cases). Thus, the assumption taken here – that the clarification request must be confirmed for the concepts to be considered as correct – may be implausible.

Of the reactions that imply that the request was understood correctly by the user, it is interesting to note that far from all started with simple "yes" or "no" answers. Especially after misunderstandings, the user often corrects the system without starting with "no, ...". For many reactions, it is not obvious if they should be interpreted as answers to the preceding request, even if they relate to it. This supports the previously discussed assumption that clarification requests should not be treated as some sort of "sub-dialogue", but rather as a signal from the system that it lacks understanding, in the form of a request that makes a fragmentary response possible to resolve. An interesting observation is the existence of some "no" answers after clarification requests based on correct understanding. These are cases where the user changes her mind, possibly because the system's request is interpreted as if it was doubting the correctness of the user's description.

Table 10 shows the distribution of understanding of the user reactions. For requests based on correct interpretations, the understanding of the responses seems to be similar to that of CAs in general (compare with Table 7). However,

Table 10. The system's understanding of the user reactions to fragmentary clarification requests. The second column shows the distribution for cases where the clarified concepts were correctly understood by the system, and the third column cases where the clarified concepts were incorrect (i.e., after a misunderstanding).

Understanding	*Correct*	*Incorrect*
FULLUND	67.2%	43.3%
PARTUND	3.1%	13.3%
MISUND	6.3%	6.7%
NONUND	15.6%	30.0%
NONHEAR	7.8%	6.7%

the performance is poorer for responses to requests based on misunderstandings, reflecting the fact that these were more unpredictable.

6.2.6 Late error detection. The log files from the action manager showed that there was a total of 78 cases where there was no place where the user could be when matching the discourse model against the database – indicating that a misunderstanding had occurred. In 45 of these cases, removing concepts with low grounding status made it possible for the system to continue positioning the user. This looks promising, but it does not tell us how many of the correct or incorrect concepts actually were removed. Future work will focus on answering this question, as well as finding methods for improving late error detection.

7. Conclusions and Future Work

In this chapter, the discourse modeller GALATEA has been presented. It models the grounding status of concepts mentioned during the course of the discourse by storing confidence scores for individual concepts, together with information about when the concepts have been mentioned and by whom. It has been shown how this information may be used by an action manager for display of understanding, clarification requests, and late error detection, all on the concept level. By integrating this with ellipsis and anaphora resolution, the system may track how individual concepts may be grounded, depending on the surface realisation of utterances.

An evaluation of the system showed that the performance of GALATEA and the rest of the HIGGINS system looks promising, not only when utterances are correctly recognised, but also when ASR errors are introduced. Further analysis of the results will also take the subjects' answers from the questionnaires into account to draw conclusions on how the error handling techniques affect user satisfaction.

Fragmentary clarification requests may contribute to a more natural and efficient dialogue. However, there has previously not been much study on the use of such utterances in spoken dialogue systems interacting with real users. The results from the evaluation showed that users responded to fragmentary clarification requests in a number of different ways. When the requests were correctly understood by the users, the user responses to them seemed to be correctly understood by the system. However, there seemed to be a lot of cases where the requests were not understood by the users. This is partly due to the limited prosodic model that was used. We are currently investigating how prosodic features affect the pragmatic consequences of these fragmentary clarification requests, and working to improve such a model (Skantze et al., 2006). Further analysis of the results will also look at user responses after display of understanding, and compare them to reactions to clarification requests.

This chapter has focused on how to model the way all utterances may provide evidence of understanding, or feedback, while simultaneously operating on the domain level. Clarification requests, for example, are treated in the same way as other requests and are not seen as special types of utterances. There is also a limited set of domain-unrelated utterance-level feedback utterances, not specifying concepts, such as "pardon?", that have not been discussed here. For a discussion of the use of such utterances in dialogue systems, see, for example, Larsson (2003).

Currently, the thresholds used for different strategies and tasks have been manually tuned. The notion of high and low grounding status is also very simplistic. An important topic for future research is to investigate machine learning techniques for choosing strategies, as well as for late error detection, similar to the "belief updating" approach in Bohus and Rudnicky (2005). To improve such classification, the grounding status could be enriched with more information on how the concept was grounded, such as prosodic and lexical information.

An important question is to what extent the techniques described in this chapter may apply to other domains. GALATEA, as well as other HIGGINS components, is currently being tested in CONNECTOR, a dialogue system acting as an automatic switchboard and secretary (Edlund and Hjalmarsson, 2005). CONNECTOR is part of the EU-funded CHIL-project (http://chil.server.de/) – a project investigating automatic tracking and support of interactions in meeting rooms. The HIGGINS components have also been used in the conversational training game DEAL (Hjalmarsson et al., 2007). DEAL is a dialogue system for second language learners, where the user talks to an embodied conversational agent in a flea market domain in order to train conversational skills.

Acknowledgements This research was carried out at the Centre for Speech Technology, a competence centre at KTH, supported by VINNOVA (The Swedish Agency for Innovation Systems), KTH and participating Swedish companies and organisations. The author would like to thank the other participants in the HIGGINS project, Jens Edlund and Rolf Carlson, for valuable ideas and discussion. Thanks also to Johan Boye and David House for commenting on the text.

References

Allen, J. F., Ferguson, G., and Stent, A. (2001). An Architecture for More Realistic Conversational Systems. In *Proceedings of Intelligent User Interfaces (IUI)*, pages 1–8, Santa Fe.

Allen, J. F., Miller, B. W., Ringger, E. K., and Sikorski, T. (1996). Robust Understanding in a Dialogue System. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 62–70, Santa Cruz.

Allwood, J., Nivre, J., and Ahlsen, E. (1992). On the Semantics and Pragmatics of Linguistic Feedback. *Journal of Semantics*, 9(1):1–26.

Anderson, A., Bader, M., Bard, E., Boyle, E., Doherty, G., Garrod, S., Isard, S., Kowtko, J., McAllister, J., Miller, J., Sotillo, C., Thompson, H., and Weinert, R. (1991). The HCRC Map Task Corpus. *Language and Speech*, 34(4):351–366.

Baus, J., Kray, C., Krüger, A., and Wahlster, V. (2002). A Resource-Adaptive Mobile Navigation System. In *Proceedings of International Conference on Intelligent User Interfaces (IUI)*, pages 15–22, San Franscisco.

Bohus, D. and Rudnicky, A. (2005). Constructing Accurate Beliefs in Spoken Dialog Systems. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, San Juan.

Carbonell, J. G. (1983). Discourse Pragmatics and Ellipsis Resolution in Task-Oriented Natural Language Interfaces. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 164–168, Cambridge.

Clark, H. H. (1996). *Using Language*. Cambridge University Press, Cambridge.

Dutoit, T., Pagel, V., Pierret, N., Bataille, F., and van der Vreken, O. (1996). The MBROLA Project: Towards a Set of High-Quality Speech Synthesizers Free of Use for Non-Commercial Purposes. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 1393–1396, Philadelphia.

Edlund, J. and Hjalmarsson, A. (2005). Applications of Distributed Dialogue Systems: The KTH Connector. In *Proceedings of COST278 and ISCA*

Tutorial and Research Workshop (ITRW) on Applied Spoken Language Interaction in Distributed Environments (ASIDE), Aalborg.

Edlund, J., House, D., and Skantze, G. (2005). The Effects of Prosodic Features on the Interpretation of Clarification Ellipses. In *Proceedings of 9th European Conference on Speech Communication and Technology (INTERSPEECH)*, pages 2389–2392, Lisbon.

Edlund, J., Skantze, G., and Carlson, R. (2004). HIGGINS – a Spoken Dialogue System for Investigating Error Handling Techniques. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 229–231, Jeju Island.

Evermann, G. and Woodland, P. C. (2000). Large Vocabulary Decoding and Confidence Estimation Using Word Posterior Probabilities. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2366–2369, Istanbul.

Gabsdil, M. (2003). Clarification in Spoken Dialogue Systems. In *Proceedings of AAAI Spring Symposium on Natural Language Generation in Spoken and Written Dialogue*, pages 28–35, Stanford.

Gabsdil, M. and Bos, J. (2003). Combining Acoustic Confidence Scores with Deep Semantic Analysis for Clarification Dialogues. In *Proceedings of International Workshop on Computational Semantics (IWCS)*, pages 137–150, Tilburg.

Gabsdil, M. and Lemon, O. (2004). Combining Acoustic and Pragmatic Features to Predict Recognition Performance in Spoken Dialogue Systems. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 343–350, Barcelona.

Heisterkamp, P. and McGlashan, S. (1996). Units of Dialogue Management: An Example. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 200–203, Philadelphia.

Hirst, D. and Di Cristo, A. (1998). A Survey of Intonation Systems. In Hirst, D. and Di Cristo, A., editors, *Intonation Systems*, pages 1–45. Cambridge University Press, Cambridge.

Hirst, G., McRoy, S., Heeman, P., Edmonds, P., and Horton, D. (1994). Repairing Conversational Misunderstandings and Non-Understandings. *Speech Communication*, 15(3-4):213–230.

Hjalmarsson, A., Wik, P., and Brusk, J. (2007). Computer Assisted Conversation Training for Second Language Learners. In *Proceedings of Fonetik*, Stockholm.

Kilpelainen, P. (1992). *Tree Matching Problems with Applications to Structured Text Databases*. PhD thesis, Department of Computer Science, University of Helsinki, Finland.

Knight, S., Gorrell, G., Rayner, M., Milward, D., Koeling, R., and Lewin, I. (2001). Comparing Grammar-Based and Robust Approaches to Speech

Understanding: A Case Study. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 1779–1882, Aalborg.

Krahmer, E., Swerts, M., Theune, M., and Weegels, M. (2001). Error Detection in Spoken Human-Machine Interaction. *International Journal of Speech Technology*, 4(1):19–29.

Larsson, S. (2002). *Issue-Based Dialogue Management*. PhD thesis, Göteborg University, Sweden.

Larsson, S. (2003). Interactive Communication Management in an Issue-Based Dialogue System. In *Proceedings of 7th Workshop on the Semantics and Pragmatics of Dialogue (DiaBruck)*, pages 75–82, Saarland University, Saarbrücken.

Larsson, S. and Traum, D. R. (2000). Information State and Dialogue Management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering: Special Issue on Best Practice in Spoken Language Dialogue Systems Engineering*, pages 323–340.

McTear, M., O'Neill, I., Hanna, P., and Liu, X. (2005). Handling Errors and Determining Confirmation Strategies - An Object-Based Approach. *Speech Communication*, 45(3):249–269.

Pfleger, N., Engel, R., and Alexandersson, J. (2003). Robust Multimodal Discourse Processing. In *Proceedings of 7th Workshop on the Semantics and Pragmatics of Dialogue (DiaBruck)*, pages 107–114, Saarland University, Saarbrücken.

Purver, M. (2004). *The Theory and Use of Clarification Requests in Dialogue*. PhD thesis, University of London, UK. http://www.speech.kth.se/prod/publications/files/100068.pdf.

Rieser, V. (2004). Confidence-Based Fragmentary Clarification on Several Levels for Robust Dialogue Systems. Master's thesis, University of Edinburgh, Edinburgh.

Schegloff, E. A. (1992). Repair after Next Turn: The Last Structurally Provided Defense of Intersubjectivity in Conversation. *American Journal of Sociology*, 97(5):1295–1345.

Schlangen, D. (2003). *A Coherence-Based Approach to the Interpretation of Non-Sentential Utterances in Dialogue*. PhD thesis, School of Informatics, University of Edinburgh, Edinburgh.

Schlangen, D. (2004). Causes and Strategies for Requesting Clarification in Dialogue. In *Proceedings of SIGdial Workshop on Discourse and Dialogue*, pages 136–143, Boston.

Skantze, G. (2005). Exploring Human Error Recovery Strategies: Implications for Spoken Dialogue Systems. *Speech Communication*, 45(3):325–341.

Skantze, G. and Edlund, J. (2004a). Early Error Detection on Word Level. In *Proceedings of COST278 and ISCA Tutorial and Research Workshop (ITRW) on Robustness Issues in Conversational Interaction*, Norwich.

Skantze, G. and Edlund, J. (2004b). Robust Interpretation in the HIGGINS Spoken Dialogue System. In *Proceedings of COST278 and ISCA Tutorial and Research Workshop (ITRW) on Robustness Issues in Conversational Interaction*, Norwich.

Skantze, G., House, D., and Edlund, J. (2006). User Responses to Prosodic Variation on Fragmentary Grounding Utterances in Dialogue. In *Proceedings of the 9th International Conference on Spoken Language Processing (Interspeech 2006 – ICSLP)*, Pittsburgh.

Traum, D. (1994). *A Computational Theory of Grounding in Natural Language Conversation*. Ph.D. thesis, University of Rochester, New York.

Appendix

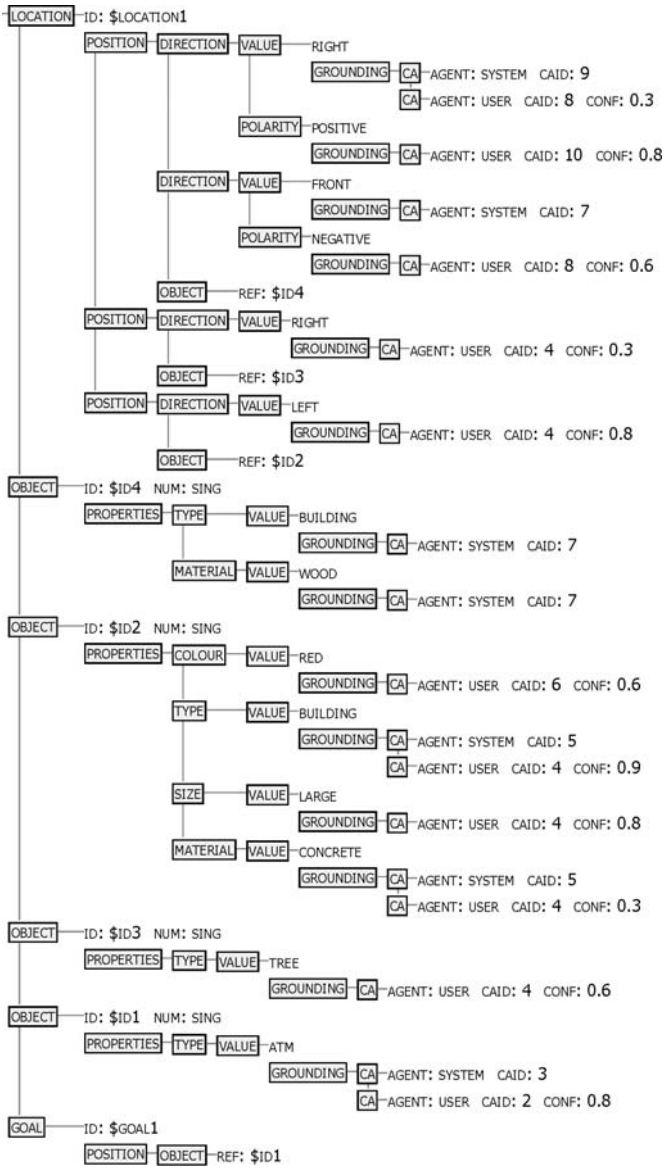


Figure 7.A.1. The complete entity list after turn U.10 in Table 1, before late error detection.

Table 7.A.1. The same dialogue as in Table 1. The third column shows the interpretation of the utterance after ellipsis resolution. Concepts that are grounded, i.e., displayed, in the same turn are written in uppercase. The fourth column shows the error handling strategies that are used.

Turn	*Transcription and ASR result*	*After ellipsis resolution*	*Error handling*
S.1	*Where do you want to go?*	Where do you want to go?	
U.2	To an ATM. **TO AN ATM**	I want to go to an ATM.	
S.3	*OK,* *an ATM.*	OK, You want to go to an ATM.	Separate display of understanding (ATM)
	Can you describe where you are now?	Can you describe where you are now?	
U.4	I have a large concrete building on my left **I HAVE LARGE HELLO CONCRETE BUILDING ON MY LEFT**	I have a LARGE CONCRETE BUILDING on my LEFT	
	and a tree in front of me. **AND TREE ON RIGHT**	I have a TREE on my RIGHT.	
S.5	*What colour is the concrete building?*	What COLOUR is the CONCRETE BUILDING?	Integrated display of understanding (CONCRETE, BUILDING)
U.6	Red. **RED**	The concrete building is RED.	
S.7	*OK,* *can you see a wooden building in front of you?*	OK, can you see a WOODEN BUILDING in FRONT of you?	
U.8	No, **NO** but I have one on my right. **I HAVE ONE ON RIGHT**	I can not (NEGATIVE) see a wooden building in front of me I have a wooden building on my RIGHT.	
S.9	*On your right?*	Do you have a wooden building on your RIGHT?	Clarify RIGHT
U.10	Yes. **YES**	I do (POSITIVE) have a wooden building on my right.	
S.11	*OK,* *I think I know where you are.* *Walk a little bit forward and take a left after the red building.*	OK, I think I know where you are. Walk a little bit FORWARD and take a LEFT AFTER the RED BUILDING.	Late error detection of RIGHT in U.4

Chapter 8

PARTIALLY OBSERVABLE MARKOV DECISION PROCESSES WITH CONTINUOUS OBSERVATIONS FOR DIALOGUE MANAGEMENT

Jason D. Williams
AT&T Labs - Research
Florham Park, NJ, USA
jdw@research.att.com

Pascal Poupart
School of Computer Science, University of Waterloo
Ontario, Canada
ppoupart@cs.uwaterloo.ca

Steve Young
Department of Engineering, University of Cambridge
Cambridge, UK
sjy@cam.ac.uk

Abstract This work shows how a spoken dialogue system can be represented as a Partially Observable Markov Decision Process (POMDP) with composite observations consisting of discrete elements representing dialogue acts and continuous components representing confidence scores. Using a testbed simulated dialogue management problem and recently developed optimisation techniques, we demonstrate that this continuous POMDP can outperform traditional approaches in which confidence score is tracked discretely. Further, we present a method for automatically improving handcrafted dialogue managers by incorporating POMDP belief state monitoring, including confidence score information. Experiments on the test-bed system show significant improvements for several example handcrafted dialogue managers across a range of operating conditions.

Keywords: Spoken dialogue systems; partially observable Markov decision processes; dialogue management; decision theory

L. Dybkjær and W. Minker (eds.), Recent Trends in Discourse and Dialogue, 191–217.

1. Introduction

Dialogue management is a difficult problem for several reasons. First,speech recognition errors are common, corrupting the evidence available to the machine about a user's intentions. Second, users may change their intentions at any point – as a result, the machine must decide whether conflicting evidence has been introduced by a speech recognition error, or by a new user intention. Finally, the machine must make trade-offs between the "cost" of gathering additional information (increasing its certainty of the user's goal, but prolonging the conversation) and the "cost" of committing to an incorrect user goal. That is, the system must perform planning to decide what sequence of actions to take to best achieve the user's goal despite having imperfect information about that goal. For all these reasons, dialogue management can be cast as planning under uncertainty.

In this context, making use of available information about speech recognition accuracy ought to improve the performance of a dialogue manager. One key piece of information typically provided by the automatic speech recognition process is a confidence score, which provides a real-valued estimate of the probability that the recognition hypothesis is correct. In a traditional spoken dialogue system, a confidence score is used to decide whether to accept or reject a speech recognition hypothesis: if a hypothesis has a high confidence score, it is accepted; otherwise it is rejected. More nuanced approaches create confidence buckets which subcategorise the accept category into N "buckets" such as low, medium and high. Confidence bucket information can then be incorporated into the dialogue state, and the dialogue manager can subsequently use this information when choosing actions, for example when deciding whether or not to confirm an element of the dialogue state.

This process is illustrated in the first two columns of Figure 1, which shows a conversation with a spoken dialogue system in the pizza-ordering domain. The first column indicates the words spoken by the user and the machine; the bracketed text shows the (possibly erroneous) results from the speech recognition process, followed by the confidence score. The second column shows how a typical spoken dialogue system might track dialogue state. In the last turn, a speech recognition error is made, and it is unclear how this evidence should be incorporated into the form in column 2 – should the new information replace the old information, or should it be ignored?

In this chapter we consider a different model for dialogue management: a partially observable Markov decision process (POMDP, pronounced "pomdp"). Rather than tracking one explicit dialogue state, a POMDP maintains a probability distribution over all possible dialogue states, called a belief state. As the dialogue progresses, the belief state is updated. This belief state update provides a principled method for interpreting confidence score: intuitively,

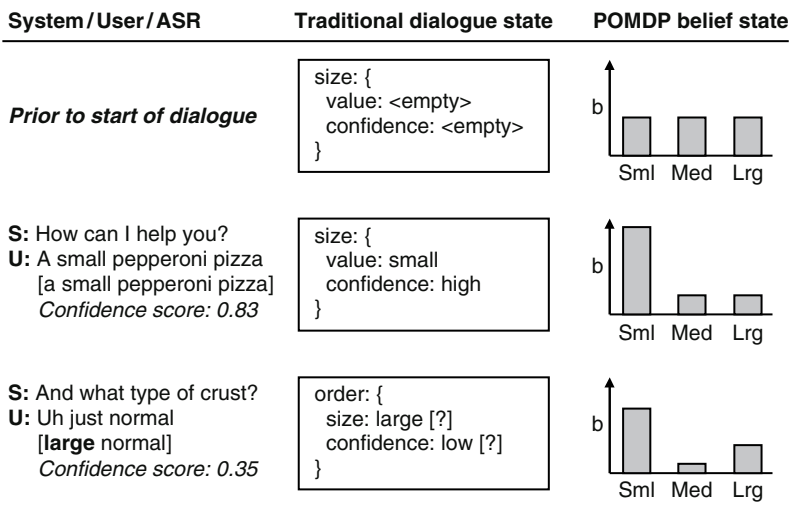


Figure 1. Example conversation with a spoken dialogue system in the pizza-ordering domain. The first column shows the words spoken by the user and the machine. The text in brackets shows the results from the speech recognition process, and the following line shows the resulting confidence score provided by the speech recognition engine. The second column shows how a typical dialogue manager might track a dialogue state, including a "confidence bucket" for the "size" field. The third column shows how the "POMDP belief state" would track the same conversation. Note how the traditional method struggles to account for the conflicting evidence in the last turn, whereas in the POMDP, the confidence score simply scales the magnitude of the update.

the confidence score simply scales the magnitude of the update. This process is illustrated in the third column of Figure 1 – note how the (first) higher-confidence recognition causes a large movement of belief mass, whereas the (second) lower-confidence recognition causes a smaller movement of belief mass.

The goal of this chapter is to explain this process in detail and show it represents significant gains over a traditional "confidence bucket" approach through two central contributions. First, we show how a confidence score can be accounted for exactly in a POMDP-based dialogue manager by treating confidence score as a continuous observation. Using a test-bed simulated dialogue management problem, we show that recent optimisation techniques produce policies which outperform traditional MDP-based approaches across a range of operating conditions.

Second, we show how a handcrafted dialogue manager can be improved automatically by treating it as a POMDP policy. We then show how a confidence score metric can be easily included in this improvement process. We illustrate the method by creating three handcrafted controllers for the test-bed dialogue manager, and show that our technique improves the performance of each

controller significantly across a variety of operating conditions. This chapter is organised as follows. Section 2 briefly reviews background on POMDPs. Section 3 casts the dialogue management problem as a POMDP, showing how to incorporate a confidence score, and reviewing previous work. Section 4 outlines our test-bed dialogue management simulation, and compares policies produced by our method to a baseline on the test-bed problem which uses the traditional "confidence bucket" approach. Section 5 shows how a handcrafted policy can be improved using confidence score, and provides an illustration, again using the test-bed problem. Section 6 briefly concludes.

2. Overview of POMDPs

Formally, a POMDP is defined as a tuple $\{S, A_m, T, R, O, Z\}$, where S is a set of states, A_m is a set of actions that an agent may take,[1] T defines a transition probability $p(s'|s, a_m)$, R defines the expected (immediate, real-valued) reward $r(s, a_m)$, O is a set of observations, and Z defines an observation probability, $p(o'|s', a_m)$. In this chapter, we will consider POMDPs with discrete S and continuous O. The POMDP operates as follows. At each time-step, the machine is in some unobserved state s. The machine selects an action a_m, receives a reward r, and transitions to (unobserved) state s' , where s' depends only on s and a_m. The machine receives an observation o' which is dependent on s' and a_m. Although the observation gives the system some *evidence* about the current state s, s is not known exactly, so we maintain a distribution over states called a "belief state", b. We write $b(s)$ to indicate the probability of being in a particular state s. At each time-step, we update b as follows:

$$\begin{aligned} b'(s') &= p(s'|o', a_m, b) \\ &= \frac{p(o'|s', a_m, b)p(s'|a_m, b)}{p(o'|a_m, b)} \\ &= \frac{p(o'|s', a_m)\sum_{s\in S} p(s'|a_m, b, s)p(s|a_m, b)}{p(o'|a_m, b)} \\ &= \frac{p(o'|s', a_m)\sum_{s\in S} p(s'|a_m, s)b(s)}{p(o'|a_m, b)} \end{aligned} \tag{8.1}$$

The numerator consists of the observation function, transition matrix, and current belief state. The denominator is independent of s', and can be regarded as a normalisation factor; hence:

$$b'(s') = k \cdot p(o'|s', a_m) \sum_{s\in S} p(s'|a_m, s)b(s) \tag{8.2}$$

[1]In the literature, the system action set is often written as an un-subscripted A. In this work, we will model both machine and user actions, and have chosen to write the machine action set as A_m for clarity.

We refer to maintaining the value of b at each time-step as "belief monitoring". The immediate reward is computed as the expected reward over belief states:

$$\rho(b, a_m) = \sum_{s \in S} b(s) r(s, a_m) \tag{8.3}$$

A POMDP policy specifies which action should be taken given a belief state.[2] The goal of policy learning is then to find a policy which maximises the cumulative, infinite-horizon, discounted reward called the return:

$$\sum_{t=0}^{\infty} \lambda^t \rho(b_t, a_{m_t}) = \sum_{t=0}^{\infty} \lambda^t \sum_{t=0}^{\infty} r(s, a_{m_t}) \tag{8.4}$$

where b_t indicates the distribution over all states at time t, $b_t(s)$ indicates the probability of being in state s at time-step t, and λ is a geometric discount factor, $0 < \lambda < 1$. Because belief space is real-valued, an optimal infinite-horizon policy may consist of an arbitrary partitioning of S-dimensional space in which each partition maps to an action. In fact, the size of the policy space grows exponentially with the size of the (discrete) observation set and doubly exponentially with the distance (in time steps) from the horizon (Kaelbling et al., 1998). A continuous observation space compounds this further. Nevertheless, real-world problems often possess small policies of high quality.

In this work, we make use of approximate solution methods. The first, a point-based value iteration algorithm called Perseus (Spaan and Vlassis, 2004), operates on problems with discrete observation sets and is capable of rapidly finding good yet compact policies (when they exist). Perseus heuristically selects a small set of representative belief points, and then iteratively applies value updates to just those points, instead of all of the belief space, achieving a significant speed-up. Perseus has been tested on a range of problems, and found to outperform a variety of other methods, including grid-based methods (Spaan and Vlassis, 2004).

Perseus (like all value-iteration optimisation algorithms) produces a value function represented as a set of N vectors each of dimensionality $|S|$. We write $v_n(s)$ to indicate the s_{th} component of the n_{th} vector. Each vector represents the value, at all points in the belief space, of executing some "policy tree" which starts with an action associated with that vector. We write $\hat{\pi}(n) \in A$ to indicate the action associated with the n_{th} vector. If we assume that the policy trees have an infinite horizon, then we can express the optimal policy at all time-steps as:

$$\pi(b) = \hat{\pi}(\operatorname*{argmax}_n \sum_{s=1}^{|S|} v_n(s) b(s)) \tag{8.5}$$

[2]We will assume the planning horizon for a policy is infinite unless otherwise stated.

In simple terms, a value function provides both a partitioning of the belief space (where each region corresponds to an action which is optimal in that region), as well as the expected return of taking that action. In this chapter we will also make use of an extension to Perseus proposed by Hoey and Poupart (2005) which operates on POMDPs with continuous or very large discrete observation sets. This method exploits the fact that different observations may lead to identical courses of action to discretise continuous observations without any loss of information. In the context of dialogue management with a continuous confidence score, it implicitly and adaptively finds optimal lossless buckets of confidence that are equivalent to using the original continuous confidence score.[3]

3. Casting Dialogue Management as a POMDP

Williams et al. (2005) cast a spoken dialogue system as a factored POMDP, and this model will be used as the general framework for the techniques presented here. In this model, the POMDP state variable $s \in S$ is separated into three components: (1) the user's goal, $s_u \in S_u$; (2) the user's action, $a_u \in A_u$; and (3) the history of the dialogue, $s_d \in S_d$. The POMDP state s is given by the tuple (s_u, a_u, s_d). We note that, from the machine's perspective, all of these components are unobservable.

The user's goal, s_u, gives the current goal or intention of the user. Examples of a complete user goal include a complete travel itinerary, a desired appointment to make in a calendar, or a product the user would like to purchase. The user's goal persists over the course of the dialogue, and in general it will remain static although it is possible for it to change (e.g., if the machine indicates that there are no direct flights, the user's goal might change to include indirect flights).

The user's action, a_u, gives the user's most recent actual action. Examples of user actions include specifying a place the user would like to travel to, responding to a yes/no question, or a "null" response indicating the user took no action. User actions may convey a portion of the user's goal (such as requesting a flight "to London"), or may serve a communicative role (such as answering a yes/no question).

The history of the dialogue s_d indicates any relevant dialogue history information. For example, s_d might indicate that a particular slot has not yet been stated, has been stated but not grounded, or has been grounded. s_d enables a policy to make decisions about the appropriateness of behaviours in a dialogue – for example, if there are ungrounded items, a dialogue designer might wish to penalise asking an open question (vs grounding an item).

[3]The actual implementation used in this chapter approximates some integrals by Monte Carlo sampling, which means that the confidence buckets are not exactly lossless.

Note that we do not include a state component for *confidence* associated with a particular user goal. The concept of confidence is naturally captured by the distribution of probability mass assigned to a particular user goal in the belief state.

The POMDP action $a_m \in A_m$ is the action the machine takes in the dialogue. For example, machine actions might include greeting the user, asking the user where he or she wants to go "to", or confirming that the user wants to leave "from" a specific place.

To factor the model, we decompose the POMDP transition function as follows:

$$\begin{aligned} p(s'|s, a_m) &= p(s'_u, s'_d, a'_u|s_u, s_d, a_u, a_m) \\ &= p(s'_u|s_u, s_d, a_u, a_m) \cdot \\ &\quad p(a'_u|s'_u, s_u, s_d, a_u, a_m) \cdot \\ &\quad p(s'_d|a'_u, s'_u, s_u, s_d, a_u, a_m) \end{aligned} \tag{8.6}$$

We then assume conditional independence as follows. The first term – which we call the user goal model – indicates how the user's goal changes (or does not change) at each time step. We assume the user's goal at a time step depends only on the previous goal and the machine's action:

$$p(s'_u|s_u, s_d, a_u, a_m) = p(s'_u|s_u, a_m) \tag{8.7}$$

The second term – which we call the *user action model* – indicates what actions the user is likely to take at each time step. We assume the user's action depends on his/her (current) goal and the preceding machine action:

$$p(a'_u|s'_u, s_u, s_d, a_u, a_m) = p(a'_u|s'_u, a_m) \tag{8.8}$$

The third term – which we call the *dialogue history model* – indicates how the user and machine actions affect the dialogue history. The current history of the dialogue depends on the previous history combined with the most recent user and machine actions:

$$p(s'_d|a'_u, s'_u, s_u, s_d, a_u, a_m) = p(s'_d|a'_u, s'_u, a_m) \tag{8.9}$$

In sum, our transition function is given by:

$$p(s'|s, a_m) = p(s'_u|s_u, a_m) \cdot p(a'_u|s'_u, a_m) \cdot p(s'_d|a'_u, s_d, a_m) \tag{8.10}$$

This factored representation reduces the number of parameters required for the transition function, and allows groups of parameters to be estimated separately. For example, we could estimate the *user action model* from a corpus by counting user dialogue acts given a machine dialogue act and a user goal,

or use a "generic" distribution and adapt it to a particular problem once data becomes available.[4] We could then separately specify the dialogue history model using a handcrafted function such as "Information State" update rules as in, for example, Larsson and Traum (2000).

The POMDP observation o is decomposed into two elements: the speech recognition hypothesis $\tilde{a}_u \in A_u$ and the confidence score $c \in R$. The observation function is given by:

$$p(o'|s', a_m) = p(\tilde{a}'_u, c'|s'_u, s'_d, a'_u, a_m) \tag{8.11}$$

The observation function accounts for the corruption introduced by the speech recognition engine, so we assume the observation depends only on the action taken by the user, and by the grammar g selected by the dialogue manager:

$$p(\tilde{a}'_u, c'|s'_u, s'_d, a'_u, a_m) = p(\tilde{a}'_u, c'|a'_u, g) \tag{8.12}$$

The observation function can be estimated from a corpus or derived analytically using a phonetic confusion matrix, language model, etc. This distribution expresses the probability density of observing recognition hypothesis $\tilde{a}'_u$ with confidence score c when the user actually took action a_u and recognition grammar g was activated. As such, the observation function can be viewed as a model of the errors introduced by the speech recognition channel.

Together equations 8.10 and 8.12 represent a statistical model of a dialogue. The transition function allows future behaviour to be predicted and the observation function provides the means for inferring a distribution over hidden user states from observations. The factoring is general-purpose in that the user goal component s_u allows the user to have a hidden, persistent state which emits unobserved actions a_u that are corrupted into observations $\tilde{a}_u$ by the speech recognition process. Further, the dialogue history component s_d enables actions to be selected with an awareness of dialogue history. Figure 2 summarizes the factored model, depicted as an influence diagram.

The reward function is not specified explicitly in this proposal since it depends on the design objectives of the target system. We note that the reward measure could contain incentives for dialogue speed (by using a per-turn penalty), appropriateness (through rewards conditioned on dialogue state), and successful task completion (through rewards conditioned on the user's goal). Weights between these incentives could be estimated through formalisms like PARADISE (Walker et al., 2000), and then adapted to the needs of a particular domain – for example, accuracy in performing a financial transaction is

[4]To appropriately cover all of the conditions, the corpus would need to include variability in the strategy employed by the machine – e.g., using a Wizard-of-Oz framework with a simulated ASR channel (Stuttle et al., 2004).

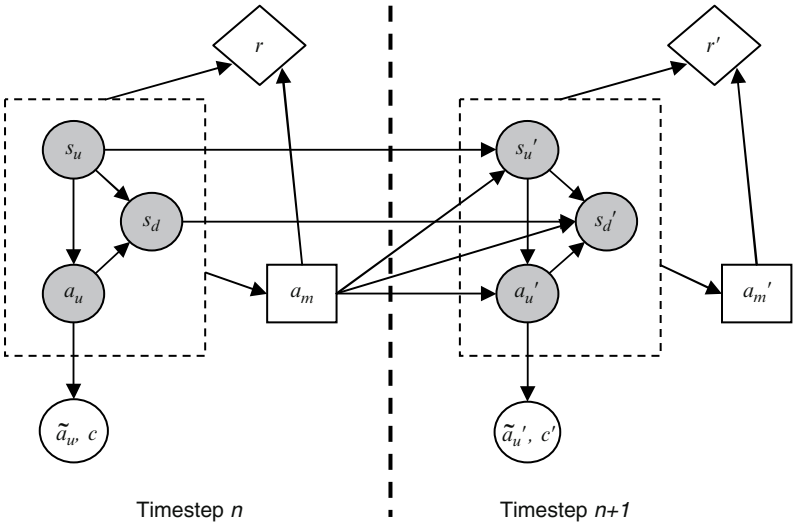


Figure 2. Influence diagram for the factored model. The dotted box indicates the composite state s is comprised of three components, s_u, s_d, and a_u. Shading indicates a component is unobservable. Arcs into circular chance nodes and diamond-shaped utility nodes show influence, whereas arcs into square decision nodes are informational, as in Jensen (2001, p. 140). The arc from the dotted box to a_m indicates that a_m is chosen based on the belief state – i.e., a distribution over s_u, s_d, and a_u.

arguably more important than accuracy when obtaining weather information. As described in the previous section, actions are selected based on the belief state to maximise cumulative long-term reward.

Finally, we update the belief state at each time step by substituting equations 8.10 and 8.12 into 8.2 and simplifying:

$$b'(s'_u, s'_d, a'_u) = k \cdot p(\tilde{a}'_u, c'|a'_u, g)p(a'_u|s'_u, a_m) \cdot \sum_{s_u \in S_u} p(s'_u|s_u, a_m) \cdot \quad (8.13)$$

$$\sum_{s_u \in S_u} p(s'_u|s_u, a_m) \cdot \sum_{s_d \in S_d} p(s'_d|a'_u, s_d, a_m) \cdot \sum_{a_u \in A_u} b(s_u, s_d, a_u)$$

The belief monitoring update equation 8.13 exemplifies the key difference between conventional approaches to dialogue management and the POMDP approach. In conventional approaches, a single state vector is maintained which encodes the system's "best guess" about all of the information needed to determine the next system action. For example, a state vector might include a record of all of the informational items supplied by the user, their grounding state, dialogue history, etc. Both the user's input and the subsequent system output are dependent on this state vector, but since there can be errors in this state vector, the system will often make mistakes and must then enter some

form of recovery procedure. This is essentially a depth-first greedy search with backtracking.

In the POMDP approach, all possible states are maintained rather than the single most likely state. Each user input (i.e., the observation) is then interpreted in the context of each possible new state via the observation term $p(\tilde{a}'_u, c'|a'_u, g)$. If the new observation is likely given s', then the subsequent belief in s' will be high and vice versa. The new state s' itself will only be plausible if there is a non-zero likelihood of making a transition from some previous state s to the new state, and since the previous state is unknown, all possible transitions are considered, weighted by the beliefs at the previous turn. In search terms, this is breadth-first search. It has the advantage over depth-first that both inputs and outputs can be determined from a knowledge of all of the alternative interpretations.

In practice the observation function $p(\tilde{a}'_u, c'|a'_u, g)$ will be difficult to estimate directly from data, so we will decompose the distribution by assuming that confidence scores are drawn from just two distributions – one for "correct" recognitions and another for "incorrect" recognitions:

$$p(\tilde{a}'_u, c'|a'_u, g) \approx \begin{cases} p_{correct}(c') \cdot p(\tilde{a}'_u|a'_u, g), & if \quad \tilde{a}'_u = a'_u \\ p_{incorrect}(c') \cdot p(\tilde{a}'_u|a'_u, g) & if \quad \tilde{a}'_u \neq a'_u \end{cases} \quad (8.14)$$

where $p(\tilde{a}'_u|a'_u, g)$ expresses the *confusion matrix* – i.e., probability of observing hypothesis $\tilde{a}'_u$ given that the user took action a'_u and grammar g was active; and $p_{correct}(c')$ and $p_{incorrect}(c')$ express the probability density function of the confidence scores associated with correct and incorrect recognitions. To perform policy improvement on this POMDP we have two options. First, we can use an optimisation method which accounts for the continuous observations, such as that by Hoey and Poupart (2005). This method creates a policy which takes the expected additional information in the confidence score into account, and we call this the *continuous-POMDP* solution. Alternatively, there is still benefit to using the confidence score information for belief state monitoring (as in 8.13) even if it was not used during policy optimisation. Thus a second option for performing policy improvement is to marginalise the confidence score, i.e.:

$$p(\tilde{a}'_u|a'_u, g) = \int_{c'} p(\tilde{a}'_u, c'|a'_u, g) \quad (8.15)$$

and to then optimise the resulting POMDP using a technique such as *Perseus*. At runtime, the full observation function $p(\tilde{a}'_u, c|a'_u, g)$ is used for belief state monitoring. We call this the *discrete-POMDP* solution.

Stated alternatively, the *continuous-POMDP* technique uses infinitely many confidence buckets during planning and belief monitoring, whereas the *discrete-POMDP* technique uses no confidence information during planning,

but infinitely many confidence buckets during belief monitoring. By contrast, MDP methods (in the literature, and our baseline, presented below) use a handful of confidence buckets for planning, but do not perform any belief monitoring.[5]

In the literature, casting dialogue management as planning under uncertainty has been attempted using both (fully observable) Markov decision processes (MDPs) and POMDPs. The application of MDPs was first explored by Levin and Pieraccini (1997). Levin et al. (2000) provide a formal treatment of how an MDP may be applied to dialogue management, and Singh et al. (2002) show application to real systems. However, MDPs assume the current state of the environment (i.e., the conversation) is known exactly, and thus they do not naturally capture the uncertainty introduced by the speech recognition channel.

Partially observable MDPs (POMDPs) extend MDPs by providing a principled account of noisy observations. Roy et al. (2000) compare an MDP and a POMDP version of the same spoken dialogue system, and find that the POMDP version gains more reward per unit time than the MDP version. Further, the authors show a trend that as speech recognition accuracy degrades, the margin by which the POMDP outperforms the MDP increases. Zhang et al. (2001) extend this work in several ways. First, the authors add "hidden" system states to account for various types of dialogue trouble, such as different sources of speech recognition errors. Second, the authors use Bayesian networks to combine observations from a variety of sources (including confidence score). The authors again show that the POMDP-based methods outperform MDP-based methods. In all previous work (using both MDPs and POMDPs), confidence score has been incorporated by dividing the confidence score metric into discrete confidence "buckets". For example, in the MDP literature, Singh et al. (2002) track the confidence bucket for each field as "high, medium, or low" confidence. The authors do not address how to determine an "optimal" number of confidence buckets, nor how to determine the "optimal" thresholds of the confidence score metric that divide each bucket. In the POMDP literature, Zhang et al. (2001) use Bayesian networks to combine information from many continuous and discrete sources, including confidence score, to compute probabilities for two metrics called "Channel Status" and "Signal Status". Thresholds are then applied to these probabilities to form discrete, binary observations for the POMDP. Again, it is not clear how to set these thresholds to maximise POMDP return. Looking outside the (PO)MDP framework, Paek

[5]In theory, one could create an MDP with continuous components in its state space, and use these components to track confidence score. While this avoids "binning" the confidence score, it does not aggregate evidence over time: in order to do this in an MDP, state components for "most recent confidence score", "2nd more recent confidence score", etc. would be required, causing rapid growth in the state space. By contrast, a POMDP frames a sequence of confidence scores as observations and naturally accumulates evidence over time through belief monitoring.

and Horvitz (2003) suggest using an influence diagram to model user and dialogue state, and selecting actions based on "Maximum Expected [immediate] Utility". This proposal can be viewed as a POMDP with continuous observations that greedily selects actions – i.e., which selects actions based only on immediate reward. By choosing appropriate utilities, the authors show how local grounding actions can be automatically selected in a principled manner. In this work, we are interested in POMDPs as they enable planning over any horizon.

4. Comparison with Traditional Approach

To assess the benefits of the POMDP approach versus traditional "confidence bucket" approaches, we created a test-bed dialogue management problem in the travel domain. This test-bed problem enables direct comparisons between dialogue managers produced by casting the problem as a POMDP with continuous observations, and dialogue managers produced by adding "confidence buckets" and casting the problem as an MDP. In both the POMDP and MDP, dialogue managers are produced automatically. Assuming that these represent optimal solutions, then this comparison gives a quantitative indication of the value of the POMDP approach.

4.1 POMDP Test-Bed Dialogue Management Problem

In the test-bed dialogue management problem, the user is trying to buy a ticket to travel from one city to another city. The machine asks the user a series of questions, and then "submits" a ticket purchase request, ending the dialogue. The machine may also choose to "fail". In the test-bed problem, there are three cities, $\{a, b, c\}$. The machine has 16 actions available, including *greet*, *ask-from/ask-to*, *conf-to-x/conf-from-x*, *submit-x-y*, and *fail*, where $x, y \in \{a, b, c\}$. As above, the POMDP state is given by the tuple (s_u, a_u, s_d). The user's goal $s_u \in S_u$ specifies the user's desired itinerary. There are a total of six user goals, given by $s_u = (x, y) : x, y \in \{a, b, c\}, x \neq y$. The dialogue state s_d contains three components. Two of these indicate (from the user's perspective) whether the from place and to place have not been specified (n), are unconfirmed (u), or are confirmed (c). A third component z specifies whether the current turn is the first turn (1) or not (0). There are a total of 18 values of s_d, given by:

$$s_d = (x_d, y_d, z); \quad x_d, y_d \in \{n, u, c\}; \quad z \in 1, 0 \tag{8.16}$$

The user's action $a_u \in A_u$ and the observation $\tilde{a}_u \in A_u$ are drawn from the set *x*, *from-x*, *to-x*, *from-x-to-y*, *yes*, *no*, and *null*, where $x, y \in \{a, b, c\}, x \neq y$. These state components yield a total of 1,944 states, to which we add one additional, absorbing end state. When the machine takes the *fail* action or a

submit-x-y action, control transitions to this end state, and the dialogue ends. The initial (prior) probability of the user's goal is distributed uniformly over the six user goals. In the test-bed problem the user has a fixed goal for the duration of the dialogue, and we define the *user goal model* accordingly.

We define the *user action model* $p(a'_u|s'_u, a_m)$ to include a variable set of responses – for example: the user may respond to *ask-to/ask-from* with x, *to-x/from-x*, or *from-x-to-y*; the user may respond to *greet* with *to-y*, *from-x*, or *from-x-to-y*; the user may respond to *confirm-to-x/confirm-from-x* with *yes/no*, x, or *to/from-x*; and at any point the user might not respond (i.e., respond with $null$). The probabilities in the user action model were chosen such that the user usually provides cooperative but varied responses, and sometimes doesn't respond at all. The probabilities were handcrafted, selected based on experience performing usability testing with slot-filling dialogue systems.

We define the *dialogue model* $p(s'_d|a'_u, s_d, a_m)$ to deterministically implement the notions of dialogue state above – i.e., a field which has not been referenced by the user takes the value n; a field which has been referenced by the user exactly once takes the value u; and a field which has been referenced by the user more than once takes the value c. For example, at the beginning of the dialogue, the dialogue state s_d is $(n, n, 1)$. If the user were to say "I'd like to go to b" in his/her first utterance, the resulting dialogue state would be $(n, u, 0)$. If the system were to reply "To b – is that right?", and the user replied "Yes, from a to b", then the resulting dialogue state would be $(u, c, 0)$. We define the confusion matrix $p(\tilde{a}'_u|a'_u, g)$ to encode the probability of making a speech recognition error to be p_{err}. Further, we assume that one recognition grammar is always used:

$$\begin{aligned} p(\tilde{a}'_u, c'|a'_u, g) &= p(\tilde{a}'_u, c'|a'_u) \\ &= \begin{cases} p_{correct}(c') \cdot (1 - p_{err}) & if \quad \tilde{a}_u = a_u \\ p_{incorrect}(c') \frac{p_{err}}{|A_u|-1} & if \quad \tilde{a}_u \neq a_u \end{cases} \end{aligned} \tag{8.17}$$

Below we will vary p_{err} to explore the effects of speech recognition errors.

Past work has found the distribution of confidence scores to be exponential (Pietquin, 2004), and here we define the confidence score probability density functions $p_{correct}(c')$ and $p_{incorrect}(c')$ to be exponential probability density functions normalised to the region [0,1], i.e.:

$$p_{correct}(c) = \begin{cases} \frac{a_{correct} e^{c \cdot a_{correct}}}{e^{a_{correct}} - 1}, & a_{correct} \neq 0 \\ 1, & a_{correct} = 0 \end{cases}$$

$$p_{incorrect}(c) = \begin{cases} \frac{a_{incorrect} e^{(1-c) \cdot a_{incorrect}}}{e^{a_{incorrect}} - 1}, & a_{incorrect} \neq 0 \\ 1, & a_{incorrect} = 0 \end{cases} \tag{8.18}$$

Table 1. Minimum classification error rate possible for various concept error rates (P_{err}) and levels of confidence score informativeness (a_x).

	Concept error rate (P_{err})		
a_x	0.10 (%)	0.30 (%)	0.50 (%)
0	10	30	50
1	10	30	38
2	9	23	27
3	9	16	18
4	6	11	12
5	4	7	8
∞	0	0	0

where $a_{correct}$ and $a_{incorrect}$ are constants defined on $(-\infty, \infty)$. We note that as a_x approaches positive or negative infinity, $p_x(c)$ becomes deterministic and conveys complete information; when $a_x = 0$, $p_x(c)$ is a uniform density and conveys no information. Since we expect the confidence value for correct recognition hypotheses to tend to 1, and for incorrect recognition hypotheses to tend to 0, we would expect $a_x > 0$. To illustrate the meaning of $a_{correct}$ and $a_{incorrect}$, a small classification task was created in which a confidence score is used as a decision variable to classify $\tilde{a}_u$ as either *correct* or *incorrect*. Various concept error rates (values of p_{err}) and a_x were considered and for each pair of values, the confidence threshold which minimised classification error rate was used. Table 1 shows the results. When $a_x = 0$, all hypotheses are classified as *correct* and the classification error rate is the same as p_{err}. As a_x is increased, the classification error rate decreases. Intuitively, Table 1 shows the minimum possible classification error rate achievable with a given a_x, and comparing this with the prior error rate p_{err} gives an indication of the *informativeness* of a_x.

The reward measure for the test-bed dialogue problem includes components for both task completion and dialogue "appropriateness", including: a reward of -3 for confirming a field before it has been referenced by the user; a reward of -5 for taking the *fail* action; a reward of $+10$ or -10 for taking the *submit-x-y* action when the user's goal is *(x,y)* or not, respectively; and a reward of -1 otherwise. The reward measure reflects the intuition that behaving inappropriately or even abandoning a hopeless conversation early are both less severe than getting the user's goal wrong. The per-turn penalty of -1 expresses the intuition that, all else being equal, short dialogues are better than long dialogues. The reward measure also assigned -100 for taking the *greet* action when not in the first turn of the dialogue. This portion of the reward function effectively expresses a design decision: the greet action may only be taken in the first turn. A discount of $\gamma = 0.95$ was used for all experiments.

Both the *Perseus* and the *Hoey–Poupart* algorithms required parameters for the number of belief points and number of iterations. Through experimentation,

we found that 500 belief points and 30 iterations attained asymptotic performance for all values of P_{err}. In addition, the *Hoey–Poupart* algorithm required a parameter specifying the number of observations to sample at each belief point. Through experimentation, we found that 300 samples produced acceptable results and reasonable running times.

4.2 MDP Baseline

To test whether the method for incorporating confidence score outperforms current methods, an MDP was constructed to assess performance of a model which does not track multiple dialogue states, and which does not make use of an explicit user model. The MDP was patterned on systems in the literature (Pietquin, 2004). The MDP state contains components for each field which reflect whether, from the standpoint of the machine, (a) a value has not been observed, (b) a value has been observed but not confirmed, or (c) a value has been confirmed. The MDP state also tracks which confidence bucket was observed for each field, as well as for the confirmation. Finally, two additional states – *dialogue-start* and *dialogue-end* – are included in the MDP state space.

The "confidence bucket" is determined by dividing the confidence score into M buckets. Ideally the confidence score bucket sizes would be selected so that they maximise average return. However, it is not obvious how to perform this selection – indeed, this is one of the weaknesses of the "confidence bucket" method. Instead, a variety of techniques for setting the confidence score threshold were explored. It was found that dividing the probability mass of the confidence score c evenly between buckets produced the largest average returns among the techniques explored.[6] That is, we define

$$cThresh_0 = 0 < cThresh_1 < \cdots < cThresh_{M-1} < cThresh_M = 1 \tag{8.19}$$

and then find the values of $cThresh_m$ such that:

$$\int_{cThresh_{m-1}}^{cThresh_m} p(c)dc = \int_{cThresh_m}^{cThresh_m+1} p(c)dc, \quad m \in 1, 2, \cdots, M-1 \tag{8.20}$$

where $p(c)$ is the prior probability of a confidence score. We find this prior for our test-bed problem as follows. We first find the distribution $p(c|a_u)$ as:

$$p(c|a_u) = \sum_{h \in A} p(h, c|a_u) \tag{8.21}$$

$$= p_{correct}(c|a_u)(1 - p_{err}) + p_{incorrect}(c|a_u)(p_{err}) \tag{8.22}$$

[6]The other techniques included dividing the *range of confidence* scores equally (e.g., for two buckets, using a threshold of 0.5), and dividing the *range of error rates* equally (e.g., for two buckets, setting a threshold such that p (observation is correct | confidence score) = 0.5).

In the MDP context, we assume the confidence score buckets are formed without access to a prior $p(a_u)$. From this assumption, we find:

$$p(c) = p_{correct}(c)(1 - p_{err}) + p_{incorrect}(c)(p_{err}) \tag{8.23}$$

from which the values of $cThresh_m$ can be derived.

Because the confidence bucket for each field (including its value and its confirmation) is tracked in the MDP state, the size of the MDP state space grows with the number of confidence buckets. For $M = 2$ confidence buckets, the resulting MDP called *MDP-2* has 51 states.[7]

Given the current MDP state, the MDP policy selects an MDP action, and the MDP state estimator then maps the MDP action back to a POMDP action. Because the MDP learns through experience with a simulated environment, an on-line learning technique, (Watkins, 1989) Q-learning, was used to train the MDP baseline. A variety of learning parameters were explored, and the best-performing parameter set was selected: initial Q values set to 0, exploration parameter $\varepsilon = 0.2$, and the learning rate α set to $1/k$ (where k is the number of visits to the $Q(s, a)$ being updated). *MDP-2* was trained with approximately 125,000 dialogue turns. To evaluate the resulting MDP policy, 10,000 dialogues were simulated using the learned policy.

4.3 Results

Figure 3 shows the average returns for the *continuous-POMDP*, *discrete-POMDP*, and *MDP-2* solutions vs p_{err} ranging from 0.00 to 0.65 for $a_{correct} = a_{incorrect} = a = 1$. (At each data point, an error rate p_{err} was set, and errors and confidence scores were generated synthetically according to equations 8.16 and 8.18.). Figure 3 also shows curves for *noconf-POMDP* a POMDP which ignores confidence score information and *MDP*, an MDP which ignores confidence score information (i.e., an MDP with just one confidence score bucket). The error bars show the 95% confidence interval for return assuming a normal distribution. Note that return decreases consistently as p_{err} increases for all solution methods, but the POMDP solutions attain larger returns than the MDP method at all values of p_{err}.[8] From this plot it can be seen that the addition of confidence score information improves both the POMDP and MDP solutions. This plot shows that, at $a = 1$, the addition of confidence score information has a large improvement in performance for the MDP, and a modest but significant improvement on the POMDP.

As the informativeness of the confidence score increases, it would be expected that the performance of both the MDP and POMDP would continue

[7]For reference, $M = 1$ produces an MDP with 11 states, and $M = 3$ produces an MDP with 171 states.
[8]The *MDP-3* system was also created but we were unable to obtain better performance from it than we did from the *MDP-2* system.

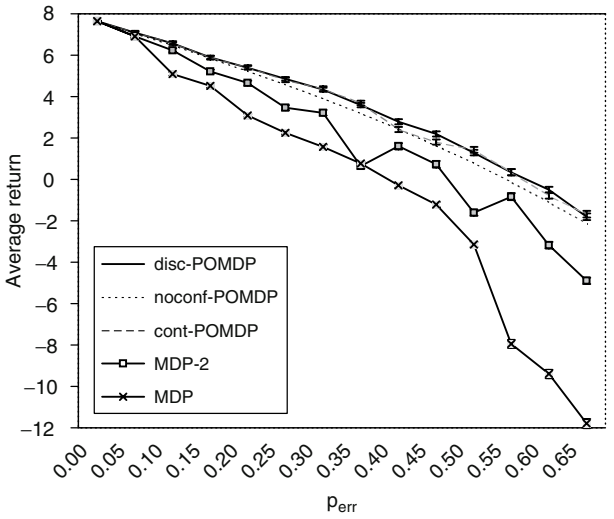


Figure 3. *Average return* for continuous-POMDP, discrete-POMDP, noconf-POMDP, MDP-2 and MDP methods for $a = 1$.

to improve. This is confirmed in Figures 4, 5, and 6 which show average returns for the *discrete-POMDP* and *continuous-POMDP* methods and *MDP-2* method vs. a for $p_{err} = 0.3$, 0.4, and 0.5, respectively. The error bars show the 95% confidence interval for return assuming a normal distribution. In these figures, we again define $a_{correct} = a_{incorrect} = a$. The POMDP methods outperform the baseline MDP method consistently. Note that increasing a increases average return for all methods, and that the greatest improvements are for $p_{err} = 0.5$ – i.e., the information in the confidence score has more impact as speech recognition accuracy degrades. These figures also provide a quantitative illustration of the benefit of belief monitoring vs the benefit of the confidence score information. For example, in Figure 6 (in which $p_{err} = 0.5$), at $a = 0$, the POMDP achieves an average of 0.7 units of reward/dialogue whereas the MDP achieves an average of -3.1 units of reward/dialogue. In other words, ignoring confidence score altogether, the belief monitoring provided by the POMDP results in an increase from -3.1 to 0.7. The addition of a very informative confidence score (i.e., $a = 5$) to the POMDP results in an increase from 0.7 units of reward/dialogue to 3.2 units of reward/dialogue.

In Figures 3 through 6, the discrete-POMDP and continuous-POMDP methods performed similarly.[9] In this task, use of the confidence score *during*

[9]Additional experiments were performed (not shown here) which performed POMDP optimisation with 2, 4, and 8 "buckets" and continuous belief monitoring during evaluation, and these produced very similar results.

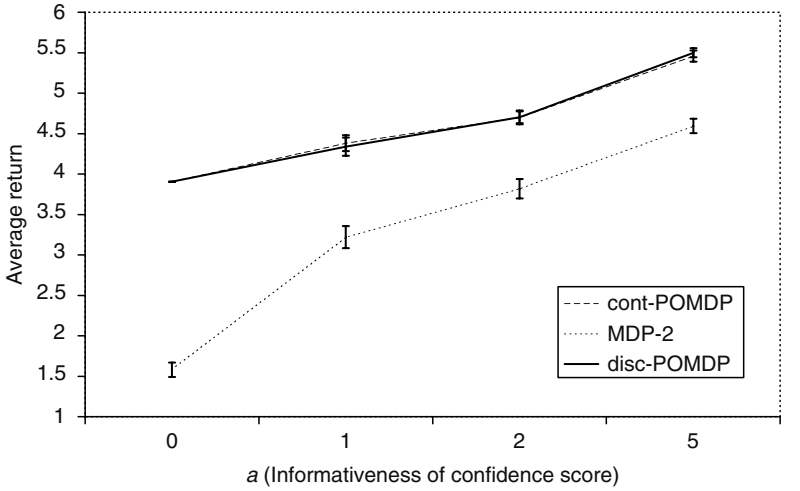


Figure 4. Average return vs *a (informativeness of confidence score)* at $p_{err} = 0.30$ for continuous-POMDP, discrete-POMDP, and MDP-2 methods.

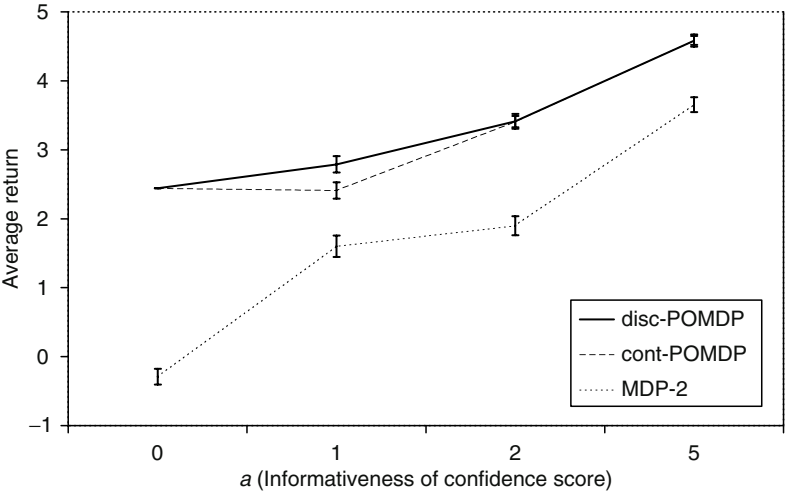


Figure 5. Average return vs *a (informativeness of confidence score)* at $p_{err} = 0.40$ for continuous-POMDP, discrete-POMDP, and MDP-2 methods.

planning does not improve performance of the POMDP. This could be due to the relatively short horizon in the test-bed problem, as most of the dialogues spanned only a handful of turns. We intend to explore this issue with larger dialogue management problems in future work.

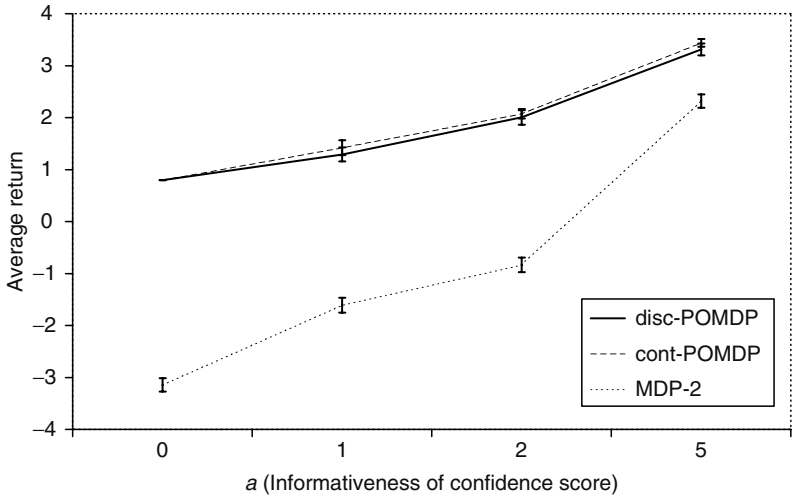


Figure 6. Average return vs *a (informativeness of confidence score)* at $p_{err} = 0.50$ for continuous-POMDP, discrete-POMDP, and MDP-2 methods.

5. Improving Handcrafted Policies

In the previous section, a designer specified a reward function, and actions were selected to maximise reward using automated planning. In traditional approaches to dialogue management, the designer specifies actions directly, a process often called *handcrafting*.[10]

Automated planning is appealing because adding confidence score information to the dialogue state space increases its size dramatically, complicating the work of a human designer. This section presents an alternative approach in which a human designer produces a handcrafted dialogue manager which *does not include confidence score information*. Rather, the spoken dialogue system is viewed as a POMDP, belief monitoring (which takes confidence score into account) is performed, and the handcrafted controller is executed in conjunction with the belief state. Concretely, the handcrafted policy is evaluated by constructing its value function, and is then executed in the style of the *discrete-POMDP* above.

Intuitively, a policy specifies what action to take in a given situation. In the previous section, we relied on the representation of a POMDP policy produced by value iteration – i.e., a value function, represented as a set of N vectors each of dimensionality $|S|$. We write $v_n(s)$ to indicate the sth component of the nth vector.

[10]In both POMDP and traditional approaches, the designer creates a dialogue model; the focus here is how actions are selected given a dialogue model.

A second way of representing a POMDP policy is as a "policy graph" – a finite state controller consisting of N nodes and some number of directed arcs. Each controller node is assigned a POMDP action, and we will again write $\hat{\pi}(n)$ to indicate the action associated with the nth node. Each arc is labelled with a POMDP observation, such that all controller nodes have exactly one outward arc for each observation. $l(n, o)$ denotes the successor node for node n and observation o. A policy graph is a general and common way of representing handcrafted dialogue management policies (Pieracinni and Huerta, 2005). More complex handcrafted policies – for example, those created with rules – can usually be compiled into a (possibly very large) policy graph. That said, a policy graph does not make the expected return associated with each controller node explicit, but as pointed out by Hansen (1998), we can find the expected return associated with each controller node by solving this system of linear equations in v:

$$v_n(s) = r(s, \hat{\pi}(n)) + \gamma \sum_{s' \in S} \sum_{o \in O} p(s'|s, \hat{\pi}(n)) p(o|s', \hat{\pi}(n)) v_{l(n,o)}(s') \quad (8.24)$$

Solving this set of linear equations yields a set of vectors – one vector $v(s)$ for each controller node, $v_n(s)$. To find the expected value of starting the controller in node n and belief state b we compute:

$$\sum_{s=1}^{|S|} v_n(s) b(s) \quad (8.25)$$

To improve the performance of the controller, we use $v_n(s)$ at *run-time*, as follows. At the beginning of the dialogue, we find the node with the highest expected return for b_0 and execute its action. Throughout the dialogue, we perform belief state monitoring – i.e., we maintain the current belief state at each time-step as given in equation 8.13. At each time-step, rather than following the policy specified by the finite state controller, we *re-evaluate* which node has the highest expected return for the current b. We then take the action specified by that node. Because the node-value function and belief state are exact, this style of execution is guaranteed to perform at least as well as the original handcrafted controller. Note that, in this style of execution, transitions may occur which are not arcs in the handcrafted policy.

This style of execution is distinct from *policy iteration*, in which the nodes and links of the controller are changed and the controller is re-evaluated (using e.g., equation 8.24) to iteratively improve the controller's expected return. We do not explore policy iteration in this chapter; however, we note that a handcrafted controller could be used to bootstrap a policy iteration process. Since

a finite state controller is more intuitive for a (human) designer to understand, we intend to explore policy iteration in future work.

Three handcrafted policies were created for the test-bed dialogue management problem, called HC1, HC2, and HC3. All of the handcrafted policies first take the action *greet*. *HC1* takes the *ask-from* and *ask-to* actions to fill the *from* and *to* fields, performing no confirmation. If the user does not respond, it re-tries the same action. If it receives an observation which is inconsistent or nonsensical, it re-tries the same action. If it fills both fields without receiving any inconsistent information, it takes the corresponding *submit-x-y* action. A logical diagram showing *HC1* is shown in Figure 7.[11]

HC2 is identical to HC1 except that if the machine receives an observation which is inconsistent or nonsensical, it immediately takes the *fail* action. Once it fills both fields, it takes the corresponding *submit-x-y* action.

$HC3$ employs a similar strategy to HC1 but extends HC1 by confirming each field as it is collected. If the user responds with "no" to a confirmation, it re-asks the field. If the user provides inconsistent information, it treats the new information as "correct" and confirms the new information. If the user does not respond, or if the machine receives any nonsensical input, it re-tries the same action. Once it has successfully filled and confirmed both fields, it takes the corresponding *submit-x-y* action.

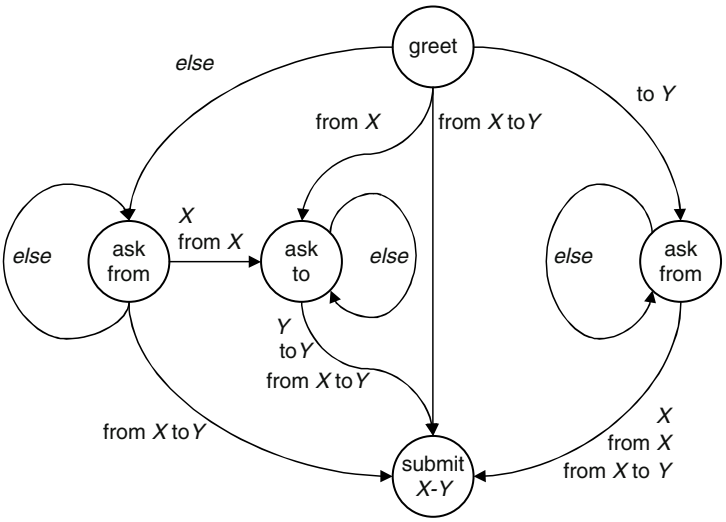


Figure 7. HC1 handcrafted controller.

[11] A logical diagram is shown for clarity: the actual controller uses the real values a, b, and c, instead of the variables X and Y, resulting in a controller with 15 states.

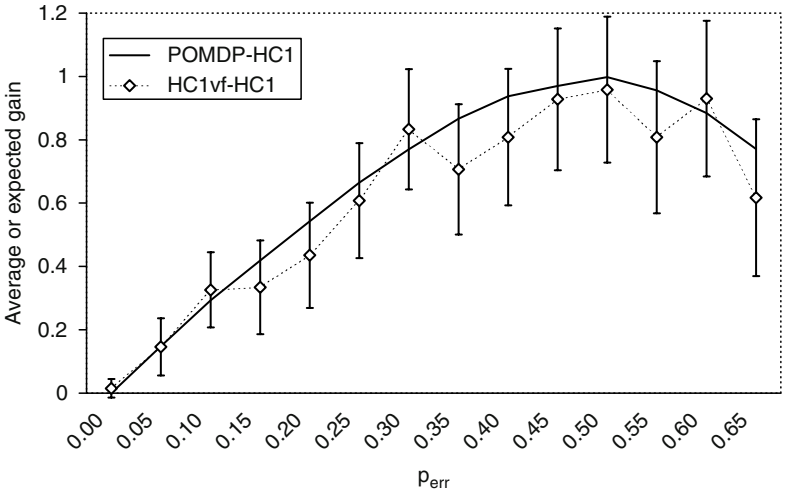


Figure 8. Gain in average/expected return for HC1 executed using belief state monitoring vs p_{err} for $a = 0$. (The POMDP policy, which we take to be our practical upper bound, is shown for reference in Figures 8 through 10.)

We first studied the operation of the greedy improvement method without access to confidence score information. We executed 10,000 dialogues for each handcrafted policy at values of p_{err} ranging from 0.05 to 0.65. Figure 8 gives results for $HC1$. To make the gain of the greedy improvement method explicit, Figure 8 shows the difference between the proposed method and the expected value of executing the handcrafted policy directly. For reference, Figure 8 also includes the difference between the handcrafted policies executed normally and the POMDP policy, which we take to be a practical upper bound for the test-bed problem. Error bars show the 95% confidence interval for the true expected return assuming normal distribution. We note that in almost all cases, the greedy improvement method results in a significant improvement. In many cases, the improved handcraft controller is close to the POMDP policy – our assumed practical upper bound. Results for $HC2$ and $HC3$ are shown in Figures 9 and 10.

We next studied the operation of the greedy improvement method when confidence score information is present. Figures 11, 12, and 13 show average returns for the *discrete-POMDP* and improved handcraft methods vs a for $p_{err} = 0.3$, 0.4, and 0.5, respectively. a is defined as in Section 4.2 – i.e., $a = a_{correct} = a_{incorrect}$. Error bars are negligible and are not shown. For each of the three handcrafted controllers in each of the three values of p_{err}, increasing a consistently increases average return.

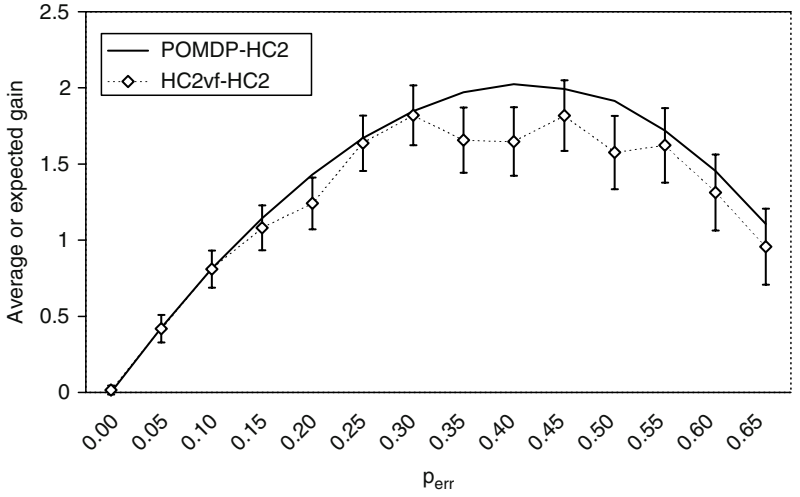


Figure 9. Gain in average/expected return for HC2 executed using belief state monitoring vs p_{err} for $a = 0$.

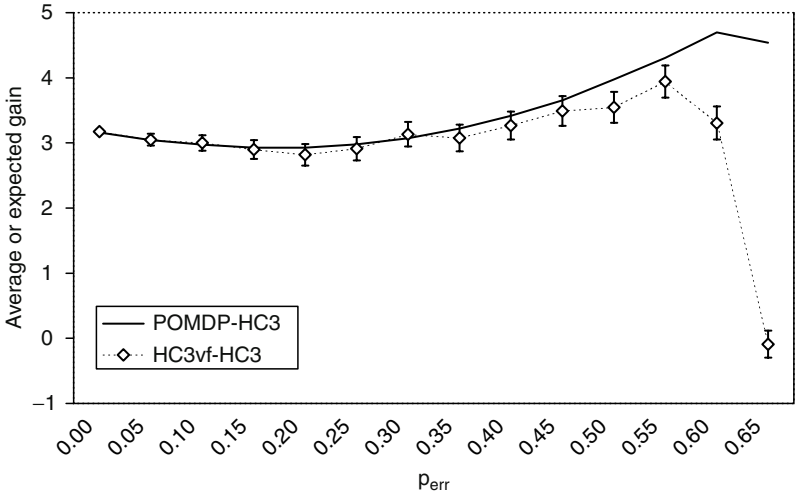


Figure 10. Gain in average/expected return for HC3 executed using belief state monitoring vs p_{err} for $a = 0$.

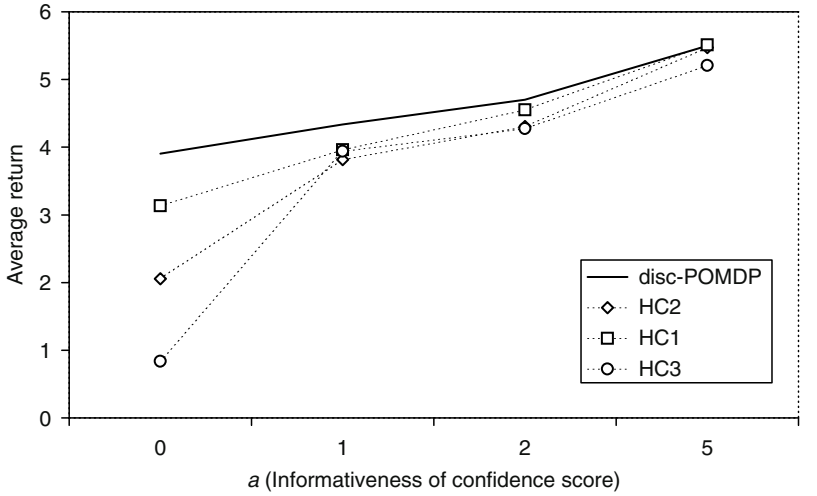


Figure 11. *Average return* vs a (informativeness of confidence score) for $p_{err} = 0.30$ for discrete-POMDP and handcrafted policies executed with belief state monitoring.

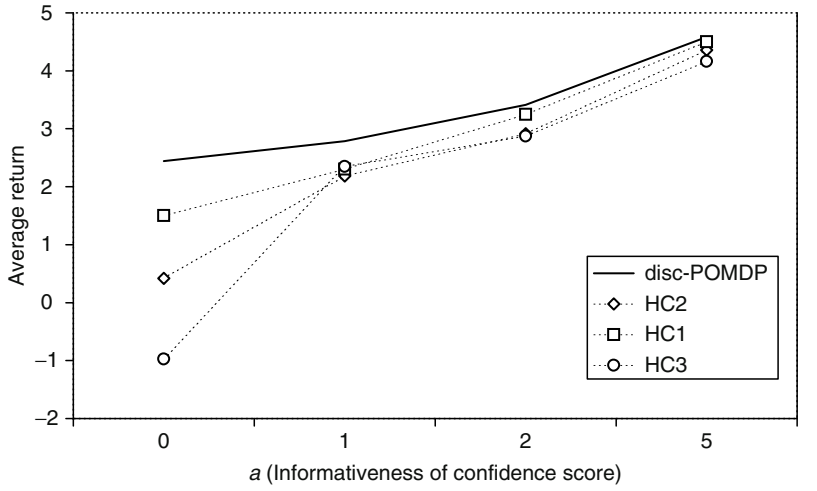


Figure 12. *Average return* vs a (informativeness of confidence score) for $p_{err} = 0.40$ for discrete-POMDP and handcrafted policies executed with belief state monitoring.

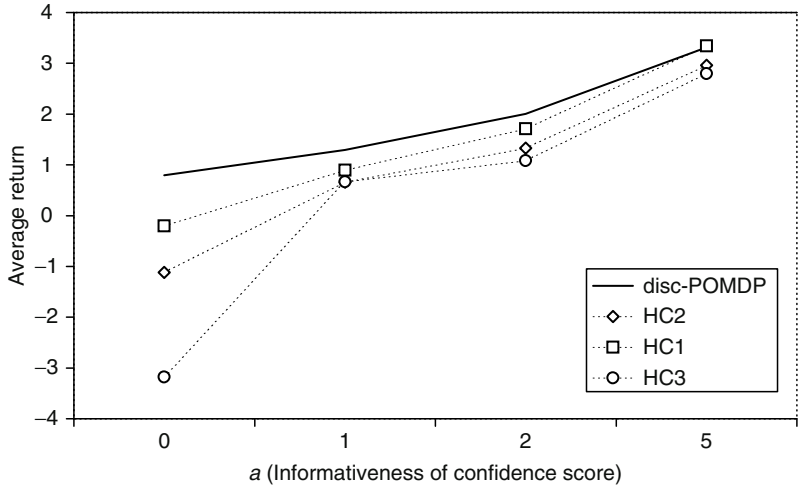


Figure 13. *Average return* vs a (informativeness of confidence score) for $p_{err} = 0.50$ for discrete-POMDP and handcrafted policies executed with belief state monitoring.

6. Conclusions

This chapter has shown how a confidence score can be directly incorporated into the dialogue model represented as a Partially Observable Markov Decision Process (POMDP) used for dialogue management. Unlike traditional approaches which maintain a single dialogue state at each time step, in effect a POMDP considers all possible dialogue states, and maintains a probability distribution over these called a *belief state*. This representation allows a confidence score to be tracked in the dialogue state in a principled fashion, and optimising the POMDP produces a dialogue manager which exploits this representation when selecting actions. In evaluation, the POMDP significantly outperforms a baseline MDP, which tracks only one hypothesis for the dialogue state.

This chapter has also presented a second approach to policy production in which a handcrafted controller which does not account for confidence score information can be improved to automatically account for confidence score information.

The problems considered here were unrealistically small for real-world deployment, and recent work has shown how to scale POMDPs to slot-filling problems of a realistic size (Williams and Young, 2005). Also, this chapter has considered only the top recognition hypothesis and confidence score. A natural extension would to be consider more complex hypothesis representations such as *N-Best* lists or word lattices, and more recognition features such as prosodic information, parse scores, or acoustic metrics.

Acknowledgements This work was supported in part by the European Union Framework 6 TALK Project (507802).

References

Hansen, E. A. (1998). Solving POMDPs by Searching in Policy Space. In *Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 211–219, Madison.

Hoey, J. and Poupart, P. (2005). Solving POMDPs with Continuous or Large Discrete Observation Spaces. In *Proceedings of the Joint International Conference on Artificial Intelligence (IJCAI)*, pages 1332–1338, Edinburgh.

Jensen, F. (2001). *Bayesian Networks and Decision Graphs*. Springer Verlag.

Kaelbling, L., Littman, M., and Cassandra, A. (1998). Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101(1-2):99–134.

Larsson, S. and Traum, D. (2000). Information State and Dialogue Management in the Trindi Dialogue Move Engine Toolkit. *Natural Language Engineering*, 5(3-4):323–340.

Levin, E. and Pieraccini, R. (1997). A Stochastic Model of Computer-Human Interaction for Learning Dialogue Strategies. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 1883–1886, Rhodes.

Levin, E., Pieraccini, R., and Eckert, W. (2000). A Stochastic Model of Human-Machine Interaction for Learning Dialogue Strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23.

Paek, T. and Horvitz, E. (2003). On the Utility of Decision-Theoretic Hidden Subdialog. In *Proceedings of International Speech Communication Association (ISCA) Workshop on Error Handling in Spoken Dialogue Systems*, pages 95–100, Switzerland.

Pieracinni, R. and Huerta, J. (2005). Where do we Go from Here? Research and Commercial Spoken Dialog Systems (invited talk). In *Proceedings of SIGdial Workshop on Discourse and Dialogue*, pages 1–10, Lisbon.

Pietquin, O. (2004). *A Framework for Unsupervised Learning of Dialogue Strategies*. PhD thesis, Faculty of Engineering, Mons.

Roy, N., Pineau, J., and Thrun, S. (2000). Spoken Dialogue Management Using Probabilistic Reasoning. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 93–100, Michigan.

Singh, S., Litman, D., Kearns, M., and Walker, M. (2002). Optimizing Dialogue Management with Reinforcement Leaning: Experiments with the NJFun System. *Artificial Intelligence*, 16:105–133.

Spaan, M. and Vlassis, N. (2004). Perseus: Randomized Point-Based Value Iteration for POMDPs. Technical report, Informatics Institute, University of Amsterdam.

Stuttle, M., Williams, J., and Young, S. (2004). A Framework for Dialogue Data Collection with a Simulated ASR Channel. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 241–244, Jeju Island.

Walker, M., Kamm, C., and Litman, D. (2000). Towards Developing General Models of Usability with PARADISE. *Natural Language Engineering*, 6:363–377.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. Ph.D. thesis, Cambridge University.

Williams, J. D., Poupart, P., and Young, S. (2005). Factored Partially Observable Markov Decision Processes for Dialogue Management. In *Proceeedings of 4th Workshop on Knowledge and Reasoning in Practical Dialog Systems, International Joint Conference on Artificial Intelligence (IJCAI)*, pages 76–82, Edinburgh.

Williams, J. D. and Young, S. (2005). Scaling up POMDPs for Dialog Management: The "Summary POMDP" Method. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 177–182, San Juan.

Zhang, B., Cai, Q., Mao, J., Chang, E., and Guo, B. (2001). Spoken Dialogue Management as Planning and Acting under Uncertainty. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, volume 2, pages 2169–2172, Aalborg.

Chapter 9

DOES THIS ANSWER YOUR QUESTION?

Towards Dialogue Management for Restricted Domain Question Answering Systems

Matthias Denecke and Norihito Yasuda
Communication Science Laboratories
NTT Corporation, Kyoto, Japan
deneckecslab.kecl.ntt.co.jp, n-yasuda@cslab.kecl.ntt.co.jp

Abstract Interactive Restricted Domain Question Answering Systems combine the interactivity of dialogue systems with the information retrieval features of question answering systems. The main problem when going from task-oriented dialogue systems to interactive restricted domain question answering systems is that the lack of task structure prohibits making simplifying assumptions as in task-oriented dialogue systems. In order to address this issue, we propose a solution that combines representations based on keywords extracted from the user utterances with machine learning to learn the dialogue management function. More specifically, we propose to use Support Vector Machines to classify the dialogue state containing the extracted keywords in order to determine the next action to be taken by the dialogue manager. Much of the content selection for clarification question usually found in dialogue managers is moved to an instance-based generation component. The proposed method has the advantage that it does not rely on an explicit representation of task structure as is necessary for task-oriented dialogue systems.

Keywords: Restricted domain question answering system; statistical dialogue management; instance-based generation

1. Introduction

Question answering is the task of providing natural language answers to natural language questions using an information retrieval engine. Question answering systems work well in the presence of highly specific keywords in the queries, but deteriorate in case of vague or ambiguous questions.

L. Dybkjær and W. Minker (eds.), Recent Trends in Discourse and Dialogue, 219–246.

Dialogue systems address the problem of accessing information from a structured database (such as timetable information) or controlling appliances by voice. The responsibilities of task-oriented dialogue managers can be broadly grouped into eliciting task-relevant information from the user and managing error recognition and correction to compensate for speech recognition errors (see Zue and Glass, 2000) for a more detailed analysis. In order to accomplish the first task, namely deciding whether and how to elicit information from the user, a dialogue manager can rely on a structured domain model. In the simplest case, the domain model can consist of a set of slots to be filled, and the dialogue manager prompts the user until all slots are actually filled.

Restricted domain question answering systems can be deployed in interactive problem solving solutions, for example, software troubleshooting. An overview of potential applications is given in Minock (2005). In these scenarios, interactivity becomes a necessity. This is because it is highly unlikely that all facts relevant to retrieving the appropriate response are stated in the query. For example, in the software troubleshooting task described in Kiyota et al. (2002), a frequent system generated information seeking question is for the version of the software. Therefore, there is a need to inquire additional problem relevant information from the user, depending on the interaction history and the problem to be solved, and therefore, there is a need to equip question answering systems with dialogue capabilities.

However, a straightforward adaptation of techniques used in spoken dialogue systems to question answering systems is elusive due to the absence of a structured task model. To illustrate the point, consider a simple database as the back-end of a dialogue manager. The database schema, containing the field names and their data types, defines the information seeking questions a dialogue manager can ask. The result set from the database for any given query dictates whether and what sort of disambiguation questions the user needs to be asked; prompting the user for fillers for those slots that have multiple fillers will eventually reduce the size of the result set to 1. Ferrieux and Sadek (1994) describe an algorithm that performs the task described informally here. Denecke and Waibel (1997) describe an algorithm to determine the contents of prompts based on an ontology, and Rudnicky and Wu (1999) derive the order of prompts from an agenda. All three approaches rely critically on the presence of a structured task model that contains at least the slots the user needs to be prompted for. See Section 2 for a more detailed discussion.

Once the database engine is replaced with an information retrieval engine, a structured task model is not available. Therefore, the task to decide whether to prompt the user for more information, and, if so, how to prompt for more information, becomes difficult. First, the scope of the dialogue is determined by the text documents in the question answering system. Consequently, it is not straightforward to determine a list of slots that should be known to the system.

Even if that list of slots could be determined, it is a non-trivial task to extract the relevant fillers from a retrieved text document and make them available to the dialogue management algorithms described above.

1.1 Addressed Problem

We investigate how a dialogue manager can decide whether and how to prompt a user for more information in the absence of a structured task model, with the goal of equipping a restricted domain question answering system with dialogue capabilities. To achieve this goal, we relate information relevant for text retrieval (such as question types, or key words) with the texts retrieved by the question answering system. We use statistical classifiers to determine the action taken by the dialogue manager.

The focus of our approach is in contrast to current research on statistical methods in the area of spoken dialogue systems which typically focuses on optimising dialogue strategies in the presence of recognition errors. For example, Markov Decision Processes and Partially Observable Markov Decision Processes optimised with reinforcement learning have been shown to reduce dialogue length or increase user satisfaction compared to handcrafted systems. While optimising dialogue strategies for recognition errors is an important research topic, we focus in this work on the problem of determining whether, and how, the user should be prompted for more information.

1.2 Our Approach

Our central idea to dialogue management for restricted domain question answering systems is to defer much of the work to the natural language generation component. The dialogue manager is only responsible for choosing one among a set of predefined actions (the equivalent for a task-oriented dialogue manager would be just to determine whether to prompt for a new slot filler or to confirm a filled slot, but without deciding which slot to prompt or to confirm). This allows us to cast the dialogue management problem as a multi-class classification problem, with a small number of classes. The concrete realisation of the action to be taken is carried out by an instance-based generation component that modifies example sentences from that chosen class to the given context and presents them to the user. The generation component is described in detail in Denecke and Tsukada (2005).

Besides the absence of a structured task model we need to address the problem of data sparseness. It is difficult to obtain sufficient data such that a complex dialogue function can be learned reliably (see also Section 3). For this reason, we intend to reduce the complexity of dialogue management, and assign some of the tasks usually performed by a dialogue manager to the generation component.

It could be argued that moving responsibility from the dialogue manager to the natural language generation component is simply an engineering concern. However, we exploit the fact that an instance selection mechanism is already part of any instance-based generation component. We abuse this selection mechanism by assuming that the selection will not only appropriately determine the form, but also the content of the prompt to the user.

Since the action to be taken depends both on the dialogue context and the retrieved documents of the question answering system, both need to be taken into account by the learning algorithm. More specifically, we are interested in learning a function

$$a = f(c, l)$$

that, given a dialogue context c and an n-best list of retrieved documents l, decides an appropriate action a for the dialogue manager. In particular, we wish to learn a function that answers the two questions:

1 Does the user need to be prompted for more information?

2 If so, how should the user be prompted?

For classification, we use Support Vector Machines (Vapnik, 1995). The choice of Support Vector Machines is motivated by two factors. First, they are capable of supervised learning in high dimensional feature spaces, a fact that has been amply demonstrated in the literature. Second, it is possible to introduce a learning bias, or domain knowledge, by means of the choice of the kernel function. As we would like to investigate how best to learn the dialogue function, we compare the efficacy of several kernel functions.

Levin et al. (2000) argue that supervised learning is not appropriate for optimising a dialogue strategy, but see Henderson et al. (2005) for a hybrid approach to learning dialogue strategies, including supervised learning. We emphasise again that our goal is not the optimisation of a dialogue strategy. Instead, our goal is to determine whether the current dialogue context is lacking information, and if so, how it best can be obtained, given past experience.

To summarise, our approach can be described as follows. After processing the current user utterance, a multi-class classifier assigns one out of a few labels to the current dialogue state. The label constrains the form of action to be generated by the dialogue manager. Subsequently, an instance-based generation component retrieves an utterance from a corpus that is similar to the one to be generated. The retrieved utterance is adapted to the current context by replacing content words and presented to the user.

1.3 Our System

We implemented an interactive restricted domain question answering system that combines features of question answering systems with those of spoken

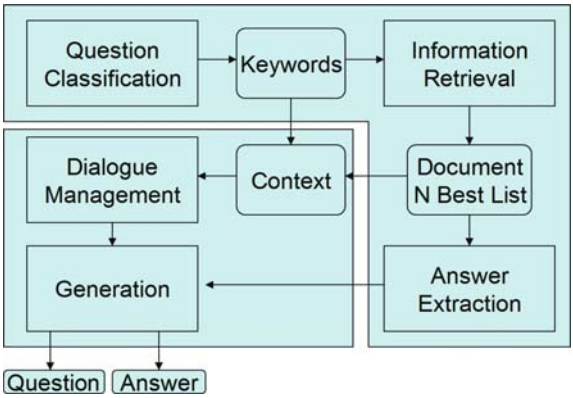


Figure 1. Overview of the system.

dialogue systems. The system has been developed for Japanese, and the task we chose is to inform users on several aspects of travel destinations in Japan. This resulted in a system that, from the standpoint of vocabulary size, is located somewhere between question answering systems and spoken dialogue systems. It is so large that domain modelling as done in spoken dialogue systems typically becomes impractical due to the required manual labour, but not large enough to answer all questions one might think of.

We show an overview of our system in Figure 1. In particular, the relationship between the question answering system and the dialogue manager can be seen. The user input is passed on to the question answering system which extracts keywords and classifies the question. The determined information is integrated with the context and passed on to the information retrieval engine which returns an n-best list of documents that best fit the question. Subsequently, the dialogue manager needs to determine whether to present the answer extracted from the highest ranking document to the user or whether to inquire additional information from the user.

2. Background

Task-oriented dialogue systems were among the first commercially available natural language processing systems. The manageability of task-oriented systems stems from the fact that natural language processing, when limited to a specific task, can impose several simplifying assumptions. Thus, the natural language processing aspect of it becomes more manageable.

However, the simplifying assumptions are not limited to the size of grammars or the possibility to use template-based natural language generation. Dialogue management proper benefits from the fact that the purpose of the dialogue is the interaction of the user with a back-end system with

limited functionality. Interactive question answering systems whose back-end application is a generic information retrieval system do not have the benefit of structure in the way task-oriented dialogue systems have.

2.1 Structured and Unstructured Information for Dialogue Processing

In this section, we point out the ways in which the structure of the back-end application has been exploited in dialogue managers. In the following section, we discuss different approaches to interactive question answering systems.

Given the nature of the application, database schemata of relational databases are one of the most frequently used sources of structured information in dialogue managers. A database schema can be seen as a form of type information on the information stored in databases, much the same way as an XML Schema is a typing scheme for XML documents. For example, Ferrieux and Sadel (1994) describe an algorithm that determines the slots to prompt the user for. The algorithm is based on the database schema and the records retrieved from the database. It is the structure of the database, as represented in the schema, together with the number of potential fillers, that determines which values to prompt for. Similarly, Denecke and Waibel (1997) propose a data driven approach to determining which questions to ask based on the structure of an ontology. The difference to the work by Ferrieux and Sadek (1994) is that the ontology adds structure to the retrieved record sets so that cascading follow-up questions, leading to the filler of only one slot, can be asked. Also, using subsumption information, potential fillers can be inferred from the ontology in case database access is not possible, for example because the result set is too large. Flycht-Eriksson and Jönsson (2003) propose to use a domain specific ontology to enhance the information contained in free form text. A different problem was tackled by Rudnicky and Wu (1999) who introduced task models to dialogue management. Here, it is the structure of the domain that guides the sort of questions that are asked. A task model is a hierarchical representation of tasks that make up the dialogue. At any point, one task is active. Depending on the interaction with the user, the system moves to another task, a subtask, or repeats the current task. In all three examples, task-specific structure helps to answer the two questions outlined in the introduction.

2.2 Question Answering Systems and Dialogue Systems

One of the earliest systems in which a free text database could be queried in natural language was proposed by Wilensky et al. (1984). The queried text consisted of the Unix manual pages, and while the system had limited dialogue capabilities, contextual information could be processed.

More recently, Kiyota et al. (2002) proposed an interactive question answering system that helps users troubleshoot problems with computer systems. In case the user does not present all information necessary to determine the correct help text, a dialogue manager detects vague questions and, if necessary, prompts the user for additional information. The prompting is based on so-called *dialogue cards* which can be seen as simplified dialogue scripts. If the user's question matches one of a list of questions on the dialogue card, predetermined information seeking questions associated with the dialogue card are presented to the user. However, the approach using dialogue cards cannot be easily extended to an open-domain interactive question answering system since the cost of creating the dialogue cards would be prohibitive.

In an updated version of the system, an information gain criterion is proposed to decide which question to ask (Misu and Kawahara, 2005). As in the case of the dialogue cards, a set of pre-prepared questions is provided.

One approach to determine whether to pursue a dialogue is to match the question with the retrieved answer. In case the match is poor, it is assumed that the information provided by the user is not specific enough; therefore the system engages in dialogue. Matching between questions and answers, albeit with a different motivation, is described in Brill et al. (2001). In order to avoid question classification, Brill et al. (2001) proposes to reformulate the query into a declarative sentence and to rank documents retrieved by a search engine based on whether (and how often) a given document contains a sentence with similar structure. The query reformulation takes place *en lieu* of question classification.

Hori et al. (2003) propose an interactive voice question answering system in which a list of information seeking questions is hypothesized for each user input. The information seeking questions are generated by using templates in which chunks of the user question are inserted. The template depends on the type of the question as determined by the question answering system. For each hypothesized question, an ambiguity score is calculated. This score depends on the result set and the phrase inserted in the template. If there is a disambiguation question with a score higher than a given threshold, the question is asked; otherwise the answer is generated.

In order to increase the structure of the documents returned from the question answering system, Small et al. (2004) propose to extract information using various means from the retrieved documents. The extracted information is then represented in frame-like structures which allows the application of methods known from task-oriented dialogue management.

The work on interactive question answering systems cited above and proposed here is different from the work proposed in De Boni and Manandhar (2003) in that here, the authors are interested in detecting whether a question from a user is a follow-up question to a previous question or not. This contrasts

with our desire to determine the need for system initiated clarification dialogues and the generation thereof. Of course, from an interaction perspective, the differences in approaches result in different degrees of system initiative, but the underlying technologies are also different.

2.3 Support Vector Machines

In recent years, Support Vector Machines have been applied successfully in several pattern recognition applications (Vapnik, 1995). Their popularity is due to the fact that they can be applied in high dimensional feature spaces without actually incurring the high cost of explicitly computing the feature map. Support Vector Machines are instances of supervised learning algorithms. A supervised learning algorithm attempts to learn a decision function $f : \mathcal{X} \rightarrow \mathcal{Y}$ from labelled examples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$. In the case of Support Vector Machines, we have $\mathcal{Y} = \{-1, +1\}$.

Given a set of labelled training examples, a Support Vector Machine attempts to determine hyperplanes separating the positive from the negative examples such that the margin between the classes is maximized. In order to improve classification performance, the training examples are separated not in the input space but in some high-dimensional feature space. The reason this can be done efficiently is that instead of determining the image $\phi(\mathbf{x}_i)$ of the training examples in the feature space and calculating their inner product $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ there, the distance is calculated implicitly by the *kernel function* $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$.

2.3.1 Multi-class classification. As described in Section 1.2, we would like to classify the dialogue and the user input into one out of n classes. The classification is used to determine what kind of output to the user is to be generated. Section 3 gives an overview of how the number and meaning of the classes is determined.

However, standard Support Vector Machines provide only binary classification. There are several proposals on how to do multi-class classification using Support Vector Machines. *One-vs-all* classification is a simple approach in which one binary classifier is trained separately for each class, whereby the members of that class are given as positive examples and members of all other classes are given as negative examples. During classification, each binary classifier outputs a distance of the input to the closest decision boundary. This distance can be used as an indicator of "how sure" the classifier is regarding the classification. Therefore, the classifier outputting the largest positive number wins and determines the class of the input.

An alternative is to follow the approach of Weston and Watkins (1999), where a form of voting takes place during classification.

2.3.2 Convolution kernels. Initially, kernel-based learning algorithms have been applied to various tasks in attribute-value learning in which attributes are represented as components x_i of vector $\mathbf{x}$. This approach does not scale well to domains such as natural language processing in which the representation of structure is necessary in order to achieve good classification performance. While *Bag-of-Words* techniques can be applied as an approximation to derive features for classifiers, the loss of structure is not desirable. To address this problem, Haussler (1999) proposed *convolution kernels* that are capable of processing structured objects x and y consisting of components $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$. The convolution kernel of x and y is given by the sum of the products of the components' convolution kernels. This approach can be applied to structured objects of various kinds, and results have been reported for string kernels and tree kernels.

The idea behind convolution kernels is that the kernel of two structures is defined as the sum of the kernels of their parts. Formally, let D be a positive integer and $X, X_1, \ldots, X_D$ separable metric spaces. Furthermore, let x and y be two structured objects, and $\mathbf{x} = x_1, \ldots, x_D$ and $\mathbf{y} = y_1, \ldots, y_D$ their parts. The relation $R \subseteq X_1 \times \ldots \times X_D \times X$ holds for $\mathbf{x}$ and x if $\mathbf{x}$ are the parts of x. The inverse R^{-1} maps each structured object onto its parts, i.e. $R^{-1}(x) = \{\mathbf{x} : R(\mathbf{x}, x)\}$. Then the kernel of x and y is given by the following generalised convolution:

$$K(x, y) = \sum_{\mathbf{x} \in R^{-1}(x)} \sum_{\mathbf{y} \in R^{-1}(y)} \prod_{d=1}^{D} K_d(x_d, y_d)$$

Informally, the value of a convolution kernel for two objects X and Y is given by the sum of the kernel value for each of the sub-structures, i.e. their convolution.

Collins and Duffy (2001) described the application of convolution kernels to several natural language processing tasks, focusing on the case where the substructures convoluted by the kernel are trees.

Suzuki et al. (2003) proposed *Hierarchical Directed Acyclic Graph* kernels in which the substructures contain nodes which can contain graphs themselves. The hierarchy of graphs allows extended information from multiple components to be represented and used in classification. In addition, nodes may be annotated with attributes, such as part of speech tags, in order to add information. For example, in a question answering system, components such as named entity extraction, question classification, chunking and so on may each add to the graph. Figure 2 shows the graph structure of a sentence after having been preprocessed; this graph structure is the structure that is processed by the HDAG kernel.

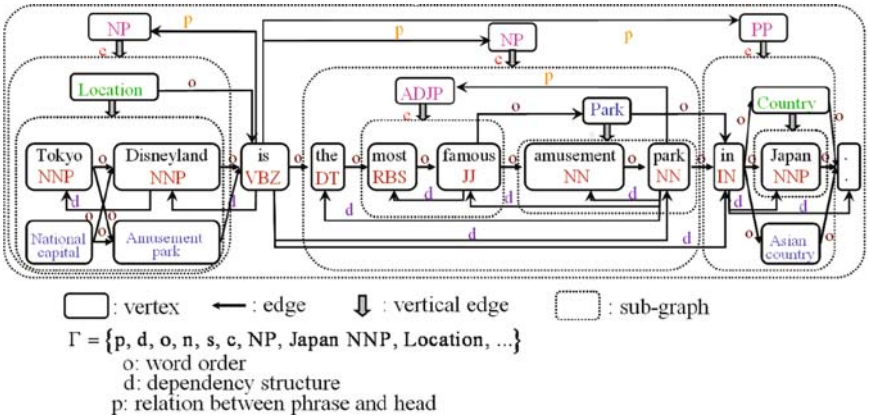


Figure 2. An example of hierarchical sentence structure.

The way to arrive at the graph structures (and the one we follow) is to subject each sentence to an extensive analysis. Since in Japanese, words are not separated by spaces, the input sentence is broken down into words and analysed morphologically by ChaSen (Asahara and Matsumoto, 2000). Based on this analysis, dependency analysis is done using CaboCha (Kudo and Matsumoto, 2002). With each node a word stem is associated with semantic information extracted from "Goi-Taikei" (Ikehara et al., 1997), a semantic network similar to the English WordNet.

3. Corpus

We collected a corpus for our instance based generation system as follows. We set up communications between a wizard and users.

3.1 Data Collection

The subjects were instructed to ask travel-related questions. The subjects' utterances were recorded and later transcribed. The wizard, placed in a different room, would engage in the dialogue by typing her answers and displaying it on the subjects' screen.

The questions the subjects were instructed to ask comprise what kind of accommodation is available, if the destination is known for special food (various areas in Japan are renowned for food that can be had only there), famous sightseeing spots and events, if the destination is important for the history of Japan, and if so, how. The restriction on the kinds of question was motivated by the fact that certain kinds of questions are not well suited for interactive restricted domain question answering systems. In particular, simple factual questions (such as: "How much is a stay in the hotel for three nights?") are better

handled by task-oriented dialogue systems, a problem we did not want to address in this study. Altogether, 20 users participated in the data collection effort. The participating users were not computer experts, and did not participate in similar research experiments before.

The wizard was instructed to "act like the system" we intend to build, that is, she was required to interact with the user either by prompting for more information or by giving the user the information she thought he wanted.

3.2 Corpus Properties

Each of the 20 users contributed 8 to 15 dialogues. The length of the dialogues varies between 11 and 84 turns, the median being 34 turns. Altogether, the corpus consists of 201 dialogues. The corpus consists of 6,785 turns, 3,299 of which are user turns and the remaining 3,486 are wizard turns.

The instructions imposed on the wizard in the onset of the data collection caused each dialogue to consist either of an equal number of user and wizard turns (in case the user ends the dialogue; 14 cases) or one wizard turn more than user turns in case the wizard ends the dialogue (187 cases). For the same reason, misunderstandings were rare, and corrections did not occur often. In the implemented system, no special form of discourse processing takes place, although, if suitably implemented, it is expected to increase performance. Due to the nature of the Japanese language, phenomena interesting to discourse processing, such as ellipses and anaphora, are mostly realised through omission (implicit contextual understanding is often required in Japanese).

Figure 3 shows the first part of a dialogue from the corpus.

W: こちらは旅行ガイドです
This is the travel guide.
U: 京、京都で１日ゆったりとお寺巡りをしようと思っているんですが、京都駅から、お寺を、
お寺巡りをする、効率的というか、こうしたらいいんじゃないかっていうコースはありますか
I was thinking to take a small trip to Kyoto to see some temples, starting
from Kyoto station, is there something like an efficient course to do this?
W: はい、それでは京都駅周辺でお調べ致します
京都駅周辺をゆったりと回る、でよろしいですか
Yes then I will look for something around Kyoto station.
A quiet trip around Kyoto station, would that be okay?
U: はい
Yes.
W: それではJR京都駅から東寺、梅小路蒸気機関車館、門屋もてなしの文化美術館、西本願寺、
コウセイジ、京都タワー、そしてJR京都駅に戻ってくるというコースはいかがでしょうか
Then starting from Kyoto station, going to the Touji temple, to a steam train museum,
Kadoya museum, Nishi-Hongan temple, Kousei temple, Kyoto tower and back to Kyoto station,
would that be okay?

Figure 3. An extract from the dialogue corpus used. The letter *'U'* identifies user utterances, the letter *'W'* identifies wizard utterances.

3.3 Corpus Annotation

We recall from the introduction that the purpose of the system-generated questions is to integrate additional information from the user in the information retrieval. However, due to the absence of a structured task model in restricted domain question answering systems, it is not obvious how to decide whether the system should generate an information seeking question, or return the highest ranking answer from the question answering system to the user. The information based on which this decision can be made consists of the user question and the retrieved documents, together with the answers as extracted by the question answering system.

In task-oriented dialogue processing, the *specificity* of the information acquired from the user can be stated explicitly in terms of the task model (all slots are filled, or confirmed etc.). This is not possible in our situation. Therefore, we attempt to capture the specificity of the acquired information indirectly. We assume that the wizards' reaction to the user's utterance expresses the specificity of the acquired information: If the wizard asks a question, more information is needed, if the user gives the desired answer, the acquired information was sufficient to complete the task. Therefore, the key idea of our approach is to observe the user–wizard exchanges, label user-wizard utterance pairs according to specificity and learn a classifier that approaches the behaviour of the wizard.

To this end, each dialogue is divided in utterance pairs. Each utterance pair contains a user turn followed by a wizard turn. Each utterance pair is manually classified into one out of 8 classes according to the schema shown in Figure 4.

```
if (wt is not question)
    if (ut is yes/no question) assign 1
    else if (ut is enumeration question) assign 2
    else if (ut is wh question) assign 3
    else assign 4
else
    if (wt is yes/no question) assign 5
    else if (wt is enumeration question) assign 6
    else if (wt is wh question) assign 7
    else assign 8
```

Figure 4. Annotation algorithm used to annotate the dialogue corpus (ut and wt refer to *user utterance* and *wizard utterance*, respectively).

Table 1. Distribution of utterance pair classes before and after cleaning. The classes are sorted according to specificity of the question-answer pair, i.e. class 1 is a yes-no question asked by the user that is answered by the wizard, whereas class 8 corresponds to an interaction in which the user asks some question which is followed up with a counter question by the wizard.

Label	1	2	3	4	5	6	7	8
Raw	530	2	2,209	3	454	23	76	2
Clean	530		2,211	0	454		99	0

According to this annotation scheme, the exchange *"I would like to take a weekend trip to Hakone." – "Would you like to stay in a hot spring resort or in a standard hotel?"* would be classified as class six.

After annotation we found that the distribution of labels is extremely skewed, as instances of classes two and eight have been observed only twice, respectively, in the entire corpus. For that reason, we merged classes two and three, and classes six and seven, respectively. Moreover, we removed instances of classes four and eight from the corpus. The raw and clean counts of the classes in the dialogue corpus are distributed as shown in Table 1.

To the best of our knowledge, the proposed annotation scheme is different from other schemes that are usually adopted for dialogue annotation. The motivation behind this annotation scheme is to use the obtained data to learn the dialogue classifier. Therefore, we deliberately ignore information such as dialogue structure which is in contrast to other annotation schemes.

4. Representations and Dialogue Management

Before we discuss our choice of kernel functions, we detail the function f we would like to learn. The representations to be classified need to capture two aspects of the dialogue state: (1) how specific is the information gathered so far, and (2) how well does the highest ranking document from the current n-best list answer the users' question? The first aspect is expressed by the context representation, the second by the relationship between the gathered information and the highest-ranking document in the n-best list.

In the following sections, we describe how features for the representation of context c and the answer list l are extracted.

4.1 Context Representation

We choose a simple discourse representation in which the user turns as well as information extracted from them are stored. More specifically, we memorise the user turn ut_T at time T, the set of keywords $kw(ut_T)$ extracted from ut_T by the question analysis module and the question type $qt(ut_T)$ determined by

the question classifier. At time T, we also have access to an n-best list of documents $A_T = \langle a_T^1, \ldots, a_T^n \rangle$ which are returned from the question answering system.

The dialogue state at time T is then given by the three lists $\langle ut_1, \ldots, ut_t \rangle$, $\langle kw(ut_1), \ldots, kw(ut_T) \rangle$ and $\langle A_1, \ldots, A_T \rangle$. A dialogue state contains too much information to be used directly for classification purposes. We discuss several ways of extracting features relevant for classification in the following sections.

User input and context in spoken dialogue systems is often represented as fillers of a predefined set of slots. A straightforward adaptation to interactive restricted domain question answering systems is not straightforward, because it is difficult to determine the set of slots necessary for the application at hand. Since we intend to use the representations for classification purposes, we adopt a *bag of words* approach that is often used in kernel-based methods.

4.1.1 Pure context. While in task-oriented dialogue systems, the representation of context is an accumulation of slot-filler parts of some form, we cannot resort to this technique, because a predefined set of slots is not given. Therefore, we propose to generalise slot-filler representations such that the *information relevant to the information retrieval process* is memorised. We arrive at a bag-of-word representation of a sentence by tokenizing and tagging the sentence (Japanese does not use space between the words). Then, all content words are added to the bag of words. However, the chosen content words may be too specific or too vague for classification purposes. For that reason, we add all hypernyms of all present nouns to the set.

In other words, our first proposal for context representation at time T is given by

$$C_T = Cl\left(\bigcup_{t=1}^{T} kw(ut_t)\right) \tag{9.1}$$

where Cl is the closure function mapping a set of words to a larger set that also contains all hypernyms. Informally, this is the set of all keywords extracted from the user utterances up until now, augmented by the closure of concepts to include hypernyms.

4.1.2 Filtered through results. In information retrieval systems using large corpora co-occurrence between terms is used as an additional source of information. In other words, the fact that two terms appear in the same documents more often than chance is used to address problems such as synonyms. When representing the context for classification, we would like to take advantage of this concept as well. We can do this in an indirect way by extracting relevant terms from the retrieved documents and add them to our

context representation. This can be done as follows. Let N be the set of all nouns occurring in the dialogue training corpus. Furthermore, the document a_T^1 refers to the highest ranking newspaper article retrieved at time T. An approximation of those nouns in the document relevant to the travel domain can be given by the intersection of the set of all nouns in the document with the set N. This leads to the following equation.

$$C_T = Cl\left(\bigcup_{t=1}^{T} kw(ut_t) \cap N \cap Cl(a_T^1)\right) \tag{9.2}$$

4.2 Answer Representation

In addition to a representation of the context, it is possible to represent the degree to which the retrieved documents answer the question asked by the user. We assume that this feature can be expressed by a relationship between the representation of the context and the retrieved documents.

4.2.1 Content word intersection. A straightforward approach to extract a relationship between the informational content in the context and in the retrieved document is to determine bag-of-words representations for each and calculate the intersection:

$$A_T = Cl\left(\bigcup_{t} kw(ut_t) \cap answer(a_T^1)\right) \tag{9.3}$$

where $answer$ is a function that takes a document and extracts the sentence that (according to the question answering system) constitutes the answer to the user's question.

4.3 Alignment

A potential drawback to the bag-of-words representation described above is the loss of information that occurs as sentence structure is ignored. For this reason, our second approach is to determine an alignment between the user input sentence and the answer sentence extracted from the document. Given a question and a list of answer candidates, we submit the question and each answer candidate to several steps of linguistic analysis, and try to match the resulting structures of the question with each of the structures of the answer candidates. If there is a close match, the answer candidate is considered a valid answer. This approach is motivated by the work by Brill et al. (2001) described above.

The purpose of this approach is to build a classifier that determines whether the sentence shown in Figure 5 (b) is an answer to the query shown in

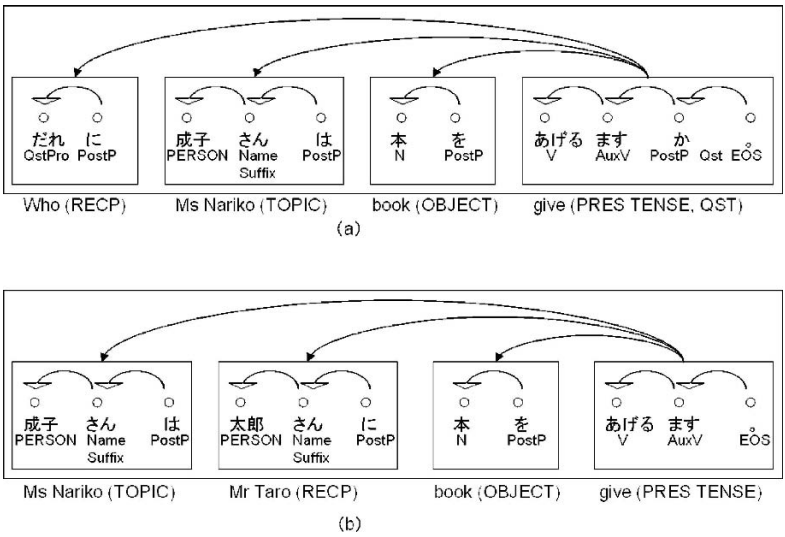


Figure 5. Analysis.

Figure 5 (a). In other words, we are interested in classifying the relation between sentence 5 (a) and sentence 5 (b). Therefore, we need to add structure to 5 (a),(b) that makes explicit the relationship between the dependency structures. We do this by aligning the dependency trees for the question and the answer with each other. Then, we form a graph consisting of the dependency tree of the question, the dependency tree of the answer and the edges representing the alignment. In the following, we refer to the surface form of the question and answer as q and a, respectively, their dependency trees as $t(q)$ and $t(a)$, and a possible alignment of $t(q)$ with $t(a)$ as $A(q, a)$. Figure 6 shows a resulting alignment. In contrast to the features described above, we cannot use a standard kernel to classify the extracted structure. Instead, we use the Hierarchical Directed Acyclic Graph kernel described above for classification purposes.

It counts as a positive example if the presented answer is indeed the correct answer to the question and as a negative example if not. In the following, we present the preprocessing before the alignment and propose two different ways to perform the alignment.

4.3.1 Preprocessing. Before the alignment can take place, we need to perform some preprocessing in order to address the differences between question and answer. We are primarily concerned with two issues. First, we need to address the fact that nodes containing the question word (such as *who, what, where* and so on) need to be treated differently. We do

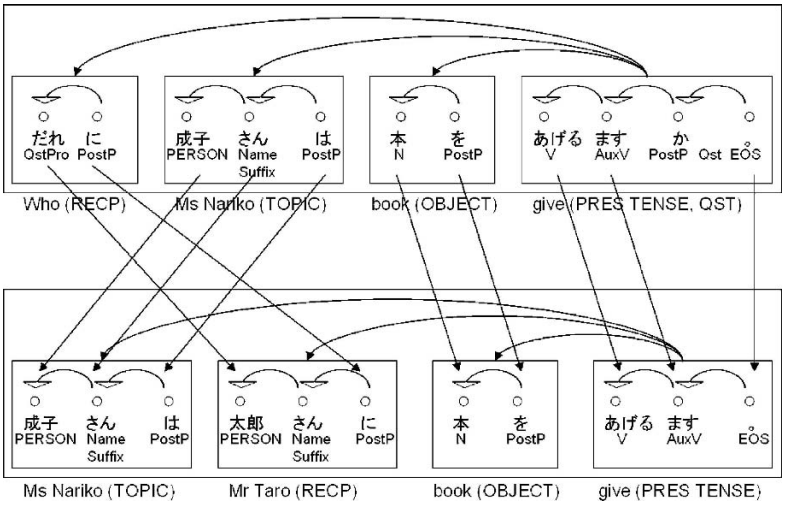


Figure 6. One possible alignment of the dependency trees shown in Figure 5. The alignment of the chunk nodes is not shown for clarity, but is added to the graph as well.

this as follows. From the training corpus, we associate with each question type the used question words. Then, in order to identify the node containing the question word in the dependency tree of the question, we determine the nodes that are associated with a word in the list.

4.3.2 Detailed alignment. The detailed alignment is an attempt at aligning as many nodes as possible in $t(q)$ with nodes in $t(a)$. In order to abstract away the particular realisation of question and answer, we do not consider nodes whose part of speech tag is not a noun, verb, adjective or adverb. In the following we refer to a node from the question or answer dependency tree as n_q, and n_a, respectively. The ith attribute of a node n is indicated by $attr_n(i)$. The part of speech of n is written as $pos(n)$. The ith node contained in n can be accessed by $node(n, i)$. We distinguish between word and chunk nodes. A word node is a node containing a word appearing in the surface form of the question or answer, but does not contain other words. Examples of word nodes include the nodes labelled with the words "Nariko" or "book" in Figure 5. A chunk node is a node containing other nodes. Chunk nodes are shown as square boxes in Figure 5. We are interested in determining the value $V(n_q, n_a)$ of aligning node n_q with node n_a. We distinguish three cases: the value of the alignment if both nodes are word nodes, the value of the alignment if both nodes are chunk nodes, and the remaining case. A value of $V(n_q, n_a) = -1$ represents the fact that n_q and n_a are not aligned.

We first consider cases in which both n_q and n_a contain the representation of a word. The nodes carry information on the surface string, the uninflected word, the part of speech, and semantic tag. The value of the alignment between two word nodes is given by the number of attributes contained in the question node that are also contained in the answer node. This characteristic is intended to capture the fact that information present in the answer but not in the question does not hurt the association, while the other way round it does.

$$V(n_q, n_a) = \begin{cases} |\{i : attr_{n_q}(i) = attr_{n_a}(i)\}| & \text{if } pos(n_q) \in list \\ -1 & \text{otherwise} \end{cases}$$

A chunk node contains references to word nodes, and, since the graphs are hierarchical, other chunk nodes. For alignment purposes, we consider a chunk node a set of nodes. We do not take those word nodes into account. Given two chunk nodes, the cost of associating between them is calculated recursively according to

$$V(n_q, n_a) = \max_{\pi \in \Pi(s)} \sum_j V(node(n_q, j), node(n_a, \pi(j)))$$

where $\Pi(s)$ is the set of all permutations of $\{1, \ldots, s\}$ and s is the number of nodes contained in n_q.

The value $V(n_q, n_a)$ equals 0 if one node is a word node while the other is a chunk node.

Dynamic programming is used to determine a minimal association between the structures of the query and the potential. A solution to the example sentences from Figure 5 is shown in Figure 6. Once the alignment has been determined, the edges $\langle n_q, n_a \rangle$ for which $V(n_q, n_a) > 0$ are added to the graph. In addition, for each node n in $t(q)$ that is not aligned with some node in $t(a)$, we add a new node n' and align n with n'. The resulting structure is shown in Figure 6.

4.3.3 Sloppy alignment. Sloppy alignment is identical to detailed alignment except that after alignment only those edges $\langle n_q, n_a \rangle$ for which $V(n_q, n_a) > 0$) such that n_q is a chunk node are added to the graph. Figure 7 shows the sloppy alignment.

4.3.4 Thinning. In order to avoid memorisation, we remove features from the graph structures. We propose to thin out the graph and make the alignment structure more visible. The resulting graph of the example is shown in Figure 8.

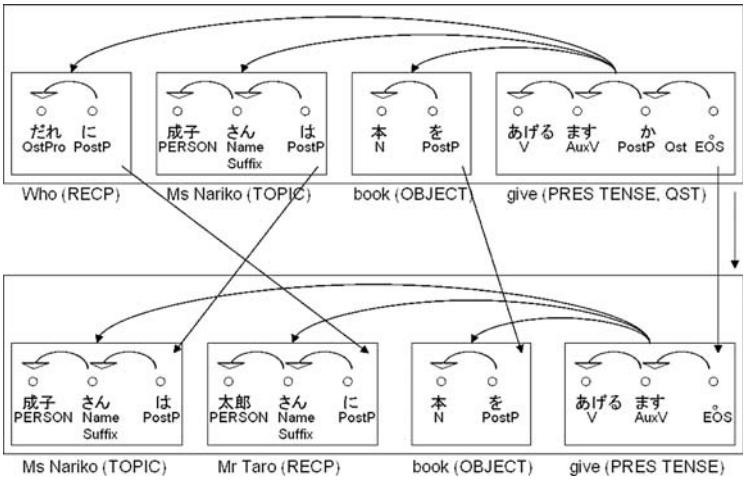


Figure 7. One possible sloppy alignment of the dependency trees shown in Figure 5.

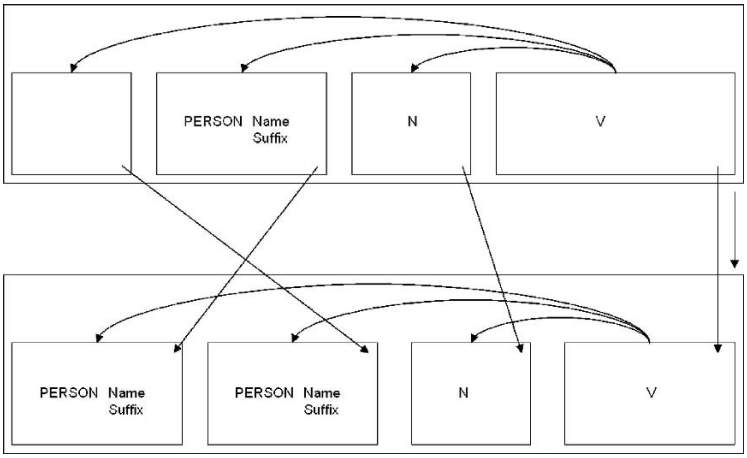


Figure 8. Alignment shown in Figure 7 after thinning.

5. Dialogue Management

After having discussed the choice of representations, we proceed to describe dialogue management based on the representations. We recall from the introduction that our approach to dialogue management is to classify the dialogue state such that the resulting class is an indicator of the kind of action to be taken. The responsibility for carrying out the chosen action lies entirely with the generation component. Generation is an instance-based approach which takes sentences from the corpus and adapts them to the current dialogue situation as necessary.

5.1 Classifiers and Kernels

For our experiments with binary classifiers, we use TINYSVM as an implementation of the SVM classifier. To choose the winning class, we use *one-vs-all* classification. For our experiments using multi-class classification, we use an implementation of the classifier described in Weston and Watkins (1999).

5.2 Dialogue Management Algorithm

The dialogue management consists mainly of updating the chosen context representation, generating the input representation to the classifier and passing on the result to the instance-based generation system. The dialogue management algorithm is shown in Figure 9.

5.3 Instance-Based Generation

Given a query q asked by the user, and an answer n-best list $\langle a_1, \ldots, a_n \rangle$ generated by the question answering system, the classifier determines for each alignment $A(q, a_i)$ whether a_i is an answer candidate for q, and if so, the answer is added to the content for the information seeking question.

In order to turn the action a as determined by the classifier, into an acceptable utterance, we are pursuing an approach of instance-based generation. In this approach, appropriate sentences are selected from the dialogue corpus, adapted to the current context where necessary, and finally presented to the user. In short, the instance-based method for information seeking question generation works as follows. A dialogue with "similar context" is retrieved from the example corpus, the information-seeking question following the "similar context" is chosen, the content words in this question are determined and

Input User query
Output Utterance type

Step 1: Perform standard question classification
and keyword extraction on user query
Step 2: Add extracted features to dialogue context
Add hypernyms to dialogue context
Step 3: Perform standard IR using result of step 1
Add retrieved documents to dialogue context
Step 4: Classify dialogue context to obtain utterance type
and pass on result to instance-based generation

Figure 9. The dialogue management algorithm.

exchanged with content words from the current corpus, and the question is presented to the user.

As a consequence of our choice to defer specific content selection to natural language generation, our instance-based generation algorithm receives much less specific information compared to task-oriented dialogue systems. It is fairly typical to have the dialogue manager decide at least the slot to be filled, if not the template to choose for generation. In our approach, however, the input to the generation algorithm is only the output of the classifier. Likewise, the dialogue state is accessible to the generation algorithm for sentence selection and adaptation.

We refer the reader to Denecke and Tsukada (2005) for details on the generation algorithm.

5.4 An Example Dialogue

The example dialogue transcript shown in Figure 10 was obtained during a user study with a user that had not used the system before.

5.5 Comparison to Related Work

We compare our implementation with the three interactive question answering systems DIALOGNAVIGATOR, HITIQA and SPIQA shown in Table 2. Table 2 summarizes features of the system relevant for our discussion.

In the case of the DIALOGNAVIGATOR, the users' input is matched against the questions on the dialogue cards. The earlier version of the DIALOGNAVIGATOR relies only on the information on the dialogue cards to determine what to ask whereas the later version uses an information gain criterion to select

U：横浜の観光施設を教えてください
：Please tell me about sightseeing attractions in Yokohama.
S：どういった観光施設をお期待するんですか
：What kind of sightseeing attractions are you looking for?
U：家族で楽しめるところを
：Something to enjoy with my family.
S：それではシルク市場や、横浜開港資料館はいかがでしょうか
：In that case, how about silk market or Yokohama Port Museum?
U：横浜開港資料館はどんなところですか
：What is Yokohama Port Museum?
S：じゃ横浜明治の冊にはけっこういろいろあるって訳ですよね
：Then you mean that Yokohama Meiji has many <unintelligible>?
U：横浜にはどんなイベントがありますか
：What kind of events are currently taking place in Yokohama?
S：乗船会
：Meeting for sailboat fans.

Figure 10. Example dialogue. The words replaced by the instance-based generation are highlighted in grey. In one instance, the generation component generated an utterance part of which were nonsensical.

Table 2. Features of implemented interactive open domain question answering systems.

System name	**Whether to ask a question?**	**How to decide what to ask?**	**How to generate?**	**Structure**
Dialog navigator	Match against list of questions	Dialogue cards Information gain (New version)	Dialogue cards	Handcoded
Hitiqa	Conflicts, missing fillers in frames, scoring	Conflicts, missing fillers in frames	Templates?	Extracted WordNet
Spiqa	Scoring	Wh question, specialising ambiguous phrase	Templates	Extracted
Our system	SVM classifier	Instance selection in NLG Adaptation to current context	Instance-based NLG	Keyword extraction Ontology

among multiple questions. Finally, generation of the prompt is template-based and done by presenting the question associated with the selected dialogue card. The consequence of these design decisions is that the behaviour of the DIALOGNAVIGATOR is easier to control than in our implementation as dialogue cards can be added and removed as needed. On the other hand, this approach requires a substantial amount of handcrafting and testing. While it is possible in our case to add or remove (handcrafted) questions to the instance base to force the system into a certain behaviour, such tweaking is not possible for the dialogue manager as it relies on the output of the classifier. Another difference is that DIALOGNAVIGATOR relies on explicitly coded templates for generation while in our approach the prompts can be generated from a corpus without the need to code templates.

HITIQA employs several text processing techniques to arrive at a slot/filler representation where the vocabulary for slots and fillers is determined by an ontology (WordNet). The slot/filler representation allows HITIQA to use dialogue algorithms similar to the ones used in spoken dialogue systems. For instance, multiple fillers for one slot trigger a clarification question. In our system, we use the information contained in an ontology indirectly by relying on the question type and key word extraction algorithms of the question answering system, and directly by adding hypernyms of the extracted key words to the context as described above.

In the case of SPIQA, whether to ask for more information is decided based on the scores associated with the retrieved texts. The prompts are generated based on sentence fragments extracted from the retrieved text. Strong assumptions are made on the form of the prompts; only wh-questions are permitted and it is assumed that missing information can be concatenated to the sentence fragments by means of the Japanese particle "no". In our case, the form of the prompts is only limited by the sentences in the corpus, as there are no *a priori* limitations in the generation algorithm.

6. Evaluation

6.1 Methodology

The purpose of this work is to determine how best to decide on the action the system should take. Therefore, we train different classifiers on the representations described in Section 4 and determine how well they perform on an unseen section of the corpus. We combine the representations obtained by equations 9.1 and 9.2, respectively, with the representations obtained by 9.3 to yield the input to our classifier. Using these data, we train two one-vs-all systems and two multi-class classifiers, respectively, yielding a total of four different classifiers.

We compare those classifiers trained on unstructured data against two classifiers trained on hierarchical graph structures. In order to combine the alignment structure obtained by the algorithm described above with the contextual representation, we introduce a new graph node in the structure and add all contextual features as children to this node. This results in six classifiers to compare.

We used a training set consisting of 151 dialogues to train the four different versions of Support Vector Machines. The Support Vector Machines were then used to predict the labels of the unseen corpus. The parameter C (a training parameter for Support Vector Machines indicating sloppiness) was set to 1 in all experiments.

6.2 The Question Answering System

For our experiments, we used the Japanese question answering system SAIQA (Sasaki, 2002) as the building block. The text resources the system accesses are the complete documents of the *Mainichi Shimbun* (a Japanese newspaper) dating from 1991 to 2002.

6.3 Results

We obtained a total of six different classifiers. The results of the classifiers are shown in Table 3. It can be shown that the context feature extraction according to equation (9.2) combined with the answer feature extraction according to equation (9.3) generalises best to the unseen section of the corpus. We believe the reason that the structured alignment does not work better is that the alignment is between one user utterance and one sentence in the retrieved document. In order to work well, more context should have been taken into account.

We also tried radial and neural kernels instead of the polynomial kernel, but found problems with both of them. The radial basis kernels showed excellent accuracy and precision, but had recall below 30% for underrepresented classes. The training of SVMs with neural kernels would not converge, even when setting the parameter C to unusually large values.

Table 3. Accuracy of the classifiers.

Classifiers	**Context (%)**	
	Eq. (9.1)	Eq. (9.2)
One-vs-all	76.3	75.3
Multi-class	84.2	80.4
Alignment	62.2	61.4

In initial experiments, we tried to classify the structures resulting from the alignments as shown in Figures 6 and 7. While the performance on the training set was around 90%, performance on previously unseen data was not much better than chance. This suggests that generalisation could not take place. We hypothesize that this is due to the fact that the number of nodes and edges in the linguistic structures of question and answer outweighs the number of nodes and edges of the alignment. In other words, we suspect that overfitting occurs, and therefore the classifier does not generalise well. One way to overcome this problem is to remove some of the information from the alignment structures. We do this by removing all word related information, such as part of speech tag and semantic information. Yet, the simpler bag-of-words (or bag-of-concept, more appropriately) approach still outperforms the structured approach.

Transcriptions of dialogues between naive users and the end-to-end system indicates that the proposed system works well as long as keywords added to the discourse are aligned with the information the user is interested in. However, if the instance-based generation algorithm chooses a keyword in its prompt to the user that is not representative for the information the user is interested in, there is a danger that the user repeats that keyword (in the simplest case saying something like *"No, I did not mean X"*. Subsequently, this keyword will be introduced into the dialogue state. This, in turn, will rank those texts higher that contain this keyword.

7. Summary and Future Work

In this chapter, we presented dialogue management techniques for interactive restrictive domain question answering systems. We proposed to learn classifiers for dialogue management based on representations extracted from dialogue context and retrieved documents from the question answering machines.

The main problem when going from task-oriented dialogue systems to interactive restricted domain question answering systems is that the lack of task structure prohibits making simplifying assumptions as in task-oriented dialogue systems. In order to address this issue, we proposed a solution that consists of two parts: First, we use representations based on keywords extracted from the user utterances and concepts extracted from an ontology. These representations can be seen as a generalisation of the slot/filler representations Future work includes evaluation with new users as opposed to trying to predict the actions in an existing dialogue corpus. Second, we use machine learning to learn the dialogue management function. We are currently preparing an evaluation with users to test the dialogue classification under realistic circumstances.

Acknowledgements We would like to thank Takuya Suzuki for implementing parts of the instance-based generation system. Our thanks go to Hideki Isozaki and the members of the Knowledge Processing Group for discussions

and support. We would also like to thank the anonymous reviewers for helpful comments on an earlier draft of this paper.

References

Asahara, M. and Matsumoto, Y. (2000). Extended Models and Tools for High-Performance Part-of-Speech Tagger. In *Proceedings of International Conference on Computational Linguistics (COLING)*, pages 21–27, Saarbrücken.

Brill, E., Lin, J., Banko, M., Dumais, S., and Ng, A. (2001). Data-Intensive Question Answering. In *Proceedings of the Tenth Text REtrieval Conference (TREC)*, pages 393–400.

Collins, M. and Duffy, N. (2001). Convolution Kernels for Natural Language. In *Proceedings of Conference on Neural Information Processing Systems (NIPS)*, pages 625–632, Vancouver.

De Boni, M. and Manandhar, S. (2003). An Analysis of Clarification Dialogue for Question Answering. In *Proceedings of Annual Meeting of the Human Language Technology Conference/North American chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 48–55, Edmonton.

Denecke, M. and Tsukada, H. (2005). Instance-Based Generation for Interactive Restricted-Domain Question-Answering Systems. In *Proceedings of Second International Joint Conference on Natual Language Processing*, volume 3651, pages 486–497. LNCS, Springer, Jeju Island.

Denecke, M. and Waibel, A. (1997). Dialogue Strategies Guiding Users to their Communicative Goals. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 1339–1342, Rhodes. Available at http://www.opendialog.org/info_papers.html.

Ferrieux, A. and Sadek, M. (1994). An Efficient Data-Driven Model for Cooperative Spoken Dialogue. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 979–982, Yokohama.

Flycht-Eriksson, A. and Jönsson, A. (2003). Some Empirical Findings on Dialogue Management and Domain Ontologies in Dialogue Systems - Implications from an Evaluation of BirdQuest. In *Proceedings of SIGdial Workshop on Discourse and Dialogue*, pages 158–167, Sapporo.

Haussler, D. (1999). Convolution Kernels on Discrete Structures. Technical report, UC Santa Cruz, California.

Henderson, J., Lemon, O., and Georgila, K. (2005). Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR Data. In *Proceedings of IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 68–75, Edinburgh.

Hori, C., Hori, T., Tsukada, H., Isozaki, H., Sasaki, Y., and Maeda, E. (2003). Spoken Interactive ODQA System: SPIQA. In *Proceedings of Annual*

Meeting of the Association for Computational Linguistics (ACL), pages 153–156, Sapporo.

Ikehara, S., Miyazaki, M., Shirai, S., Yokoo, A., Nakaiwa, H., Ogura, K., Oyama, Y., and Hayashi, Y. (1997). *The Semantic Attribute System, Goi-Taikei? A Japanese Lexicon, Volume 1 (in Japanese)*. Iwanami Publishing, Tokyo.

Kiyota, K., Kurohashi, S., and Kido, F. (2002). Dialog Navigator: A Question Answering System based on Large Text Knowledge Base. In *Proceedings of International Conference on Computational Linguistics (COLING)*, pages 460–466, Taipei.

Kudo, T. and Matsumoto, Y. (2002). Japanese Dependency Analysis using Cascaded Chunking. In *Proceedings of 6th Conference on Natural Language Learning (CoNLL)*, pages 63–69, Taipei.

Levin, E., Pieraccini, R., and Eckert, W. (2000). A Stochastic Model of Human Machine Interaction for Learning Dialog Strategies. *IEEE Transactions on Speech and Audio Processing*, 8:11–23.

Minock, M. (2005). Where are the 'Killer Applications' of Restricted Domain Question Answering? In *Proceedings of Workshop on Knowledge and Reasoning for Answering Questions*, pages 98–101, Edinburgh.

Misu, T. and Kawahara, T. (2005). Dialogue Strategy to Clarify User's Queries for Document Retrieval System with Speech Interface. In *Proceedings of 9th European Conference on Speech Communication and Technology (INTERSPEECH)*, pages 637–640, Lisbon.

Rudnicky, A. and Wu, X. (1999). An Agenda-Based Dialog Management Architecture for Spoken Language Systems. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 337–341, Keystone, Colorado.

Sasaki, Y. (2002). NTT's QA Systems for NTCIR QAC-1. In *Working Notes of the Third NTCIR Workshop Meeting*, pages 63–70, Tokyo.

Small, S., Strzalkowski, T., Liu, T., Ryan, S., Salkin, R., Shimizu, N., Kantor, P., Kelly, D., Rittman, R., and Wacholder, N. (2004). HITIQA: Towards Analytical Question Answering. In *Proceedings of International Conference on Computational Linguistics (COLING)*, pages 1291–1297, Geneva.

Suzuki, J., Hirao, T., Sasaki, Y., and Maeda, E. (2003). Hierarchical Directed Acyclic Graph Kernel: Methods for Structured Natural Language Data. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 32–39, Sapporo.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer, New York.

Weston, J. and Watkins, C. (1999). Support Vector Machines for Multi-Class Pattern Recognition. In *Proceedings of Seventh European Symposium on Artificial Neural Networks*, pages 219–224, Bruges.

Wilensky, R., Arens, Y., and Chin, D. (1984). Talking to UNIX in English: An Overview of UC. *Communications of the ACM*, 27(6):574–593.

Zue, V. and Glass, J. (2000). Conversational Interfaces: Advances and Challenges. In *Proceedings of IEEE, Special Issue on Spoken Language Processing*, pages 1166–1180.

Chapter 10

MEETING STRUCTURE ANNOTATION

Annotations Collected with a General Purpose Toolkit

Alexander Gruenstein
Spoken Language Systems Group
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology, MA, USA
alexgru@csail.mit.edu

John Niekrasz and Matthew Purver
Center for the Study of Language and Information
Stanford University, CA, USA
niekrasz@csli.stanford.edu, mpurver@csli.stanford.edu

Abstract We describe a generic set of tools for representing, annotating, and analysing multi-party discourse, including: an ontology of multimodal discourse, a programming interface for that ontology, and *NOMOS* – a flexible and extensible toolkit for browsing and annotating discourse. We describe applications built using the *NOMOS* framework to facilitate a real annotation task, as well as for visualising and adjusting features for machine learning tasks. We then present a set of *hierarchical topic segmentations* and *action item subdialogues* collected over 56 meetings from the ICSI and ISL meeting corpora using our tools. These annotations are designed to support research towards automatic meeting understanding.

Keywords: Topic segmentation; action items; annotation; media; discourse; dialogue; meetings; NOMOS

1. Introduction

The automatic processing and understanding of multi-party meetings has emerged recently as a major area of research. Technically, meetings present

L. Dybkjær and W. Minker (eds.), Recent Trends in Discourse and Dialogue, 247–274.

many interesting multidisciplinary challenges; for instance, they have multiple interacting participants and contain spontaneous speech, movement, and gesture. Commercially, they are interesting as they often involve important decisions, yet they are usually poorly documented. Several major projects studying meetings are underway, including Mapping Meetings,[1] M4,[2] AMI,[3] ISL,[4] IM2,[5] and CHIL.[6]

In this discussion, we view meetings from the perspective of building meeting understanding components which comprise part of the *cognitive personal office assistant* being designed for the CALO project.[7] The types of assistance envisioned include summarising the meeting, actively bringing attention to relevant documents, and helping the collaborative creation of documents in the course of the meeting. Additionally, the content of meetings will be presented in a *meeting browser* which will allow a user to browse a top-level summary, locate pertinent portions, and "drill down" into more detailed structure as desired.

In order to summarise meeting structure in a useful way, it is therefore critical to first understand what sort of structure best assists humans in browsing or reviewing the contents of meetings. With this in mind, we describe an *application-driven* approach undertaken to annotate a set of meetings with relatively coarse structural annotations with the hopes of spurring development of automatic structural segmentation algorithms in this difficult domain. This approach encompasses both the development of a novel framework for manipulating and annotating recordings of multiparty discourse, and annotations performed on meeting corpora.

In this chapter, we first describe the architecture developed in the course of the project for both collecting annotations over, and performing research tasks involving, multi-party discourse. While this architecture was developed in the context of working with meetings, it is more generally applicable to multiparty discourse. In particular, we discuss an *ontology of multimodal discourse*, along with its corresponding *ontology programming interface*. We then present *NOMOS*, an *audiovisual toolkit* built on top of this programming interface. *NOMOS*, in turn, was used to develop a tool used to perform annotations, as well as several other tools designed for manipulating meetings.

We then discuss how the tools developed were used to create a new set of annotations of the ICSI (Janin et al., 2003) and ISL (Burger et al., 2002)

[1] http://labrosa.ee.columbia.edu/mapmeet/.
[2] http://www.m4project.org.
[3] http://www.amiproject.org.
[4] http://penance.is.cs.cmu.edu/meeting_room/.
[5] http://diuf.unifr.ch/im2/.
[6] http://chil.server.de/.
[7] http://www.ai.sri.com/project/CALO.

meeting corpora that mark *hierarchical topic segmentation* and *action items*. Finally, we describe the characteristics of these annotations, and analyse inter-annotator agreement.

The annotations and tools described in this chapter, as well as technical documentation, can be downloaded from the World Wide Web at http://godel.stanford.edu under *Software*.

2. Architecture for Meeting Annotation, Research, and Browsing

We begin our discussion by describing the flexible architecture we have developed for working with multi-party discourse. The architecture has grown out of three major threads of research: (1) performing and viewing annotations of discourse, (2) working toward automatic discourse segmentation, and (3) integrating our work with other components comprising a digital office assistant – including components responsible for vision, gesture, and high-level reasoning. In this section, we discuss a *multimodal discourse ontology* (MMDO) which has resulted from these efforts, as well as *NOMOS* – an *audiovisual toolkit* for manipulating multi-party discourse and annotations of that discourse.

2.1 MMDO and Ontology Programming Interface

In order to generically represent both corpora and annotations of those corpora, we have devised a *multimodal discourse ontology* (MMDO). The MMDO is fully described in Niekrasz et al. (2005) and Niekrasz and Purver (2006); here, we give a brief overview focusing on how the ontological framework allows us to unify several research threads. In accordance with our principles of *application-driven* annotations, the MMDO is a suitable representation on top of which to build agents capable of integrating with others into a digital personal assistant.

The MMDO follows recent trends in information technology which put *semantics* in the limelight of data-driven research, the most significant being the Semantic Web (Berners-Lee et al., 2001) which brings ontology and knowledge engineering in contact with the World Wide Web. Following this trend, research in annotation of both linguistic and multimedia resources has begun to shift away from the paradigm of *markup* toward that of *semantic annotation* (Farrar, 2007; Geurts et al., 2003). While the former are commonly schematised in a manner similar to an XML DTD, the latter is grounded in a formal ontology, providing an expressive semantics to the annotation and allowing inference.

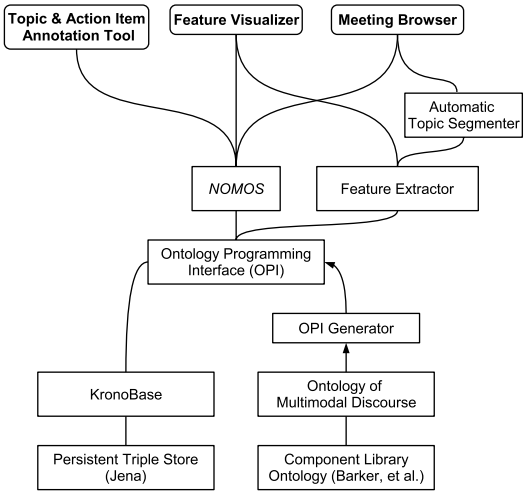


Figure 1. Architecture Diagram: The OPI Generator produces an Ontology Programming Interface through which applications can manipulate a knowledge base constrained by an arbitrary ontology. *NOMOS* is built on top of an extensible OPI generated from an ontology of multimodal discourse. *NOMOS*, in turn, serves as the audiovisual backbone of several tools, shown at the top. The OPI is also utilised by the feature extractor, which produces features for automatic topic segmentation.

The MMDO can be found as part of the software architecture in Figure 1. At the core is a general upper ontology called the Component Library (Barker et al., 2001), the core ontology used in the CALO project. This provides the most abstract level of semantics to the annotation schema such as events, entities, and roles. Building from these general concepts, we have designed an ontology of multimodal discourse. This layer encodes the concepts important to understanding discourse, such as utterances, words, speaking events, writing events, linguistic constituents, gesturing, etc. In its design, we place an emphasis on unifying our multiple research threads (e.g. human–computer dialogue, open-domain parsing, meeting modelling, and lexical semantics) both theoretically and pragmatically where possible, as well as on capturing as many of the commonly-held concepts in natural language research as possible.

Using this ontology, we create a custom-made Java API – which we call an *ontology programming interface* (OPI) – via an algorithm which encodes the hypernymic relations in the ontology as Java class inheritance and encodes the class relations (attributes) as Java methods. The OPI is written to interface with a triple-store database back-end, which supports persistent access to annotations, currently implemented using the Jena Semantic Framework. *Kronobase* is a layer we have developed for meta-annotation, which allows the recording

of important aspects of annotation, including who performed it, when it was performed, and on which resources (other annotations) it is dependent.

2.2 *NOMOS*: An Audiovisual Toolkit for Meeting Annotation, Research, and Browsing

Leveraging the OPI is *NOMOS*,[8] a generic *audiovisual toolkit* for displaying and playing recorded discourses (or, in fact, any type of media recording), and for manipulating and visualising associated annotations. *NOMOS* provides functionality for graphically displaying information stored in the annotation knowledge base, thus creating a generic platform in which any discourse can be loaded so long as it can be converted to the appropriate format. Moreover, since *NOMOS* is built using the OPI infrastructure, it can easily leverage the same set of underlying ontologies used internally by the CALO systems, including the MMDO. This makes it easy to use *NOMOS* as a platform underlying end-user components of the CALO systems.

NOMOS is a highly customisable environment, serving as the primary ingredient in building the annotation-related software tools discussed in Section 3. In particular, both the *Feature Visualiser* and the *Topic and Action Item Annotation Tool* are composed entirely of a set of plugins and templates developed within the *NOMOS* framework; screenshots of these tools can be found in Figures 5 and 6 later in this chapter. The latter is a tool for annotators, while the former is targeted at researchers. In addition, *NOMOS* serves as the basis for the *Meeting Browser* tool currently under development, with which end-users of the CALO systems will be able to browse through an automatically annotated meeting. Figure 1 shows the architectural hierarchy contributing to each piece of software. *NOMOS* is implemented entirely in Java, as are the tools built on top of it. Each has been used extensively under Windows, OS X, and Linux.

The rest of this section describes the core components which make up the *NOMOS* toolkit.

2.2.1 Query editor. Since *NOMOS* is built on top of the OPI, all annotations are stored in a knowledge base accessible via the powerful programming interface exposed by the OPI. While this is an excellent interface for software development experts, it is not necessarily suitable for annotators or end-users of other applications built on top of *NOMOS*. In order to provide an intuitive mechanism for users to interact with this powerful programming tool, *NOMOS* provides a graphical *query editor*. The query editor provides a way for users to construct and edit *queries*, which are an intuitive means of ex-

[8] *NOMOS*, is an abbreviation for "anNOtation of Media with Ontological Structure".

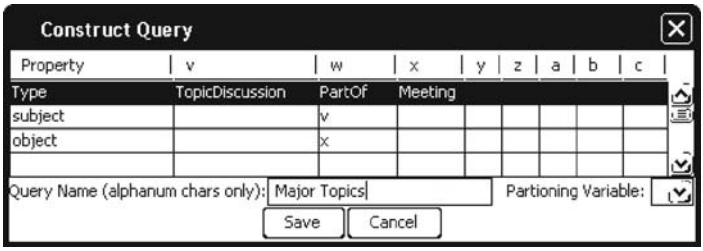


Figure 2. The graphical Query Editor.

tracting sets of annotations from the knowledge base – much in the same way that SQL queries are used to extract datasets from a database. For instance, as Figure 2 demonstrates, it is a simple matter to construct a query which extracts all of the *major topic* annotations of a particular meeting. Queries can be executed by *NOMOS*, with the results typically displayed on *tracks*, as described below. By providing such a query representation language, *NOMOS* itself can be, for the most part, agnostic with regard to the underlying ontology used to represent the annotations.

2.2.2 Tracks. At the heart of the visual representation of *NOMOS* is the notion of a *track*. Tracks appear in a vertical stack in the centre of the display, and can be clearly seen in Figures 5 and 6. The x-axis of each track is measured in time: the start and end of a track correspond to the start and end of the discourse being displayed. A track, then, is appropriate for displaying annotations which are rooted at a particular time in the discourse, for instance: transcripts, topic segments, gestures, or groupings of utterances into linguistic units. *NOMOS* provides default functionality for displaying the properties of time-based annotations as text; in addition, there is extensive plugin support so that developers are free to write custom *plugins* for graphically displaying annotations in whatever means is most appropriate. This makes it possible to develop highly customised applications, such as the *Topic and Action Item Annotation Tool* discussed in the next section. Finally, users can easily zoom in and out on tracks.

2.2.3 Tabbed panels. In addition to tracks, *NOMOS* also provides a generic mechanism for plugins to represent annotations graphically in any appropriate format as a *panel*. Any number of panels can be shown at once, each appearing as a *tab* which can be clicked. Like a *track*, a *panel* displays a set of annotations retrieved by executing a particular *query*. Unlike a track, a panel need not represent annotations *temporally*. Thus, panels serve primarily as a means of representing entities not associated with one particular time in a discourse – a typical example is the set of *Participants* of a particular dis-

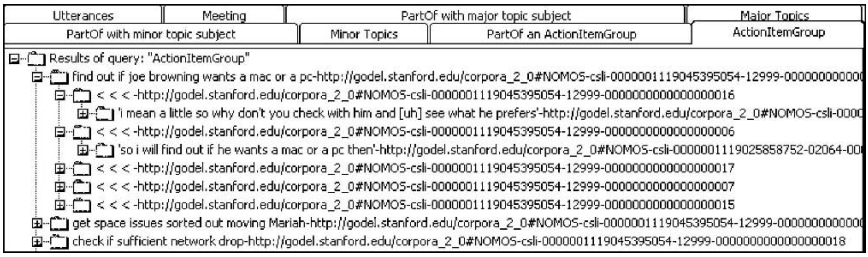


Figure 3. An example of a tree panel showing the results of a query.

course. The plugin infrastructure allows developers to create customised ways of displaying and editing entities with a panel. Distributed as part of *NOMOS* are core plugins which visualise any set of entities as a *tree*, where parent/child relationships denote that a *Relation* exists among entities; Figure 3 shows an example of such a tree. A tree display is particularly appropriate for non-time-based annotations, often in the form of *persistent annotations*. Persistent annotations are ones which "persist" across multiple discourses (or, more generally, across multiple media files); for instance, a list of discourse references might persist across several discourses, as the same entity might be referred to in the course of multiple discourses.

2.2.4 Templates. In order to customise how a particular set of annotations are displayed, *NOMOS* provides a mechanism for creating and editing *templates*. A template consists of the following:

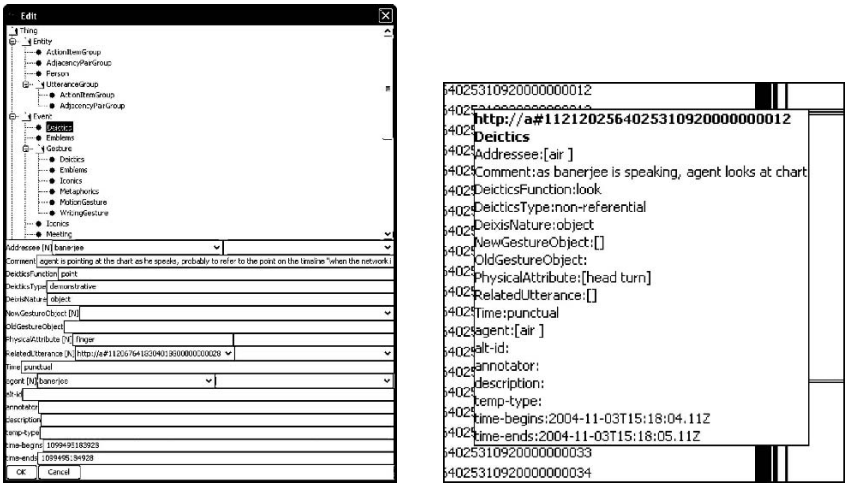
- A set of queries to be executed
- The types of tracks to be used to display the results of each query – where the type is determined by the type of plugins used to display the track itself and the entities on each track
- The set of plugins used to display annotations as tabbed panels
- Values for configuration parameters for each plugin – used to further customise behaviour according to plugin-specific parameters

Given this set of information, templates define how the user will view the annotations: the same set of annotations can be visualised in quite different ways, depending on the subset of the annotations defined through the queries and the plugins used to display the results of these queries.

2.2.5 Transcription. Capabilities for both displaying and editing transcripts are packaged as core *NOMOS* plugins, as these capabilities are desired in many annotation tasks. In the GUI, each conversational participant

is assigned a *track*, in which the transcribed utterances (or speech recognition hypotheses) of that participant are displayed – moving from left to right moves along the time axis. In the screenshot shown in Figure 5, for example, each of the top seven horizontal tracks are dedicated to the transcripts of the seven meeting participants. Each small box on a track shows the transcription of a single utterance, where the left- and right-hand sides of each box are time-aligned with the start and stop time of the utterance. Zooming in and out allows the user to adjust how much of the transcript is viewed at once; this makes it easy to move from a microscopic view of the discourse to a global one, and back. For instance, while Figure 5 displays about a minute of discourse, Figure 6 shows about an hour.

2.2.6 Creating and editing annotations. The plugin architecture implemented in *NOMOS* allows tool designers to create arbitrary mechanisms for users to interact with, modify, and create new annotations – the *Topic and Action Item Annotation Tool* described below provides an excellent example of how plugins can lead to such specialisation. However, many annotation tasks share a common flavour, so *NOMOS* includes a core set of capabilities for defining new sets of annotations, as well as modifying existing sets. Using the core architecture, time-dependent annotations (or annotations relating time-based entities to one another) are typically made on additional *tracks*; for example, in a gesture-annotation task, gestures might be shown as *events* on a track so that the start and end time of each gesture can be pinpointed. The *relation* of each gesture to a particular utterance can then be annotated via *drag and*



(a) Editing an entity's properties – the potential values for each slot are constrained by ontological constraints over the current domain.

(b) A tooltip brought up by hovering the mouse over an entity allows for quick interrogation.

Figure 4. Screenshots of *NOMOS* capabilities for viewing and editing entities.

drop: users can drag one entity on to another to set a particular entity as a value for a particular *slot* in another entity. In addition, any entity can be *interrogated* by bringing up a dialog box like the one shown in Figure 4(a) which shows the slots and values that define that entity, allowing users to directly modify the knowledge base. In both mechanisms for relating entities to each other, *ontological constraints* are enforced by *NOMOS*. For example, if the value of a particular slot can only be of a certain type, *NOMOS* will use tools associated with the OPI to do subclass inference and only allow entities of that type (or subtypes of that type) to be set as the value of a particular slot. Similarly, when editing the properties of a particular entity, only valid slot-fillers in the current domain are presented as options to fill that slot; for example, when choosing the value for a *Participant* slot on an utterance, an annotator will only be able to choose an available entity of type *Person*, since this slot can only take values of this type. These inference capabilities mean that highly customised tools can be developed quickly with little or no programming; instead, ontological constraints directly "customise" the tool.

2.2.7 Audio and video. A red vertical line overlaying the tracks represents the audio and/or video cursor. It indicates the current position of playback: as playback proceeds, it moves from left to right and the track display is automatically scrolled. Buttons along the bottom can be used to pause playback, or skip forward and back a few seconds – allowing users to quickly replay a bit of the conversation, or quickly fast forward through parts of it. The *focus* button is used to centre the display around the current media location; conversely, clicking in a particular location in a track will move the cursor to that location. An arbitrary number of audio and video streams can be synchronised at once; for instance, a video of a discourse can be played back with a separate audio track for each participant mixed together in real time.

2.2.8 Annotation comparison capabilities. It is often quite important to be able to see each annotator's annotations of a single discourse side by side. Built into *NOMOS* is the capability to partition a set of annotations based on the *annotator* who created each annotation, laying out each annotator's contributions on a separate track. This capability facilitates easy comparison of multiple annotations made to the same discourse, by stacking each distinct set of annotations on tracks one above another. When comparing topic segmentations, for example, loading each set of annotations one above another and then zooming out allows annotators to get a rough idea of where areas of disagreement and agreement lie; these areas can then be zoomed in on for more detailed discussion. The same techniques can be used to compare the output of annotations automatically generated by, for instance, machine learning

techniques. Visually comparing similarities and differences lends powerful (though perhaps anecdotal) insight into differences among algorithms.

2.3 Comparison to Similar Efforts

The architecture described in this section provides similar functionality to toolkits in development elsewhere. Of particular note, is the NITE XML Toolkit (Carletta et al., 2004; Carletta and Kilgour, 2004), which is a generic toolkit for performing linguistic annotation tasks. At a basic audiovisual level, NITE provides fairly similar functionality to the NOMOS architecture described in this chapter: synchronised audio and video playback and a plugin architecture. A key difference in the visual display is that by default transcripts in NOMOS are displayed along *tracks*, while in the NITE system they are presented as linearised text. While both approaches have their advantages in different applications, the tracked presentation provides the most natural means for emphasising *overlap*, a feature of multiparty discourse we believe is often ignored in natural language processing applications.

Greater differences between NOMOS and NITE arise as a result of fundamental differences in the way the two represent the underlying annotations. The NOMOS architecture is centred around semantic annotation, which results in annotations made according to particular schemata, and a uniform user interface built around editing the fields of entities and their relationships to one another. Moreover, the semantic framework made use of by NOMOS is meant to be interoperable with high level reasoning components currently in development, and annotations are stored in the standard OWL/RDF triples format. Other knowledge-base tools can therefore easily manipulate them, and it is straightforward to transfer knowledge to agents capable of reasoning. NITE, on the other hand, stores all annotations as XML.

Finally, our framework stands out in that the OPI is "compiled" from the annotation schema. The availability of this Java API makes it straightforward to manipulate and analyse annotations, as we have done in Section 5. The OPI makes it possible to write scripts which are forced at *compile-time* to conform to the annotation schema.

3. Tools

In this section, we describe several distinct tools we have developed using *NOMOS* as the core platform, backed by the multimodal discourse ontology and its associated ontology programming interface. All of the tools described in this section are made up entirely of a set of *NOMOS* plugins, laid out using the standard *template* mechanism described in the previous section.

The *Topic and Action Item Annotation Tool* was developed for the use of the annotators performing the annotations which will be described later in this

chapter. The *Feature Visualiser* is a tool we have developed in the course of our preliminary automatic segmentation work. And the *Meeting Browser* is a tool currently under development which is intended to be an end-user component of the CALO digital personal office assistant. Taken together, these tools demonstrate the flexibility of the architecture we have developed, showing how it can play a cross-cutting role across the tasks of meeting annotation, browsing, and research.

3.1 Topic and Action Item Annotation Tool

A screenshot of the *Topic and Action Item Annotation Tool* is shown in Figure 5. It leverages the full features of *NOMOS*, complementing them via plugins to allow for additional annotation capabilities specialised to annotating topic segments and action items. The tool is an excellent example of how the generic capabilities provided by *NOMOS* can be further specialised via plugins to make performing a specific set of annotations particularly efficient.

We briefly note here features developed in the tool (as well as in *NOMOS* in general) which particularly decrease the high cognitive load demanded by the annotation task. Notably, key capabilities revolve around simultaneously providing global and local insight into the meeting and annotations, as well as the capability to easily revise draft annotations.

Topic and action items (see Section 4) are annotated via context menus available on the tracks displaying the utterance of the discourse. Specialised tabbed *panels* (see above) show a topic hierarchy and a list of action items (shown in

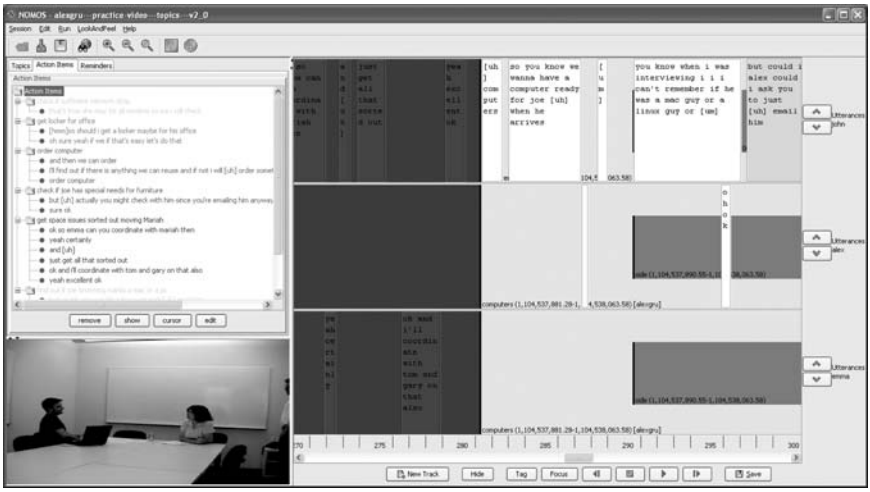


Figure 5. Screenshot of the *Topic and Action Item Annotation Tool*.

the upper left of Figure 5), giving an overview of the annotations at a global level. During the pilot period of annotation, it became clear how important it was to be able to easily modify annotations after making an initial rough pass through a discourse. As a result, capabilities for *renaming* and *deleting* both topics and action items exist, as well as the ability to *promote*, *demote*, or *merge* major and minor topics as appropriate. These capabilities provide single-click *shortcuts* to what would otherwise be somewhat involved tasks in the default *NOMOS* framework. In addition, "reminders" can be inserted at particular time points, allowing annotators to make notes to refer back to in a subsequent pass.

Specialised *track* plugins provide a task-specific visualisation of both topic segmentations and action items. *Major* topics are signalled graphically on the tracks containing the utterances by alternating the background colour. The *minor* breaks are indicated by the narrower bands of alternating light and dark gray centred vertically in the track. For instance, in Figure 5 there are two major topics visible in the time slice shown; in addition, the second major topic is a parent to one visible child minor topic. Brief descriptions assigned to each major and minor topic are displayed in each track. Finally, the entire hierarchy of topics can be shown by clicking on the appropriate tab in the upper left hand corner; clicking on any topic in this list will shift the track display to the start of that topic.

An example of annotations for *action items* is also displayed in Figure 5. Several utterances by the top and bottom speaker in the first major topic have been shaded the same colour to indicate that they are related to the same action item; similarly, an utterance on the top right has also been highlighted a different colour to indicate it is part of a *different* action item. Moreover, the panel in the upper left corner lists all of the action items marked in the entire discourse. A brief description of each appears, followed below by the transcript of each utterance comprising that action item. Clicking on an utterance will scroll the track display to show that utterance. Each action item is assigned a colour, shown both in the summary in the upper right and in the highlighted utterances in the display.

3.2 Feature Visualiser

We have developed a generic *Feature Extractor* and *Feature Visualiser* using the ontology programming interface and *NOMOS* audiovisual toolkit, as the architecture diagram in Figure 1 shows. We mean *feature* here in the sense of features which can be computed from discourse as input to machine learning algorithms for classification tasks such as topic segmentation. The *Feature Extractor* is simply a set of Java classes which provide core functionality for processing discourse, as represented by the OPI. Functionalities include:

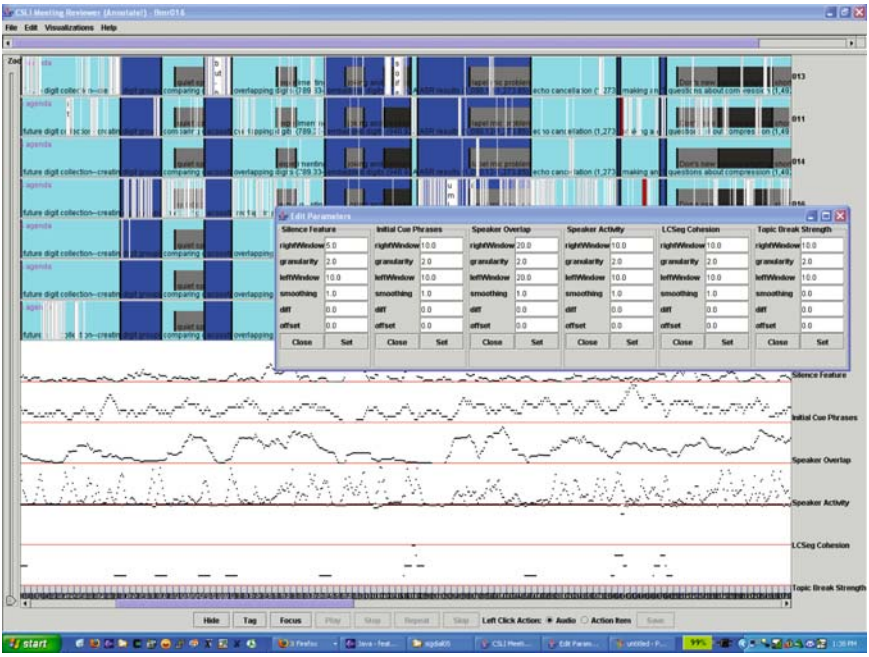


Figure 6. Screenshot of the *Feature Visualiser*.

extracting sets of utterances in a given time window, turning these utterances into bags of words per speaker, smoothing feature values, and calculating their derivatives. Moreover, generic tools are provided for iterating over discourses, processing them, and extracting sets of feature values at regular intervals which can then be piped directly into learners like decision trees, neural nets or support vector machines.

The *Feature Visualiser* is built on top of the extraction architecture, using a set of plugins to create the GUI using *NOMOS*. It displays calculated feature values alongside an annotated discourse, as shown in Figure 6. Moreover, as the popup window in Figure 6 shows, it allows the user to dynamically modify each feature's parameters (e.g. window size, smoothing, or other feature-specific parameters) and immediately observe the results. We have found the visualiser to be invaluable in debugging algorithms for feature extractors, tweaking parameter values, and hypothesizing new, interesting features.

3.3 Meeting Browser

We are currently developing a *Meeting Browser* tool, which will sit on top of both the audiovisual toolkit and the feature extractor. The eventual development of this tool is the motivation that has driven our annotations and

associated schema. The browser is meant to allow users to "drill down" through the structure of the meeting, easily pinpointing segments of interest.

4. Annotation Motivations and Schema

We now turn to describing an annotation task performed using the *Topic and Action Item Annotation Tool* described in the previous section, providing a real world example of both the sort of annotations which may be performed in the NOMOS architecture, and the type of analyses which are straightforward to perform using the OPI compiled from the annotation schema. We focus on two types of discourse structure annotations. The first, *topic segmentation*, breaks the discourse up into a (hierarchical) sequence of topics. The second, *action item subdialogues*, marks particular utterances as being relevant to the discussion or assignment of action items. In this section, we describe our motivations in studying these phenomena, related work, and the iterative process by which we refined an application-driven annotation schema.

We worked with the ICSI Meeting corpus (Janin et al., 2003) and the ISL Meeting Corpus (Burger et al., 2002) because both contain high-quality close-talking microphone recordings of conversational speech in a meeting environment, as well as word-level transcriptions and utterance-level timing information. We focused mainly on the ICSI corpus because its contents most closely matched our task of processing fairly informal, office-style meetings. In addition, extensive annotations have already been completed on the ICSI corpus, including: dialogue acts (Shriberg et al., 2004), "hot spots" (Wrede and Shriberg, 2003), and some work on topic segmentation (Galley et al., 2003; Carletta and Kilgour, 2004).

4.1 Topic Segmentations

A significant challenge in spoken discourse segmentation is providing a concrete definition of the problem – the desired concepts of both *topic* and *segmentation*. To that end, we first briefly discuss the conceptualisations – and motivations behind those conceptualisations – that have arisen in the related fields of segmenting text and monologue. We then discuss previous work in segmenting discourse, our own motivations, and finally outline an annotation schema derived from these motivations.

4.1.1 Text and monologues. The segmenting of text documents is often motivated by information retrieval tasks – for instance, so that a single appropriate segment can be returned matching a query. In some cases, topic boundaries are hand-annotated, as in Hearst (1994). However, topic boundaries are often artificially created by concatenating multiple articles together, as in Galley et al. (2003) and Choi (2000). Moreover, since text is written

linearly, usually with clearly punctuated boundaries in the form of sentences and paragraphs, it is natural to assume that topic boundaries will occur at such places. Thus, such "natural" boundaries both define and limit the search space. In addition to text, there has been much research in segmenting *non-conversational speech*; essentially monologues or series of monologues. For example, much work has been done on automatically segmenting broadcast news (e.g. Tür et al., 2001; Beeferman et al., 1999; Allan et al., 1998).

The tasks of segmenting text and monologue are similar in that both tend to have fairly well defined topic structure. In the case of artificial text corpora created through concatenation, topic boundaries can be objectively defined over the concatenated article boundaries. News broadcasts tend to consist primarily of scripted speech – with little spontaneity – produced by highly practiced professionals (though some work has also been done on more spontaneous monologues (see Passonneau and Litman, 1997). Topic boundaries in news broadcasts are designed to be obvious, with unambiguous shifts from one story to the next. In both domains, automatic segmentation algorithms tend to rely primarily on lexical co-occurrence statistics to calculate a measure of *lexical cohesion* between chunks of text (Hearst, 1994; Hearst, 1997). In the case of monologue, prosodic cues are often utilised as well (Tür et al., 2001; Hirschberg and Nakatani, 1998).

4.1.2 Discourse. When turning to spontaneous discourse, most previous work has followed this text/monologue approach: for example, when Galley et al. (2003) annotated 25 meetings in the ICSI Meeting corpus for topics, the discourse was represented *linearly* as a series of non-overlapping utterances, topics were represented as a linear sequence of segments, and topic boundaries were allowed only at *speaker changes*. Although we are aware of one project in which *hierarchical* topic annotations are being used (on the ICSI corpus using the NITE XML toolkit (Carletta and Kilgour, 2004)), no annotations are yet publicly available.

Rather than adapting the task of discourse segmentation to make it look more like a text segmentation task, we took an *application-driven* approach to segmenting discourse. Our motivation for topic segmentation was to enable broad understanding of a discourse, providing a coarse summary segmentation for broad-perspective user browsing capabilities, and allowing for selective "drill-down" and replay; for more detailed discussion of the utility of high-level segmentations, see Banerjee et al. (2005). We therefore wanted to collect annotations which can be leveraged specifically to provide such capabilities for a digital personal office assistant. Specifically, we instructed the annotators to look at the problem of providing a topic segmentation from the perspective of utility: if they were reviewing a meeting they might not have attended, what segmentation would help them quickly "drill down" to por-

tions they might be particularly interested in reviewing. While a bit vague, this description of the task avoids biasing the annotators toward relying on particular discourse phenomena or restricting them to particular boundary locations; Ries (2001) argues that such an application-driven approach, with linguistically naive coders, may help best represent end-users of meeting browser systems.

This application-driven approach proved difficult at first, resulting in low inter-annotator agreement among the two undergraduate annotators in the first five meetings that were annotated. However, through discussions of the annotations (often using annotation comparison capabilities discussed in Section 2) – discussions in which no actual concrete annotation criteria for what always must constitute a topic break were discussed – an acceptable level of inter-annotator agreement was reached for the majority of meetings (see Section 5). Agreement results eventually reached a plateau, at which point further discussion of the annotation guidelines was terminated. At this point, guidelines were then drawn summarising the result of these discussion (see Gruenstein et al., 2004). The resulting schema is discussed below.

4.1.3 Topic segmentation schema. Meetings were segmented according to a two-level hierarchical segmentation schema. In the top (*major*) level of the hierarchy, the entire meeting is wholly and contiguously segmented, where segment boundaries symbolise highly salient breaks in discourse structure and/or distinguish parts of the discourse between which there is an obvious difference in subject matter. In the second (*minor*) level of the schema, major segments are optionally subsegmented without a requirement for contiguity, but with overlapping segments forbidden. Minor segments signify either a temporary digression or a more focused discussion of the subject matter, while still remaining directly relevant to the encompassing major segment. Our pilot annotation work indicated that restricting topic breaks to speaker changes was an unnatural restriction. Instead, our schema allows topics to start and end at any point in the discourse, even in the middle of a single speaker's utterance. Some ramifications of this choice are discussed in Section 5. We note that while our choice to allow topic breaks at any time point may be "permissive", it may in fact not be permissive enough when considering multiparty discourse. In such discourses, it may sometimes be the case that while some speakers have moved along to a new topic, others may still linger on an old topic; or, a few speakers may discuss one topic amongst themselves while others discuss another. While such phenomena are interesting, we felt that given the *application-driven* nature of our annotations (with the application being a meeting reviewer tool), capturing such granularity was not necessary. Figure 7 depicts a meeting segmented according to the schema,

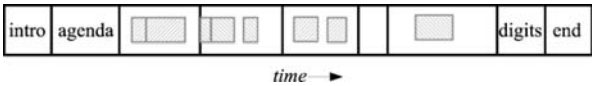


Figure 7. A sample hierarchical meeting segmentation.

with vertical lines separating major topics, and shaded areas representing minor topics.

Annotators also gave brief descriptive names to topics, though no standards were set as to the format or content of the assigned names, with the exception of the following *reserved* topic names:

- *AGENDA*: the portion of the meeting in which the agenda is presented and discussed
- *INTRO*: speech before the meeting "officially" begins (appears in every meeting, though may have zero length)
- *END*: speech after the meeting "officially" ends (appears in every meeting, though may have zero length)
- *TECHNICAL DIFFICULTIES*: a period in which there are technical difficulties with recording equipment
- *DIGITS*: the digits task in the ICSI meeting corpus, see (Janin et al., 2003)

Except for *AGENDA*, the reserved names simply serve the purpose of highlighting portions of the recording which might not be considered part of the meeting proper; below we discuss how they play a role in defining a reference segmentation. In addition, if a new topic is a continuation of a discussion of a previous topic left off earlier, the convention is used that the same descriptive text is given for both topics – implicitly linking them.

4.2 Action Items

Though the focus of the annotation work was hierarchical topic segmentation, annotators also marked *action items*. Previously, we have shown how simple task-assignment charts can be inferred from highly scripted, multimodal meetings (Kaiser et al., 2004). In moving to free-form meetings, identifying *decision points* like action items follows as a natural first step in extending this work.

For the purposes of annotation, we define an action item loosely as a task which is discussed in the meeting and then assigned to a participant (or participants) to complete at some point after the completion of the meeting. In our schema, action items are defined as sets of *utterances*, rather than start and end times: this is possible because action items are usually discussed only briefly,

so it is feasible for an annotator to pinpoint particular utterances in which the discussion occurred. Moreover, it is useful to identify as specifically as possible the utterances in which action items were discussed, as not all speech within a time window may be relevant due to the high levels of speech overlap in multi-party conversations.

Note also that while identifying the general regions of action item discussion could be useful for logging and browsing by a user, it is only by identifying the relevant utterances themselves that we will be able to move towards automatic interpretation of the action items, where interpretation might include: identifying the person it has been assigned to, its deadline and the information about the task it involves. With this goal in mind, we plan future annotation passes to further classify each utterance into specific categories such as *task*, *deadline* and *person* assignment. Furthermore, it may be useful to mark information particular to the task, such as: the person it has been assigned to, its deadline, and its relation to other tasks.

5. Analysis of Collected Annotations

We collected annotations for a total of 65 meetings, however 9 of those meetings were not annotated by both annotators, were annotated during our preliminary annotation sessions, or had other problems. Excluding this set of meetings leaves a total of 56 annotated meetings: 40 meetings from the ICSI corpus and 16 from the ISL corpus, totalling 45.9 hours. In this section, we provide a statistical analysis of our annotations of this set, along with some more qualitative observations. We describe multiple algorithms which have been applied to the data to make our analysis possible. We also provide an analysis of inter-annotator agreement using multiple metrics. Last, we compare our annotations to other similar datasets.

5.1 Pre-Processing

Every meeting recording has a beginning and end which do not actually contain meeting dialogue and which are not relevant to an analysis of topic structure. Before analysis, we therefore perform pre-processing of our annotations to produce a segmentation that does not contain these sections of the discourse. Because our annotators were asked to annotate these special cases, our pre-processing algorithm simply takes the union of the set of *INTRO* and *END* segments from both annotators and removes those portions of the discourse from both annotations. All the analyses presented below were done after this pre-processing step. While pre-processing of *DIGITS* and *TECHNICAL DIFFICULTIES* segments is necessary for training of topic detection algorithms, these segment types were not removed prior to the analysis presented in this section.

5.2 Segment and Break Classification

While most text segmentation methods constrain the number of possible segmentations by specifying a finite set of discrete locations where segment boundaries may occur (most often at sentence boundaries), our annotators were free to assign boundaries at any time during the discourse. Unfortunately, this complicates our use of standard evaluation metrics, and it does not suit iterative automatic discourse segmentation algorithms which operate at discrete intervals of time.

To overcome these obstacles we transform our annotations into a set of classifications in two ways, arriving at what we call a *segment classification* and a *break classification*. For each of the two, the first step is to divide the discourse into temporal units based on a set of possible break locations, e.g. a set of evenly spaced temporal values, utterance start times, or speaker changes. We use evenly spaced intervals of 20 seconds in our analysis.

In the case of evenly spaced windows, a discourse d is evenly divided into $i = |d|/n$ non-overlapping contiguous temporal intervals of length n, with the last window realizing any remainder and possibly being cut short. For the *segment classification*, each temporal unit is classified as to which topic segment it belongs. Temporal units which contain segment boundaries are classified simply by determining in which half of the unit the annotated boundary lies. If it lies in the later half, the unit is classified as belonging to the previous topic segment. For the earlier half, it is classified with the following topic segment. This produces segment boundaries which are between windows.

For *break classification*, each unit is classified as to whether or not it contains a topic boundary. This latter interpretation is essential for making use of the Kappa agreement statistic when the number of topic segments is unconstrained, as it is here. This may be transformed back into a set of segment boundaries by placing boundaries at the centre of windows which have been classified as containing a topic break.

5.3 Reference Segmentation

Another essential processing step is to produce a reference segmentation from our individual annotations. This is important to providing a comparison to other annotations such as those used in Galley et al. (2003), and for training automatic segmentation algorithms. Galley et al. (2003) create a reference segmentation by establishing sets of topic boundaries based on co-occurrence between annotations within 20 seconds. They then choose those sets which have been annotated by a majority and establish a boundary at each set's median time value.

In our current method, we employ the same strategy of discarding the minor segments. However, we believe benefit can be derived using our second tier of

segmentations as there are many cases where topic boundaries are annotated as a major shift by one annotator and as a minor shift by the other, suggesting some level of agreement that should be used. Also a second tier of segmentation in an automatic segmentation application would likely be useful for more localised "drill-down". Therefore, we do not believe this strategy should be a hard and fast rule: we provide our segmentations as individual annotations without establishing a defined reference. We will likely employ different strategies in the future for establishing a reference segmentation which incorporates minor boundaries.

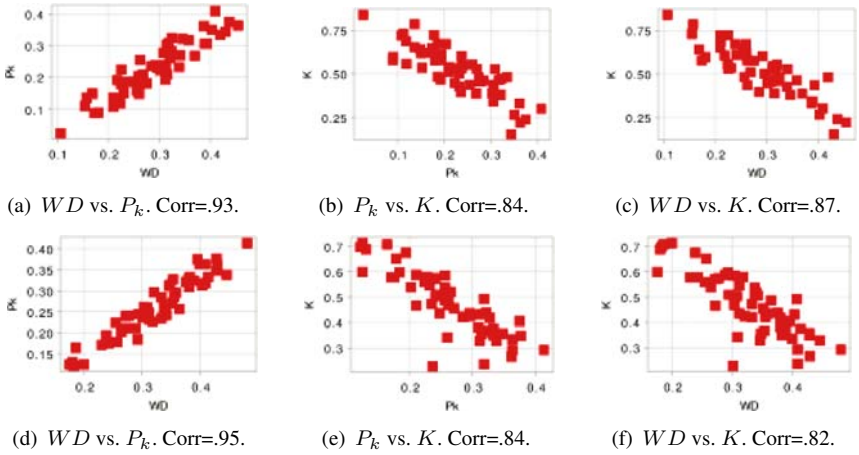
5.4 Evaluating Inter-Annotator Agreement

In this section we present the results of evaluating agreement between our two annotators and compare multiple agreement metrics. The results show variance among meetings, suggesting that the topic segmentation task may be ill-formed for certain classes of meetings.

The current standard metric for measuring inter-annotator agreement in classification tasks is the kappa statistic (K) (Carletta, 1996). While K is a good measure of how well annotators can agree on pinpointing topic breaks at time points, it does not accommodate near-miss break assignments in which annotators label different nearby time points as topic breaks. For the evaluation of segmentation algorithms specifically, two metrics are most commonly used: P_k (Beeferman et al., 1999) and *WindowDiff* (WD) (Pevzner and Hearst, 2002). These were designed principally to evaluate text segmentation algorithms that operate at sentence boundaries, but can be applied to continuous-time segmentations through the use of windowing. P_k accommodates near-miss labellings by considering how likely two time points are to be assigned to the same topic, while WD further refines this notion by measuring the difference in number of topic breaks between two time points. Each metric provides a reasonable, though different, evaluation of inter-annotator agreement. Results given in Table 1 and Figure 8 show a high degree of correlation among them.

Table 1. Mean/median agreement on topic segmentations.

	Major topics	*Major and minor topics*
WD	28.9% / 29.2%	32.7% / 33.8%
P_k	22.6% / 22.5%	26.5% / 26.2%
K	52.1% / 53.2%	47.0% / 46.9%



(a) WD vs. P_k. Corr=.93. (b) P_k vs. K. Corr=.84. (c) WD vs. K. Corr=.87.

(d) WD vs. P_k. Corr=.95. (e) P_k vs. K. Corr=.84. (f) WD vs. K. Corr=.82.

Figure 8. Segmentation inter-annotator agreement: each point represents a single meeting. (a)–(c) include major topics only; (d)–(f) include major and minor topics.

Our measurement of K follows that suggested in Carletta (1996) and described fully in Siegel and Castellan (1988):

$$K = \frac{P(A) - P(E)}{1 - P(E)} \tag{10.1}$$

This measures pairwise agreement on classification tasks, correcting for chance, where $P(A)$ is the probability of agreement and $P(E)$ is the probability of chance agreement between two annotators. Increasing values of K indicate better agreement. We use the break classification form of our annotations when calculating this metric.

Our second measurement is a variation on P_k, which is computed as follows:

$$P_k(\mathbf{a}, \mathbf{b}) = \frac{\sum_{i=1}^{N-k}(\delta_{\mathbf{a}}(i, i+k) \overline{\oplus} \delta_{\mathbf{b}}(i, i+k))}{N-k} \tag{10.2}$$

P_k estimates the probability that two randomly drawn temporal values occurring during the discourse are classified as being in *different* segments by the two segmentations $\mathbf{a}$ and $\mathbf{b}$ – thus, decreasing P_k indicates better agreement. Here, $\delta_{\mathbf{x}}(t_1, t_2)$ is an indicator function which evaluates to 1 if the segmentation $\mathbf{x}$ places the times t_1 and t_2 in the same segment. The $\overline{\oplus}$ operator represents the `XNOR` function. As mentioned in Beeferman et al. (1999), if the value k is set to half the mean topic segment length, the metric provides appropriate results for all degraded forms of segmentation, including random segmentation. We impose a slight variation on the calculation of k by not treating one annotation as a reference and the other as a hypothesis, but rather by incorporating both annotations when calculating the average segment length.

The third and final metric, WD, is the most recently proposed and is a variation on P_k intended to improve its tolerance of near-misses and varying segment size distributions:

$$WD(\mathbf{a}, \mathbf{b}) = \frac{\sum_{i=1}^{N-k} (|b_{\mathbf{a}}(i, i+k) - b_{\mathbf{b}}(i, i+k)| > 0)}{N-k} \tag{10.3}$$

Here, $b_{\mathbf{x}}(t_1, t_2)$ replaces $\delta_{\mathbf{x}}(t_1, t_2)$ from equation 10.2 and is the number of segment boundaries occurring between times t_1 and t_2 in the segmentation $\mathbf{x}$. This metric is different from P_k in that a penalty is assessed at each evaluation point if the number of segment breaks in the interval is not equal between the annotations. In P_k, the number of breaks is not counted and a penalty is only assessed if one totals 0 and the other does not. For WD, we impose the same change to the calculation of k as we do in our calculation of P_k.

Because our annotations have continuous-time boundaries, we must establish a stepping method for i. Following Galley et al. (2003), we use 20-second stepping intervals. An investigation of inter-annotator agreement for varying step sizes from 5 to 60 seconds showed no significant change in P_k or WD. An evaluation of K with varying break classification window widths showed a maximum at near 20 seconds. For the purposes of transparency and descriptiveness, we include measurements of all three of the above metrics in our evaluation, using a 20-second window width and/or step size.

5.5 Results

Multiple graphs showing results for inter-annotator agreement may be found in Figure 8. The top three plots show agreement based only on major topic boundaries. The bottom three include minor topic boundaries in the evaluation. Each of the columns rows shows a pair-wise comparison of two of the three metrics. Means and medians are provided in Table 1.

As expected, the metrics show a high level of correlation (correlation coefficients are given in the figure captions). It is difficult to say what values for our metrics signify a "good" level of reliability in the annotations. In computational linguistics, a value of $K = 0.67$ is generally used as a cut-off for reliable analysis, though it has been suggested on multiple occasions that this is not appropriate for all tasks (see Eugenio and Glass, 2004) for a discussion. Undeniably low scores do occur in our annotations. This is often found for meetings which involved presentations of visual information, which made the audio-only annotation task difficult. Some of this information may be gleaned from the available annotator notes. Poor agreement and self-evaluation by the annotators on some meetings suggest that some of the annotations should not be used. It should be noted that there are more numerous outliers in the evaluation of major segments only, which is a result of there being some meetings

which were only annotated as having as few as two major boundaries (after pre-processing).

In addition, the two annotators marked 921 and 1,267 utterances respectively as belonging to discussion about action items. We have yet to do significant analysis of these annotations and wish to produce further annotations of decision-making processes before using the data. Current analysis shows inter-annotator agreement of utterance classification at $K = 0.36$.

5.6 Comparison with Similar Annotation Sets

In Galley et al. (2003), 25 of the meetings in the ICSI Meeting corpus were hand annotated for topic breaks. A minimum of three annotators per meeting were given the task of deciding if each *speaker change* in a linearly represented meeting constituted a topic break.

Due to their process of establishing a reference segmentation, topic boundary frequency is significantly different between their annotations and our individual annotations. Our annotators produced major segments with an average length of 180 seconds, while Galley et al.'s (2003) average 684 seconds. Their annotations total 12.6 hours, while ours total 45.9.

Another noteworthy statistic is the distribution of topic boundaries over meeting duration, depicted in Figure 9. The distribution is shown for each of our annotators and from Galley et al. (2003). While the total number of meetings is different between the two sets, there are significantly more topic changes in the latter half of the meetings for each. It will be interesting to take note of this statistic in other corpora to see if the trend is universal. It is unclear if this is a by-product of the annotation process or of the meeting itself.

Finally, Figure 10 gives some further details about the characteristics of the topic segmentation annotations we have collected. The first four graphs highlight characteristics of distinct sets of meetings based on their general type. *Bed*, *Bmr*, and *Bro* are each a particular subset of the ICSI meetings which

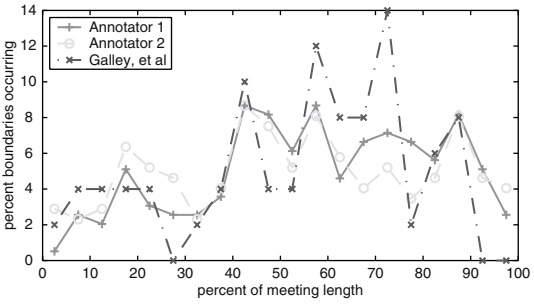
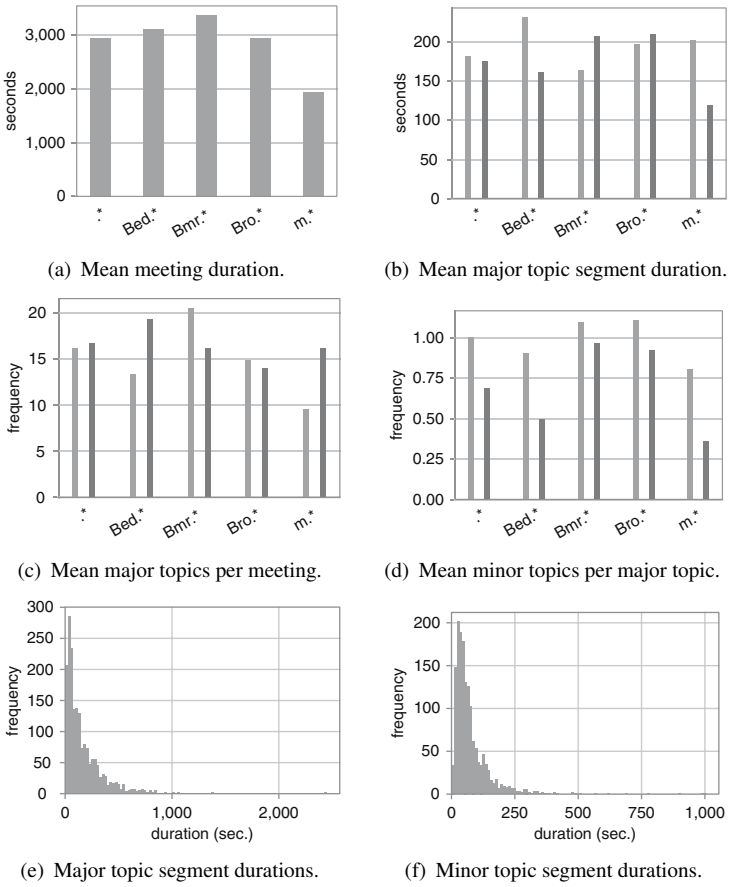


Figure 9. Distribution of boundaries over meeting duration.



(a) Mean meeting duration.

(b) Mean major topic segment duration.

(c) Mean major topics per meeting.

(d) Mean minor topics per major topic.

(e) Major topic segment durations.

(f) Minor topic segment durations.

Figure 10. Topic Segmentation Annotation Characteristics: (a) gives the mean duration of the meetings annotated of each different type, which is useful in interpreting the other graphs; (b)–(d) show per-annotator statistics broken down by meeting type, while (e)–(f) show histograms of major and minor topic segment durations of the reference annotation.

have a similar theme (see the corpus documentation for details), and *m* indicates meetings from the ISL corpus. The first graph shows mean meeting length, while the second shows mean major topic segment length. The following two show the number of major topic segments per meeting and the number of minor topics per major topic. Finally, histograms indicating the distribution of the durations of major and minor topic segments are given.

6. Current and Future Work

The work described in this chapter represents our first steps toward automatic meeting understanding for a personal office assistant. While coarse-level meeting segmentation is a useful first step, we are tackling the problem from

multiple angles: including robust natural language chunk parsing, dialogue act detection, argumentation structure analysis, and decision detection. Our first steps in these areas will likely be similar to those we have taken in topic segmentation: establishing modular additions to the annotation ontology, supporting this in the NOMOS audiovisual toolkit, coding annotation, research, and application tools for them, and then collecting annotations. Annotation of these richer structures will require greater use of the inference capabilities the ontology provides. For example, a tool designed for the annotation of argumentative structure will need to employ the constraints imposed by the ontology on that structure through the use of reasoning engines to constrain the annotations a human can make.

In parallel, we are currently developing automatic topic segmentation and action item detection tools by training classifiers on the annotations presented above while using the presented software framework for feature extraction and visualisation. For topic segmentation, initial investigation following a roughly similar approach to Galley et al. (2003) (using a decision tree trained on both lexical cohesion values and some discourse-based features – speaker activity, speaker overlap, amount of silence – and cross-validating over 25 ICSI meetings) has given average P_k error levels of around 0.35 for major topics. This is higher than Galley et al. (2003) achieved on their segmentation, but this would be expected with our finer-grained and less restricted notion of topic, and is at least comparable to our human annotator agreement. Future development will add prosodic features and chunk parser output. For action item detection, an initial n-gram-based classifier using a combination of manually and automatically extracted features is currently being developed, and has shown promising performance on a separate small test meeting corpus; future development will include the use of more structured hierarchical action item annotations.

Lastly, we expect to use the NOMOS audiovisual toolkit as a part of the CALO office assistant itself. This will involve the integration of our architecture with the CALO Desktop environment, allowing for pervasive feedback to our algorithms and online supervised learning.

Acknowledgements Thanks to our two annotators Michael Deeringer and Claire Gilbert, to our CALO associates Satanjeev Banerjee and Bill Jarrold, three anonymous SIGDIAL reviewers who commented on the version of this paper appearing in the proceedings of SIGDIAL 2005, the participants of the conference who provided invaluable feedback, and two additional reviewers who evaluated this book chapter. This work was supported by DARPA grant NBCH-D-03-0010. The information in this publication does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

References

Allan, J., Carbonell, J., Doddington, G., Yamron, J., and Yang, Y. (1998). Topic Detection and Tracking Pilot Study: Final Report. In *Proceedings of DARPA Workshop on Broadcast News Transcription and Understanding*, pages 194–218, Lansdowne.

Banerjee, S., Rose, C., and Rudnicky, A. (2005). The Necessity of a Meeting Recording and Playback System, and the Benefit of Topic-Level Annotations to Meeting Browsing. In *Proceedings of International Conference on Human-Computer Interaction (INTERACT)*, pages 643–656, LNCS, Springer, Rome.

Barker, K., Porter, B., and Clark, P. (2001). A Library of Generic Concepts for Composing Knowledge Bases. In *Proceedings of First International Conference on Knowledge Capture (K-CAP)*, pages 14–21, Victoria.

Beeferman, D., Berger, A., and Lafferty, J. D. (1999). Statistical Models for Text Segmentation. *Machine Learning*, 34(1–3):177–210.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Scientific American*, 285(5):34–43.

Burger, S., MacLaren, V., and Yu, H. (2002). The ISL Meeting Corpus: The Impact of Meeting Type on Speech Style. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 301–304, Denver.

Carletta, J. (1996). Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics*, 22(2):249–255.

Carletta, J. and Kilgour, J. (2004). The NITE XML Toolkit Meets the ICSI Meeting Corpus: Import, Annotation, Browsing. In *Proceedings of Joint AMI/PASCAL/IM2/M4 Workshop on Multimodal Interaction and Related Machine Learning Algorithms*, pages 111–121, Martigny.

Carletta, J., McKelvie, D., Isard, A., Mengel, A., Klein, M., and Møller, M. B. (2004). A Generic Approach to Software Support for Linguistic Annotation using XML. In Sampson, G. and McCarthy, D., editors, *Readings in Corpus Linguistics*. Continuum International, London.

Choi, F. Y. (2000). Advances in Domain Independent Linear Text Segmentation. In *Proceedings of Conference of the North American chapter of the Association for Computational Linguistics (NAACL)*, pages 26–33, Seattle.

Eugenio, B. D. and Glass, M. (2004). The Kappa Statistic: A Second Look. *Computational Linguistics*, 30(1):95–101.

Farrar, S. (2007). Using 'Ontolinguistics' for Language Description. In Schalley, A. and Zaefferer, D., editors, *Ontolinguistics*, pages 175–192. Mouton de Gruyter, Berlin.

Galley, M., McKeown, K., Fosler-Lussier, E., and Jing, H. (2003). Discourse Segmentation of Multi-Party Conversation. In *Proceedings of Annual*

Meeting of the Association for Computational Linguistics (ACL), pages 562–569, Sapporo.

Geurts, J., Bocconi, S., van Ossenbruggen, J., and Hardman, L. (2003). Towards Ontology-Driven Discourse: From Semantic Graphs to Multimedia Presentations. In *Proceedings of International Semantic Web Conference (ISWC)*, pages 597–612, Sanibel Island.

Gruenstein, A., Niekrasz, J., Deeringer, M., and Gilbert, C. (2004). *Discourse Annotation using Annotate!*
godel.stanford.edu/twiki/bin/view/Public/TopicActionItemDataset.

Hearst, M. (1997). TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages. *Computational Linguistics*, 23(1):33–64.

Hearst, M. A. (1994). Multi-Paragraph Segmentation of Expository Text. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 9–16, Las Cruces.

Hirschberg, J. and Nakatani, C. (1998). Acoustic Indicators of Topic Segmentation. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 976–979, Sydney.

Janin, A., Baron, D., Edwards, J., Ellis, D., Gelbart, D., Morgan, N., Peskin, B., Pfau, T., Shriberg, E., Stolcke, A., and Wooters, C. (2003). The ICSI Meeting Corpus. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 364–367, Hong Kong.

Kaiser, E., Demirdjian, D., Gruenstein, A., Li, X., Niekrasz, J., Wesson, M., and Kumar, S. (2004). A Multimodal Learning Interface for Sketch, Speak and Point Creation of a Schedule Chart. In *Proceedings of International Conference on Multimodal Interfaces (ICMI)*, pages 329–330, State College, ACM Press, Pennsylvania.

Niekrasz, J. and Purver, M. (2006). A Multimodal Discourse Ontology for Meeting Understanding. In Renals, S. and Bengio, S., editors, *Machine Learning for Multimodal Interaction: 2nd International Workshop, (MLMI 2005), Revised Selected Papers*, volume 3869 of *Lecture Notes in Computer Science*, pages 162–173. Springer.

Niekrasz, J., Purver, M., Dowding, J., and Peters, S. (2005). Ontology-Based Discourse Understanding for a Persistent Meeting Assistant. In *Proceedings of AAAI Spring Symposium on Persistent Assistants*, pages 26–33, Stanford.

Passonneau, R. J. and Litman, D. J. (1997). Discourse Segmentation by Human and Automated Means. *Computational Linguistics*, 23(1):103–139.

Pevzner, L. and Hearst, M. (2002). A Critique and Improvement of an Evaluation Metric for Text Segmentation. *Computational Linguistics*, 28(1):19–36.

Ries, K. (2001). Segmenting Conversations by Topic, Initiative and Style. In *Proceedings of ACM SIGIR Workshop on Information Retrieval Techniques for Speech Applications*, pages 51–66, New Orleans.

Shriberg, E., Dhillon, R., Bhagat, S., Ang, J., and Carvey, H. (2004). The ICSI Meeting Recorder Dialog Act (MRDA) Corpus. In *Proceedings of SIGdial Workshop on Discourse and Dialogue*, pages 97–100, Cambridge.

Siegel, S. and N. J. Castellan, J. (1988). *Nonparametric Statistics for the Behavioral Sciences*, 2nd edition, McGraw-Hill, New York.

Tür, G., Hakkani-Tür, D., Stolcke, A., and Shriberg, E. (2001). Integrating Prosodic and Lexical Cues for Automatic Topic Segmentation. *Computational Linguistics*, 27(1):31–57.

Wrede, B. and Shriberg, E. (2003). Spotting "Hot Spots" in Meetings: Human Judgements and Prosodic Cues. In *Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 2805–2808, Geneva.

Chapter 11

ANALYZING DEPENDENCIES BETWEEN STUDENT CERTAINNESS STATES AND TUTOR RESPONSES IN A SPOKEN DIALOGUE CORPUS

Kate Forbes-Riley
University of Pittsburgh
Learning Research and Development Center
Pittsburgh, PA, USA
forbesk@cs.pitt.edu

Diane J. Litman
University of Pittsburgh
Department of Computer Science & Learning Research and Development Center
Pittsburgh, PA, USA
litman@cs.pitt.edu

Abstract This study presents an approach for developing more empirically motivated affective dialogue tutorial systems. In particular, we use n-gram techniques from statistical natural language processing to identify dependencies between student affective states of certainty and subsequent tutor dialogue acts, in an annotated corpus of human–human spoken tutoring dialogues. We first represent our dialogues as bigrams of annotated student and tutor turns. We next use χ^2 analysis to identify dependent bigrams, i.e., where the student certainty and tutor dialogue act annotations are related in some way other than predicted by chance. Our results show dependencies between many student states and subsequent tutor dialogue acts. We then analyse the dependent bigrams both with respect to differences between observed and expected counts and with respect to correlations with learning; these analyses suggest ways that our current computer tutor can be enhanced to adapt its dialogue act generation based on these dependencies.

Keywords: Affect and attitudes; spoken dialogue systems; intelligent tutoring systems; corpus-based techniques and analysis; adaptive dialogue modelling

L. Dybkjær and W. Minker (eds.), Recent Trends in Discourse and Dialogue, 275–304.

1. Introduction

There has been increasing interest in *affective dialogue systems* (André et al., 2004), motivated by the belief that in human–human dialogues, conversational participants seem to be (at least to some degree) detecting and responding to the emotional states of other participants. Affective dialogue research addresses topics such as emotion recognition and synthesis, annotation, evaluation, and agents, and is being pursued in many application areas, including *intelligent tutoring systems* (Aist et al., 2002; Craig and Graesser, 2003; Bhatt et al., 2004; Johnson et al., 2004; Moore et al., 2004; Heylen et al., 2004; Zhang et al., 2004; Pon-Barry et al., 2004). However, while it seems intuitively plausible that human tutors do in fact vary their responses based on the detection of student affect,[1] to date this belief has largely been theoretically rather than empirically motivated. We propose using bigram-based techniques from statistical natural language processing, as a data-driven method for identifying relationships between student affect and tutor responses in a corpus of human–human spoken tutoring dialogues. Analysis of these relationships suggests strategies for enhancing our current computer tutor to adapt its dialogue act generation based on student affect.

To investigate affect and tutorial dialogue systems, we have built ITSPOKE (**I**ntelligent **T**utoring **SPOKE**n dialogue system) (Litman and Silliman, 2004), which is a *speech-enabled* version of the *text-based* Why2-Atlas conceptual physics tutoring system (VanLehn et al., 2002).[2] Our long term goal is to have this system detect and adapt to student affect, and to investigate whether such an affective version of our system improves learning and other measures of performance. To date we have collected corpora of both human–human and human–computer tutoring dialogues, and we have demonstrated the feasibility of annotating and recognising student emotions from lexical, acoustic–prosodic, and dialogue features automatically extractable from these corpora (Litman and Forbes-Riley, 2004a, b, 2006b; Forbes-Riley and Litman, 2004).

Here, we assume viable emotion recognition and move on to the next step: providing an empirical basis for enhancing our computer tutor to adaptively respond to student affect. We first show how to apply n-gram techniques used in other areas of computational linguistics to mine human–human dialogue corpora for dependent bigrams of student states and tutor responses. We then use our bigram analysis to show: (1) statistically significant dependencies exist between students' emotional states and our *human* tutor's dialogue act responses,

[1]We use the terms "affect" and "emotion" loosely to cover emotions and attitudes believed to be relevant for tutoring.

[2]We also use ITSPOKE to examine the utility of building *spoken* (as opposed to *typed*) dialogue tutors (e.g. Litman et al., 2004, 2006).

(2) by comparing differences between observed versus expected counts, the dependent bigrams identified as statistically significant suggest empirically-motivated adaptive strategies for future implementation in our *computer* tutor. This method should generalise to any domain with dialogue corpora labelled for user state and system response.

Because human–human tutoring generally yields higher student learning than human–computer tutoring, human behaviour is often considered the gold-standard when designing intelligent tutoring systems. However, whether replicating human adaptive strategies will indeed improve system performance can only be determined once such strategies have been isolated and implemented. We shed light on these issues by analysing the relationship between our dependent bigrams and student learning in both our human–human and human–computer corpora. Our results suggest first that not all human adaptive strategies are necessarily effective for increasing learning in our human–human tutoring corpus, and second, that these adaptive strategies may have differing effectiveness in a human–computer tutoring corpus.

In Section 2 we present our corpora of spoken tutoring dialogues and our annotation schemes for labelling student affect and tutor responses. In Section 3 we describe how bigram representations of annotated student and tutor turns are extracted from our human tutoring corpus. We then use the Chi Square (χ^2) test to identify and analyse dependent bigrams. In Section 4 we use Pearson's correlation to analyse relationships between dependent bigrams and student learning. Section 5 discusses related work, and Section 6 concludes and discusses current directions.

2. Spoken Tutoring Data and Annotation

2.1 The Spoken Tutoring Dialogue Corpora

Our data for this study comes from spoken interactions between a student and a human tutor, through which students learn to solve qualitative physics problems, i.e., thought-provoking "explain" or "why" type physics problems that can be answered without doing any mathematics.

Our *human–human* spoken dialogue tutoring corpus was collected from November 2003–April 2004, as part of a larger evaluation comparing student learning across typed and spoken human–human and human–computer dialogue tutoring conditions (Litman et al., 2004, 2006). For that evaluation, we also collected a *human–computer* corpus using our ITSPOKE spoken dialogue tutoring system (Litman and Silliman, 2004); the human tutor and ITSPOKE performed the same task.

The experimental procedure for collecting both corpora was as follows: (1) students were given a pre-test measuring their knowledge of physics, (2) students used a web and voice interface to work through a set of 5–10 qualitative

physics training problems with the (human or computer) tutor, and (3) students were given a post-test that is similar to the pre-test. Before working through the training problems, students were asked to read through a small document of background physics material.[3] The experiment typically took no more than 7 hours per student, and was performed in 1–2 sessions. Students were University of Pittsburgh students who had never taken a college level physics course, and who were native speakers of American English.

Each spoken dialogue involves one physics problem. The dialogue begins after a student types an essay answering the physics problem. The (human or computer) tutor analyses the student essay, then engages the student in dialogue to correct misconceptions and elicit more complete explanations. The student then revises the essay, thereby ending the dialogue or causing another round of dialogue/essay revision. Dialogue interaction between student and tutor was mediated via a web interface supplemented with a high-quality audio link. The student and the human tutor were separated by a partition, and spoke through head-mounted microphones. Each participant's speech was digitally recorded on a separate channel. In the human–human corpus, transcription and turn-segmentation of the student and tutor speech were done by a paid transcriber. The transcriber added a turn boundary when: (1) the speaker stopped speaking and the other party in the dialogue began to speak, (2) the speaker asked a question and stopped speaking to wait for an answer, (3) the other party in the dialogue interrupted the speaker and the speaker paused to allow the other party to speak. In the human–computer corpus, transcription and turn segmentation were performed by the ITSPOKE system.

Our *human–human* corpus, which is the focus of this study, consists of 128 transcribed and annotated spoken tutoring dialogues between 14 different students and one human tutor. On average, the dialogues last 18.1 minutes and contain 46.5 student and 43.0 tutor turns. In total, there are 5,778 student turns and 5,493 tutor turns in the corpus. Annotated[4] dialogue excerpts from this corpus are shown in Figures 1, 2, and 7 (punctuation is added for readability).

2.2 Annotating Student Certainty

In previous work, (Litman and Forbes-Riley, 2004a, b, 2006b; Forbes-Riley and Litman, 2004), we described and evaluated a scheme for manually annotating the student turns in our tutoring corpora for intuitively perceived, positive and negative emotions such as (un-)certainty, frustration, boredom, and

[3]In the human tutoring experiment, the pretest was given before the reading. In the computer tutoring experiment, the pretest was moved to after the reading, to allow us to measure learning gains caused by the experimental manipulation without confusing them with gains caused by the background reading. Also, in the computer tutoring experiment, all students worked through only five physics problems with ITSPOKE.
[4]All annotations were performed from both audio and transcription within a speech processing tool.

PHYSICS PROBLEM: Suppose a man is running in a straight line at constant speed. He throws a pumpkin straight up. Where will it land? Explain.

. . . dialogue excerpt at 16.6 minutes into session. . .

STUDENT$_{68}$: So the- when you throw it up the acceleration will stay the same? *(UNCERTAIN)*

TUTOR$_{69}$: Acceleration uh will always be the same because there is- that is being caused by force of gravity which is not changing. *[RS, EX]*

STUDENT$_{70}$: mm-k *(NEUTRAL)*

TUTOR$_{71}$: Acceleration is- it is in- what is the direction uh of this acceleration- acceleration due to gravity? *[SAQ]*

STUDENT$_{72}$: It's- the direction- it's downward. *(CERTAIN)*

TUTOR$_{73}$: Yes, it's vertically down. *[POS, RS]*

STUDENT$_{74}$: mm-k. *(NEUTRAL)*

. . . 53 seconds pass while student revises their essay. . .

STUDENT$_{75}$: So in relation to the velocity, when it's being thrown up the velocity increases- well no, it decreases and when it goes down it increases. *(MIXED)*

TUTOR$_{76}$: Yeah. *[POS]*

Figure 1. Annotated human–human corpus excerpt #1.

PHYSICS PROBLEM: Suppose a man is in a free-falling elevator and is holding his keys motionless right in front of his face. He then lets go. What will be the position of the keys relative to the man's face as time passes? Explain.

. . . dialogue excerpt at 4.0 minutes into session. . .

TUTOR$_{16}$: Um, ok, so now we are thinking in terms of the factors that govern this displacement. Um, now they are- the elevator is in freefall. So does what does that tell you about the motion of the keys and the person? *[RD, LAQ]*

STUDENT$_{17}$: Um, that they're only under one force and that's gravity. *(UNCERTAIN)*

TUTOR$_{18}$: Ok, that is the force. But what does it tell you- that's right and that is about the force, but what does that tell you about their motion? *[POS, LAQ]*

STUDENT$_{19}$: Ok, uh, the motion is- oh, is they're moving in the same direction con- they're constant. *(UNCERTAIN)*

TUTOR$_{20}$: Uh, ok. They are also in freefall. *[BO]*

Figure 2. Annotated human–human corpus excerpt #2.

enthusiasm; we also showed that once annotated, these emotion classes could be predicted significantly better than a majority class baseline using a variety of (para-) linguistic features automatically extractable from our corpora. However, because this emotion annotation was very labour-intensive, only 10 human–human dialogues and 15 human–computer dialogues from our corpora were annotated using this method.

- **uncertain**: Use this label only when you feel the student is clearly uncertain about what they are saying.
- **certain**: Use this label only when you feel the student is clearly certain about what they are saying.
- **mixed**: Use this label if you feel that the speaker conveyed some mixture of uncertain and certain utterances within the same turn.
- **neutral**: Use this label when you feel the speaker conveyed no sense of certainness. In other words, the speaker seemed neither clearly uncertain nor clearly certain (nor clearly mixed). This is the default case.

Figure 3. "Certainness" annotation.

In order to widen the scope of our current investigation into emotion *adaptation*, the present study makes use of a related but more limited emotion annotation scheme. In particular, student states of "Certainness" (Liscombe et al., 2005) are more prevalent than other student emotional states in our annotated dialogues, and are also of interest in other recent tutorial dialogue research (Bhatt et al., 2004; Moore et al., 2004; Pon-Barry et al., 2004). Thus for this study, we manually annotated all of the student turns in our human–human corpus for "Certainness".[5] This annotation uses one of four labels, defined in our manual as shown in Figure 3. Although evidence for these "Certainness" labels can come from many knowledge sources, such as lexical items (e.g., "I don't know") and/or acoustic-prosodic features (e.g., rising pitch, pausing), to avoid influencing the annotator's intuitive understanding of "Certainness" expression, and because particular cues are not used consistently or unambiguously across speakers, our annotation manual does not associate particular cues with particular labels. Rather, as shown in Figure 3, the annotator's decision is based wholly on intuitive judgement. Examples of *uncertain* student turns are shown in Figure 1 ($\mathbf{STUDENT}_{68}$) and Figure 2 ($\mathbf{STUDENT}_{17}$, $\mathbf{STUDENT}_{19}$). Examples of *certain* student turns are shown in Figure 1 ($\mathbf{STUDENT}_{72}$) and Figure 7 ($\mathbf{STUDENT}_{99}$, $\mathbf{STUDENT}_{101}$, $\mathbf{STUDENT}_{103}$). An example of a *mixed* student turn is shown in Figure 1 ($\mathbf{STUDENT}_{75}$). Examples of *neutral* student turns are shown in Figure 1 ($\mathbf{STUDENT}_{70}$, $\mathbf{STUDENT}_{74}$).

One annotator labelled all the student turns in our human–human corpus for "Certainness", yielding the distribution shown in Table 1.

We tested inter-annotator agreement using the 10 dialogues (505 student turns) in our human–human corpus that were previously labelled by a separate annotator using the more labour-intensive scheme described above. The

[5]Liscombe et al. (2005) show that using only acoustic-prosodic features as predictors, these student certainness annotations can be predicted with 76.42% accuracy.

Table 1. Student "Certainness" totals.

Certain	Mixed	Neutral	Uncertain
1,158	235	3,529	856

Table 2. Confusion matrix for Student Certainness inter-annotation.

	Uncertain	Neutral	Certain	Mixed
Uncertain	65	20	24	14
Neutral	5	246	14	5
Certain	4	50	43	6
Mixed	2	3	3	1

confusion matrix in Table 2 summarises the inter-annotator agreement across the Certainness labels, which yielded 0.50 Kappa. We view this as a lower bound for our inter-annotator agreement, since the annotation tasks being compared are non-identical. Although Kappa value interpretation is somewhat controversial and varies depending on the application field, Landis and Koch (1977) and others use the following agreement standard: 0.21–0.40 = "Fair"; 0.41–0.60 = "Moderate"; 0.61–0.80 = "Substantial"; 0.81–1.00 = "Almost Perfect". Other studies of emotion annotation in naturally occurring dialogues in other domains have also yielded Kappas in the range of "Moderate" agreement (Ang et al., 2002; Narayanan, 2002; Shafran et al., 2003).

2.3 Annotating Tutor Dialogue Acts

Also prior to the present study, each *tutor turn* in our corpus had been manually annotated for tutoring-specific dialogue acts as part of a project comparing tutor and student dialogue behaviour in human versus computer tutoring, and examining the relationships of these behaviours to student learning (Litman and Forbes-Riley, 2006a; Forbes-Riley et al., 2005). Our tagset of "Tutor Dialogue Acts" is shown and briefly defined in Figures 4–6. As shown, we distinguish three main types of Tutor Acts.[6] The "Tutor Feedback Acts" in Figure 4 label feedback based on the presence of *lexical items* in the tutor turn. Although these tags often coincide with the correctness of a student turn, they can also convey encouragement, or relate to the discourse level (e.g., "no I didn't say that") or to the student's earlier essay. Examples

[6]An additional tag, "NonSubstantive" (NS), was used for utterances that did not contribute to the physics discussion (e.g., social coordinations such as "Are you ready to begin?"). These utterances were removed from this analysis.

- **Positive Feedback (POS)**: positive feedback *lexical item* is present in the turn.
- **Negative Feedback (NEG)**: negative feedback *lexical item* is present in the turn.

Figure 4. Tutor Feedback Acts.

- **Short Answer Question (SAQ)**: concerns basic quantitative relationships.
- **Long Answer Question (LAQ)**: requires a definition or interpretation of concepts.
- **Deep Answer Question (DAQ)**: requires reasoning about causes and/or effects.

Figure 5. Tutor Question Acts.

- **Restatement (RS)**: repetitions and rewordings of prior student statement.
- **Recap (RC)**: restating student's overall argument or earlier-established points.
- **Request/Directive (RD)**: directions summarising expectations about student's overall argument.
- **Bottom Out (BO)**: complete answer supplied after student answer is incorrect, incomplete or unclear.
- **Hint (HN)**: partial answer supplied after student answer is incorrect, incomplete or unclear.
- **Expansion (EX)**: novel details about student answer supplied without first being queried to student.

Figure 6. Tutor State Acts.

of *Positive Feedback* are shown in Figure 1 (**TUTOR**$_{73}$, **TUTOR**$_{76}$), Figure 2 (**TUTOR**$_{18}$) and Figure 7 (**TUTOR**$_{98}$). Examples of *Negative Feedback* are shown in Figure 7 (**TUTOR**$_{100}$, **TUTOR**$_{104}$). Additional examples of both types of Feedback Acts are shown in the appendix.

The "Tutor Question Acts" in Figure 5 label the type of question that the tutor asks, in terms of their content and the expectation that the content presupposes with respect to the type of student answer required. Examples of *Short Answer Questions* are shown in Figure 1 (**TUTOR**$_{71}$) and Figure 7 (**TUTOR**$_{98}$, **TUTOR**$_{100}$). Examples of *Long Answer Questions* are shown in Figure 2 (**TUTOR**$_{16}$, **TUTOR**$_{18}$). Examples of *Deep Answer Questions* are shown in Figure 7 (**TUTOR**$_{102}$, **TUTOR**$_{104}$).

The "Tutor State Acts" in Figure 6 summarise or clarify the current state of the student's argument, based on the prior student turn(s). Examples of *Restatements* are shown in Figure 1 (**TUTOR**$_{69}$, **TUTOR**$_{73}$) and Figure 7 (**TUTOR**$_{100}$, **TUTOR**$_{102}$). An example of a *Recap* is shown in Figure 7 (**TUTOR**$_{98}$). An example of a *Request/Directive* is shown in Figure 2 (**TUTOR**$_{16}$). Examples of *Bottom Outs* are shown in Figure 2 (**TUTOR**$_{20}$)

PHYSICS PROBLEM: Two closed containers look the same, but one is packed with lead and the other with a few feathers. How could you determine which had more mass if you and the containers were floating in a weightless condition in outer space? Explain.

. . . dialogue excerpt at 16.5 minutes into session. . .

TUTOR$_{98}$: Yes, we are all learning. Ok, so uh now uh you apply the same push for the same amount of time for on both the containers. Then what would you compare to distinguish between them? *[POS, RC, SAQ]*

STUDENT$_{99}$: I would be comparing their rate of velocity. *(CERTAIN)*

TUTOR$_{100}$: Not rate. You will be comparing their velocity, you see, rate will imply that something is changing which there is no change, velocity is constant. So you will surely compare their velocities- which one will be faster? *[NEG, HN, RS, SAQ]*

STUDENT$_{101}$: The feathers. *(CERTAIN)*

TUTOR$_{102}$: The feathers- why? *[RS, DAQ]*

STUDENT$_{103}$: Because there's less matter. *(CERTAIN)*

TUTOR$_{104}$: No no because they accelerated more- because for the same amount of time, the acceleration in the container which contains feather was more. Therefore it acquired a larger velocity- uh, and why did it acquire lar-larger uh, why did it have greater acceleration? *[NEG, BO, EX, DAQ]*

Figure 7. Annotated human–human corpus excerpt #3.

and Figure 7 (**TUTOR**$_{104}$). An example of a *Hint* is shown in Figure 7 (**TUTOR**$_{100}$). Examples of *Expansions* are shown in Figure 1 (**TUTOR**$_{69}$) and Figure 7 (**TUTOR**$_{104}$).

This tagset was developed based on pilot annotation studies using similar tagsets previously applied in other tutorial dialogue projects, e.g., as in Graesser and Person (1994), Graesser et al. (1995), Pilkington (1999) and Johnson et al. (2004). Note that since tutoring dialogues have a number of tutoring-specific dialogue acts (e.g., hinting), researchers working on tutorial dialogue typically use tutoring-specific dialogue act tagsets rather than more domain-independent schemes such as DAMSL (Core and Allen, 1997) (although DAMSL is still somewhat biased towards task-oriented dialogue). Rickel et al. (2001) present a first step towards bridging this gap, by integrating an initial set of tutoring-specific acts into a more general collaborative discourse framework. Similarly, Wolska et al. (2004) are extending DAMSL to address the needs of tutoring. Our Feedback and Question Acts have primarily backward- and forward-looking functions respectively, in DAMSL.

As our corpus dialogue excerpts in Figures 1, 2, and 7 illustrate, most tutor turns are labelled with multiple Tutor Acts. Applying the Dialogue Act coding scheme to our tutor turns yielded 7,201 Tutor Acts on the 5,493 tutor turns in our human–human corpus, distributed across our Tutor Dialogue Acts as shown in Table 3.

Table 3. Tutor Dialogue Act totals.

SAQ	LAQ	DAQ	POS	NEG	RS	RC	RD	BO	HN	EX
1,205	167	586	902	203	1,273	305	298	336	865	1,061

Table 4. Confusion matrix for Tutor Act inter-annotation.

	DQ	LQ	SQ	BO	EX	HN	RD	RC	RS	NEG	POS	NS
DQ	25	1	14	0	0	2	2	0	0	0	0	4
LQ	1	4	1	0	0	0	0	0	0	0	0	1
SQ	3	9	60	0	0	2	1	0	1	0	0	7
BO	0	0	1	12	2	0	1	1	3	0	0	4
EX	0	0	0	0	18	3	6	11	0	0	0	20
HN	0	1	0	2	8	9	11	7	0	16	0	17
RD	0	0	0	0	0	0	5	0	0	0	0	1
RC	0	0	1	0	2	0	0	14	1	0	0	4
RS	0	0	0	3	1	1	2	13	35	1	1	21
NEG	0	0	0	0	0	0	0	0	0	9	0	4
POS	0	0	0	0	0	0	0	0	0	1	34	18
NS	0	0	1	1	0	0	10	0	2	3	0	68

While one annotator labelled the entire corpus, a second annotator separately annotated 8 human–human dialogues containing 330 tutor turns, for the purposes of an inter-annotator agreement study. Since the unit of analysis that ITSPOKE currently processes is *the turn*, the studies herein are focused at the turn-level. However, since our "Tutor Dialogue Acts" coding scheme labels utterances within the turn, we measured inter-annotator agreement on the 548 Tutor Acts within the 8 inter-annotated dialogues. The confusion matrix in Table 4 summarises the inter-annotator agreement across these Tutor Acts,[7] which yielded 0.48 Kappa ("Moderate"); this agreement improves to 0.63 Kappa ("Substantial") if we collapse the categories into their more general types (Question Acts, State Acts, and Feedback Acts). These results suggest that the annotation is reliable.

[7]Note that we include the NS (NonSubstantive) tag here in the confusion matrix and in the Kappa computation, although this tag was excluded from our analyses since it labels utterances that do not contribute to the physics discussion.

3. Data Analysis

We hypothesize that there are dependencies between student emotional states (as represented by the "Certainness" labels) and subsequent tutor responses (as represented by "Tutor Dialogue Act" labels), and that analysing these dependencies can suggest ways of incorporating techniques for adapting to student emotions into our computer tutor. We test these hypotheses by extracting a bigram representation of student and tutor turns from our annotated dialogues, computing the dependencies of the bigram permutations using Chi Square (χ^2) analyses, and drawing conclusions from the significant results.

3.1 Dialogue Bigrams

We view the sequence: "Student Turn, Tutor Turn" as our bigram unit, whose individual elements constitute "words" in the bigram. In Figure 7 there are three such units: **STUDENT**$_{99}$ - **TUTOR**$_{100}$, **STUDENT**$_{101}$ - **TUTOR**$_{102}$, and **STUDENT**$_{103}$ - **TUTOR**$_{104}$. Because our goal in this study is to analyse tutor responses, we extract all and only these units from our dialogues for analysis. In particular, we do not extract bigrams of the form: "Tutor Turn$_n$ – Student Turn$_{n+1}$", although we will do so in a future study when we analyse student responses to tutor actions. This decision is akin to disregarding word-level bigrams that cross sentence boundaries. Here, the sequence: "Student Turn, Tutor Turn" is our "dialogue sentence", and we are interested in all possible permutations of our student and tutor turn annotations in our data that combine to produce these dialogue sentences.

After extracting the annotated "Student Turn, Tutor Turn" bigrams, we sought to investigate the dependency between student emotional states and tutor responses. Although each of our student turns was labelled with a single "Certainty" tag, frequently our tutor turns were labelled with multiple "Tutor Act" tags, as noted above. Because there are 11 "Tutor Act" tags, and no limits on tag combinations per turn, it is not surprising that in our 4,921 extracted bigrams, we found 478 unique tag combinations in the tutor turns, 294 of which occurred only once. Treating each tagged tutor turn as a unique "word" would thus yield a data sparsity problem for our analysis of bigram dependencies. Due to this data sparsity problem, a question we can ask instead, is: is the tutor's *inclusion* of a particular Tutor Act in a tutor turn dependent on the student's state of certainness in the prior turn?

That is, we decided to approach the dependency analysis by considering the presence or absence of each Tutor Act tag separately. In other words, we performed 11 different analyses, one for each Tutor Act tag T, each time asking the question: is there a dependency between student emotional state and a tutor response containing T? More formally, for each analysis, we took our set of "Student Turn, Tutor Turn" bigrams, and replaced all annotated tutor turns

TUTOR$_{98}$: *[POS, RC, SAQ]* $\longrightarrow$ *[POS]*
TUTOR$_{100}$: *[NEG, HN, RS, SAQ]* $\longrightarrow$ *[notPOS]*
TUTOR$_{102}$: *[RS, DAQ]* $\longrightarrow$ *[notPOS]*
TUTOR$_{104}$: *[NEG, BO, EX, DAQ]* $\longrightarrow$ *[notPOS]*

Figure 8. Tutor turns from excerpt #3 in POS/notPOS analysis.

containing T with only **T**, and all not containing T with only **not T**. The result was 11 different sets of 4,921 "Student Turn, Tutor Turn" bigrams. As an example, we show in Figure 8 how the tutor turns in Figure 7 are converted within the "POS/notPOS" analysis: As shown, we have replaced all tutor tagged turns containing POS with only "POS", and we have replaced all tutor tagged turns not containing POS with only "notPOS".

The benefit of these multiple analyses is that we can ask specific questions directly motivated by what our computer tutor can do. For example, in the POS/notPOS analysis, we ask: should student emotional states impact whether the computer tutor generates positive feedback? Currently, there is no emotion adaptation by our computer tutor – it generates positive feedback independently of student emotional states, and independently of any other Tutor Acts that it generates. The same is true for each Tutor Act generated by our computer tutor.

3.2 Chi Square (χ^2) Analyses

We analysed bigram dependency using the Chi Square (χ^2) test.[8] In this section we illustrate our analysis method, using the set of "Certainness – POS/notPOS" bigrams. In the next section we discuss the results of performing this same analysis on all 11 sets of "Student Certainness – Tutor Act" bigrams.

χ^2 tests the statistical significance of the relationship between two variables in a dataset. Our observed "Certainness – POS" bigram permutations are reported as a bivariate table in Table 5. For example, we observed 252 **neutral – POS** bigrams, and 2,517 **neutral – notPOS** bigrams. Row totals show the number of bigrams containing the first bigram "word" (e.g., 2,769 bigrams contained "neutral" as the first token, followed by "POS" or "notPOS"). Column totals show the number of bigrams containing the second bigram "word" (e.g., 781 bigrams contained "POS" as the second token).

χ^2 compares these observed counts with the counts that would be expected if there were no relationship at all between the two variables in a larger pop-

[8] A good tutorial for using the χ^2 test of statistical significance is found here: http://www.georgetown.edu/faculty/ballc/webtools/web_chi_tut.html.

Table 5. Observed student "Certainness" – tutor "Positive Feedback" bigrams.

Certainness	POS	notPOS	Total
Neutral	252	2,517	2,769
Certain	273	831	1,104
Uncertain	185	631	816
Mixed	71	161	232
Total	781	4,140	4,921

Table 6. Expected student "Certainness" – tutor "Positive Feedback" bigrams.

Certainness	POS	notPOS	Total
Neutral	439.46	2,329.54	2,769
Certain	175.21	928.79	1,104
Uncertain	129.51	686.49	816
Mixed	36.82	195.18	232
Total	781	4,140	4,921

ulation (the *null* hypothesis). For each cell c in Table 5, the expected count is computed as: (c's row total * c's column total)/(total bigrams). Expected counts for Table 5 are shown in Table 6.

A χ^2 value assesses whether the differences between observed and expected counts are large enough to conclude that a statistically significant relationship exists between the two variables. The χ^2 value for the table is computed by summing the χ^2 value for each cell, which is computed as follows: (observed value – expected value)^2/expected value. The total χ^2 value for Table 5 is 225.92. χ^2 would be 0 if observed and expected counts were equal. However some variation is required (the "critical χ^2 value"), to account for the table's degree of freedom and one's chosen probability of exceeding any sampling error (typically 0.05–0.001). Our actual χ^2 value must be larger than this critical value in order to conclude that there is a statistically significant relationship between our two variables. Table 7 shows the critical χ^2 values at different error thresholds for tables having 1 and 3 degrees of freedom, where degrees of freedom is computed as (#rows - 1) * (#columns - 1). Critical χ^2 values are listed in most statistics textbooks.

Table 5 has 3 degrees of freedom, and its χ^2 value of 225.92 greatly exceeds the critical value for 3 degrees of freedom even at $p \leq 0.001$. We thus conclude that there is a statistically significant dependency between Certainness and Positive Feedback.

Table 7. Critical χ^2 values.

Degrees of Freedom	$p \leq 0.05$	$p \leq 0.01$	$p \leq 0.001$
1	3.84	6.64	10.83
3	7.82	11.35	16.27

Table 8. NSP output: **Certainness – POS** bigrams.

Bigram	Rank	χ^2	Total	Token_1	Token_2
Neutral - POS	1	217.35	252	2,769	781
Certain - POS	2	83.63	273	1,104	781
Mixed - POS	3	39.58	71	232	781
Uncertain - POS	4	33.88	185	816	781

We can look more deeply into this overall dependency by calculating the statistical significance of the dependencies between each specific "Certainness" tag and the Positive Feedback tag. The freely available Ngram Statistics Package (NSP) (Banerjee and Pedersen, 2003) computes these χ^2 values automatically when we input each set of our "Student Certainness – Tutor Act" bigrams. Table 8 shows the resulting NSP output for the POS/notPOS analysis. Each row shows: (1) the bigram, (2) its rank (according to highest χ^2 value), (3) its χ^2 value, (4) the total number of occurrences of this bigram, (5) the number of times the first token in this bigram occurs first in any bigram, (6) the number of times the second token in this bigram occurs last in any bigram.

Each row in Table 8 can alternatively be viewed as a 2×2 table of observed counts. For example, the table for the **neutral – POS** bigram has a "neutral" row (identical to that in Table 5) and a "non-neutral" row (computed by summing all the non-neutral rows in Table 5). This table has one degree of freedom.

As shown, all of the bigrams in Table 8 have χ^2 values exceeding the critical value for 1 degree of freedom even at $p \leq 0.001$. We thus conclude that there are statistically significant dependencies between each of the Certainness tags and Positive Feedback.[9] In the next section we will see cases where there is an *overall* significant dependency, but significant dependencies only for *a subset* of the four Certainness tags.

[9]Note that the χ^2 value for each of the bigrams in Table 8 is identical to its "Certainness – notPOS" counterpart. This can be understood by observing that the 2 X 2 observed (and expected) table for each "Certainness – POS" bigram is identical to its "notPOS" counterpart, *except* that the columns are flipped. That is, "not notPOS" is equivalent to "POS".

Finally, we can compare the difference between observed and expected values for the statistically significant dependent bigrams that have been identified. For example, by comparing Tables 5 and 6, we see that the human tutor responds with positive feedback more than expected after emotional turns, and less than expected after neutral turns. This suggests an adaptive technique for our computer tutoring system too: ITSPOKE could adapt to non-neutral emotional states by generating more positive feedback (independently of whether the Certainness value is certain, uncertain, or mixed).

3.3 Results and Discussion

In essence, for each of the 11 Tutor Acts described in Figures 4–6, the first part of our χ^2 analysis determines whether or not there is an overall dependency between Student Certainness and that specific Tutor Act. The second part then determines how this dependency is distributed across individual Student Certainness states. In this section, we present and discuss our results of the χ^2 analysis across all 11 sets of our "Certainness – Tutor Act" bigrams. Note that the tables present only our best results, where the χ^2 value exceeded the critical value at $p \leq 0.05$ (7.82 and 3.84 for 3 and 1 degrees of freedom, respectively). If a bigram's χ^2 value did not exceed this critical value, it is not shown.

Table 9 presents our best results across our two sets of "Certainness – Feedback Act" bigrams. The first four columns show the bigram, along with its observed and expected counts and its χ^2 value. The first row for each set shows the χ^2 value for the overall dependency between Certainness and Feedback (e.g. 225.92 for **CERT – POS**). The remaining rows per set are ranked

Table 9. Observed, expected, and χ^2 values for dependent "Certainness" – "Feedback" bigrams ($p \leq 0.05$).

Bigram	Observed	Expected	χ^2	# Students
CERT - POS	781	781	225.92	12
Neutral - POS	252	439.46	217.35	13
Certain - POS	273	175.21	83.63	9
Mixed - POS	71	36.82	39.58	6
Uncertain - POS	185	129.51	33.88	6
CERT - NEG	196	196	135.96	11
Neutral - NEG	34	110.29	125.67	7
Uncertain - NEG	68	32.5	48.41	7
Mixed - NEG	24	9.24	25.77	5
Certain - NEG	70	43.97	20.69	3

according to the χ^2 values for the specific dependencies between each "Certainness" tag and the "Feedback" tag (e.g., 217.35 for **neutral – POS**). Note that, while all bigrams shown are statistically significant at $p \leq 0.05$, as the χ^2 values increase above the critical value, the results become more significant. Finally, recall that our bigrams were extracted from all the student turns in our corpus, and thus these dependent bigrams represent dependencies that hold across all 14 students in our corpus. For comparison, the final column in the table shows the total number of individual students (out of the 14 possible) for which this same dependency also holds at $p \leq 0.05$, when individual χ^2 values are computed for each student's individual observed and expected bigram counts.[10]

As shown, there are strong overall dependencies between Student Certainness and both Positive and Negative Tutor Feedback. There are also dependencies between every specific Certainness tag and both Positive and Negative Feedback. Most of these dependencies also hold for a majority of students individually. Moreover, for both types of feedback, the tutor responds with more feedback than expected after all emotional student turns (non-neutral), and with less feedback than expected after neutral student turns. This suggests that an increased use of feedback is a viable adaptation to non-neutral emotional states. Of course, the type of feedback adaptation (POS or NEG) will also be influenced by whether the student answer is correct, which we will return to below.

Table 10 presents our best results across our three sets of "Certainness – Question Act" bigrams. As shown, there are overall dependencies between Student Certainness and all of the Tutor Question Act types. Within the Short Answer Question (SAQ) bigrams, the tutor responds to certain and uncertain turns with slightly more Short Answer Questions than expected, and to neutral turns with slightly fewer Short Answer Questions than expected. Similarly, within the Long Answer Question (LAQ) bigrams, the tutor responds to mixed and uncertain turns with slightly more Long Answer Questions than expected, and to neutral turns with slightly fewer Long Answer Questions than expected. The dependency between Student Certainness and Tutor Deep Answer Questions (DAQ) is wholly explained by the **uncertain – DAQ** bigram, where the tutor responds to uncertain turns with slightly fewer Deep Answer Questions than expected. However, most of these χ^2 values barely exceed the critical value even at $p \leq 0.05$, and are dependent for very few students individually. These results suggest that "Question Acts" aren't highly relevant overall for adaptation to student Certainness; we hypothesize that they will play a more significant role when we analyse student emotional responses to tutor actions.

[10]We thank the attendees at SIGdial 2005 for suggesting this comparison.

Table 10. Observed, expected, and χ^2 values for dependent "Certainness" – "Question Act" bigrams ($p \leq 0.05$).

Bigram	Observed	Expected	χ^2	# Students
CERT - SAQ	1,135	1,135	18.06	1
Neutral - SAQ	588	638.65	11.94	3
Certain - SAQ	290	254.63	8.23	4
Uncertain - SAQ	213	188.21	5.09	2
CERT - LAQ	160	160	11.57	2
Mixed - LAQ	14	7.54	6.00	1
Neutral - LAQ	76	90.03	5.17	2
Uncertain - LAQ	36	26.53	4.19	0
CERT - DAQ	550	550	9.15	0
Uncertain - DAQ	67	91.2	8.67	0

Table 11 presents our best results across our six sets of "Certainness – State Act" bigrams. As shown, there is a strong overall dependency between Student Certainness and Tutor Restatements, explained by the strong dependencies of the **certain – RS** and **neutral – RS** bigrams, where the tutor responds to certain turns with more Restatements than expected, and to neutral turns with less Restatements than expected. These dependencies hold across most students individually as well. There is a much weaker dependency between Student Certainness and Tutor Recaps, explained by the **neutral – RC** and **certain – RC** bigram, where the tutor responds to neutral turns with slightly more Recaps than expected and to certain turns with slightly less Recaps than expected. There are no dependencies at all between Student Certainness and Tutor Request Directives (**RD**). Although these three Tutor State Acts all serve a summary purpose with respect to the student's argument, RC and RD are defined as more general acts whose use is based on the overall discussion so far. Only RS addresses the immediately prior student turn; thus it is not surprising that its use shows a stronger dependency on the prior student certainness. The tutor's increased use of RS after certain turns suggests an adaptation strategy of increasing or maintaining student certainty by repeating information about which the student has already shown certainty.

The remaining three bigram sets contain Tutor Acts that clarify the prior student answer. First, there is an overall dependency between Student Certainness and Tutor Bottom Outs, largely explained by the dependencies of the **neutral – BO** and **uncertain – BO** bigrams, which both also hold for a number of students individually. After uncertain turns, the tutor "Bottoms Out" (supplies

Table 11. Observed, expected, and χ^2 values for dependent "Certainness" – "State Act" bigrams ($p \leq 0.05$).

Bigram	Observed	Expected	χ^2	# Students
CERT - RS	1,102	1,102	169.18	12
Certain - RS	402	247.23	160.96	12
Neutral - RS	477	620.08	97.29	11
CERT - RC	289	289	20.15	1
Neutral - RC	199	162.62	19.77	4
Certain - RC	43	64.84	10.07	1
CERT - BO	308	308	82.52	6
Neutral - BO	103	173.31	69.58	7
Uncertain - BO	97	51.07	52.82	4
Certain - BO	87	69.10	6.38	1
CERT - HN	779	779	37.07	4
Mixed - HN	64	36.73	25.25	4
Neutral - HN	383	438.34	18.98	3
Certain - HN	201	174.76	6.03	1
CERT - EXP	998	998	47.08	5
Neutral - EX	651	561.57	40.86	4
Uncertain - EX	109	165.49	29.00	4
Certain - EX	197	223.90	5.23	2

the complete answer) more than expected, and after neutral turns, less than expected. This suggests a straightforward adaptive technique for student uncertainty. The χ^2 value of the **certain – BO** bigram barely exceeds the critical value even at $p \leq 0.05$ and only holds for one student individually.

There is also an overall dependency between Student Certainness and Tutor Hints, largely explained by the dependencies of the **mixed – HN** and **neutral – HN** bigrams. After mixed turns, the tutor "Hints" (supplies a partial answer) more than expected, and after neutral turns, less than expected. This suggests an adaptive technique similar to the BO case, except the tutor gives less of the answer because there is less uncertainty (i.e., there is more certainty because the student turn is mixed). Again, the χ^2 value of the **certain – HN** bigram barely exceeds the critical value even at $p \leq 0.05$ and only holds for one student individually.

Finally, there is an overall dependency between Student Certainness and Tutor Expansions, largely explained by the dependencies of the **neutral – EX** and **uncertain – EX** bigrams. In this case, however, the tutor responds with an "Expansion" (supplying novel details) more than expected after neutral turns, and less than expected after uncertain turns. This suggests another adaptive technique to uncertainty, whereby the tutor avoids overwhelming the uncertain student with unexpected details. Again, the χ^2 value of the **certain – EX** bigram barely exceeds the critical value even at $p \leq 0.05$.

4. Correlations between Dependent Bigrams and Student Learning

Because human–human tutoring generally yields higher student learning than human–computer tutoring, human behaviour is often considered the gold-standard when designing intelligent tutoring systems. Student learning is often considered the primary evaluation metric for evaluating the effectiveness of intelligent tutoring systems.

The dependent bigrams discussed in the previous section showed how our human tutor responds differently to different states of student certainness. Whether replicating these dependencies as adaptive strategies in ITSPOKE will improve system effectiveness can only be determined after implementation, when we can compare the performance of the adaptive system with its non-adaptive counterpart.

However, we can shed light on the effectiveness of these dependencies in our human–human corpus. As noted above, student learning is a primary evaluation metric in the tutoring domain. In prior work we used Pearson's partial correlations to analyse the relationship between student learning and shallow dialogue measures such as student turn length (Litman et al., 2004, 2006), as well as deeper measures using our Dialogue Act annotations (Litman and Forbes-Riley, 2006a; Forbes-Riley et al., 2005). For the pilot study discussed in this section, we used Pearson's partial correlations to analyse the relationships between student learning and our dependent bigrams in our human–human corpus. First, for each dependent bigram in Tables 9–11, we computed a *per-student total*: we counted the total number of times that each dependent bigram occurred across all the dialogues of each student, for the 14 students in the corpus. Second, we computed a Pearson's partial correlation across all students between each of these total bigram counts and post-test score, controlling for pre-test score. The human–human means for the (multiple-choice) pre- and post-tests were 0.42 and 0.72, respectively.

Briefly, a Pearson's correlation measures the strength (**R**) of the linear relationship between two variables across a population: in our case, the bigram counts and post-test scores across our human–human corpus. Positive **R** val-

ues indicate that as values of one variable (X) increase, so do values of the other variable (Y); negative **R** values indicate that as values of variable X increase, values of variable Y decrease. A **p**-value is then computed to assess the significance of this relationship. A Pearson's partial correlation is the correlation between two variables with the influence of a third variable removed from both. In our case, pre-test and post-test scores are already significantly correlated with each other in our human–human corpus (R = 0.65, p = 0.01); thus we are interested in determining the correlation between our bigram counts and post-test scores after removing the influence of pre-test scores.

Table 12 presents our significant results ($p < 0.05$) on these correlations. The first column lists the dependent bigram, the next two columns show the mean and standard deviation (across all students) for the bigram count, and the last two columns give the strength (R) and significance (p) of the Pearson's partial correlation between post-test score and the bigram count, controlled for pre-test score. The first row shows that the human tutor response of Negative Feedback to student Certainness overall (the **CERT – NEG** bigram) occurred on average 14 times per student, and there is a significant (p = 0.03) negative correlation ($R = -0.60$) between this bigram and student learning. The next two rows also show significant negative correlations between student learning and the human tutor response of Negative Feedback to both certain and uncertain student turns. Since Negative Feedback often follows an incorrect student turn, we hypothesize that these negative correlations may reflect a more general relationship between student incorrectness and learning. The last row shows a significant negative correlation between student learning and the human tutor response of Long Answer Questions to student Certainness overall. Note that in essence, the **CERT – NEG** and **CERT – LAQ** bigrams can be viewed as tutor act unigrams, in that they represent the occurrences of the tutor act (NEG or LAQ) immediately preceded by any student turn. So for example, the fact that the **CERT – LAQ** bigram negatively correlates with student learning suggests that an increased use of Long Answer Questions to respond to any student turn

Table 12. Correlations between student learning and the dependent "Student Certainness – Tutor Act" bigrams: Human–Human Corpus (14 students).

Bigram	Mean	Std. Dev.	R	p
CERT - NEG	14.00	7.51	−0.60	0.03
Certain - NEG	5.00	4.10	−0.58	0.04
Uncertain - NEG	4.86	2.96	−0.67	0.01
CERT - LAQ	11.43	7.91	−0.59	0.04

is related to decreased learning. Of course, as with all of our correlation results, further experimentation would be required in order to claim that an increased use of this type of question is causally related to student learning.

Finally, the majority of dependent bigrams in Tables 9–11 do not correlate significantly with learning, and thus do not appear in Table 12. Together, these results show that although there are statistically significant dependencies between tutor Feedback, Question and State Acts and prior student Certainness states, these dependencies do not in and of themselves have a significant positive impact on student learning in our human–human tutoring corpus.

Taken in isolation, these results also suggest that these dependencies may not yield effective adaptive techniques for increasing student learning in our ITSPOKE tutoring system, either. However, we have previously shown that our human–human and human–computer tutoring corpora are very different with respect to how both shallow and deep dialogue measures relate to student learning (Litman et al., 2004, 2006; Litman and Forbes-Riley, 2006a; Forbes-Riley et al., 2005). Thus, our next step in this pilot study was to shed light on the *potential* effectiveness of adding our dependencies as adaptive strategies to ITSPOKE, by evaluating the *current* relationship between the dependent bigrams and student learning in our *human–computer* corpus. For although ITSPOKE currently does not adapt to student emotions, the student and tutor turns in our human–computer corpus have already been annotated for both "Certainness" and Tutor Acts, respectively; thus we can study the current relationship between student learning and the bigrams that were found to be dependent in our human–human corpus. As above, for each dependent bigram in Tables 9–11, we computed the total number of times that bigram occurred per student, for each of the 20 students in our human–computer corpus. Again, pre-test and post-test scores were already significantly correlated in our human–computer corpus ($R = 0.46$, $p = 0.04$). Thus we computed a Pearson's partial correlation between each bigram total and post-test score across all students, controlled for pre-test score. The human–computer means for the (multiple-choice) pre- and post-tests were 0.48 and 0.69, respectively.

Table 13 presents our significant results ($p < 0.05$) on these correlations. The first row shows a significant positive correlation between student learning

Table 13. Correlations between student learning and the dependent "Student Certainness – Tutor Act" bigrams: Human–computer corpus (20 students).

Bigram	**Mean**	**Std. Dev.**	**R**	**p**
Uncertain - LAQ	1.85	1.50	0.48	0.04
Neutral - RC	8.95	2.91	−0.72	0.00

and ITSPOKE Long Answer Questions occurring after uncertain student turns. This suggests that the use of Long Answer Questions to adapt to uncertain student turns *would be* an effective adaptation strategy for increasing student learning in ITSPOKE, even though Table 12 showed a negative correlation between the **CERT – LAQ** bigram and student learning in our human–human corpus.

The second row in Table 13 shows a significant negative correlation between student learning and ITSPOKE Recaps occurring after neutral student turns. However, as was shown in Table 11, we found that the human tutor responded with significantly *more* Recaps than expected after neutral student turns in our human–human corpus. Since the correlation between this bigram and learning is already negative in our human–computer corpus, increased use of this bigram as an adaptive strategy may not be effective at improving student learning in ITSPOKE. Finally, none of the other dependent bigrams currently show a significant correlation with learning in our human–computer corpus.

Overall, this pilot study showed that although we found numerous dependent bigrams in our human–human corpus representing statistically significant human tutor responses to student Certainness, only a handful showed a significant correlation with student learning in our human–human corpus. However, our results suggest that these adaptive strategies may have a different impact on student learning in ITSPOKE. A useful extension to this work will be to investigate which of the "Student Certainness – Tutor Act" bigrams show dependencies in our ITSPOKE corpus. For although the computer tutor is not adapting to student Certainness, these dependencies will give insight into how different the computer tutor currently is from the human tutor, and will also give further insight into the bigram correlations that were or were not found in our ITSPOKE corpus.

Finally, we hypothesize that the human tutor responses to Certainness are not solely intended to improve student learning. For example, they may also be aimed at increasing student motivation (Baylor et al., 2003) or persistence (Aist et al., 2002) or even tutor likeability. Student learning is thus only one way of measuring the effectiveness of adaptive strategies. As in other types of spoken dialogue systems (e.g., Walker et al., 2002), these and other evaluation metrics (e.g., user satisfaction) can also be measured and used to evaluate the effectiveness of our adaptive strategies after implementation in ITSPOKE.

5. Related Work

While there have been other approaches to using dialogue n-grams (e.g., Stolcke et al., 2000, Reithinger et al., 1996), such n-grams have typically consisted of only dialogue acts, although Higashinaka et al. (2003) propose computing bigrams of dialogue state and following dialogue act. Moreover,

these methods have been used to compute n-gram probabilities for implementing statistical components. We propose a new use of these methods: to mine corpora for only the significant n-grams, for use in designing strategies for *adapting to student affect* in a computational system. Previous Ngram Statistics Package (NSP) applications have focused on extracting significant *word* n-grams (Banerjee and Pedersen, 2003), while our "dialogue" bigrams are constructed from multiple turn-level annotations of *student certainness* and *tutor dialogue acts*. Although Shah et al. (2002) have mined a human tutoring corpus for significant "dialogue" bigrams to aid in the design of adaptive dialogue strategies, their goal is to generate appropriate tutor responses to student *initiative*. Their bigrams consist of manually labelled student initiative and tutor response in terms of mutually exclusive categories of communicative goals.

In the area of affective tutorial dialogue, Bhatt et al. (2004) have coded (typed) tutoring dialogues for student hedging and affect. Their focus, however, has been on identifying differences in human versus computer tutoring, while our focus has been on analyzing relationships between student states and tutor responses. Conversely, Johnson et al. (2004) have coded their tutoring dialogue corpora with tutoring-specific dialogue acts, but have not annotated student affect, and to date have performed only qualitative analyses. Finally, while our research focuses on dialogue acts, others are studying affect and different linguistic phenomena such as lexical choice (Moore et al., 2004).

6. Conclusions and Current Directions

This study proposes an empirically motivated approach to developing techniques for adapting to student affect in our intelligent tutoring spoken dialogue system (ITSPOKE). Furthermore, our method of extracting and analysing dialogue bigrams to develop adaptation techniques generalises to other domains that seek to use user affective states to trigger system adaptation. We first extract "dialogue bigrams" from a corpus of human–human spoken tutoring dialogues annotated for student Certainness states and tutor Dialogue Acts. We then use the Chi Square (χ^2) test to determine which bigrams are dependent, such that there is a significant relationship between the use of a Tutor Act and a prior state of Student Certainness.

Analysis of our dependent bigrams indicates specific strategies for emotion adaptation that we can implement in ITSPOKE. Specifically, we found many dependencies between student states of certainness and subsequent tutor dialogue acts, which suggest ways that our computer tutor can be enhanced to adapt dialogue act generation to student affective states. In particular, our results suggest that "Bottoming Out" and avoiding "Expansions" are viable adaptations to student uncertainty, whereas "Hinting" is a viable adaptation to a student state of mixed certainness, and adapting by "Restatements" may help

maintain a state of student certainty. Positive and Negative Feedback occur significantly more than expected after all the non-neutral student states, and thus seem to be a generally "human" way of responding to student emotions.

Further analysis of the correlations between these dependent bigrams and student learning suggests that these dependencies are not necessarily related to increased student learning in our human–human tutoring corpus. In particular, although we found numerous dependent bigrams, only a handful showed a significant correlation with student learning, and these were negative. However, our correlation analyses also suggest that these adaptive strategies may have a different relationship to student learning in ITSPOKE. For example, although our human tutor's use of Long Answer Questions to respond to student Certainness states was negatively correlated with learning in our human–human corpus, we found a positive correlation between student learning and Long Answer Questions that occurred after student uncertain turns in our (non-adaptive) human–computer corpus. This suggests that using Long Answer Questions to adapt to student uncertainty would nevertheless be an effective adaptive strategy for our computer tutor.

Finally, we emphasise that student learning is only one way of measuring the effectiveness of adaptive strategies. We hypothesize that our human tutor responses to student Certainness are not solely intended to improve student learning. For example, they may also be aimed at increasing student motivation (Baylor et al., 2003) or persistence (Aist et al., 2002) or even tutor likeability. As in other types of spoken dialogue systems (Walker et al., 2002), these and other evaluation metrics (e.g. user satisfaction) can also be used to evaluate the effectiveness of adaptive strategies after implementation in ITSPOKE.

Our approach for developing adaptive strategies is currently based on one human tutor's responses across dialogues with multiple students. As Chi et al. (2001) note, different tutors have different teaching styles, both based on their training and based on their individual differences. Moreover, it is an open question in the tutoring community as to whether, and why, one tutor is better than any other with respect to student learning or other evaluation metrics. Analysing a different tutor's responses may yield different dependencies between student emotions and tutor responses. Analysing the responses of multiple tutors would yield a broader range of responses from which common responses could be extracted. However, the common responses of multiple tutors are not necessarily better for improving student learning than those of a human tutor who responds differently. Moreover, such a "mix and match" approach would not necessarily yield a *consistent* generalisation about adaptive strategies for student emotion. We have already demonstrated that students learned a significant amount with our human tutor (Litman et al., 2006). Thus, although it is an open question as to *why* these students learn, analysing our

tutor's responses across multiple students enables a consistent generalisation about one successful tutor's responses to student emotion.

Again, it is important to emphasise that we do not know yet if these adaptive techniques will be "effective", i.e., that they will improve student learning or improve other performance measures when implemented in our *computer* tutor. Our next step will thus be to use these human tutor responses as a *guideline* for implementing adaptive techniques in ITSPOKE. We can then compare the performance of the adaptive system with its non-adaptive counterpart, to see whether or not system effectiveness is improved. Currently ITSPOKE adaptation is based only on the correctness of student turns.

Further, in this paper we assumed "independence" of the different tutor acts, due to the fact that considering the combination of Tutor Acts in each turn yielded a data sparsity problem. For example, as noted in Section 3, we had 478 unique Tutor Act combinations, 294 of which occurred only once. However, there are counter-examples to this independence assumption. For example, we found 84 tutor turns containing the sequence "Positive Feedback, Hint", with or without additional Tutor Acts; this intuitively seems like an effective combination. Examples of other high-frequency combinations of two Tutor Acts include: 68 instances of "Expansion, Short Answer Question", 24 instances of "Hint, Deep Answer Question", and 81 instances of "Negative Feedback, Hint". In future work we will investigate dependencies and correlations for such combinations of Tutor Acts, beginning with combinations of two Acts.[11]

We will also investigate how other factors interact with student emotional states to determine subsequent Tutor Acts. For although our results demonstrate significant dependencies between emotion and our human tutor responses, only a small amount of variance is accounted for in our results, indicating that other factors play a role in determining tutor responses. One such factor is student "correctness", which is not identical to student "certainness" (as measured by "hedging" (Bhatt et al., 2004)); for example, a student may be "certain" but "incorrect". Other factors include the dialogue act that the student is performing. We have annotated student turn "correctness" as well as "Student Dialogue Acts" (in tandem with our Tutor Dialogue Acts), and we have begun investigating relationships between these annotations (Litman and Forbes-Riley, 2006a). Annotation of student "Frustration" and "Anger" categories has also recently been completed. We plan to extend our bigram analysis by looking at other n-grams combining these new student turn annotations with our tutor turn annotations.

[11] We thank an anonymous reviewer for this interesting suggestion.

Acknowledgements We thank Julia Hirschberg, Jennifer Venditti, Jackson Liscombe, and Jeansun Lee at Columbia University for certainness annotation and discussion. We thank Pam Jordan, Amruta Purandare, Ted Pederson, and Mihai Rotaru for their helpful comments. This research is supported by ONR (N00014-04-1-0108), and NSF (0325054, 0328431).

References

Aist, G., Kort, B., Reilly, R., Mostow, J., and Picard, R. (2002). Experimentally Augmenting an Intelligent Tutoring System with Human-Supplied Capabilities: Adding Human-Provided Emotional Scaffolding to an Automated Reading Tutor that Listens. In *Proceedings of ITS Workshop on Empirical Methods for Tutorial Dialogue Systems*, pages 16–28, San Sebastian.

André, E., Dybkjær, L., Minker, W., and Heisterkamp, P., editors (2004). *Affective Dialogue Systems*, volume 3068 of *Lecture Notes in Computer Science*. Springer, Kloster Irsee.

Ang, J., Dhillon, R., Krupski, A., Shriberg, E., and Stolcke, A. (2002). Prosody-Based Automatic Detection of Annoyance and Frustration in Human-Computer Dialog. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 2037–2040, Denver.

Banerjee, S. and Pedersen, T. (2003). The Design, Implementation, and Use of the Ngram Statistics Package. In *Proceedings of 4th International Conference on Intelligent Text Processing and Computational Linguistics (CICling)*, pages 370–381, Mexico City.

Baylor, A. L., Ryu, J., and Shen, E. (2003). The Effects of Pedagogical Agent Voice and Animation on Learning, Motivation, and Perceived Persona. In *Proceedings of ED-MEDIA*, pages 452–458, Honolulu.

Bhatt, K., Evens, M., and Argamon, S. (2004). Hedged Responses and Expressions of Affect in Human/Human and Human/Computer Tutorial Interactions. In *Proceedings of 26th Annual Meeting of the Cognitive Science Society (COGSCI)*, Chicago.

Chi, M. T. H., Siler, S. A., Jeong, H., Yamauchi, T., and Hausmann, R. G. (2001). Learning from Human Tutoring. *Cognitive Science*, 25:471–533.

Core, M. G. and Allen, J. F. (1997). Coding Dialogues with the DAMSL Annotation Scheme. In D. Traum, editor, *Working Notes: AAAI Fall Symposium on Communicative Action in Humans and Machines*, pages 28–35, Menlo Park.

Craig, S. D. and Graesser, A. (2003). Why am I confused: An Exploratory Look into the Role of Affect in Learning. In Mendez-Vilas, A. and Gonzalez, J. A. M., editors, *Advances in Technology-Based Education: Towards a Knowledge-Based Society*, volume 3, pages 1903–1906. Badajoz. Junta de Extremadura.

Forbes-Riley, K. and Litman, D. J. (2004). Predicting Emotion in Spoken Dialogue from Multiple Knowledge Sources. In *Proceedings of Annual Meeting of the Human Language Technology Conference/North American chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 201–208, Boston.

Forbes-Riley, K., Litman, D. J., Huettner, A., and Ward, A. (2005). Dialogue-Learning Correlations in Spoken Dialogue Tutoring. In *Proceedings of International Conference on Artificial Intelligence in Education (AIED)*, pages 225–232, Amsterdam.

Graesser, A. and Person, N. (1994). Question Asking During Tutoring. *American Educational Research Journal*, 31(1):104–137.

Graesser, A., Person, N., and Magliano, J. (1995). Collaborative Dialog Patterns in Naturalistic One-on-One Tutoring. *Applied Cognitive Psychology*, 9:495–522.

Heylen, D., Vissers, M., op den Akker, R., and Nijholt, A. (2004). Affective Feedback in a Tutoring System for Procedural Tasks. In (André et al., 2004), pages 244–253.

Higashinaka, R., Nakano, M., and Aikawa, K. (2003). Corpus-Based Discourse Understanding in Spoken Dialogue Systems. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 240–247, Sapporo.

Johnson, W. L., Rizzo, P., Bosma, W., Kole, S., Ghijsen, M., and van Welbergen, H. (2004). Generating Socially Appropriate Tutorial Dialog. In (André et al., 2004), pages 254–264.

Landis, J. R. and Koch, G. G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33:159–174.

Liscombe, J., Hirschberg, J., and Venditti, J. (2005). Detecting Certainness in Spoken Tutorial Dialogues. In *Proceedings of 9th European Conference on Speech Communication and Technology (INTERSPEECH)*, pages 1837–1840, Lisbon.

Litman, D. J. and Forbes-Riley, K. (2004a). Annotating Student Emotional States in Spoken Tutoring Dialogues. In *Proceedings of SIGdial Workshop on Discourse and Dialogue*, pages 144–153, Boston.

Litman, D. J. and Forbes-Riley, K. (2004b). Predicting Student Emotions in Computer-Human Tutoring Dialogues. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 351–358, Barcelona.

Litman, D. J. and Forbes-Riley, K. (2006a). Correlations between Dialogue Acts and Learning in Spoken Tutoring Dialogues. *Journal of Natural Language Engineering: Special Issue on Educational Applications*, 12(2):161–176.

Litman, D. J. and Forbes-Riley, K. (2006b). Recognizing Student Emotions and Attitudes on the Basis of Utterances in Spoken Tutoring Dialogues with both Human and Computer Tutors. *Speech Communication*, 48(5):559–590.

Litman, D. J., Rose, C. P., Forbes-Riley, K., VanLehn, K., Bhembe, D., and Silliman, S. (2004). Spoken versus Typed Human and Computer Dialogue Tutoring. In *Proceedings of Conference on Intelligent Tutoring Systems (ITS)*, pages 368–379, Maceió, Alagoas, Brazil. *Lecture Notes in Computer Science*, Springer, Kloster Irsee.

Litman, D. J., Rose, C. P., Forbes-Riley, K., VanLehn, K., Bhembe, D., and Silliman, S. (2006). Spoken versus Typed Human and Computer Dialogue Tutoring. *International Journal of Artificial Intelligence in Education*, 16:145–170.

Litman, D. J. and Silliman, S. (2004). ITSPOKE: An Intelligent Tutoring Spoken Dialogue System. In *Companion Proceedings of the Human Language Technology Conference/North American chapter of the Association for Computational Linguistics annual meeting (HLT-NAACL)*, pages 233–236, Boston.

Moore, J. D., Porayska-Pomsta, K., Varges, S., and Zinn, C. (2004). Generating Tutorial Feedback with Affect. In *Proceedings of 17th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, AAAI Press, Miami Beach.

Narayanan, S. (2002). Towards Modeling User Behavior in Human-Machine Interaction: Effect of Errors and Emotions. In *Proceedings of ISLE Workshop on Dialogue Tagging for Multimodal Human Computer Interaction*, Edinburgh.

Pilkington, R. M. (1999). Analysing Educational Discourse: The DISCOUNT Scheme. Computer-Based Learning Unit 99/2, University of Leeds.

Pon-Barry, H., Clark, B., Bratt, E. O., Schultz, K., and Peters, S. (2004). Evaluating the Effectiveness of SCoT: a Spoken Conversational Tutor. In *Proceedings of ITS Workshop on Dialogue-Based Intelligent Tutoring Systems: State of the Art and New Research Directions*, pages 23–32, Maceio.

Reithinger, N., Engel, R., Kipp, M., and Klesen, M. (1996). Predicting Dialogue Acts for a Speech-To-Speech Translation System. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 654–657, Philadelphia.

Rickel, J., Lesh, N. B., Rich, C., Sidner, C. L., and Gertner, A. (2001). Building a Bridge between Intelligent Tutoring and Collaborative Dialogue Systems. In *Proceedings of International Conference on Artificial Intelligence in Education (AIED)*, pages 592–594, San Antonio.

Shafran, I., Riley, M., and Mohri, M. (2003). Voice Signatures. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 31–36, Virgin Islands.

Shah, F., Evens, M., Michael, J., and Rovick, A. (2002). Classifying Student Initiatives and Tutor Responses in Human Keyboard-to-Keyboard Tutoring Sessions. *Discourse Processes*, 33(1):23–52.

Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Martin, R., Meteer, M., and Van Ess-Dykema, C. (2000). Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. *Computational Linguistics*, 26(3):339–373.

VanLehn, K., Jordan, P. W., Rosé, C. P., Bhembe, D., Böttner, M., Gaydos, A., Makatchev, M., Pappuswamy, U., Ringenberg, M., Roque, A., Siler, S., Srivastava, R., and Wilson, R. (2002). The Architecture of Why2-Atlas: A Coach for Qualitative Physics Essay Writing. In *Proceedings of Conference on Intelligent Tutoring Systems (ITS)*, pages 158–167, Biarritz.

Walker, M., Rudnicky, A., Prasad, R., Aberdeen, J., Bratt, E., Garofolo, J., Hastie, H., Le, A., Pellom, B., Potamianos, A., Passonneau, R., Roukos, S., Sanders, G., Seneff, S., and Stallard, D. (2002). DARPA Communicator: Cross-System Results for the 2001 Evaluation. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, volume 1, pages 269–272, Denver.

Wolska, M., Vo, B. Q., Tsovaltzi, D., Kruiff-Korbayová, I., Karagjosova, E., Horacek, H., Fiedler, A., and Benzmuller, C. (2004). An Annotated Corpus of Tutorial Dialogs on Mathematical Theorem Proving. In *Proceedings of International Conference on Language Resources and Evaluation (LREC)*, pages 1007–1010, Lisbon.

Zhang, T., Hasegawa-Johnson, M., and Levinson, S. E. (2004). Children's Emotion Recognition in an Intelligent Tutoring Scenario. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 1441–1444, Jeju Island.

Appendix: More Examples of Tutor Turns Containing Positive and Negative Feedback

Each example below comes from a different human tutoring dialogue. These examples further illustrate how Tutor Positive and Negative Feedback Acts label feedback based on the presence of *lexical items* in the tutor turn. As the examples below show, these labels can be used to refer to tutor feedback concerning the correctness of the student's prior turn, but they can also be used to refer to instances of tutor encouragement, and they can also be used to refer to tutor feedback regarding the student's prior essay, or regarding other aspects of the tutoring session.

1 **STUDENT**$_{64}$: The force of gravity is in proportion to the mass.

TUTOR$_{65}$: Yes, uh, but that alone is not enough. *[POS, NEG]*

2 **STUDENT**$_{1}$: Ok. (*student has just submitted first essay*)

TUTOR$_{2}$: Hm, yeah, you are thinking in the right direction but uh, I think needs a little more few more clarifications. *[POS, NEG]*

3 **STUDENT**$_{65}$: I said the head would apply force on the neck. *(referring to prior essay revision)*

TUTOR$_{66}$: Uh, no, uh, right in the beginning, for example, what you say in the beginning of the essay is all correct, but what you are saying is- you are telling why uh head is left behind now. *[NEG, POS, HN]*

4 **STUDENT**$_{8}$: Oh, it says does the earth pull equally. *(referring to the problem statement)*

TUTOR$_{9}$: No not that that that- you have- what you have said is correct- your answer is correct. There is no-nothing wrong with it, the only thing is that one could have been bothered a little uh by the fact that the- when force is exerted on a body the body must accelerate. That is the Newton's second law of motion. *[NEG, POS, EX]*

5 **STUDENT**$_{98}$: Should I try and click it again? *(referring to the essay submit button on the interface)*

TUTOR$_{99}$: No no, that's not the problem. Perhaps there is something in the interface. So, after you have finished writing just indicate by words so I'll know. Um, now uh, you have stated everything here... Um, one suggestion is that uh you uh separate the reason and... You see, for each one of these three factors which determine the time taken to fall, there is a sep- different reason. For example, acceleration is same, uh, from where did you conclude it? *[NEG, POS, RD, DAQ]*

Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
ASR	Automatic Speech Recognition
ATM	Automated Teller Machine
AVM	Attribute-Value Matrix
AVP	Attribute-Value Pair
CA	Concept Accuracy
CA	Contextual Appropriateness
CA	Communicative Act
CA	Conversational Analysis
CAN	Controller Area Network
CCML	Call Control Markup Language
CDDB	Compact Disc Database
CER	Concept Error Rate
CHIL	Computers in the Human Interaction Loop
CL	Association for Computational Linguistics
CSR	Customer Service Representative
DTD	Document Type Definition
D2B	Domestic Digital Data Bus
DAAD	Deutscher Akademischer Austauschdienst
DAMSL	Dialog Act Markup in Several Layers
DARPA	Defence Research Project Agency
EC	European Community
ELSNET	European Network of Human Language Technologies
Embassi	Elektronische Multimediale Bedien- und Service Assistenz
EMMA	Extensible Multi Modal Annotation
EU	European Union
FDR	False Discovery Rate
FIA	Form Interpretation Algorithm
GDML	Generic Dialogue Modelling Language
GUI	Graphical User Interface

HDAG	Hierarchical Directed Acyclic Graph
HMI	Human-Machine-Interaction
HTML	Hypertext Markup Language
ID3	Identify an MP3
IEC	International Electrotechnical Commission
IEE	Institution of Electrical Engineers
IEEE	Institute of Electrical and Electronics Engineers
INSPIRE	INfotainment management with SPeech Interaction via REmote microphones and telephone interfaces
IP	Internet Protocol
ISCA	International Speech Communication Association
IST	Information Society Technologies
ITU	International Telecommunication Union
ITU-T	International Telecommunication Union (Standardization Sector)
IVR	Interactive Voice Response
J2EE	Java Secure Socket Extension
JSP	Java Server Pages
MDP	Markov Decision Process
MOST	Media Oriented Systems Transport
MP3	MPEG-1 Audio Layer 3
MRCP	Media Resource Control Protocol
NL	Natural Language
NLG	Natural Language Generation
NLU	Natural Language Understanding
NSF	National Science Foundation
NSP	Ngram Statistics Package
ONR	US Navy Office of Naval Research
OPI	Ontology Programming Interface
OWL/RDF	Web Ontology Language / Resource Description Framework
PARADISE	PARAdigm for DIalogue System Evaluation
PIN	Personal Identification Number
POMDP	Partially Observable Markov Decision Process
PTT	Push-To-Talk
SA	Sentence Accuracy
SCM	Support Vector Machine
SDS	Spoken Dialogue System
SER	Sentence Error Rate
SIGDIAL	Special Interest Group on Discourse and Dialogue
SRGS	Speech Recognition Grammar Specification
SSML	Speech Synthesis Markup Language

SVM	Support Vector Machine
TCL	Tool Command Language
TTS	Text-To-Speech
UA	Understanding Accuracy
USB	Universal Serial Bus
VoiceXML	Voice Extended Markup Language
VUI	Voice User Interface
WA	Word Accuracy
WER	Word Error Rate
WOZ	Wizard-of-Oz
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

Index