KTH
VETENSKAP
OCH KONST

**KTH Computer Science
and Communication**

Error Handling in Spoken Dialogue Systems

Managing Uncertainty, Grounding and Miscommunication

GABRIEL SKANTZE

Doctoral Thesis
Stockholm, Sweden 2007

KTH Computer Science and Communication
Department of Speech, Music and Hearing
100 44 Stockholm, Sweden

GSLT Graduate School of Language Technology
Faculty of Arts, Göteborg University
405 30 Göteborg, Sweden

TRITA-CSC-A 2007:14
ISSN 1653-5723
ISRN KTH/CSC/A–07/14–SE
ISBN 978-91-7178-788-0

Akademisk avhandling som med tillstånd av Kungliga Tekniska Högskolan framlägges till offentlig granskning för avläggande av filosofie doktorsexamen fredagen den 23 november klockan 10.00 i sal F3, Kungliga Tekniska Högskolan, Lindstedtsvägen 26, Stockholm.

Printed by Universitetsservice US AB

Abstract

Due to the large variability in the speech signal, the speech recognition process constitutes the major source of errors in most spoken dialogue systems. A spoken dialogue system can never know for certain what the user is saying, it can only make hypotheses. As a result of this uncertainty, two types of errors can be made: over-generation of hypotheses, which leads to misunderstanding, and under-generation, which leads to non-understanding. In human-human dialogue, speakers try to minimise such miscommunication by constantly sending and picking up signals about their understanding, a process commonly referred to as grounding.

The topic of this thesis is how to deal with this uncertainty in spoken dialogue systems: how to detect errors in speech recognition results, how to recover from non-understanding, how to choose when to engage in grounding, how to model the grounding process, how to realise grounding utterances and how to detect and repair misunderstandings. The approach in this thesis is to explore and draw lessons from human error handling, and to study how error handling may be performed in different parts of a complete spoken dialogue system. These studies are divided into three parts.

In the first part, an experimental setup is presented in which a speech recogniser is used to induce errors in human-human dialogues. The results show that, unlike the behaviour of most dialogue systems, humans tend to employ other strategies than encouraging the interlocutor to repeat when faced with non-understandings. The collected data is also used in a follow-up experiment to explore which factors humans may benefit from when detecting errors in speech recognition results. Two machine learning algorithms are also used for the task.

In the second part, the spoken dialogue system HIGGINS is presented, including the robust semantic interpreter PICKERING and the error aware discourse modeller GALATEA. It is shown how grounding is modelled and error handling is performed on the concept level. The system may choose to display its understanding of individual concepts, pose fragmentary clarification requests, or risk a misunderstanding and possibly detect and repair it at a later stage. An evaluation of the system with naive users indicates that the system performs well under error conditions.

In the third part, models for choosing when to engage in grounding and how to realise grounding utterances are presented. A decision-theoretic, data-driven model for making grounding decisions is applied to the data from the evaluation of the HIGGINS system. Finally, two experiments are presented, which explore how the intonation of synthesised fragmentary grounding utterances affect their pragmatic meaning.

The primary target of this thesis is the management of uncertainty, grounding and miscommunication in conversational dialogue systems, which to a larger extent build upon the principles of human conversation. However, many of the methods, models and results presented should also be applicable to dialogue systems in general.

Acknowledgements

First of all, I would like to thank my supervisor Rolf Carlson, who has guided me into the field of speech communication, encouraged me to pursue my research interests, discussed research ideas, and provided valuable comments and suggestions all along the way to finishing this thesis.

Second, I want to thank my co-worker and roommate Jens Edlund. Many of the ideas that have led to the research presented in this thesis originate from our discussions, and parts of the work have been done in collaboration with him. Thanks also for reading and commenting on parts of the thesis.

It has also been very rewarding to collaborate with David House on the experiments presented in Chapter 9. Furthermore, David has read the thesis closely and suggested many improvements at the final stages of writing the thesis.

I also wish to thank Johan Boye for reading the thesis, as well as papers I have written, and for giving me encouraging comments as well as critical questions that have forced me to develop my thoughts and arguments.

I have really enjoyed working at the Department of Speech, Music and Hearing, and I wish to thank all the people working there. Special thanks to Björn Granström and Rolf Carlson for heading the speech group and for all hard work to find funding for our research, Alexander Seward for explaining and discussing speech recognition issues, Jonas Beskow for help on speech synthesis issues, and Preben Wik, Anna Hjalmarsson, Mattias Heldner and Joakim Gustafson for stimulating discussions.

I would not be doing research on dialogue systems if it were not for Arne Jönsson. By working as a programmer in his research group in the summers during my MSc studies, I was introduced to the field and got valuable hands-on experience on developing dialogue systems.

This thesis is to a large extent based on empirical material, and I am very thankful towards all patient subjects participating in the many (sometimes quite boring) experiments.

I must also thank my anonymous reviewers around the world who have read and commented on papers and articles I have written.

Finally, I want to thank my supportive family: my wife Linda, my daughters Agnes and Miranda, my parents Patrik and Margareta, and my brothers Alexander and Valdemar.

This study was carried out at KTH with the support of the Swedish Graduate School of Language Technology (GSLT) and the Centre for Speech Technology (CTT). The Knut and Alice Wallenberg Foundation, as well as the Ragnar and Astrid Signeul Foundation have also contributed with some of the travelling and conference expenses.

Publications and contributors

A large part of this thesis is based on work already published in conference and workshop proceedings, journal articles and book chapters. This represents work partly done in collaboration with others, as indicated in the following list.

Chapter 4

The experiment presented in this chapter was conducted by the author. The chapter is based on the following publications:

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* (pp. 71-76). Chateau-d'Oex-Vaud, Switzerland.

Skantze, G. (2005). Exploring human error recovery strategies: implications for spoken dialogue systems. *Speech Communication, 45*(3), 325-341.

Chapter 5

The experiment on human error detection was conducted by the author together with Jens Edlund. The machine learning study was done by the author. The chapter is based on the following publication:

Skantze, G., & Edlund, J. (2004). Early error detection on word level. In *Proceedings of ISCA Tutorial and Research Workshop (ITRW) on Robustness Issues in Conversational Interaction.* Norwich, UK.

Chapter 6-7

The HIGGINS system and its components were conceived and specified by the author together with Jens Edlund. The major part of the implementation of the system was done by the author. The evaluation of the interpreter PICKERING was done by the author together with Jens Edlund. The evaluation of GALATEA and the HIGGINS system was done by the author. The chapters are based on the following publications:

Edlund, J., Skantze, G., & Carlson, R. (2004). Higgins - a spoken dialogue system for investigating error handling techniques. In *Proceedings of the International Conference on Spoken Language Processing, ICSLP 04* (pp. 229-231). Jeju, Korea.

Skantze, G. (2005). Galatea: a discourse modeller supporting concept-level error handling in spoken dialogue systems. In *Proceedings of SigDial* (pp. 178-189). Lisbon, Portugal.

Skantze, G. (in press). Galatea: A discourse modeller supporting concept-level error handling in spoken dialogue systems. To be published in Dybkjær, L., & Minker, W. (Eds.), *Recent Trends in Discourse and Dialogue.* Springer.

Skantze, G., & Edlund, J. (2004). Robust interpretation in the Higgins spoken dialogue system. In *ISCA Tutorial and Research Workshop (ITRW) on Robustness Issues in Conversational Interaction.* Norwich, UK.

Chapter 8

The proposed method was conceived and applied to the collected data by the author. The chapter is based on the following publication:

Skantze, G. (2007). Making grounding decisions: Data-driven estimation of dialogue costs and confidence thresholds. In *Proceedings of SigDial* (pp. 206-210). Antwerp, Belgium.

Chapter 9

The experiments were performed together with Jens Edlund and David House. The chapter is based on the following publications:

Edlund, J., House, D., & Skantze, G. (2005). The effects of prosodic features on the interpretation of clarification ellipses. In *Proceedings of Interspeech 2005* (pp. 2389-2392). Lisbon, Portugal.

Skantze, G., House, D., & Edlund, J. (2006). User responses to prosodic variation in fragmentary grounding utterances in dialogue. In *Proceedings of Interspeech 2006 - ICSLP* (pp. 2002-2005). Pittsburgh PA, USA.

Contents

PART I

Introduction and Background

DAVE: How would you account for this discrepancy between you and the twin 9000?
HAL: Well, I don't think there is any question about it. It can only be attributable to human error. This sort of thing has cropped up before, and it has always been due to human error.
FRANK: Listen HAL. There has never been any instance at all of a computer error occurring in the 9000 series, has there?
HAL: None whatsoever, Frank. The 9000 series has a perfect operational record.

2001 – A Space Odyssey
screenplay by *Stanley Kubrick* and *Arthur C. Clark*

To err is human, but it takes a computer to really louse it up.

Anonymous

CHAPTER 1

Introduction

Most of today's computer applications have a graphical user interface (GUI), which allows the user to perform actions through direct manipulation of graphical elements, such as icons, text and tables, presented to the user on a display device. As an alternative (or complement) to this kind of interaction, a *spoken dialogue system* offers the possibility to interact by means of spoken language. Possible applications which may benefit from spoken language interfaces include flight booking over the telephone, human-robot interaction, speech controlled music players and conversational computer games. Compared to a GUI, a spoken language interface frees the user's hands and eyes for other tasks. Moreover, human-human conversation is generally a natural, intuitive, robust and efficient means for interaction. A lot of effort is being invested in trying to make spoken dialogue systems also benefit from these properties.

To be able to engage in conversation, a spoken dialogue system has to attend to, recognise and understand what the user is saying, interpret utterances in context, decide what to say next, as well as when and how to say it. To achieve this, a wide range of research areas and technologies must be involved, such as automatic speech recognition, natural language understanding, dialogue management, natural language generation and speech synthesis.

One of the greatest challenges when building dialogue systems is to deal with *uncertainty* and *errors*. Uncertainty comes partly from the ambiguity of natural language itself. In addition, in the case of spoken dialogue systems, a great deal of uncertainty comes from the error-prone speech recognition process. Speech recognition errors arise from speaker variability, background noise and unexpected language use, which are all hard (if not impossible) to model exhaustively. However, as Brown (1995) points out, apparently satisfactory communication may often take place between humans without the listener arriving at a full interpretation of the words used. One explanation for this is the redundancy and context-dependence of language use; the same information may be conveyed in different ways in the same utterance, or may be repeated by the speakers in order to ensure understanding, and the context may be used to reduce uncertainty or fill in the gaps. Furthermore, when humans speak to each other, there is a collaborative process of avoiding and recovering from miscommunication that often goes unnoticed (Clark, 1996).

The topic of this thesis is how to draw lessons from this seemingly smooth handling of uncertainty and miscommunication in human-human dialogue, and how to use this knowledge to improve error handling in spoken dialogue systems.

1.1 Error handling issues

Due to the error prone speech recognition process, a dialogue system can never know for certain what the user is saying, it can only make *hypotheses*. Errors in spoken dialogue systems may be classified into two broad categories: under-generation and over-generation of interpretation hypotheses. In terms of miscommunication, these categories correspond to the notions of *non-understanding* and *misunderstanding*. *Misunderstanding* means that the listener obtains an interpretation that is not in line with the speaker's intentions. If the listener fails to obtain any interpretation at all, or is not confident enough to choose a specific interpretation, a *non-understanding* has occurred. One important difference between non-understandings and misunderstandings is that non-understandings are noticed immediately by the listener, while misunderstandings may not be identified until a later stage in the dialogue. Some misunderstandings might never be detected at all. The same utterance may, of course, give rise to both misunderstanding and non-understanding, that is, parts of an utterance may be misunderstood while others are not understood.

As is argued in this thesis, error handling is not a separate processing step in a spoken dialogue system, like a speech recogniser or a dialogue manager. Instead, error handling should be regarded as a set of issues that should be considered in all parts of the system to handle the consequences of under-generation and over-generation. As an example, take a dialogue system which is supposed to understand the user's description of her location in a city. This might be the output of the speech recogniser[1]:

(1) User: **I CAN SEE A BLUE BUILDING**

The system must now consider the possibility that this is not what the user actually said – the speech recogniser might have over-generated or under-generated words. One important error handling issue is then to detect such errors in the speech recognition result, so that misunderstanding may be avoided. We will call this *early error detection*. If all words are deemed to be incorrect, or if the speech recogniser does not deliver any hypotheses at all, we may say that a non-understanding has occurred.

[1] The greyscale represent the speech recognition word confidence scores. Dark colour represents high confidence and light colour represents low confidence.

Another important issue is then how to *recover from non-understanding*. This means that the system must gain understanding of future contributions as efficiently as possible, after a non-understanding has occurred.

But, as Clark (1996) points out, the handling of uncertainty and problems in dialogue is not just an individual task. Speakers deal with uncertainty and problems together, by providing and evaluating positive and negative *evidence of understanding* – a process commonly referred to as *grounding*. By this process, the speakers share their private knowledge and beliefs with each other and make sure that what was private can now be regarded as *common ground*. The following example[2] illustrates some alternative responses from the system that all provide different kinds of evidence:

(2) U.1: **I CAN SEE A BLUE BUILDING**
S.2 (alt. a): *Ok, can you see a tree?*
S.2 (alt. b): *Blue?*
S.2 (alt. c): *How many stories does the blue building have?*

As can be seen in the example, the system may provide evidence of its understanding in different ways and for different parts of the original utterance. It may for example assume understanding and provide an acknowledgement (a), it may suspect that some part of the utterance is incorrect and pose a clarification request (b), or it may assume understanding but provide a verbatim repetition as part of the next contribution (c). A key problem for spoken dialogue systems is to choose *which* evidence to provide and *how* it should be realised. It would not be efficient to provide evidence on everything that is said; neither would it be efficient to not provide any evidence at all, since this would lead to a lot of misunderstandings that would need to be repaired later on.

If the system assumes understanding of an incorrect hypothesis, this misunderstanding may still be repaired. For example, the positive evidence that the system provides (such as alt. c above) might let the user detect such misunderstandings and raise an objection:

(3) U.1: **I CAN SEE A BLUE BUILDING**
S.2: *How many stories does the blue building have?*
U.3: **I SAID A BROWN BUILDING**

However, the system must be able to understand such objections and repair the error that was made. This will be referred to as *late error detection* (i.e., detection of misunderstandings) and *misunderstanding repair*. Another opportunity for late error detection is if the system detects inconsistencies in its model of what has been said and what it knows about the world.

[2] In all examples, U refers to the user and S to the system (A and B are used for human interlocutors). The number after the dot indicates the order of the turn. Italics are used to separate the speakers and enhance readability. In the case of human-computer dialogue examples, user utterances written in greyscale capital letters represent speech recognition results, not what the user actually said.

1.2 Thesis contributions

The work presented in this thesis consists of experiments on several of the issues outlined above and the development of models and methods for handling them. Of course, all aspects cannot be covered in the scope of this thesis. However, the ambition has been to not focus on only one particular aspect of error handling, but to study how error handling may be performed in different parts of a complete spoken dialogue system.

There are two general themes in this thesis. The first theme concerns how to explore and draw lessons from *human-like error handling strategies* and how to apply these to human-computer dialogue. The second theme, which is related to the first, is to explore *concept-level error handling.* Most approaches to error handling in spoken dialogue systems have focused on whole utterances as the smallest unit. Typically, whole utterances are assigned confidence scores and decisions are made whether to reject, verify or accept them as correct. Such utterance-level error handling is often feasible in command-based dialogue systems where utterances are relatively short and predictable. However, in conversational dialogue systems, utterance-level error handling is too simplistic. Humans engaging in conversation often focus on parts of utterances to, for example, pose fragmentary clarification requests (as exemplified in alt. b in example (2) above), and thereby increase the efficiency of the dialogue. In dialogue systems that are targeted towards more human-like conversation, speech recognition results and the semantic interpretations of them may often be partly correct. This calls for error handling on a "sub-utterance" level – to base error handling on individual words or concepts in utterances.

This thesis relies to a large extent on an *empirical approach.* In order to understand how humans manage problems in dialogue, such data must be collected and analysed. To understand what kind of problems arise in spoken dialogue systems, we also need data containing real speech recognition errors. The models and guidelines derived from the data have been used to develop a dialogue system that is in turn evaluated with naive users.

The main contributions of this thesis can be summarised as follows:

- An experimental setup that allows us to study the way humans deal with speech recognition errors.
    - Results indicating that humans to a large extent employ other strategies than encouraging the interlocutor to repeat when faced with non-understandings.
- Results confirming previous findings that errors in speech recognition results may be detected by considering confidence scores as well as other knowledge sources that were not accessible to the speech recogniser. While previous studies have shown this to be true for utterance-level error detection, these results show that it is also true for word-level error detection.
- A practical model for how the grounding status of concepts gets updated during the discourse in a spoken dialogue system, how this updating is affected by the use of

anaphora and ellipses, and how this information may be used for various error handling strategies.
    - An implementation of the model in a complete spoken dialogue system in a non-trivial domain.
    - An evaluation of the system with naive users, which indicates that the system and model performs well under error conditions.
- A decision-theoretic, data-driven model, based on task-analysis and bootstrapping, for making grounding decisions in spoken dialogue systems.
- A tentative model for the intonation of synthesised fragmentary grounding utterances in Swedish and their associated pragmatic meaning.

1.3 Thesis overview

In the rest of Part I, the brief initial overview of error handling given in 1.1 above is developed into a detailed discussion of the issues, and research on how they may be handled is reviewed. This will serve as a background for presenting the contributions of this thesis in Part II, III and IV. The thesis ends with a concluding summary and discussion in Part V.

Chapter 2: Spoken dialogue systems

This chapter discusses some basic properties of spoken dialogue and the techniques and issues involved in developing spoken dialogue systems. It is argued that two general types of dialogue systems may be distinguished, command-based and conversational, and that this thesis is targeted towards the latter, that is, dialogue systems that to a larger extent build upon the principles of human conversation. The basic building blocks of spoken dialogue systems are discussed: automatic speech recognition, natural language understanding, dialogue management, natural language generation and text-to-speech synthesis.

Chapter 3: Miscommunication and error handling

This chapter starts with a general discussion on the concepts of miscommunication, grounding, repair, clarification and error in the context of human-human and human-computer dialogue. This is followed by reviews and discussions on previous research related to error handling in spoken dialogue systems, including early error detection, grounding and late error detection. Places where the contributions of this thesis fit in and extend this body of knowledge will be pointed out.

Part II: Exploring Human Error Handling

Chapter 4: Exploring non-understanding recovery

In this chapter, a method for collecting data on human error handling strategies is presented. An experiment was conducted based on this method, in which pairs of subjects were given a joint task for which they needed to engage in dialogue. A speech recogniser was used to introduce errors in one direction – thereby simulating the roles of human and computer in a dialogue system. The analysis is focussed on how the subjects recovered from non-understanding.

Chapter 5: Early error detection on word level

One interesting result from the experiment presented in Chapter 4 was that humans were extremely good at *early error detection*, that is, to understand which recognition hypotheses were incorrect in order to avoid misunderstandings. Most studies on automatic early error detection have focused on detecting whether a given hypothesis of a user utterance contains any errors at all or is error-free. For concept level error handling, it would be more useful to detect which words or concepts in the hypothesis that are erroneous, and it is obvious that this is what the human subjects did. This chapter explores which factors humans rely on when detecting such errors. A machine learning experiment is also presented where the data from Chapter 4 is used for automatic detection.

Part III: The Higgins Spoken Dialogue System

Chapter 6: Concept-level error handling in Higgins

As part of the work presented in this thesis, a complete spoken dialogue system, called HIGGINS, has been developed to serve as a test-bed for implementing and evaluating error handling methods and models. For this system, a set of modules have been developed, most notably a robust interpreter called PICKERING and a discourse modeller called GALATEA, which models the grounding status of concepts. This chapter describes how concept level error handling is done in the different parts of the system. In most previous accounts, a special set of grounding actions are used to provide evidence of understanding for complete user utterances. In this chapter, it is shown how all utterances instead may contribute to the grounding process on the concept level.

Chapter 7: Higgins evaluation

This chapter presents two separate evaluations. First the performance of the robust interpreter PICKERING is studied under different error conditions. Second, a data collection is presented in which naive users interact with the HIGGINS system. The data is used to evaluate the system in general and the discourse modeller GALATEA in particular, as well as the users' reactions to fragmentary clarification.

Part IV: Deciding and Realising Grounding Actions

Chapter 8: Making grounding decisions

The different grounding strategies supported by HIGGINS leave the system with decisions that have to be made: what kind of evidence should it provide and which recognition hypotheses should it accept as correct? These grounding decisions should ideally be based on the system's uncertainty in its hypothesis, the cost of misunderstanding, and the cost of making a grounding action. In this chapter, the framework of decision making under uncertainty is applied to the problem, where the utility and costs of different actions are weighted against the probabilities of different outcomes. The data collected in Chapter 7 is used to derive a data-driven, dynamic model for making these decisions.

Chapter 9: Prosody in fragmentary grounding

Since fragmentary grounding utterances (such as alt. b in example (2) above) lack syntactic and lexical information, their interpretation depends to a large extent on their prosodic realisation. This chapter presents two experiments which show how the prosody of synthesised grounding utterances affects their interpretation, as well as users' behaviour in a human-computer dialogue setting.

Part V: Conclusion

Chapter 10: Summary and discussion

In the final chapter, the contributions made in this thesis are summarised, followed by a discussion on how the methods and models presented may be extended and integrated further. Finally, the generalisation of the results to other dialogue systems and domains is discussed.

CHAPTER 2

Spoken dialogue systems

In this chapter, we will start with a broad classification of spoken dialogue systems in order to narrow down the class of systems that are targeted in this thesis. This is followed by a brief overview of the structure and properties of human-human conversation from a linguistic perspective, in order to derive a set of terms and concepts that are useful when discussing spoken dialogue systems. The final, and major, part of this chapter consists of a description of the different technologies and research areas that are involved in a spoken dialogue system.

2.1 Classifications of spoken dialogue systems

Spoken dialogue systems is a label that denotes a wide range of systems, from simple weather information systems ("say the name of your city") to complex problem-solving, reasoning, applications. The division between "simple" command-based systems and "complex" systems targeted towards spontaneous language can be roughly associated to systems developed within the industry and academia, respectively (Pieraccini & Huerta, 2005). However, as Pieraccini & Huerta (2005) point out, this distinction is somewhat misleading, since commercial systems have to meet usability requirements to a much larger extent and deal with "real" users, taking the technological limitations into account. This may indeed be "complex", but in another sense. Academic researchers on dialogue systems, on the other hand, often have the goal of exploring how systems may allow more spontaneous language use. They often do not have the resources to make large-scale usability studies, nor do they typically have access to "real" users.

This has lead to a distinction between two types of dialogue systems; we may call them *conversational* systems and *command-based* systems. It should be stressed that these are prototypical categories, and all dialogue system do not fit neatly into one of them. One way of viewing this distinction is to regard it as two metaphors that may be exploited – the "human metaphor" and the "interface metaphor" (Edlund et al., 2006). In the first case, the user regards the system as a conversational partner. In the second case, the user regards it as a "voice interface" with some options and slots that may be activated and filled by speech commands. One metaphor isn't necessarily better than the other – they may just meet different needs – but the metaphor should be consistent so that the user may act accordingly.

It should be noted that not all academic research is targeted towards conversational systems. Command-based systems and conversational systems have different challenges. Regarding usability, command-based systems have the problem of making the user understand what can be said. One approach to this is the "universal speech interface" or "speech-graffiti" – that is, to define and agree on a universal set of speech commands that can be used in the same way in all dialogue systems (Harris & Rosenfeld, 2004). Since the goal of conversational systems is to allow spontaneous speech, the challenge is instead how to model everything that people may say. This leads to the challenge of how to model the language used. In command-based systems, the language models are often more rigid, assuming that users understand this and that the interface metaphor will constrain the use of disfluencies, etc. Here, the challenge may instead be to handle a very large vocabulary (such as all the streets in London). In conversational systems, the vocabulary is often made smaller, but the grammar less strict, in order to model the less predictable language use. Also, longer utterances with more complex semantics may be expected, mixed with shorter, context-dependent fragmentary utterances.

In targeting conversational dialogue, the goal is not to handle all aspects of human language use. As Allen et al. (2001a) points out, full natural-language understanding by machine is an enormously complex (if not impossible) problem that will not be solved in the foreseeable future. A fundamental constraint for most dialogue systems, both command-based and conversational, has therefore been that they should operate within a given *domain*, or handle a specific task. The domain will constrain the user's expectations and behaviour into something that could be modelled by the computer.

An argument for moving towards conversational dialogue, as opposed to a command-based, is that human-like conversation generally is considered to be a natural, intuitive, robust and efficient means for interaction. Thus, the advantage of command-based speech interfaces over traditional graphical user interfaces is often restricted to the fact that users may use the hands and eyes for other tasks, and their usefulness may thus be limited to special contexts of use, such as when driving a car. Conversational dialogue systems hold the promise of offering a more intuitive and efficient interaction. Whether this promise will be met remains to be seen.

Table 2.1 summarises the distinction between command-based and conversational dialogue systems. Again, these are prototypical categories, and a given dialogue system does not have to exhibit all properties from one column.

The focus of this thesis is on error handling in conversational dialogue systems. However, many of the results should be applicable to dialogue systems in general.

Table 2.1: Two prototypical classes of dialogue systems and their associated properties.

	Command-based	**Conversational**
Metaphor	Voice interface metaphor.	Human metaphor.
Language	Constrained command-language.	Unconstrained spontaneous language.
Utterance length	Short utterances.	Mixed.
Semantics	Simple semantics. Less context dependence.	Complex semantics. More context dependence.
Syntax	More predictable.	Less predictable.
Language models	Strict grammar, possibly large vocabulary.	Less strict grammar, possibly smaller vocabulary.
Language coverage challenge	How to get the user to understand what could be said.	How to model everything that people say in the domain.

2.2 The structure and properties of conversation

Before discussing the components and implementation of dialogue systems, we will briefly describe some fundamental properties of human-human conversation from a linguistic perspective. This overview will help to understand the phenomena that need to be considered in a conversational dialogue system.

Spoken dialogue is the most basic and primary form of language use – it is the setting in which we first learn to use language as children. But there are, of course, other forms of language use – such as written text and monologues – and spoken dialogue has some unique characteristics. Speech differs from (most forms of) written language in that it is produced in a linear fashion and cannot be post-edited. Moreover, there are no punctuation marks or paragraphs, but instead a prosodic layer which may carry non-lexical information. Dialogue differs from monologue in that it is a joint activity in which people take turns in the roles of speaker and listener, and have to coordinate their language use to achieve mutual understanding (Clark, 1996).

2.2.1 Communicative acts

A useful unit for analysis of written text is the *sentence*. Sentences are delimited by punctuation marks, where each sentence commonly express one or more *propositions*. For spoken dialogue, on the other hand, such units are much less adequate for analysis. Spoken dialogue contains no punctuation marks and constitutes largely of non-sentential, fragmentary linguistic constructions. Furthermore, a linguistic construct may span over several turns, with other constructs in between, as in the following example:

(4) A.1: I have a red building …
B.2: *mhm, a red building*
A.3: … on my left.

A unit that is commonly used for segmenting spoken dialogue is instead the *utterance*. In dialogue, speakers exchange utterances, with the intent of affecting the other speaker in some way. The segmentation of speech into utterances is not trivial. The term "utterance" is sometimes used to denote "complete" constructs such as A.1 and A.3 together (i.e., an utterance may span several turns), and sometimes used to denote an uninterrupted sequence of speech from one speaker (i.e., A.1 and A.3 are separate utterances). In this thesis, we use the terms *utterance* or *turn* to refer to an uninterrupted sequence of speech from one speaker. We use the term *communicative act* (CA) to refer to a segment of speech from one speaker that has a main *communicative function*. A single turn may contain several CA's, but one CA may also span several turns. The term CA may also include communicative gestures that are used in face-to-face dialogue, although non-spoken communication is not addressed in this thesis.

A distinction can be made between the *form* and *function* of a CA. The form is the words which are spoken and their prosodic realisation. The function is the effect the CA has (or is supposed to have) on the listener in the context that it is signalled. This is related to Austin's distinction between the *locutionary* and *perlocutionary* acts that are performed when speaking (Austin, 1962). Take the following example:

(5) A: What is your name?
B: *Ben*

B can be said to do at least two things here: saying the word "Ben" (the locutionary act) and informing A that his name is Ben (the perlocutionary act). The same communicative function can (in a certain context) be achieved by using very different forms. Take the following examples, where the intended effect is that the listener should repeat the last utterance:

(6)	a. What did you say?	(INTERROGATIVE)
	b. Please repeat.	(IMPERATIVE)
	c. Sorry, I didn't hear what you said.	(DECLARATIVE)

In order to analyse CA's, it is useful to group form and functions into some sort of categories. The forms may be easier to categorise – a rough grouping can be done by sentence-type, as in example (6) above – but the functions are more difficult, since the functions depend on the context to a large extent, and the number of possible contexts is infinite and hard to characterise. To solve this, a level in-between locutionary and perlocutionary was defined by Austin, under the name of *illocutionary act* or *speech act*. This should be understood as the conventional function certain kinds of utterances have in accord with a conventional procedure. Austin developed a taxonomy of illocutionary acts, which was later modified by Searle (1979) into a classification of five basic types of actions that may be performed when speaking an utterance:

- ASSERTIVE: committing the speaker to something being the case.
- DIRECTIVE: attempting to get the listener to do something.
- COMMISSIVE: committing the speaker to some future course of action.
- EXPRESSIVE: expressing the psychological state of the speaker about a state of affairs.
- DECLARATION: bringing about a different state of the world via the utterance.

A problem with the concept of illocutionary acts is that it is not always clear whether it really refers to form or function, since speech acts may also have an *indirect* effect. For example, the examples in (6) above could all be considered to be DIRECTIVES, but c. could also be classified as an ASSERTIVE with an indirect DIRECTIVE effect. It is also possible that a CA may have several simultaneous functions. Indeed, the whole concept of illocutionary act has been questioned for various reasons that will not be described in detail here (see Levinson (1983) for a discussion).

Other schemes for classifying the functions of CA's have been proposed under the names of *dialogue acts* and *conversational moves*. One such scheme is DAMSL (Allen & Core, 1997), where each CA has a forward-looking function (such as STATEMENT, INFO-REQUEST, THANKING) and a backward-looking function (such as ACCEPT, REJECT, ANSWER). This scheme is also more detailed than Searle's, containing functions such as SIGNAL-NON-UNDERSTANDING, which may be applied to all examples in (6).

These classifications are necessary and useful to make, if one wants to analyse patterns in dialogue or build dialogue systems, but it should be noted that there will probably never be an exhaustive scheme that divides all possible functions of CA's for all possible domains into clearly delimited categories. The usefulness of a given scheme depends on what kind of application or analysis it will be used for. The reliability of a scheme can be measured by inter-annotator agreement.

2.2.2 Discourse

A sequence of communicative acts forms a *discourse*. If many records of conversations are collected and analysed, patterns will emerge in the discourses. For example, questions tend to be followed by answers, offers by acceptances or rejections, greetings by greetings, etc. This phenomenon is called *adjacency pairs* (Schegloff & Sacks, 1973). Between the two parts of an adjacency pair, *insertion sequences* of other adjacency pairs may also appear (often called *sub-dialogues* in spoken dialogue systems), resulting in more complex hierarchical patterns. Here is an example:

(7)	A.1: What does the red one cost?	(Raise Q.1)
	B.2: *The red one?*	(Raise Q.2)
	A.3: Yes	(Answer to Q.2)
	B.4: *It costs 100 crowns.*	(Answer to Q.1)

Other patterns will also emerge. For example, conversations often start with an opening section with greetings and end with a closing section with farewells.

It is tempting to conclude that there is some sort of generative "grammar" for conversational discourse, comparable to the grammar of a sentence. But, as Levinson (1983) points out, conversation is not a structural product in the same way that a sentence is – its outcome is spontaneously created in cooperation between two speakers with different goals and interests. In the words of Clark (1996): "People may have general goals on entering a conversation, but they cannot prepare specific plans to reach them. They must achieve what they do contribution by contribution". The reason that people tend to give answers after questions is that they often want to be compliant and have appropriate answers to give, not that there is some grammar that tells them to answer. If a speaker responds to a question with a partial answer, or with a rejection of the presuppositions of the question, or simply ignores it, it would not result in an "ungrammatical" or "ill-formed" discourse. Moreover, what counts as adjacency pairs is not obvious. If A would instead have said "The red one looks expensive" in A.1, it is not obvious if A.1 and B.4 should be considered to constitute an adjacency pair, although the perlocutionary effect of A.1 would have been (almost) the same. This shows that the structures and patterns that emerge depend on how we assign functions to communicative acts, which is indeed problematic, as discussed above. If we replace A.3 in the example above with a paraphrase which should have the same perlocutionary effect, the structure of the dialogue changes into one without any insertion sequence:

(8)	A.1: What does the red one cost?	(Raise Q.1)
	B.2: *The red one?*	(Raise Q.2)
	A.3: What does **the red one** cost?	(Raise Q.3)
	B.4: *It costs 100 crowns.*	(Answer to Q.3)

2.2.3 Ellipsis and anaphora

While the structure of the discourse does not govern *which* perlocutionary acts can be performed, it may provide constraints for how an act must be *formulated* to achieve a given perlocutionary effect. These constraints are perhaps most obvious when speakers want to reduce their communicative effort.

One way of making utterances more efficient is to leave things out that can be recovered by making inferences from the discourse. This phenomenon has been called *ellipsis* (e.g., Carbonell, 1983), *non-sentential utterances* (e.g., Schlangen, 2003), *fragmentary utterances* (e.g., Ginzburg et al., 2001), or *information enriched constituents* (Ericsson, 2005). B.2 and A.3 in (7) above are examples of this. The way ellipses "borrows" from the context can be understood by replacing them with paraphrases:

(9)	A.1: What does the red one cost?	
	B.2: *The red one?*	→ *Did you say "the red one"?*
	A.3: Yes	→ I said "the red one".
	B.4: *It costs 100 crowns.*	

In this example, B.2 borrows from A.1 and A.3 borrows from B.2. Note that A.3 can only be resolved correctly once B.2 has been resolved. By making such paraphrases, we may also assign a specific reading of the intentions behind the utterances.

When B in B.4 wants to assert the price of "the red one" in the examples above, he could use the more efficient ellipsis "100". However the use of such a construct is highly constrained by the preceding discourse in a way that a sentential utterance isn't. Suppose A had instead asked "how old is the red one" in A.1. B could then still answer "it costs 100 crowns" and be understood, while the ellipsis "100" (with the intended perlocutionary effect) would be directly misleading. There has been much work at understanding the ways in which the discourse constrains the use of ellipses, but we will not go into more detail here. Recent computational accounts of ellipsis resolution in dialogue systems are given in Schlangen (2003) and Ericsson (2005).

Another way of reducing the communicative effort is to use different forms of anaphora, that is, references to objects that have already been referred to in the same discourse. When an object is mentioned in the discourse for the first time, we may say that a representation of it – an *entity* – is *evoked*. This entity may then correspond to zero, one or more *referents* in the real world. Once an entity has been evoked, it may be referred to again, or *accessed*, by the use of anaphora. An example of this is the use of "it" to refer to "the red one" in B.4 in the example above.

Just as for ellipses, the use of anaphora is constrained by the discourse. If another entity would have been evoked between A.1 and B.4, the pronoun "it" would have been misunderstood.

Indefinite descriptions are often used to mark that an entity is *new* and should be evoked (e.g., "I saw a cat"). Definite descriptions or pronouns are often used to mark that an entity is *given* and should be accessed (e.g., "I saw the cat", "I saw him"). While these are general patterns, there are of course exceptions (see Brown & Yule (2004) for an overview). A computational model of reference resolution should be capable of recognising whether a linguistic construct is used to evoke or access an entity, and in the latter case find the accessed entity in the discourse history. Several such models have been proposed, such as the Centering algorithm (Grosz et al., 1995).

2.2.4 Spontaneous speech

Conversational speech is typically spontaneous and not planned beforehand. Such speech is to a large extent characterised by *disfluencies*, such as:

- Filled pauses: "I have a eeh red building on my left"
- Repetitions: "I have a red red building on my left"
- Repairs: "I have a red buil- *blue* building on my left"
- False starts: "I have – I have a red building on my left"

Disfluencies arise partly due to the cognitive limitations of the speakers, but they may also be utilised by speakers to, for example, mark focus or signal uncertainty. Shriberg (1994) has shown that disfluencies occur in spontaneous speech at rates higher than every 20 words, and can affect up to one third of all utterances, across many corpora and languages. Due to the less predictable word order and the truncated words caused by disfluencies, spontaneous speech is likely to give rise to more errors in automatic speech recognition. Unfilled pauses, such as "I have a … red building on my left", are not always considered to be disfluencies, but they arise for the same reasons. Such mid-utterance pauses are problematic in spoken dialogue systems, since they make it hard for the system to know when the user has ended the turn (Ferrer et al., 2002; Edlund & Heldner, 2005; Bell et al., 2001).

2.3 Spoken dialogue system components

Spoken dialogue systems are indeed complex systems, incorporating a wide range of speech and language technologies, such as speech recognition, natural language understanding, dialogue management, natural language generation and speech synthesis. These technologies do not only have to operate together in real-time, but they also have to operate together with a user, which may have individual needs and behaviours that should be taken into account.

A straightforward and well-known approach to dialogue system architecture is to build it as a chain of processes (a pipeline), where the system takes a user utterance as input and generates a system utterance as output. Such a processing chain is shown in Figure 2.1.

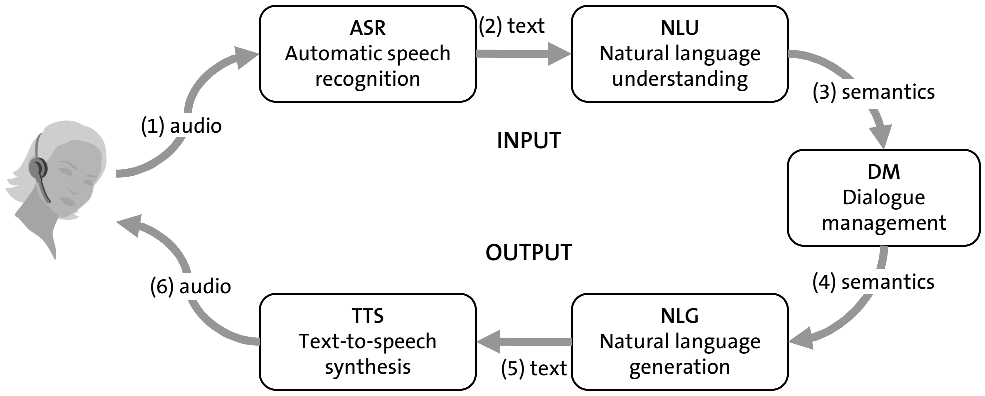


Figure 2.1: A typical processing chain in a spoken dialogue system.

In this chain, the speech recogniser (ASR) takes a user's spoken utterance (1) and transforms it into a textual hypothesis of the utterance (2). The natural language understanding (NLU) component parses the hypothesis and generates a semantic representation of the utterance (3), normally without looking at the dialogue context. This representation is then handled by the dialogue manager (DM), which looks at the discourse and dialogue context to, for example, resolve anaphora and interpret elliptical utterances, and generates a response on a semantic level (4). The natural language generation (NLG) component then generates a surface representation of the utterance (5), often in some textual form, and passes it to a text-to-speech synthesis (TTS) which generates the audio output (6) to the user.

These may be the most important components of most dialogue systems. Still, this is just a rough sketch of how a prototypical dialogue system may be composed. There are some possible alternatives and extensions to this architecture:

- There may be additional components, such as prosodic analysis, discourse modelling, deep and surface language generation, etc.
- Instead of passing all data along the pipe, it is possible to have a *shared information storage* or a *blackboard* that all components write and read to, and from which they may subscribe to events (Turunen, 2004).
- The components may send other messages, and not just according to this pipeline. For example, the ASR may send messages about whether the user is speaking or not directly to the TTS.
- The components might operate asynchronously and incrementally. Asynchronicity means that, for example, the ASR may recognise what the user is saying while the DM is planning the next thing to say. Incrementality means that, for example, the ASR recognises the utterance word by word as they are spoken, and that the NLU component simultaneously parses these words. Allen et al. (2001b) argues that these are crucial issues for conversational dialogue systems.

Another approach is of course to not divide the dialogue system into components at all, since much information and efficiency is lost when the result of one component is serialized into a message that is sent to another component. For example, ASR and NLU may be done in the same processing step (Esteve et al., 2003). The advantage of the division into components – especially for research systems – is that the components can be developed individually by different developers working with different approaches (and possibly different programming languages), as long as the interfaces between the components are well defined.

2.3.1 Automatic speech recognition

The task of the automatic speech recogniser (ASR) is to take an acoustic speech signal and decode it into a sequence of words. To do this, the speech signal must first be segmented into utterances that may be decoded. This segmentation is typically called *endpointing* or *voice ac-*

tivity detection (VAD), and is most often done based on the discrimination between speech and silence. When a certain amount of speech, followed by a certain amount of silence (a *silence threshold*, for example 750 ms) is detected, this segment is considered to be an utterance. This method is not unproblematic. First, the discrimination between speech and silence is not trivial, since the signal may also contain background noise. Second, silence is not a very good indicator that someone has finished a turn. People typically make pauses mid-utterance (as discussed in 2.2.4), at least in conversational dialogue, resulting in the system interrupting the user. To cope with this, the silence threshold may be increased, but this will instead lead to very slow turn-taking from the system. Because of this, researchers have looked at other features than silence for determining endpoints, such as intonation (see for example Ferrer et al., 2002: Edlund & Heldner, 2005).

Once the speech signal has been segmented, the decoding task can be described as follows: Take the acoustic input, treat it as a noisy version of the source sentence, consider all possible word sequences, compute the probability of these sequences generating the noisy input, and choose the sequences with the maximum probability (Huang et al., 2001). To do this, a set of models for computing these probabilities are needed, as well as a method for parameterizing the audio signal into features, and an efficient search algorithm, since the amount of possible word sequences to consider is huge.

To understand which models are needed, the problem can be summarized as: "What is the most likely word sequence, W_H, out of all possible word sequences ($W_1, W_2, W_3, \ldots W_n$) in the language *L,* given some acoustic observation *O*". This problem can be mathematically described in the following equation:

(10) $$W_H = \underset{W \in L}{\arg\max}\, P(W \mid O)$$

P(W|O) is not so easy to compute. However, using Bayes theorem, we can rewrite the equation as:

(11) $$W_H = \underset{W \in L}{\arg\max}\, \frac{P(O \mid W)P(W)}{P(O)}$$

P(O) is not so easy to compute either, but since *P(O)* is constant, regardless of which word sequence is considered, this equation can be rewritten as:

(12) $$W_H = \underset{W \in L}{\arg\max}\, P(O \mid W)P(W)$$

We end up with two probabilities that are much easier to compute. The probability of the observation given the word sequence, *P(O|W)*, is called the *acoustic model*, and the probability of the word sequence, *P(W)*, the *language model.* The parameterisation of the audio signal, the acoustic model and the search algorithm will not be described here, but see Huang et al. (2001) for an overview.

There are basically two common types of language models typically used in dialogue systems: context-free grammar models (CFG), and n-gram models. A CFG consists of a number of (hand-written) rules and a lexicon. A very small CFG may look like this:

(13) Grammar rules: `S->[NP VP]; VP->[V NP]; NP->[N]`
Lexicon: `V->"loves"; N->"John"; N->"Mary"`

This grammar states that a sentence (S), such as "John loves Mary", may consist of a noun phrase (NP) and a verb phrase (VP), and that the verb phrase ("loves Mary") may consist of a verb (V), such as "loves", and a noun phrase. A noun phrase may in turn consist of just a noun, such as "John" or "Mary". Such a grammar describes all legal word strings in a given language, but does not contain any statistical information. A corpus with texts may be used to infer the probabilities of the different rules and thus make the CFG stochastic.

N-gram models are purely stochastic; they model the probability of a sequence of words as a product of the probabilities of each word, given the N-1 preceding words. For example, a trigram model may contain the probability that the word "house" occurs after the string "have a".

The general characteristics of these two types of models, in the context of spoken dialogue systems, are fairly well known (see for example Knight et al., 2001). CFG models typically perform better than n-gram models, given that the user knows what can be said and speaks in-grammar, and are thus better suited for command-based dialogue systems. N-gram models typically need large amounts of data to perform well, but are better at covering spontaneous, less predictable, speech. N-gram models also tend to degrade more gracefully when the user's utterances are poorly covered by the grammar, but they may also give rise to ASR results which are often only partially correct. Such models may therefore be more suitable for conversational dialogue systems.

A method for increasing the ASR accuracy is to train the acoustic models on the specific user, in a so-called enrolment procedure. This is used in all state-of-the-art dictation systems, since such systems need very large vocabularies. Typically, the user uses the same dictation application frequently on the same computer, and an enrolment procedure is therefore acceptable. Since dialogue systems are often used more infrequently and often by first-time users, enrolment is very seldom used.

The output from the ASR may consist of a single hypothesis containing a string of words, but an ASR may also deliver multiple hypotheses, in the form of an *n-best list* or a *word lattice*. An n-best list is simply a list of the top hypotheses (containing *n* items). Since similar word sequences achieve similar probabilities, many of the different hypotheses are often just one-word variations of each other (Huang et al., 2001), as can be seen in Table 2.2. Especially for longer utterances, n-best lists need to be very long in order to contain useful information. N-best lists are therefore also very inefficient representations. For example, a 10-word utterance with 2 different word hypotheses in each position would need an n-best list of 2^{10}=1024 items to include all hypotheses. Word lattices are much more compact representations, in which the possible word sequences are represented in graph, as can be seen in Figure 2.2.

Table 2.2: An example 6-best list (derived from Huang et al., 2001).

I will tell you would I think in my office
I will tell you what I think in my office
I will tell you when I think in my office
I would sell you would I think in my office
I would sell you what I think in my office
I would sell you when I think in my office

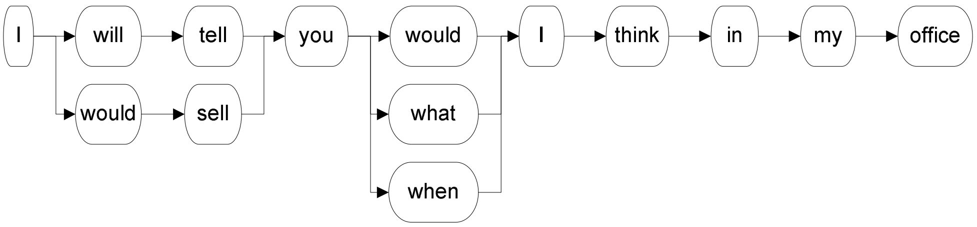


Figure 2.2: A word lattice representation of the 6-best list in Table 2.2.

The hypotheses may also be annotated with confidence scores that carry information about how well the data fit the models. Confidence scores can be provided for the whole string, but also for the individual words in it. The calculation of confidence scores is described and discussed in the next chapter.

The ASR is usually the only component that processes the speech signal, with the main task of recognising the words used. However, there is an important aspect of the speech signal – prosody – that is typically ignored. Systems doing prosodic analysis therefore need some other component that takes the speech signal as input.

2.3.2 Natural language understanding

The task of the natural language understanding (NLU) component is to parse the speech recognition result and generate a semantic representation. Such components may be classified according to the parsing technique used and the semantic representations that are generated.

A classic parsing approach is to use a CFG- based grammar which is enhanced with semantic information. The same CFG may then be used for both speech recognition and NLU. This is done in many command-based dialogue systems, such as those implemented in the W3C-standard VoiceXML (McGlashan et al., 2004). This approach may not be applicable to conversational dialogue systems, since they often require n-gram language models in the ASR, which typically generate partly erroneous and less predictable word sequences that are hard to cover with a CFG. A more robust approach is to use *keyword* or *keyphrase spotting* (i.e., some sort of pattern-matching), where each word or phrase is associated with some semantic construct (Ward, 1989; Jackson et al., 1991). The problem with this approach is that more complex syntactic constructs (such as negations) are easily lost. It can also lead to an over-generation of solutions on erroneous input. Another approach to parse less predictable input is to use a (possibly modified) CFG, but extend the parsing algorithm with robustness techniques (e.g., Mellish, 1989; Satta & Stock, 1994; Kasper et al., 1999; van Noord et al., 1999). The main potential drawbacks with this approach is that it may be inefficient (depending on how much robustness techniques are applied), and that it may over-generate solutions that are hard to choose among.

There has also been a lot of effort in data-driven approaches to NLU. One of the main rationales is to gain robustness. Examples of this include Segarra et al. (2002) and He & Young (2003). Data-driven approaches to NLU have mainly been used in academic dialogue systems, with the exception of call-routing, which is a successful commercial application. In such systems, the user is asked to make an initial description of her problem using free speech, so-called "open prompts". A large corpus of such descriptions is collected and machine-learning is used to classify them, so that the user may be routed to an appropriate operator. The "How May I Help You"-system developed at ATT is one of the most well-known examples (Gorin et al., 1997).

The purpose of the NLU component is to extract the "meaning" of the utterance. It is not obvious how this should be understood in the context of spoken dialogue systems. Bangalore et al. (2006) define this as "a representation that can be executed by an interpreter in order to change the state of the system". To know how the state should be changed, some sort of communicative function should be extracted, together with a description of the propositional content.

The simplest form of semantic result from a NLU component is a list of key-value pairs, also called *frame*. As an example, the interpretation of the utterance "I want to go from Paris to London at 8 am" may look like this:

```
(14)   {from: Paris
        to:   London
        time: 8am}
```

This example shows that many aspects of the semantics may be left out, if they are not needed for the system to update its state. In this case, the semantic representation does not tell which communicative function is being performed, nor does it tell who wants to make the booking. Frames are commonly used in command-based systems together with CFG parsing (for example in VoiceXML applications). A drawback with such frames is that they are inadequate for representing more complex semantics, such as relations between objects. Frames are suitable for keyword-spotting approaches, which cannot parse complex syntactic and semantic relations in utterances anyway. In order to represent more complex semantics, deep (or nestled) frames may be used (also called feature structures or attribute-value matrices). For example, a representation of the utterance "I want three Capricciosa and one Calzone", may look like this:

```
(15)   {order: [{item: pizza
                  type: Capricciosa
                  qty:  3}
                 {item: pizza
                  type: Calzone
                  qty:  1}]}
```

In many academic dialogue systems, first-order predicate calculus (FOPC) is used to represent semantics. Using FOPC, objects and their properties and relations may be represented, as well as quantifications of these. To be able to represent the semantics of fragments, such as verbs, and not just full propositions, FOPC is usually extended with lambda calculus. The strength of FOPC is that it is a well-defined, expressive formalism that also supports inferences, which may be utilised in dialogue systems (Bos & Oka, 2002). For an introduction to the use of FOPC for semantic representation of natural language, see Jurafsky & Martin (2000). An early example of a dialogue system using FOPC is TRAINS (Allen & Ferguson, 1994), and a later example is NICE (Boye et al., 2006). A practical hindrance for FOPC as a format for semantic representations in dialogue systems is that it is not a common data structure that is straightforward to use in most programming languages. Thus, it is often used in dialogue systems implemented in logic programming languages, such as Prolog.

2.3.3 Dialogue management

The most common tasks of the dialogue manager can be divided into three groups:

- Contextual interpretation: Resolve for example ellipses and anaphora.
- Domain knowledge management: Reason about the domain and access information sources.
- Action selection: Decide what to do next.

In other words, the dialogue manager can be regarded as the executive controller of the dialogue system, it is this component that holds the current state of the dialogue and makes decisions about the system's behaviour.

2.3.3.1 Contextual interpretation

The result of the NLU component is a semantic representation of the user's utterance. However, the NLU component does not, in most cases, have access to the dialogue history, and thus can only make a limited interpretation that in some cases has to be enriched. Take the following example:

(16) S.1: *When do you want to go?*
U.2: On Wednesday
S.3: *Where do you want to go?*
U.4: Paris

If the system is limited to simple command-based travel booking dialogues, the NLU component may directly infer from U.2 that the user wants to travel on Wednesday, since there is no other possible interpretation of "On Wednesday" within the domain. However, in U.4, the NLU component can only generate a semantic description which represents the city Paris. It cannot know if the user wants to go from Paris or to Paris. This has to be inferred by the dialogue manager. In command-based dialogue systems, there is usually less demand for the dialogue manager to make such inferences, compared to a conversational system, which may contain a lot of elliptical utterances that are dependent on context.

The system may resolve elliptical utterances by transforming them into their full propositional counterpart (Carbonell, 1983). In the example above, U.4 would be transformed into the semantic representation of "I want to go to Paris". Another solution is to open a semantic "slot" when an elliptical utterance is expected. For example, after S.3 in the example above, the system may open the slot "destination", in which the elliptical utterance U.4 will fit. An example of this approach is VoiceXML. This solution is more viable when there are not so many different elliptical utterances that are expected. In any case, to resolve ellipses, the system must somehow track the current state of the dialogue, which will be discussed later on.

Apart from the resolution of fragmentary utterances, the dialogue manager may also have to resolve anaphora, as the following example illustrates (with co-referring expressions underlined):

(17) S: *You should see <u>a red building</u> and a blue building. Position yourself so that you have <u>the red building</u> on your left.*
U: Ok, I have <u>it</u> on my left now.

In many dialogue system domains, anaphora does not need to be handled. An example of this is the extensively studied travel booking domain. In this domain, the user and system mainly talk about entities such as cities, dates and times, which are most often referred to with expressions that do not need to be resolved using the preceding discourse. For a dialogue system to resolve anaphora, it must somehow track which objects are talked about in its model of the dialogue state. Boye et al. (2004) argue that general purpose semantic reasoning for anaphora resolution might be unnecessarily complex and computationally demanding, given that the dialogue systems acts within limited domains. Instead, they propose a strategy where the most recently referred object of a compatible type is searched for.

2.3.3.2 Domain knowledge management

According to Flycht-Eriksson (2001), the different knowledge sources needed by a dialogue manager can be separated into dialogue and discourse knowledge, task knowledge, user knowledge and domain knowledge. The first two are needed for contextual interpretation and action selection. User knowledge can be used to adapt the system's behaviour to the user's experience and preferences. *Domain knowledge management* includes models and mechanisms for reasoning about the domain and for accessing external information sources, such as an SQL database or a graphical information system (GIS). In order to do this, the system must have a *domain model*, that is, a semantic representation of the world which can be mapped to natural language (which is done in the NLU and NLG processes) and to queries of the information sources. In simpler domain models, the semantic representation of utterances can be mapped directly to database queries. In more complex models, the system may have to translate vague representations (such as "large") into more precise queries, but it may also have to draw inferences (for example understanding that a house is also a building). More complex descriptions of how concepts in the domain model are related (which may be used for inference) are often referred to as *ontologies*.

Domain knowledge management is often integrated into the other tasks of the dialogue manager, but it may also be separated from the other tasks into an individual process (Flycht-Eriksson, 2001).

2.3.3.3 Action selection

The third main task of the dialogue manager is to make the decisions on what the dialogue system should do next, which is often a communicative act that is to be generated and synthesised by the output components. Different approaches to making this decision have been proposed. A general idea is to separate an "engine" from a declarative model, in order to make the transition between applications and domains easier.

Central to action selection (just as for contextual interpretation) is that it should somehow be based on the *context*, in other words the dialogue *state*. A fairly simple method is to model the dialogue as a set of pre-defined dialogue states in a finite state machine (FSM). FSM is a well-known formalism in which the model is separated from the engine.

The FSM approach is appropriate when the semantics of the dialogue simply can be represented by the current state in the machine. However, in many applications, more complex semantics is needed. For example, in a travel booking system, a frame with feature-value pairs may be used, such as:

```
(18)   {from: _
        to:   _
        date: _
        time: _}
```

If this frame is just filled one slot at a time, controlled by the system's initiative, an FSM is sufficient. But if the user should be allowed to fill any field of his choice, or multiple fields at the same time (so-called *mixed-initiative*), the number of different combinations of filled slots grows exponentially as more slots are added. Thus, the FSM grows very large and becomes incomprehensible for the dialogue designer. To solve this problem, the dialogue state is instead commonly modelled as a *store* containing variables. A pre-defined or custom *control algorithm* is then used to operate on the store. Depending on the current state of the store, different actions will be taken. The store need not represent the complete dialogue history. It only needs to represent the parts that are determined to be needed for the application.

An example of this approach is VoiceXML, where the store is modelled as a flat frame, as in example (18) above. A pre-defined control algorithm, called the *form interpretation algorithm* (FIA), is used, which operates in the following way:

1. Start from the top, stop at the first slot that is not filled.
2. Play a prompt that is associate with that slot, such as "Where do you want to go?"
3. Wait until the user fills one or more slots (for example by saying "I want to go from Paris to London"), or a timeout occurs.
4. If all slots are filled, end the form. Otherwise, restart with step 1.

This simple algorithm will ensure that all slots will be filled, but will also allow the user to fill other slots than the one asked for, or multiple slots at the same time (i.e., mixed-initiative). It is also possible to extend the control mechanism with events that are triggered when certain slots are filled.

An example of a more complex model is the *information state approach* implemented in TrindiKit (Larsson & Traum, 2000), where the store (called the *information state*) is a deep feature structure with variables. The control algorithm consists of a set of *update rules* which have preconditions (on the information state) and effects. Effects may be communicative acts, but also changes to the information state itself, triggering other rules to apply. The order in which update rules are applied is controlled by an *update algorithm*. The structure of the information state, the update rules and the update algorithm may all be customised for the specific application. The approach is similar to that of *production systems* in AI, in which the store is called the *working memory* and the update rules *productions* (Klahr et al., 1987).

In grammar-based approaches to dialogue management, a grammar is used to describe how speech acts can (and should) be sequenced and structured. An example of this approach is the LinLin system (Jönsson, 1997), where a context-free dialogue grammar is used. The grammar is used for both contextual interpretation and for action selection. As the dialogue is parsed using the grammar, the resulting tree structure becomes a model of the dialogue history. As discussed in 2.2.2 above, one may doubt whether dialogue really can be described using a grammar similar to the way sentences can, or if selection of perlocutionary acts really should be done with the help of a grammar. Even if the dialogue can be parsed with a grammar in hindsight, it may be more problematic to do it online, as the past dialogue probably has to be reinterpreted in light of new utterances. The approach may therefore be more useful for modelling simpler command-based dialogues than for spontaneous, conversational dialogue.

In plan-based approaches, utterances are treated in the same way as actions in a planning system that are performed in order to achieve some goal (Allen & Perrault, 1980). By recognising the user's intentions and goals, the system may infer which (communicative) acts need to be performed for the goal to be reached. Each act is associated with preconditions, constraints and effects. The idea is that the system has access to a library of such acts and may flexibly put these together in a chain that forms a plan that may meet the desired goal. One of the problems with this approach is that this sort of planning quickly becomes combinatorically intractable as the number of possible acts increase. It is also questionable whether these kinds of generalised and flexible methods for generating plans are really needed for most dialogue systems that operate within limited domains. Often, plans can be represented in a more straightforward way and defined in advance, given that a limited number of plans are really needed in a given domain. For example, the travel booking frame in example (18) above, together with the FIA, may be regarded as a plan for collecting information that is needed to book a trip. It is very simple and fixed, but nevertheless efficient.

Recently, there have been many efforts at making dialogue management (especially action selection) data-driven, by using statistical methods and machine learning approaches. The rationale behind this effort is that other language technologies have moved in this direction with great success (Young, 2002). Data-driven approaches to action selection will be discussed in Chapter 8.

2.3.4 Natural language generation

The NLG component takes the semantic representation of a communicative act from the system and generates a textual representation, possibly with prosodic markup, that is to be synthesised by a speech synthesiser. The simplest approach to this is so-called *canned answers*, that is, a simple mapping between a discrete set of communicative acts and their realisations. To make the answers more flexible, templates may be used that contain slots for values, such as "Ok, you want to go from <from> to <to>, <date> at <time>".

Research into more advanced and flexible models for NLG has mainly been concerned with the production of written language or coherent texts to be read, dealing with issues such as aggregation and rhetorical structure (Reiter & Dale, 2000). Although these issues may be important for dialogue systems as well, there are a number of perhaps more important issues that are specific for spoken dialogue systems, where utterances have to fit into an ongoing conversation.

Spoken dialogue is, as opposed to a written text, produced "online" and cannot be post-edited. Since the whole discourse cannot be planned beforehand, the semantics of each utterance should be integrated into the system's discourse model, since they may affect the realisation of future utterances. If the system is interrupted in the middle of an utterance, it should ideally know and remember which parts of the intended utterance were actually said.

There are several ways of realising a communicative act, for example by selecting between full utterances and elliptical constructions, or by using different anaphoric expressions. As shown in the discussion in the next chapter, the choice of realisation is important in that different realisations will signal understanding in different ways, so-called grounding. Another aspect is that a flexible choice between different forms of realisations may increase the naturalness and efficiency of the system.

Humans engaging in a dialogue tend to coordinate their linguistic behaviour with each other. Garrod & Anderson (1987) have shown in experiments on human-human conversation that speakers build conceptual pacts during the dialogue, which are specific for the discourse shared between them. Several studies have shown that users of spoken dialogue systems adjust the vocabulary and expressions to the system (see for example Brennan, 1996; Gustafson et al., 1997; Skantze, 2002). Also, if users consistently choose some lexical realisations themselves, they will probably have a greater confidence in a system that adopts these choices.

2.3.5 Text-to-speech synthesis

A simple and straightforward way of generating the audio output is to use pre-recorded speech, which is commonly used in commercial applications, since text-to-speech synthesis (TTS) is often considered lacking in quality. The drawback is, of course, lack of flexibility, and pre-recorded speech is therefore only suitable if utterances are produced as canned answers, and to some extent templates. To gain more flexibility, a TTS process is needed. TTS can generally be divided into two separate problems – the mapping from an orthographic text to a phonetic string with prosodic markup, and the generation of an audio signal from this string. The first problem has been addressed with both knowledge-driven (Carlson et al., 1982) and data-driven (van den Bosch & Daelemans, 1993) approaches. To the second problem, there are three common approaches: *formant synthesis*, *diphone synthesis* and *unit selection*. A formant synthesiser models the characteristics of the acoustic signal, which results in a very flexible model. However, the speech produced is not generally considered very natural-sounding. In diphone synthesis, a limited database of phoneme transitions is recorded, and the synthesised speech is concatenated directly from this database. Prosodic features are then added in a post-processing of the signal. Unit selection is also based on concatenation; however, the chunks are not limited to phoneme transitions. Instead, a large database is collected, and the largest chunks that can be found in the database are concatenated. Unit selection needs large amounts of data to achieve acceptable quality. This may be a problem if the dialogue system agent should have its own voice, or the system should have many personas with different characteristics, for example in game applications. A solution to this is to use a limited domain synthesis, which is basically a unit selection synthesis based on a smaller data set collected with the intended application in mind (Black & Lenzo, 2000). If the output is to be generated by an embodied conversational agent (ECA), facial (and possibly bodily) animation is also needed (see for example Beskow, 2003).

The interface between the NLG component and the TTS component is important, since it should not only consist of a text – it should also be possible to represent other properties of utterances, such as prosody. GESOM (Beskow et al., 2005) and the W3C standard SSML (Burnett et al., 2004) are examples of such interfaces.

Many utterances are not composed of words, but are *non-lexical* (or *non-verbal*) utterances, such as “uh-huh”, “um”, and “hmm”. According to Campbell (2007), approximately half of the speech sounds used in normal everyday conversational speech are non-lexical. In non-lexical utterances, much of the meaning is conveyed by prosody, rather than by the phonetic content. Non-lexical utterances are commonly used in dialogue as feedback, disfluency markers and the like. Their primary functions are turn-taking control, negotiating agreement, signalling recognition and comprehension, and expressing emotion, attitude, and affection

(Ward, 2004). It is important to understand how prosody and phonetic content affects the pragmatics of synthesised non-lexical utterances (Wallers et al., 2006), if a dialogue system should be able to generate them.

2.4 Summary

This chapter has described some basic properties of spoken dialogue and the techniques and issues involved in developing spoken dialogue systems. It was argued that two general types of dialogue systems may be distinguished, command-based and conversational, and that this thesis is targeted towards the latter, that is, dialogue systems that to a larger extent build upon the principles of human conversation. In dialogue, speakers take turns in the roles of speaker and listener and exchange communicative acts (CA's), varying in form and function. A sequence of CA's forms a discourse.

To be able to engage in conversation, a spoken dialogue system has to attend to, recognise and understand what the user is saying, interpret utterances in context, decide what to say next, as well as when and how to say it. To achieve this, a wide range of research areas and technologies must be involved, such as automatic speech recognition, natural language understanding, dialogue management, natural language generation and speech synthesis.

A more detailed description of a complete spoken dialogue system is provided in Chapter 6, where the HIGGINS system – a dialogue system developed for exploring error handling issues – is described.

CHAPTER 3

Miscommunication and error handling

In the previous chapter, conversation and spoken dialogue systems were described from a very general perspective. In this description, a fundamental issue is missing: how to deal with *uncertainty* and *errors*. Understanding is not something that speakers can take for granted, but something they constantly have to signal and monitor, and something that will sometimes fail. In this chapter, we will first review how humans ensure understanding in communication and what happens when miscommunication occurs. We will then discuss the concept of error in the contexts of human-human and human-computer dialogue, and review research done on how errors in spoken dialogue systems may be detected and repaired.

3.1 Miscommunication and grounding

3.1.1 Miscommunication

Miscommunication is a general term that denotes all kinds of problems that may occur in dialogue. One reason for miscommunication being fairly frequent in dialogue may be explained by *the Principle of Parsimony*[3] (Carletta & Mellish, 1996):

> The Principle of Parsimony states that people usually try to complete tasks with the least effort that will produce a satisfactory solution. In task-oriented dialogue, this produces a tension between conveying information carefully to the partner and leaving it to be inferred, risking a misunderstanding and the need for recovery. (p. 71)

[3] Also called Ockham's Razor.

For example, speakers may produce ambiguous referring expressions, use fragmentary utterances which can only be understood assuming a certain common ground between the speakers, and may use extremely reduced phonetic realisation of utterances. These are all different ways of increasing efficiency and introducing risk – there is always the possibility that listeners will not interpret them correctly. However, it may not be worth the effort to produce unambiguous expressions and canonical pronunciations, if the intended messages usually are interpreted correctly or if it is easy to diagnose and correct the problem when they are not.

There are different ways of analysing miscommunication phenomena. A common distinction is made between misunderstanding and non-understanding (e.g., Hirst et al., 1994; Weigard, 1999). *Misunderstanding* means that the listener obtains an interpretation that is not in line with the speaker's intentions. If the listener fails to obtain any interpretation at all, or is not confident enough to choose a specific interpretation, a *non-understanding* has occurred. One important difference between non-understandings and misunderstandings is that non-understandings are noticed immediately by the listener, while misunderstandings may not be identified until a later stage in the dialogue. Some misunderstandings might never be detected at all. The same utterance may, of course, give rise to both misunderstanding and non-understanding, that is, parts of an utterance may be misunderstood while others are not understood. Successful communication may be referred to as *correct understanding* or just *understanding*[4]. Misunderstanding and correct understanding are similar in that the listener chooses a specific interpretation and assumes understanding, which is not the case for non-understanding.

A second way of analysing miscommunication is by the *action level* with which the problem is associated. Both Allwood et al. (1992) and Clark (1996) make a distinction between four *levels of action* that take place when a speaker is trying to communicate something to a listener. The authors use different terminologies, but the levels are roughly equivalent. The terminology used here is a synthesis of their accounts. Suppose speaker A proposes an activity for listener B, such as answering a question or executing a command. For communication to be "successful", all these levels of action must succeed (listed from higher to lower):

- Acceptance: B must accept A's proposal.
- Understanding: B must understand what A is proposing.
- Perception: B must perceive the signal (e.g., hear the words spoken).
- Contact: B must attend to A.

More fine-grained analyses are of course also possible. The understanding level may for example be split into discourse-independent meaning (e.g., word meaning) and discourse-dependent meaning (e.g., referring expressions). The order of the levels is important; in order

[4] Brown (1995) prefers the term *adequate* interpretation (or understanding). According to her, every utterance is understood for a particular purpose on a particular occasion. There is, in most conversational settings, not a single interpretation which is "correct", but a number of adequate interpretations which will serve to fulfil the purpose of the speakers' joint project.

to succeed on one level, all the levels below it must be completed. Thus, we cannot understand what a person is saying without hearing the words spoken, we cannot hear the words without attending, and so on. Clark calls this the *principle of upward completion*.

Now, misunderstanding and non-understanding may occur on all these levels of action. B might correctly hear the words spoken by A, but misunderstand them or not understanding them at all. B might also attend to A speaking, but misrecognise the words spoken, or not hear them at all. As Dascal (1999) notes, this is reflected in the different names for misunderstanding in the English language, such as: *mishear*, *misrecognise*, *misinterpret*, *misinfer*, *misconclude*. In this thesis, however, we will stick to the terms misunderstanding and non-understanding to denote the general phenomena, and state which level is concerned if necessary.

It is questionable, however, whether failure on the level of acceptance really should be classified as miscommunication. If someone rejects a request or does not accept a proposal, we could easily say that the participants have succeeded in their communication. If A and B engage in a dialogue about possible activities and A suggests that they should go and see a movie, and B then rejects this proposal because he has already seen the film, we may say that they have successfully communicated that this is not an option.

A third distinction can be made depending on the *scope* of the miscommunication. Misunderstanding and non-understanding may concern not only the whole utterance, but also parts of it, resulting in *partial misunderstanding* and *partial non-understanding*:

(19) A: I have a red building on my left.
B (partial misunderstanding):
How many stories does the blue building have?
B (partial non-understanding):
What colour did you say?
Did you say red?

3.1.2 Grounding

Communication can be described as the process by which we make our knowledge and beliefs *common*, we add to our *common ground*. Clark (1996) defines the notion of common ground as follows:

> Two people's common ground is, in effect, the sum of their mutual, common, or joint knowledge, beliefs, and suppositions. (p.92)

When engaging in a dialogue, two people may have more or less in their common ground to start with. During the conversation, they try to share their private knowledge and beliefs – to add them to the common ground. As Clark (1996) points out, however, the process by which speakers add to the common ground is really a joint project, in which the speakers have to cooperatively ensure mutual understanding. A speaker cannot simply deliver a message and hope that the listener will receive, comprehend and accept it as correct. They have to constantly send and pickup signals about the reception, comprehension and acceptance of the information that is communicated. This is the process of *grounding*.

3.1.2.1 Evidence of understanding

In order to ground information, people give *positive* and *negative evidence of understanding* to each other. According to Clark, each contribution to the common ground requires a *presentation phase* and an *acceptance phase*. In the presentation phase, speaker A presents a signal for the listener B to understand; in the acceptance phase, B provides evidence of understanding. However, speaker B may in the same turn start a new presentation phase. Thus, each utterance can be said to communicate on two different tracks. On Track 1, knowledge and beliefs about the topic at hand are exchanged. At the same time, communication about understanding is (implicitly or explicitly) performed on Track 2. In Clark's words:

> Every presentation enacts the collateral question "Do you understand what I mean by this?" The very act of directing an utterance to a respondent is a signal that means "Are you hearing, identifying, and understanding this now?" (Clark, 1996, p.243)

The term *evidence of understanding* is closely related to the term *feedback*. The latter term is generally used to denote the information that an agent may receive about the consequences of the agent's actions. In this thesis, we will use the term evidence of understanding, which more precisely denotes feedback that concerns the management of understanding. For example, Allwood et al. (1992) use the term *linguistic feedback* to denote mechanisms by which interlocutors signal their understanding, but also attitudinal reactions and answers to yes/no-questions.

In Clark's account, some kind of positive evidence of understanding is required for each contribution to be considered as common. Clark & Schaefer (1989) list five different types of positive evidence:

1. The hearer shows *continued attention*.
2. An initiation of a *relevant next contribution*, for example an answer to a question.
3. An *acknowledgement* like "uh huh" or "I see".
4. A *demonstration* of understanding, for example a paraphrase.
5. A *display* of understanding, i.e., a repetition of some (or all) of the words used.

Evidence can be more or less strong. The types are listed above roughly from weak to strong: Evidence 1 and 3 only shows that the listener *thinks* that he understands; there is no real *proof* that the content of the utterance is really understood. In the words of Schegloff (1982): "'uh huh', 'mm hmm', head nods and the like at best *claim* attention and/or understanding, rather than *showing* it or *evidencing* it" (p. 78). Evidence 2 may indicate that some of the contents are understood correctly, but it is only evidence 4 and 5 that actually *prove* that (some of) the contents were correctly understood or perceived. As Traum (1994) points out, evidence 4 might actually be stronger than evidence 5, since the listener shows that he has processed the content on some deeper level.

Display of understanding may be given as *separate* communicative acts, with the main purpose of displaying understanding. These may be called *display utterances* or *echoic responses* (Katagiri & Shimojima, 2000). Here is an example:

(20) A: I have a red building on my left.
B: *A red building, ok, what do you have on your right?*

Display utterances and acknowledgements may also be given without keeping the turn, so-called *backchannels* (Yngve, 1970) or *continuing contributions* (Clark, 1996):

(21) A: I have a red building ...
B: *a red building*
A: ... on my left ...
B: *mhm*
A: ... and a blue building on my right.

An important function of such mid-utterance evidence is that it may denote which parts of the presentation utterance it concerns.

Display of understanding is very often *integrated* in the next communicative act, with its main purpose belonging to Track 1:

(22) A: I have a red building on my left.
B: *How many storeys does <u>the red building</u> have?*

3.1.2.2 The grounding criterion

In example (22) above, B could have used the pronoun "it" to refer to the building, but instead chooses the full definite description, which displays B's understanding. Thus, there is always a range of different realisations of the same propositional content (on Track 1), but which may provide different amounts of evidence (on Track 2). How do we, then, choose what strength of evidence to give? Clark (1996) defines grounding as follows:

> To ground a thing [...] is to establish it as part of common ground *well enough for current purposes*. (p.221, italics added)

Thus, the requirements on how much evidence is needed vary depending on the current purposes. Clark calls these requirements the *grounding criterion*. There are at least three important factors that should govern the choice of what evidence to give. First, the level of uncertainty is of course an important factor. The more uncertain we are, the more evidence we need. A second important factor is the *cost of misunderstanding* and *task failure*. As less evidence is given, the risk that a misunderstanding occurs will increase – thereby jeopardizing the task the speakers may be involved with. However, a task failure may be more or less serious. Consider the following example:

(23) A: *Welcome to the travel agency. Ann here. How may I help you?*
B: Hi there, I would like to book a trip to Paris.
A: *Ok, to Paris, from where do you want to go?*

In this example, B's statement about the destination requires strong evidence (such as the display in the example), since booking a ticket with the wrong destination has serious effects. On the other hand, when Ann is presenting her name in the beginning of the conversation, there is typically no need for B to provide any evidence.

Why do we not always provide strong evidence, just to be certain, then? This is explained by the Principle of Parsimony, as discussed previously – people strive to be economical and efficient in their language use. Clark (1996) calls this the *principle of least effort*:

> All things being equal, agents try to minimize their effort in doing what they intend to do. (p224)

Thus, the third important factor for choosing what evidence to provide is the cost of actually providing the evidence and the possible reactions to the evidence.

Since miscommunication may occur on different levels of actions, evidence may also be given on these different levels. For example, the utterance "I heard what you said, but I don't understand" is an explicit way of giving positive evidence on the perception level, but negative evidence on the understanding level. When positive evidence is given on one level, all the levels below it are considered complete. Clark (1996) calls this the *principle of downward evidence*.

3.1.2.3 The requirement of positive evidence

As pointed out by Traum (1994), there is a problem with Clark's strong requirement of positive evidence. Since an acceptance utterance also can be regarded as a presentation (of some evidence) and all contributions require positive evidence, not just lack of negative evidence, the acceptance should require another acceptance (with positive evidence), and so on ad infinitum. Clark's solution to this problem is that each piece of evidence provided by one speaker in turn requires less evidence from the other speaker, so that the need for evidence eventually fades out. However, it is not entirely clear when, why and how the requirement for positive evidence disappears.

This problem is due to the explicit requirement that each contribution needs some sort of positive evidence:

> What distinguishes this model is the requirement of positive evidence. In traditional accounts, Roger could assume that Nina understood him unless there was evidence to the contrary. (Clark, 1996, p. 228)

But there are a number of communicative situations when we clearly do not require positive evidence. For example, a lecturer does not need continuous positive evidence from all hearers to assume that they are listening. We may also send an email without requiring positive evidence that it is received and read. In these cases, we may instead monitor that we do not get negative evidence (such as someone in the audience falling asleep or an error message from the mail server). In other cases, we do indeed require positive evidence. This, of course, depends on the grounding criterion, as discussed previously. If lack of negative evidence may be sufficient in these situations, why would it never be sufficient in spoken dialogue? Clark states that every contribution needs positive evidence, but it is quite unclear what is meant by a contribution. Is it the whole communicative act? Or is each semantic concept a contribution? Example

(23) above illustrates that there are certainly pieces of information for which a speaker does not require positive evidence.

As indicated in the quote above, the reason that Clark puts this strong constraint into his model is to distinguish the account from the naive view that speakers always assume understanding as long as there is no negative evidence. However, there is a middle way – we could assume that people sometimes require positive evidence and sometimes just lack of negative evidence, depending on the grounding criterion. If speaker A presents some signal, he may require positive evidence of some strength (such as a display of understanding). When this evidence is given by B, the participants may determine that the signal has been grounded sufficiently, unless A gives some sort of negative evidence in return. If the grounding criterion would have been even higher, further positive evidence may have been required. It is also important to remember that once a piece of information has been considered as being grounded, there may also an option to go back and repair it later on if it turns out to be wrong.

3.1.3 Repair and recovery

Negative evidence may be given when some sort of miscommunication has occurred. If speaker B has a problem hearing, understanding or accepting a contribution from speaker A (i.e., some sort of non-understanding), speaker B may give negative evidence of understanding:

(24) A: I have a blue building on my left.
B: *What did you say?*

On the other hand, if speaker B accepts the contribution and gives some sort of positive evidence, this evidence may tell speaker A that a misunderstanding has occurred (for example if B misheard the utterance). Speaker A may then initiate a repair:

(25) A: I have a blue building on my left.
B: *How many storeys does <u>the brown building</u> have?*
A: I said a **blue** building!

Schegloff (1992) calls this latter repair type *third-turn repair*, which indicates that the error is detected and initiated in the third turn, counting from the source of the problem. This notion may also be extended to *first-turn repair*, *second-turn repair*, and *fourth-turn repair* (McRoy & Hirst, 1995). First-turn repair is the same thing as self-corrections, that is, a kind of disfluency (see 2.2.4). Second-turn repair means that the detection occurs and the repair is initiated in the second turn, as in example (24).

Hirst et al. (1994) provide a more general way of analysing the cause for repair:

> Participants in a conversation rely in part on their expectations to determine whether they have understood each other. If a participant does not notice anything unusual, she may assume that the conversation is proceeding smoothly. But if she hears some-

thing that seems inconsistent with her expectations, she may hypothesize that there has been a misunderstanding, either by herself or the other, and produce a repair - an utterance that attempts to correct the problem. (p.223)

Thus, not only direct evidence of understanding, but inconsistencies in general, may act as sources for detecting errors. This may lead to error detection and repair at later stages in the dialogue and give rise to for example fourth-turn repair:

(26) A: I am on Blackberry Street.
B: *Take to the left.*
A: Ok, now I am on Cranberry Street.
B: *Weren't you on Blueberry Street before you turned?*

"Repair", in this context, means that the speakers try to identify and remove (or correct) an erroneous assumption which is caused by a misunderstanding. In the case of non-understanding, the speakers are not trying to repair an erroneous assumption, but instead recover understanding. In this thesis, the terms *misunderstanding repair* and *non-understanding recovery* will therefore be used, which correspond to third-turn and second-turn repair, respectively.

The same factors that influence the choice of positive evidence (uncertainty, cost of task failure, and cost of providing evidence) apply, of course, to the choice of negative evidence. In other words, they apply to the choice of grounding behaviour in general.

3.1.4 Clarification

When a non-understanding recovery (or second-turn repair) is initiated with a request after partial or full non-understanding, it is often called a *clarification request*. If the clarification is due to a lack of hypotheses, the clarification can be initiated with a *request for repetition* (formed as a wh-request). If the clarification is due to a lack of confidence, it can be initiated with a *request for confirmation* (formed as y/n-request). We can also make a distinction between partial and complete clarification requests, that is, whether they concern parts of the previous utterance (concept-level clarification) or the complete previous utterance (utterance-level clarification). Examples of combinations of these are provided in Table 3.1.

Table 3.1: Categorisation of clarification requests, depending on whether they concern the complete previous utterance or parts of it, and whether they express a request for confirmation or repetition.

Scope	**Request**	**Example**
Partial	Confirm	Did you say *red*?
Partial	Repeat	What colour did you say?
Complete	Confirm	Did you say that you have a red building on your left?
Complete	Repeat	What did you say?

While a clarification request always gives some sort of negative evidence, it may also give positive evidence at the same time, concerning other parts of the utterance:

(27) A: I have a red building on my left.
B: *Did you say that the building was red?*

Clarification requests may (as other CA's) be classified based on their *form* and *function*. Purver (2004) presents a study on the different forms of clarification requests that occur in the British National Corpus. The different forms that were identified and their distributions are presented in Table 3.2.

Table 3.2: The first two columns show the distribution of different clarification forms in the British National Corpus according to Purver (2004). This is complemented with examples, as well as a mapping to the categories presented in Table 3.1.

Form	**Distr.**	**Example**	**Scope**	**Request**
Non-reprise clarifications	11.7 %	*What did you say?*	Complete	Repeat
Reprise sentences	6.7 %	*Do you have a red building on your left?*	Complete	Confirm
WH-substituted reprise sentences	3.6 %	*What can you see on your left?*	Partial	Repeat
Reprise sluices	12.9 %	*A red what?*	Partial	Repeat
Reprise fragments	29.0 %	*Red?*	Partial	Confirm
Gaps	0.5 %	*A red ...?*	Partial	Repeat
Gap fillers	4.1 %	*A: I see a red... B: building?*	Partial	Confirm
Conventional	31.1 %	*Huh? Pardon?*	Complete	Repeat
Other	0.5 %			

Different approaches have been taken to classify the functions, or readings, of clarification requests. Ginzburg & Cooper (2001) make a distinction between the *constituent* and the *clausal* reading. The following example, with paraphrases, illustrates the difference:

(28) A: Did Bo leave?
B: *Bo?*
clausal: Are you asking whether Bo left?
constituent: Who's Bo?

The clausal reading can, more generally, be understood as "Are you asking/asserting P?", or "For which X are you asking/asserting that P(X)?" and the constituent reading as "What/who is X?" or "What/who do you mean by X?". Purver et al. (2001) adds the *lexical* reading to this list, which could be paraphrased as "Did you utter X?" or "What did you utter?", that is, an attempt to identify or confirm a word in the source utterance, rather than a part of the semantic content of the utterance (as in the clausal reading).

As pointed out by Schlangen (2004), these different readings can be mapped to the different levels of action (as described in 3.1.1). Such a mapping is shown in Table 3.3, where the understanding level has been split into two levels.

Table 3.3: Mapping between the readings identified by Purver et al. (2001) and levels of action, loosely based on Schlangen (2004). The rightmost column shows the distribution in the British National Corpus according to Purver (2004).

Level		**Reading**	**Distr.**
Understanding	Understanding the meaning of fragmentary utterances. Mapping from discourse entities to referents.	constituent	14.4 %
	Understanding syntax, semantics and speech act.	clausal	47.1 %
Perception	Hearing the words that were spoken.	lexical	34.7 %
		other	3.9 %

It is also possible to imagine clarification on the acceptance level. Take the following example:

(29) A: I think we should paint the house pink.
B: *Pink?*

We could make a reading of this where B means "Pink?, that's an ugly colour I would never consider." In this case, B has no problem with hearing what was said, nor understanding what A means by "pink", he just has a problem accepting this. However, as discussed in 3.1.1, this should perhaps not be regarded as a case of miscommunication.

By the rules of upward completion and downward evidence, a clarification on one level (i.e., negative evidence) also provides positive evidence on the levels below it. For example, if B says (or implies) "Who's Bo?" in a clarification request, A gets positive evidence that B has perceived the words and understood the speech act, but negative evidence about B's abilities to find a referent to the entity "Bo".

3.2 Errors in spoken dialogue systems

Mostly due to the error prone speech recognition process, a dialogue system can never know for certain what the user is saying, it can only make *hypotheses*. Thus, it must be able to deal with *uncertainty* and *errors*. Before discussing error handling in spoken dialogue systems, we will discuss the concept of error in the context of human-human and human-computer dialogue.

3.2.1 What is an error?

In the psychological ("human factors") tradition, errors by humans have been defined in the following way:

> a generic term to encompass all those occasions in which a planned sequence of mental or physical activities fails to achieve its intended outcome, and when these failures cannot be attributed to the intervention of some chance agency. (Reason, 1990, p. 9).

In this tradition, a distinction is made between slips (unintentional action) and mistakes (intentional but mistaken action). The expression "slip of the tongue" suggests that the term "error" also may be applied to speech, and that humans indeed make errors when engaging in a dialogue. In this view, an ambiguous referring expression or a self-correction may fail to "achieve its intended outcome" or at least make the communication less efficient than the speaker ideally would wish. However, there is a problem with the concept of "error" in spoken dialogue between humans. As Clark (1996) points out, it is not at all obvious who has actually made the mistake when miscommunication occurs. Is it the speaker for his muddy pronunciation, or the listener for not listening closely enough? As discussed previously, speakers always try to cooperatively balance efficiency against risk. Thus, it may be inadequate to consider misunderstandings as mistakes – they may be part of an agreed compromise.

From a system design perspective, however, an error can be defined as a deviation from an *expected output*. From this perspective, it may be argued that it is only the system that makes errors. Human self-corrections, for instance, are not errors but just another type of input that the system should be built to handle.

The problem is that e*xpected output* is not trivial in this context. In the case of a sorting algorithm, where the input is a list of entities with some associated numeric values, the expected output can be mathematically defined. However, in the case of input such as human speech, the expected output, from for example a speech recogniser, is not possible to define in such a way. First, the mapping from speech to words is something that humans have established in informal contracts with each other. Second, the amount of information carried by the audio signal is vast and filled with noise. Third, the mapping is often ambiguous and dependent on how much context is considered. For example, if an utterance sounds like /wʌn tuː tiː/, it is not obvious what the expected output from a speech recogniser for the third word should be. Heard in isolation, it sounds like "tea", but interpreted in context, "three" is probably a better guess (maybe pronounced by someone with a foreign accent). We would probably want to ask the speaker what was actually intended. However, this person may not be available or he might not remember or be able to consciously reflect over what was actually meant. Expected output for such input is therefore often defined as what a human observer would make of the task at hand. Such a metric is problematic for several reasons, including that humans will differ in their judgement[5] and that the given input and output must be humanly comprehensible. It also leaves no room for the possibility that an automatic process may perform better than a human. Still, the metric is often used, and speech recognisers are commonly measured against a human-made gold standard.

Another way of defining expected output for a system is to relate it to usability. If a spoken dialogue system is designed to meet some human need, then it meets expectations if its users are satisfied; otherwise, it does not. A problem here is that although this is applicable to a dialogue system as a whole, it is considerably harder to relate to the different sub-processes in the system, although attempts have been made (e.g., Walker et al., 2000a). Expectation based on usability is the one that most closely relates to the over-all goal of a dialogue system, but comparing with human performance may be easier to evaluate, especially for sub-processes.

3.2.2 Under- and over-generation

Given an expected output of a process, two types of errors may be distinguished: *under-generation* and *over-generation*. Errors, then, would occur when the process fails to produce some of the expected output, or adds unexpected output, or a combination of both. For ASR, the terms *deletions* and *insertions* are often used for these kinds of errors. A combination of an insertion and a deletion (at the same point in the output) is called a *substitution*. An example is shown in Table 3.4.

Table 3.4: Example of a deletion (DEL), insertion (INS) and a substitution (SUB).

Spoken	*I*	*have*	*a*	*large*		*building*	*on*	*my*	*left*
Recognised	I	have		large	blue	building	on	my	right
			DEL		**INS**				**SUB**

As a measure of the quantity of errors, the *word error rate* (WER) is often used. It is computed by dividing the sum of all insertions, deletions and substitutions (possibly weighting these differently) with the number of words in the original utterance. Correspondingly, *concept error rate* (CER) is used for measuring the quantity of errors on the semantic level, after the utterance has been interpreted.

A process may have a tendency or be tweaked towards over-generation or under-generation. For example, an ASR under-generates if its confidence threshold is set high, and over-generates if it is set low. A rigid parser is likely to under-generate interpretations (by rejecting input that is partially flawed) and a key word spotter may over-generate (by assigning interpretations to any semantically rich word). Under- and over-generation may well occur simultaneously, but increasing one tends to decrease the other. For categorisation tasks, over-generation

[5] Lippmann (1997) reports a 4% transcription error rate for spontaneous conversations recorded over the telephone.

results in lower precision and higher recall, whereas under-generation results in the opposite. These error types result in the two types of miscommunication discussed in 3.1.1; over-generation in misunderstanding and under-generation in non-understanding.

In many classification tasks, the aim is an equal ratio of error types. In spoken dialogue systems, this may not always be optimal, since the two error types have different effects on the dialogue: non-understanding leads to more repetitions and slower progress, while misunderstanding leads to unexpected responses from the system or to wrong actions (task failure) and erroneous assumptions that may be hard to repair. These different consequences are very important to bear in mind when it comes to error handling, and we will return to this issue later on.

The characterisation of over-generation and under-generation above is summarised in Table 3.5.

Table 3.5: Two basic types of error and relating concepts.

	Over-generation	**Under-generation**
Categorisation	Low precision	Low recall
ASR error	Insertions	Deletions
Miscommunication	Misunderstanding	Non-understanding
Consequence	Task failure	Repetitions

3.2.3 Sources of uncertainty and errors

A common observation is that the speech recognition process is the main source of errors in spoken dialogue systems (e.g., Bousquet-Vernhettes et al., 2003; Bohus, 2007). The reason for this is that the input to the ASR exhibits a very large amount of variability. First, there is of course variability between speakers due to factors such as age, gender, anatomy and dialects. Factors such as speaking rate, stress, and health conditions may also vary within the same speaker. Add to this the variability in the channel, such as background noise and microphone properties, and the result is a very large spectrum of different ways the same text may be realised in the waveform that the ASR is supposed to decode. It is of course not possible to model all this variability, nor has the ASR access to all the knowledge sources that a human listener has, such as semantic relations, discourse history (beyond the current utterance) and properties of the domain. Another problem is that the vocabulary and language models used by the ASR never can cover all the things that users may say, which results in *out-of-vocabulary* (OOV) and *out-of-grammar* (OOG) problems with unpredictable results. Given its limited models, the ASR can only choose the hypothesis that is most likely.

It is important to distinguish these kinds of errors from “bugs” or “exceptions” that need error handling (or “exception handling”) in all computer systems. The source of such errors can, as soon as they are identified, be fixed (more or less easily). However, speech recognition errors cannot typically be “fixed” in a similar way. A distinction can be made here between *variable* and *constant* errors (Reason, 1990). The difference is metaphorically illustrated in Figure 3.1. Target A illustrates large variable errors, but small constant errors, that is, all shots are centred around the middle but with deviations that could be characterised as noise. There is no straightforward way of solving these errors; the sight seemed to be aligned as well as possible, but the rifleman needs more training. Target B, on the other hand, shows small variable errors, but a large constant error. Once the problem is identified (probably a misaligned sight), the error may be fixed. This doesn’t mean that constant errors always give rise to similar behaviours that are easy to discover. For example, if a computer always relied on the same sorting algorithm that always failed to consider the two last elements, this would give rise to a large number of different error forms. Nevertheless, it would be a constant error that could easily be remedied as soon as it was found.

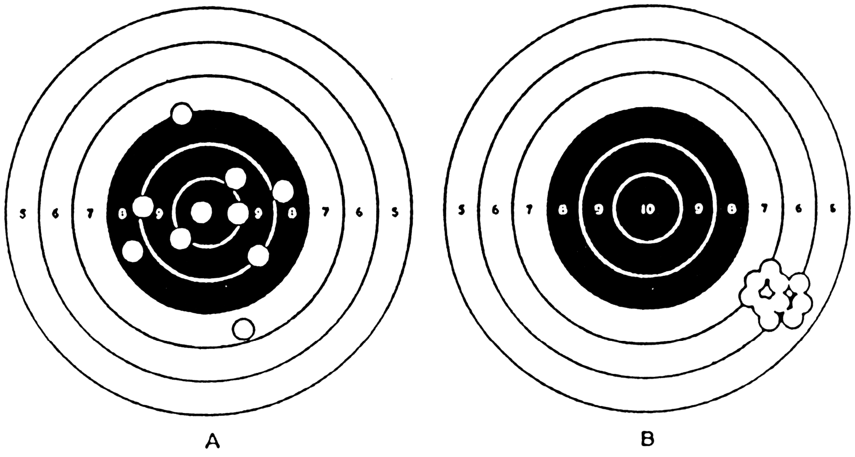


Figure 3.1: Two different target patterns. A exemplifies variable errors and B constant errors. (from Reason (1990), originally from Chapanis (1951)).

The acoustic and language models in the speech recogniser may be improved as more data is collected, and the variable error may be reduced, however probably never completely eliminated, at least not if other knowledge sources are not added to the process.

It should be noted that speech recognition may exhibit constant errors as well, that may be easily fixed once they are found. For example, a word may be incorrectly transcribed in the dictionary.

Another task that is commonly assigned to the ASR is voice activity detection (VAD). This may also be a significant source of errors, for example if the system incorrectly determines that the user has finished his turn, prepares what to say next and then starts to speak at the same time the user completes his turn.

There are of course sources of errors other than the ASR, such as NLU and dialogue management. However, the input to these processes is typically constrained by the language models used in the ASR and therefore exhibits less variability. The main challenge for these components is *error awareness* and *robust processing*, that is, to expect errors in the input and be able to do as much processing as possible despite these errors, with a performance that degrades gracefully. This leads to an error definition problem: given a partly erroneous result from the ASR, what is the expected output from these post-processes? Ideally, we would want such a process to repair the errors made by the ASR and return a result that fits the intentions of the speaker, in other words, to recover deletions and ignore insertions. However, if the number of errors is very large, this may be an unrealistic expectation. Again, it may be useful to compare with what a human could make of the task.

Given a correct result from the ASR, other processes may still make errors. Variable errors may arise in the NLU due to lexical and syntactic ambiguity and in the dialogue manager due to ambiguous elliptical and anaphoric expressions. This may lead to errors at the different levels of action discussed previously. The output processes in the dialogue system may also make errors, for example by using ambiguous referring expressions, so that the user misunderstands the system.

3.3 Error handling in spoken dialogue systems

Variable errors, due to limitations in the system's models, are inevitable in a spoken dialogue system. Even as the coverage of these models is improved, speakers (and developers of dialogue systems) will try to make the interaction more efficient by taking risks and introducing more ambiguity and uncertainty, at least in a conversational dialogue system. That said, there are ways to prevent, detect and repair errors, or minimise their negative consequences. Errors introduced in one process should not make further processing impossible – the processes should be *robust*. But errors introduced in one process may also be repaired in other processes, so that the output of the system as a whole meets the expectations. How is this possible? If we know how to repair an error in another process, why cannot the error be repaired or avoided in the process where it is introduced? There are three answers to this question. First, another process may utilise different knowledge sources which are not available in the first process. For example, the dialogue manager may have access to the dialogue history and domain knowledge which the speech recogniser doesn't have. This is true as long as we do not know how to integrate all processes into one process. Second, if we view the system and user as a joint unit, the user may be involved in the error handling process by grounding. A third, and more practical, answer is that a dialogue system developer working with a set of processes may not have knowledge or access to make the necessary modifications to fix even constant errors in the process in which they are introduced.

Error handling in a spoken dialogue system should not be seen as a single process in the system, but rather as a set of *issues* that should be regarded in all processes. The following human-computer dialogue example illustrates some error handling issues:

(30) U.1: I can see a brown building.
I CAN SEE A BLUE BUILDING
S.2: *A blue building, ok, can you see something else?*
U.3: No, a brown building.
NO A BROWN BUILDING

In this dialogue fragment, we can identify three main error handling issues which are related to the three turns. First, utterance U.1 will be recognised and interpreted, and the example illustrates an ASR substitution. If we consider the ASR to be the main source of errors, we would like to have some sort of technique for detecting potential errors in the ASR output, or in a robust interpretation of the ASR output. We call this *early error detection*. This could result in the system accepting (parts of) the hypothesis of what the user has said or rejecting it. But it could also result in an uncertainty of whether the hypothesis is correct or not. Just as humans do when faced with such uncertainty, the system may initiate a *grounding* process, which is done in S.2. In this example, the system is uncertain about the colour of the building and therefore displays its understanding ("a blue building"), as part of the next turn. This makes it possible for the user to identify the error and repair it (U.3). From the system's perspective, it must now identify and repair this error based on its understanding of U.3. Since the error was already made in U.1, but detected after U.3, we call this *late error detection*.

In the rest of this chapter, the problems involved and the research done on managing these issues are laid out.

3.3.1 Early error detection

The first important error handling issue to consider is how errors introduced in the recognition and interpretation of the user's utterance may be detected. If the recognition is poor, the ASR may give no hypothesis at all, which will inevitably result in a non-understanding. However, it is more common that the ASR will produce a result containing errors. The system must then understand which parts are incorrect and *decide* that it should be considered a (partial) non-understanding. In other words, the system must be able to *understand that it does not understand*. If this early error detection fails, it will result in a misunderstanding (which may perhaps be identified later on in late error detection).

Early error detection can be described as the task of deciding which ASR results, which words in the ASR results, or which semantic concepts in the interpretation should be considered as being correct (i.e., binary decisions), but it could also result in a set of continuous confidence scores, so that other processes may take other issues into account when making the decision. Early error detection is sometimes referred to as *recognition performance prediction* (Litman et al., 2000; Gabsdil & Lemon, 2004) or *confidence annotation* (Bohus & Rudnicky, 2002).

Most error detection techniques rely (partly) on the ASR confidence score, and we will start with a brief review of how this score is typically estimated.

3.3.1.1 ASR confidence score estimation

An ASR may be able estimate a confidence score for the whole utterance, but also for the individual words in it. The score is typically a continuous value between 0 and 1, where 0 means low confidence and 1 high confidence. If the score is only to be used for discriminating between the labels incorrect and correct – by setting a threshold for reject/accept – the only important factor for the quality of the score is how accurate such a classification can be made based on it. The standard metric used to asses the quality of a confidence scoring is the normalised cross entropy (NCE), which is an information theoretic measure of how much additional information the scores provide over the majority class baseline (i.e., assigning all words with the same (optimal) score). However, for other purposes, it could also be desirable to have a *probabilistic* score, that is, a confidence score of 0.3 would mean that there is a 30% probability that the hypothesis is correct.

According to Jiang (2005), methods for computing confidence scores in speech recognition can be roughly classified into three major categories: *predictor features*, *posterior probability* and *utterance verification*. The first approach is to collect predictor features from the recognition process, such as the n-best list, acoustic stability and language models, and then combine these in a certain way to generate a single score to indicate correctness of the recognition decision (see for example Hazen et al., 2002).

The second approach is to use the posterior probability (equation (10) on page 20) directly, which would constitute a probabilistic confidence score. However, there is a fundamental problem with this (Wessel et al., 2001). As shown in equation (11) and equation (12), the probability of the acoustic observation, *P(O),* is typically excluded from the model, since it is not needed to calculate the *relative likeliness* and choose the *most likely* hypothesis. Thus, the remaining formula, $P(O|W)P(W)$, does not describe the *absolute probability* of the hypothesis. It does not account for the fact that as the probability of the acoustic observation increases, it becomes more likely that the hypothesis is generated by something else, and the probability of the hypothesis should decrease. Methods for approximating *P(O)* have been proposed, such as using a phoneme recogniser, filler models, or deducing it from the word graph (Wessel et al., 2001).

In the third approach, utterance verification, confidence scoring is formulated as statistical hypothesis testing similar to speaker verification, using likelihood ratio testing,

3.3.1.2 Rejection threshold optimisation

Early error detection can, in the simplest case, be regarded as a choice between *reject* and *accept* by comparing the ASR confidence score against a *threshold.* If the score is above this threshold, the hypothesis is accepted, otherwise it is rejected. This threshold may be set to some default value (such as 0.3), however the performance can typically be optimised if data is collected for the specific application and analysed. Such an optimisation is shown in Figure 3.2. The lower the threshold, the greater the of number of false acceptances (i.e., over-generation). As the threshold is increased, false acceptances will be fewer, but more false rejections (i.e., under-generation) will occur. Such a graph may be used to find the optimal threshold with the lowest total number of false acceptances and false rejections (approximately 0.42 in the example).

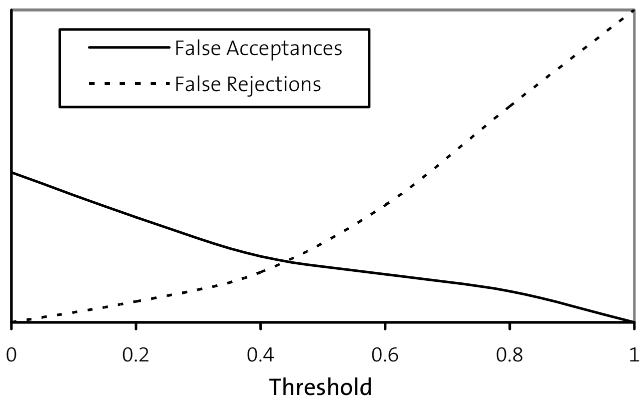


Figure 3.2: Rejection threshold optimisation.

One should bear in mind that this is only true as long as false acceptances and false rejections have the same cost – an assumption that will be questioned later on.

3.3.1.3 Other knowledge sources

To improve early error detection, machine learning has been used in many studies. A corpus of recognised utterances from the application is typically collected and annotated, and supervised learning is used to classify hypotheses as correct or incorrect, based on features from other sources than the ASR. A simple heuristic (such as accepting all hypotheses) is often used as a baseline to compare with.

An obvious argument against early error detection as a post-processing step on the ASR output is that the problems that these techniques attempt to fix should be addressed directly in the ASR. However, as argued in Ringger & Allen (1997), post-processing may consider constant errors in the language and acoustic models, which arise from mismatched training and usage conditions. It is not always easy to find and correct the actual problems in the models and a post-processing algorithm may help to pinpoint them. Post-processing may also include factors that were not considered by the speech recogniser, such as prosody, semantics and dialogue history.

Prosody is a strong candidate feature for early error detection, since people tend to hyperarticulate when they are correcting the system, which often leads to poor speech recognition performance (Oviatt et al., 1996; Levow, 1998; Bell & Gustafson, 1999). Speech recognition can also be sensitive to speaker-specific characteristics (such as gender and age), which may be reflected in prosodic features. Litman et al. (2000) examine the use of prosodic features for early error detection, namely maximum and minimum F_0 and RMS values, the total duration of the utterance, the length of the pause preceding the turn, the speaking rate and the amount of silence within the turn. A machine-learning algorithm called RIPPER (Cohen,

1995) was used. The task was to decide if a given ASR result had a word error rate (WER) greater than zero or not. Using only the ASR confidence score gave a better result than the baseline (guessing that all results were correct). However, adding the prosodic features increased the accuracy significantly. The accuracy was increased further by adding contextual features, such as information about which grammar was used in the recognition.

Other knowledge sources, not considered by the ASR, which should improve error detection are features from the NLU and dialogue manager. In Walker et al. (2000b), the usefulness of such features is studied, using data from the "How May I Help You" call centre-application. 43 different features were used, all taken from the log, which means that they could have been extracted online. The NLU and dialogue manager related features included parsing confidence, grammar coverage, and preceding system prompt. The RIPPER algorithm was used in this study also, but the task was in this case to decide if the semantic label assigned to the utterance was correct or not (i.e., early error detection was performed after interpretation). Again, using the ASR confidence score alone was better than baseline, but adding the other features improved the performance significantly.

The methods discussed above (except the raw ASR confidence score) are all based on binary decisions between correct/incorrect. This is useful if the only choice is between rejecting and accepting the hypothesis, but if other factors are to be taken into account or other options are to be considered (as will be discussed later on), a continuous (possibly probabilistic) confidence score would be more useful as a result of the early error detection. Bohus & Rudnicky (2002) investigated the use of different machine learning approaches to confidence estimation based on a number of features from the ASR, the NLU and the dialogue manager, and found that logistic regression gave the best result.

The common approach to early error detection, as the review above indicates, is to train the classifier on an annotated pre-recorded corpus. Bohus & Rudnicky (2007) present an alternative approach, where the system collects online data from clarification requests. The user's response to a clarification request indicates whether the hypothesis was correct or not. This way, training material may be collected without having a human annotating it. Thus, the system can be said to learn by its own experience. The data collected will contain more noise than manually annotated data, since users do not always act as intended after clarification requests, and their responses sometimes are misrecognised by the system. However, the study shows that the achieved confidence estimation performance is nearly (but not quite) as good as the one that is achieved with manual annotation.

In the studies presented above, whole utterances are considered. This may be useful for shorter utterances with more simple semantics. However, if utterances are longer and contain more complex semantics, it may be useful to consider individual words or concepts for early error detection. In Chapter 5, such a study is presented.

3.3.1.4 Error correction and n-best lists

Another possibility in the post-processing of the ASR result is to not only *detect* errors, but to also *correct* them, in other words not just delete insertions, but also re-insert deletions. An obvious source for such corrections is the n-best list or word lattice typically provided by the ASR, as described in 2.3.1.

In order to explore the upper limit of such an approach, Brill et al. (1998) conducted an experiment in which subjects were given the task of choosing the most likely hypothesis from 10-best lists, but were also asked to manually edit and correct the best hypothesis to improve it further if they thought it was possible. The data was from switchboard, broadcast news and Wall-street journal. The results showed that the subjects were able to improve WER by 1.3-3.1 percent units by just selecting the best hypothesis, and by 2-4 percent units by further corrections.

Examples of studies on automatic reordering of n-best lists include Rayner et al. (1994), Chotimongkol & Rudnicky (2001), Gabsdil & Lemon (2004) and Jonson (2006), which all show how the system sometimes can choose better hypotheses if knowledge sources other than those used by the ASR are considered. In short, n-best list reordering is often done by applying some sort of early error detection technique (as discussed above) to several hypotheses and then picking the one that achieves the best score.

There are some potential problems involved in processing n-best lists. First, it may be computationally challenging to consider many possible alternatives simultaneously, especially if other knowledge sources in the form of other dialogue system components, such as parsing, are to be involved. This is especially true if the dialogue system should operate incrementally, on a word by word basis. Second, as discussed in 2.3.1, n-best lists may be less fruitful to consider when utterances are longer, since many of the top hypotheses will be very similar, with perhaps just some single function words varying in a long list of semantically similar combinations. Thus, n-best lists may be more useful in command-based dialogue systems where utterances may be shorter and incrementality is not an issue.

For conversational dialogue systems, it may be more useful to explore the use of word lattices. However, this may require a more sophisticated approach than just applying an early error detection technique on a list of hypotheses. That is beyond the scope of this thesis.

3.3.1.5 Error prediction

One interesting error handling strategy is to detect the problem before it has even occurred – in other words, to *predict* errors. This way, the dialogue system could adapt its behaviour to avoid the problem. Walker et al. (2000c) report an experiment where the initial segments of a dialogue were used for error prediction. The dialogue system was the "How May I Help You" call centre-application. All dialogues were classified as "task success" or "problematic". The RIPPER machine-learning algorithm (Cohen, 1995) was trained to classify the dialogues based on online features from the ASR, NLU and discourse. By just looking at the first turn, the performance (72.3%) was significantly better than majority class baseline (64%, tagging everything as "task success"), although the improvement is not huge. By also looking at the second turn, the improvement was better (79.8%). Although 16% above baseline, just given the two first exchanges and online features, sounds impressive for predicting problematic dialogues, the question is whether 80% is good enough to be able to take appropriate actions.

3.3.2 Grounding in dialogue systems

In the simplest case, error detection leads to the choice of reject or accept. However, there are other alternatives. As discussed in 3.1.2 above, humans detect and correct errors together by giving evidence of understanding in a grounding process. If the system is uncertain about whether an error is present, it may provide evidence of understanding and detect potential errors based on the user's reaction to this evidence.

3.3.2.1 Explicit and implicit verification

The most well-known and well-tested techniques for grounding are called *explicit* and *implicit verification*. In explicit verification, the system asks a clarification request, typically in the form of a reprise sentence (see Table 3.2 on page 39). The following example is from Bouwman et al. (1999):

(31) U: I'd like to travel to Amsterdam.
S: *Do you want to go to Amsterdam?*
U: Yes. that's right.

In implicit verification, the system instead displays its understanding as in the following example (again from Bouwman et al. (1999)):

(32) U: I'd like to travel to Amsterdam.
S: *To Amsterdam. From where do you want to leave?*

Whereas explicit verification typically requires the user to confirm the hypothesis for the system to consider it as being correct, implicit verification require no such response. As long as the user does not object, the hypothesis is considered to be correct.

3.3.2.2 Other kinds of evidence

As we saw in 3.1.2, clarification in the form of reprise sentence (explicit verification) and display of understanding (implicit verification) are just two kinds of evidence that speakers use in dialogue. Are there other kinds of evidence that may be useful for error handling in spoken dialogue systems? Continued attention and relevant next contribution should perhaps not be regarded as a choice the system can make in order to give positive evidence. Instead, if the system continues with an irrelevant next contribution (due to a misunderstanding), this will probably trigger the user to initiate a repair. However, the system should be capable of understanding such a repair.

Acknowledgements may be used to actively give positive evidence. An acknowledgement like "ok" or "m" may give the user evidence that the system has heard the utterance and processed it on some level of action. However, it cannot be used to actually detect erroneous hypotheses, since the system's understanding of the involved concepts is never signalled to the user.

3.3.2.3 Evidence on different levels of action

As discussed previously, clarification requests may concern different levels of action. In explicit verification, it is normally assumed that the perception level is concerned. However, other levels may be concerned as well, as the following example illustrates:

(33) U: I have the red building on my left.
S(alt. a): *Which red building?*
S(alt. b): *Do you really mean the red building?*

In these examples, the user gets positive evidence that the system has heard and to some extent understood what the user said, but there is some problem in deeper understanding. Such clarification requests could be useful for resolving ambiguous anaphoric expressions (as in the example above), but also for ambiguous fragmentary expressions. The use of clarification on different levels of action is explored in Schlangen (2004) and Rieser (2004).

Larsson (2003) discusses the use of positive and negative evidence on different levels of action. As positive evidence, display of understanding and acknowledgements are considered. The examples on negative evidence are mostly (implicit) requests for repetition:

- Contact: “I didn’t hear anything from you”
- Perception: “I didn’t hear what you said”
- Semantic understanding: “I don’t understand”
- Pragmatic understanding: “I don’t quite understand”
- Acceptance: “Sorry, Paris is not a valid destination city”

In 3.1.1, we questioned whether failure on the acceptance level really should be classified as miscommunication. In the same way, we may question whether “evidence” on the acceptance level, as in the example above, really should be classified as evidence of understanding in the same way as the other levels – it is not caused by any uncertainty or lack of hypotheses. Thus, it should perhaps not be considered as being part of error handling.

3.3.2.4 Non-understanding recovery

Non-understandings may not only be frustrating for the user per se, they may also lead to error-spirals, that is, further non-understandings that may be hard to recover from. For example, Levow (1998) found that the probability of experiencing a recognition error after a correct recognition in a dialogue system was 0.16, but immediately after an incorrect recognition it was 0.44. Thus, if the system decides to reject the user’s last utterance it should take appropriate actions to recover understanding in subsequent turns.

One reason for non-understandings often leading to other non-understandings is that speakers usually repeat the non-understood utterance in the subsequent turn. If the ASR failed to recognise this utterance the first time (possibly because the utterance was out-of-vocabulary or that the user’s pronunciation of the utterance is uncommon), there is an increased risk that it will fail the second time too. Another reason is that people tend to hyperarticulate when making repetitions after non-understanding (Oviatt et al., 1996; Levow, 1998; Bell & Gustafson, 1999), a strategy that is useful when speaking with humans, but may worsen the performance of the ASR. Many speech recognisers lack models for hyperarticulate speech, which makes the understanding of repeated utterances even more difficult.

One approach to this problem is to look at how speech recognition can be improved after non-understanding. For example, Ainsworth & Pratt (1992) investigates how the system can eliminate the misrecognised word from the vocabulary to improve recognition of repetitions.

Another approach is to design the system response after the non-understanding more carefully. A common assumption seems to be that after non-understanding, the system has no option but to request repetition by signalling non-understanding or making a clarification request, just as in the examples on negative evidence on different levels of action listed above. While this mapping between action levels and system responses seems straightforward and intuitive, the usefulness of such signals of non-understanding for handling errors can be questioned, since they encourage repetitions. Neither is it usually fruitful to try to explain or analyse the source of the problem. For example, to use an utterance like "I didn't hear what you said" to signal that the problem is due to the ASR (and not some other processing step), will probably just encourage hyperarticulated repetitions. As Balentine et al. (2001) writes in a style guide for telephony dialogue systems:

> Avoid apologizing for problems or inadvertently blaming the user for them. Instead, simply move forward by prompting for the next appropriate user action. There are two motivations for this. First, the application can never be certain of the underlying error, so descriptions of the problem may be incorrect or misleading. Second, explaining the problem does not necessarily influence the user in a constructive way. Rather than dwelling on the condition that has led to a problem, it is better to describe what action is now expected from the user. (p. 55)

According to Balentine et al. (2001), system responses after non-understandings should encourage the user to try another wording, and provide incrementally more help on subsequent non-understandings.

The problem of non-understanding recovery is explored and discussed in more depth in Chapter 4.

3.3.2.5 Concept-level grounding

The overview of research on clarification requests in 3.1.4 above showed that humans often (in about 45% of the cases) use fragmentary constructions when making clarification requests. To improve efficiency and naturalness, a dialogue system should also be able to utilise fragmentary utterances in grounding. Fragmentary grounding utterances may not only be realised more efficiently, they may also help to pinpoint the problematic parts of the original utterance. However, as Gabsdil (2003) points out, the use of fragmentary grounding in spoken dialogue systems is not very common. The following example illustrates a possible use:

(34) U.1: I have a red building on my left.
S.2 (alt. a): *Red?*
S.2 (alt. b): *Red, ok, what do you have on your right?*

S.2(b) may look similar to the implicit verification "To Amsterdam" in example (29) above. However, there is a difference. "To Amsterdam" is, in the travel booking domain, equivalent to "I want to go to Amsterdam" or "So you want to go to Amsterdam". It does not need to be resolved. The utterance "red" on the other hand could mean many different things in the navigation domain that the example is taken from, and necessarily has to be resolved and placed in a larger semantic construct. Thus, whereas "To Amsterdam" does not help to pinpoint the problematic part of the utterance, the utterance "Red" does.

The use of fragmentary grounding utterances has some interesting challenges that are addressed in Chapter 6 and 9:

- The problematic concepts in the original utterance must be identified.
- The grounding utterance must have the right textual and prosodic realisation to be understood correctly by the user.
- The system must remember for which concepts it has provided evidence.
- The user's reaction to the request must be understood correctly. If the user negates and/or corrects the proposed concept, the system must understand that it is only parts of the original utterance that have been negated and/or corrected, not the entire contribution.

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.

The use of concept-level clarification requests in dialogue systems has received more interest than the use of concept-level display of understanding. Concepts may be displayed as separate communicative acts, as in S.2(b) in example (34) above. But, as discussed in 3.1.2.1, the display of concepts may also be integrated in the next communicative act, with the primary communicative function relating to the task at hand. The following examples with alternative system reactions are examples of this:

(35) U.1: I have a red building on my left.
S.2(alt. a): *How many stories does <u>it</u> have?*
S.2(alt. b): *How many stories does <u>the building</u> have?*
S.2(alt. c): *How many stories does <u>the red building</u> have?*

By choosing between different referring expressions, the system may display its understanding to different extents, depending on its confidence in the concepts involved. To be able to do this, the challenges listed above must be considered. The issue of choosing between these different realisations and modelling an integrated display of understanding will be addressed in Chapter 6.

3.3.2.6 Alternative clarification

There is another type of clarification request that may be referred to as an *alternative* clarification request (Gabsdil, 2003). If the system can consider several alternative hypotheses from the speech recogniser, it could make a clarification request such as this:

(36) U: I have a red building on my left.
S: *Red or blue?*

It could also be possible to make an alternative clarification request on the understanding level, for example if there are several possible ways to resolve an anaphora:

(37) U: I have the building on my left.
S: *The red or the blue one?*

In order to pose alternative clarification requests, the system must somehow be able to produce several parallel hypotheses, for example by n-best lists or word lattices. See 3.3.1.4 for a discussion on the potential problems associated with this. The use of alternative clarification will not be investigated in this thesis.

3.3.2.7 Making grounding decisions

As we have seen, there are several ways to handle uncertainty and errors in dialogue, either towards risking under-generation or over-generation. As Allen et al. (1996) points 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 system may display its understanding, request clarification on what is not understood, presuppose understanding and defer the detection of errors to a later stage in the dialogue, or simply reject the hypothesis. We refer to this choice as the *grounding decision problem*. The grounding decision concerns not only which evidence of understanding to give, but also whether the hypothesis should be regarded as common ground. A common basic approach is to use hand-crafted confidence thresholds as shown in Figure 3.3 (see for example Bouwman et al., 1999).

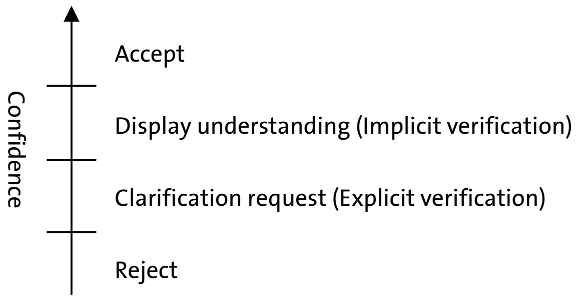

Confidence
Accept
Display understanding (Implicit verification)
Clarification request (Explicit verification)
Reject

Figure 3.3: Typical grounding decision based on a confidence score.

This division seems intuitive, but the problem is how to find optimal thresholds. Another problem is that the thresholds most often are static and not dependent on the dialogue context. Simple threshold optimisation as described in 3.3.1.2 above cannot be applied to this problem.

We will return to this issue in Chapter 8, where a dynamic, data-driven, decision-theoretic model for making grounding decisions is presented and related research is discussed.

3.3.2.8 Evidence of understanding from the user

Most studies on grounding in spoken dialogue systems, including this thesis, are focussed on how to cope with system non-understandings and misunderstanding, in particular those caused by speech recognition errors. As a consequence, the models of grounding proposed are mainly concerned with how the system should provide evidence of understanding based on its hypotheses of the user's utterances. However, it is of course also possible that users may give evidence of understanding. A dialogue system should for example be able to make a repetition after an utterance such as "what did you say?". But as dialogue systems start to utilise more sophisticated methods for providing evidence, we should expect users to do the same. For example, if we endow systems with the capabilities of making fragmentary clarification requests such as "red?" and non-verbal acknowledgements such as "uhu", users are likely to pick up on this. This poses new challenges for the recognition and interpretation of user utterances, including prosodic analysis.

3.3.3 Late error detection and misunderstanding repair

If the system accepts an incorrect hypothesis of the user's communicative acts, it may still be possible to do late error detection at later stages in the dialogue and repair the misunderstanding.

3.3.3.1 Late error detection

Late error detection may typically be performed after the system has displayed its understanding (based on an incorrect hypothesis) and the user initiates a repair:

(38) U.1: I have a blue building on my left.
S.2: *How many stories does the brown building have?*
U.3: I said blue building!

As noted previously, errors may also be detected based on inconsistencies in general, several turns after the actual error occurs. This leads to two issues that should be handled by a dialogue system. First, to facilitate late error detection, the system must be capable of detecting cues from the user in the "third turn" that something is wrong (such as the U.3 in the example above), and to detect inconsistencies in general. Second, to enable repair of misunderstandings, it must know which assumptions to remove or re-evaluate in its model of the common ground.

One problem with detecting errors in the "third turn" is that these problem signals may look very disparate and may depend on subtle prosodic cues, or the user may just ignore the problem. Here are some imaginable variants to U.3 in example (38) above:

(39) U.3a: eeh, brown building?
U.3b: I can't see any brown building.
U.3c: I don't understand.
U.3d: the brown building we talked about before?
U.3e: ehh… I can see a blue building.

Krahmer et al. (2001) calls these signals "go back" cues, as opposed to "go on" cues, which signal that the displayed hypothesis was correct. A number of possible cues are listed in Table 3.6.

Table 3.6: Possible positive and negative cues from the user after the system has displayed its understanding (from Krahmer et al., 2001).

Positive cue	**Negative cue**
Short turns	Long turns
Unmarked word order ("I want to leave from Stockholm")	Marked word order ("It is Stockholm I want to leave from")
Confirm ("Yes")	Disconfirm ("No")
Answer	No answer
No corrections	Corrections
No repetitions	Repetitions
New info	No new info

In an analysis of a hand-labelled corpus based on spoken dialogue systems providing train timetable information, Krahmer et al. (2001) found that users never gave explicit positive cues such as "yes", and rather seldom (in 15.4% of the cases) gave explicit negative cues ("no"), after an implicit verifications (display of understanding). The best determinant is instead whether the user makes any attempts to make a correction or not. It should be noted that this analysis has been performed on what the users actually said, not on the results from the ASR. The question is to what extent these results can be applied to online dialogue, since the ASR for example may miss corrected slots. It should also be noted that given a negative cue, the system has no information on what the problem actually is, just that a problem has occurred.

One approach to this problem is to use machine learning, training on a set of features, similar to early error detection, to distinguish user corrections from non-corrections. Litman et al. (2006) investigate the use of prosodic, ASR-derived, and system-specific features, both for the current turn and for contextual windows, and using summary features of the prior dialogue. An initial analysis showed that the prosody of corrections differ significantly from non-corrections, being higher in pitch, louder, longer, with longer pauses preceding them and less internal silence. As expected, they are misrecognised more frequently than non-corrections. Using machine learning, the best-performing feature set cuts the majority baseline error almost in half, from 29% to 15.7%.

The problem of detecting negative evidence in the third turn has lead many dialogue designers to avoid using display of understanding (implicit verification). A common experience is that users often don't know how to react and feel uncomfortable (see for example Weegels, 2000).

3.3.3.2 Modelling grounding status

As positive or negative evidence is given for a hypothesis and the user reacts to this evidence, we may say that the hypothesis' *grounding status* has been updated. By modelling this grounding status, the system may decide when the grounding criterion has been satisfied, that is, when the hypothesis may be considered to be common ground.

Traum (1994) shows how recursive transition networks (RTN) may be used to track the grounding status. In this account, the semantic units that get grounded are called *discourse units*, and the actions that contribute to the updating of the grounding status of these are called *grounding acts*. Grounding acts may be repairs, requests for repairs, acknowledgements and requests for acknowledgement. The discourse units can be compared to speech acts. Thus, the model can be said to track utterance-level grounding. Grounding actions are also treated as a special kind of communicative act. These two properties are common for most models of grounding status. The problem with such accounts is that they do not explain how for example different kinds of referring expressions in task-related utterances may help to provide evidence of understanding on parts of the previous utterance, as in example (35) above. A third property of many models of grounding status is that the grounding status is only tracked locally within the "subdialogue". This makes it impossible for the system to consider the grounding status later on in the dialogue if inconsistencies that indicate a misunderstanding are found. In Chapter 6, a model that deals with these shortcomings is presented.

Heisterkamp & McGlashan (1996) presents a model in which the grounding status is tracked by assigning a *contextual function* to each information unit.

- new_for_system(X).
- repeated_by_user(X).
- inferred_by_system(X).
- modified_by_user(X).
- negated_by_user(X).

A similar model is used by McTear et al. (2005) under the name of *discourse pegs*.

Another approach to late error detection and grounding status modelling is presented in Bohus & Rudnicky (2005a), where the system models its belief in concepts as a continuous confidence score that gets updated as the concept is grounded. The approach is called *belief updating* and is defined as follows:

> given an initial belief over a concept *Belief*$_t$*(C)*, a system action *SA(C)* and a user response *R*, compute the updated belief *Belief*$_{t+1}$*(C)*.

Or, in terms of confidence score for a single hypothesis:

> given an initial confidence score for the top hypothesis *h* for a concept *C*, construct an updated confidence score for the hypothesis *h*, in light of the system confirmation action *SA(C),* and the follow-up user response *R*.

Thus, the approach is similar to that of using machine learning for confidence estimation in early error detection, but it is extended to also include features from the subsequent CA's in the grounding process. In their approach, data was used to train a binary logistic regression model on features from the original hypothesis as well as the system's action and the follow-up user response. Some of the most useful features were: the initial confidence score, prosodic features, expectation match, barge-in, lexical features, the presence of repeated grammar slots, as well as the identity of the concept to be confirmed.

3.3.3.3 Misunderstanding repair

When a misunderstanding is detected, it should be repaired. To do this, the system must have some mechanisms for removing erroneous hypotheses from the common ground. For example, in Larsson (2002), a "backup" copy of the dialogue state (a "temporary storage") is kept to restore the information state if the system's hypothesis of the common ground turns out to be incorrect. A drawback with this approach is that the detection of the misunderstanding may only occur immediately after the error, in the "third turn", and that the dialogue state is completely restored, which means that individual concepts that were not erroneous are lost.

It is also possible that the system should not simply restore the state when an old error is suspected. The system could also make a late clarification request, as in example (26) on page 38.

A problem with implementing a more elaborate model of late error detection in many dialogue systems is that the result of early error detection (such as confidence scores) are most often only considered once and not stored for late error detection. In Chapter 6, this issue is discussed in more depth, and a model that supports long-term storage of confidence scores and grounding information is presented.

3.4 Summary

This chapter has reviewed the work done on error handling in spoken dialogue systems and laid-out the issues that are involved. Figure 3.4 shows a diagram which illustrates how the most important error handling issues presented above are connected, from the perspective of this thesis.

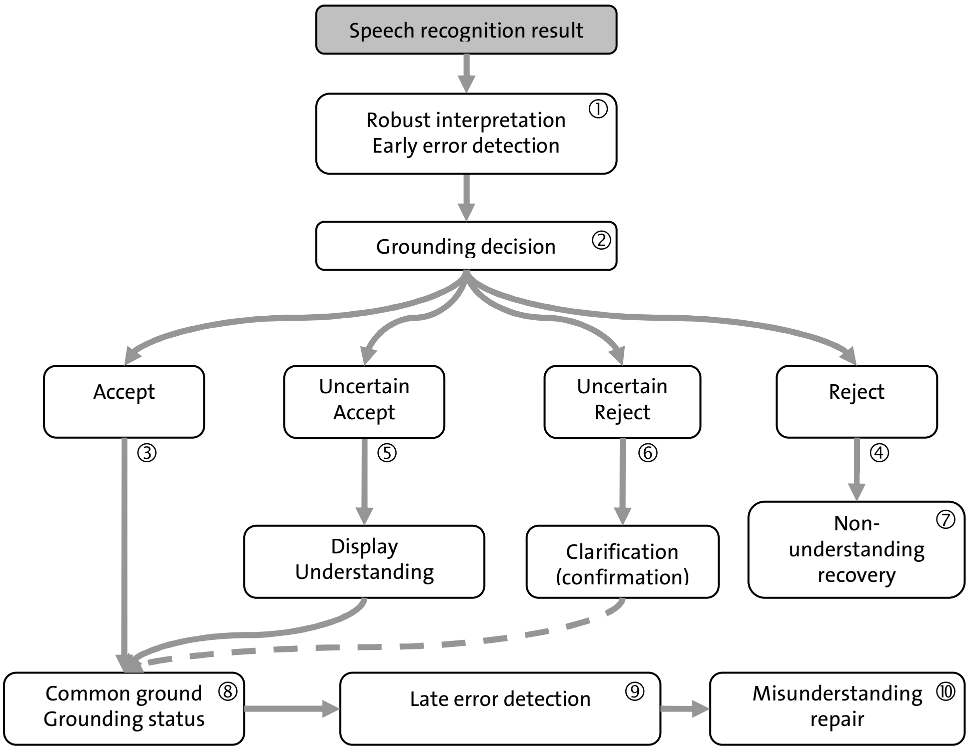

Speech recognition result
Robust interpretation Early error detection ①
Grounding decision ②
Accept
Uncertain Accept
Uncertain Reject
Reject
③
⑤
⑥
④
Display Understanding
Clarification (confirmation)
Non-understanding recovery ⑦
Common ground Grounding status ⑧
Late error detection ⑨
Misunderstanding repair ⑩

Figure 3.4: Overview of the error handling issues considered in this thesis

The diagram can be described as follows. Given a noisy speech recognition result, the system must make a *robust interpretation* of it into a hypothesis of the semantic concepts in the user's utterance ①. The system must also decide which words in the ASR output or which resulting semantic concepts should be considered to be correct, and/or decide the level of uncertainty (*early error detection*) ①. For each concept in the hypothesis, the system must make a *grounding decision* ②:

- The system could simply *accept* the concept and regard it as common ground ③.
- The system could simply *reject* the concept, i.e., treat it as a *non-understanding* ④.
- The system could accept the concept and add it to the common ground but at the same time *display its understanding*, so that the user has a chance to correct the system if the concept is incorrect ⑤.
- The system could reject the concept (i.e., not treat it as common ground), but make a *clarification request*, so that the user may confirm the concept if it is correct. If so, the concept may be treated as common ground ⑥.

If the system rejects the concept, it might perform a *non-understanding recovery* ⑦, which may be some sort of clarification request. If a concept is treated as common ground, the system's uncertainty of the concept should be stored ⑧, so that errors may be detected later on (*late error detection*) ⑨, for example if the system displays its understanding and the user objects. In such a case, a misunderstanding has been detected, which needs to be repaired ⑩. The concept should be removed from common ground, but the user may also be involved in further clarifications.

PART II

Exploring Human Error Handling

A: What's you name?
B: Watt
A: I said, what's your name?
B: Watt is my name
A: What?
B: Yes
A: Yes, what?
B: Watt, yes

Unknown source

Ignorance is preferable to error; and he is less remote from the truth who believes nothing, than he who believes what is wrong.

Thomas Jefferson

CHAPTER 4

Exploring non-understanding recovery

Despite the many sources of uncertainty in spoken communication, humans have a remarkable capability of adapting to difficulties and collaboratively prevent, detect and recover from problems. How can this seemingly smooth handling of uncertainty and miscommunication in human-human dialogue be transferred to human-computer dialogue? In this chapter, an experiment on human-human communication is presented, where the human error recovery strategies employed after miscommunication are explored and analysed.

4.1 Methods for exploring human error handling

One approach to explore human error handling strategies is to look at problems as they occur in human-human dialogue, and transfer this knowledge to human-computer dialogue. There are two potential problems with this approach. First, human-computer dialogue and human-human dialogue have been shown to have different properties (Fraser & Gilbert, 1991). This is partly due to the user's idea of the system not being a human, and partly due to the limitations of the system's conversational capabilities. A second less obvious problem is that the participants' actual understanding of what is said is not always transparent when analysing the dialogue. Just because a speaker does not give any sign of non-understanding does not necessarily mean that every word was understood correctly.

4.1.1 The Wizard-of-Oz method

In order to design dialogue systems that can handle the varieties of situations that occur in human-computer dialogue – such as miscommunication – data of such interaction needs to be collected. A method that is commonly used to collect such data before a dialogue system is actually built is the Wizard-of-Oz method, where an operator (the "wizard") is simulating parts of the system, most often assuming a perfect speech recogniser (Fraser & Gilbert, 1991). The user is not told about this setup and is supposed to believe that she is interacting with a computer. A typical Wizard-of-Oz setting is shown in Figure 4.1.

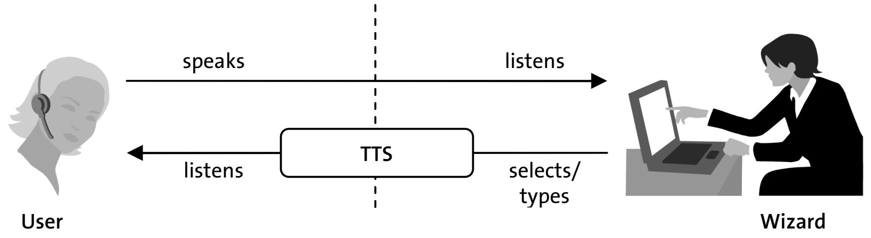


Figure 4.1: A typical Wizard-of-Oz setting.

The advantage compared to studying human-human dialogue is that people will act as if they spoke to a computer. The problem concerning studies of error handling is that it is hard to get an accurate account of what happens when speech recognition errors occur, since they are often ignored when the experiment is conducted. The optimistic assumption tends to be that these things can be added later on when the rest is solved, or that the problem will disappear automatically as speech recognisers get better. One approach to get data on miscommunication is to simulate errors, for example by randomly substituting words in the input. But, as Fraser & Gilbert (1991) points out, this is an almost impossible task in a Wizard-of-Oz environment. First, the wizard is working under time pressure and it may be hard to make the right substitutions while controlling other components. Second, the kind of errors that really do occur are hard to simulate. Just substituting random words may be too simplistic a model. Out-of-vocabulary words will often give rise to unexpected results, as the speech recogniser is trying to fit what has been said into the language model using in-vocabulary words. Another approach considered by Fraser & Gilbert (1991) is using a speech recogniser as a filter between the user and the operator, but they argue that it would be too hard for the operator to read the speech recognition results, and that the poor speech recognition performance would constrain the dialogue too much. Paek (2001) suggests that a speech recogniser could be used in a Wizard-of-Oz setting to establish a gold standard for other components, which are simulated by the operator.

A fundamental assumption behind the Wizard-of-Oz-method is that users are thought to behave differently when talking to a machine compared to talking to a human (Dahlbäck et al., 1993). There are two reasons for this. First, the system has conversational limitations that will constrain the dialogue. Second, the user has a model of the interlocutor that will affect the linguistic constructs used. Many studies that compare human-human and human-wizard conditions (see Fraser & Gilbert, 1991 for an overview) do not make this distinction and these variables are not controlled systematically and independently of one another. Thus, they do not tell us which one of them is important for the differences that appear. However, in an experiment conducted by Amalberti et al. (1993), the effect of the user's conceptions about the other speaker was tested independently of the limitations of the system. Two groups of subjects were asked to obtain information about air travel via spoken dialogue with a remote travel agent. One group was told that they were talking to a computer, while the other was told that they were talking to a human operator. In both cases, the voice of the operator was distorted. The amount of distortion was carefully tuned, so that the human group could be told that they were testing communication through a noisy channel, while the other group were told that they were talking to a computer. Thus, the experimental setting was exactly the same for the two groups, apart from their conceptions about the other speaker. The results showed that there were differences in the users' linguistic behaviour. However, the differences were most noticeable initially, and a lot of differences tended to disappear in subsequent sessions. Furthermore, the differences that were found between the groups were mainly related to problem solving, where the users in the human group were more cooperative towards the operator. This suggests that the experience the user has of a system will affect the user's behaviour in future interactions. If users are faced with more cooperative systems, they may start to take advantage of this. In order to make advances in the development of dialogue systems, it could be dangerous to adapt to users' current beliefs of the capabilities of such systems, especially users who have a very limited experience of them.

This leads us to another problem concerning explorative Wizard-of-Oz experiments. Numerous studies show that the behaviour of a dialogue system has great impact on the user's behaviour (e.g., Brennan, 1996; Gustafson et al., 1997; Skantze, 2002). Thus, the way the wizard acts will influence the data that is collected. This can be a problem, since the collected data might be based on a priori assumptions about the users' behaviour and how a system is supposed to react to them, and might not cover other, unanticipated interaction patterns. Using a speech recogniser in a controlled Wizard-of-Oz setting would also make it hard to prescribe how the operator should behave depending on different levels of comprehensibility.

All in all, it seems as if the Wizard-of-Oz method might be difficult to use for studying error handling strategies, even if a speech recogniser is included in the setting, unless the experimenter has a very clear idea of the different errors that will occur and which specific error handling strategies should be tested. In order to deceive the subject, the wizard must work fast and accurately. This does not only require a good design of the experimental setting and operator environment, but also much training of the operator. Since the method is very costly to perform, it may be unfeasible to use it for conducting more explorative experiments on such strategies.

4.1.2 The proposed method

In the experiment presented in this chapter, two humans were given a task to solve by speaking to each other. To introduce errors, a speech recogniser was used in one direction, so that the listener could only read the speech recognition results. The problems of miscommunication that these errors give rise to, and the effects they have on the dialogue and the subjects' experience of it, have been analysed.

This approach has three advantages. First, the speech recogniser imposes some limitations similar to those of a dialogue system, which makes the dialogue more similar to human-computer dialogue. Second, the kind of errors that occur are probably more similar to those that occur in spoken dialogue systems than those that occur in ordinary human-human dialogue or those that may be simulated in Wizard-of-Oz studies. Third, since the operator's understanding is limited to that of the speech recognition result, the level of understanding is more transparent when analysing the dialogue.

4.2 Experimental method

4.2.1 Experimental setting

The goal of this experiment was to get clues as to how errors may be handled in spoken dialogue systems, not to test specific error handling strategies. The task was chosen to resemble that of a fairly complex spoken dialogue system and the two subjects were given the role of operator (corresponding to the "dialogue system") and user. The operator could not hear what the user said, and had to read the speech recognition result from a screen. A number of different naive operators were used in order to get varied data on error handling strategies. As opposed to the Wizard-of-Oz method, the operator was treated as a subject as well and the users were openly informed about the setting.

As discussed previously, the problem of transferring results from human-human studies to human-computer dialogue has been shown to depend mainly on the limitations that real systems impose on the dialogue. In the current study, the users were told that a speech recogniser was used, and were therefore aware of the fact that complex utterances might not get through. Since speech recognition is regarded as the bottleneck of most complex spoken dialogue systems, the results of this study may be more easily transferred to dialogue systems than results taken from ordinary human-human dialogue.

One thing that does differ between human-human and human-computer dialogue, even when the channel is equally noisy in both cases, is the amount of common ground that the speakers have before engaging in the conversation. To minimise common ground, the operator and the user were not allowed to meet before or during the experiment. However, both subjects were fully informed about the experimental setting. A challenging issue for the setting is how the operator should reply, provided that user should not be allowed to form any assumptions about the operator. One possibility could be to let the operator type a message, synthesize it and play it back to the user, using a text-to-speech system. However, pilot studies showed that this would be too slow, and that the operator might behave in a "lazy" way, not typing the whole message as it would have been spoken. Another solution could be to let the operator choose or compose the answer from a set of templates, but this would restrict the operator's output, and unexpected behaviour would not be captured. The proposed solution is to let the operator speak freely and distort the speech through a vocoder. The final setting is illustrated in Figure 4.2.

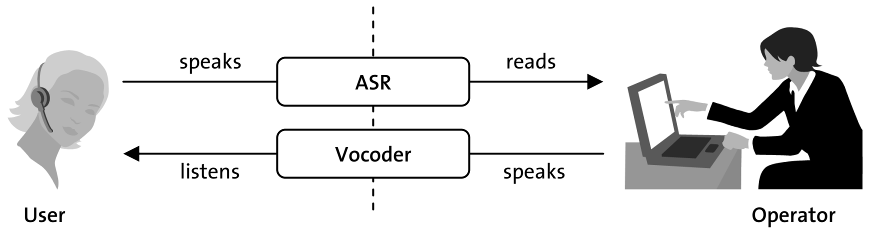


Figure 4.2: The setting used in the experiment.

It should be noted that this experimental setting lacks the control that the consistent behaviour of a trained operator would give. Still, this method may be good for explorative studies, which aim at finding new ideas on dialogue behaviour, and especially on how error situations could be handled.

4.2.2 The dialogue domain

The domain selected for the experiment was pedestrian navigation on a simulated campus. In this domain, the user's goal is to get to a specific location and the dialogue system (in this experiment the operator) is used to get directions. The system (operator) does not know where the user is, and must rely on the user's descriptions of the environment. Dialogue about route descriptions have been studied extensively in so-called Map Task experiments (see Anderson et al. (1991) for a description of a corpus, and Brown (1995) for an extensive analysis). The question is to what extent these data are applicable to dialogue systems for navigation, since the user (the "follower" in Map Task) has access to the whole map and can talk about absolute directions (such as "north", "south", "up" and "down"). For this experiment, a simulation environment, which is described in the next section, was built to prevent the user from using such information. Although miscommunication has been studied in Map Task experiments previously (Brown, 1995; Carletta & Mellish, 1996), such experiments have not, to our best knowledge, been conducted using a speech recogniser.

4.2.3 Experimental design

4.2.3.1 Subjects

16 subjects were used, 8 users and 8 operators, ranging in age from 16 to 42 years. All subjects were native speakers of Swedish. The subjects were paired in groups of operator/user. There were 8 women and 8 men, equally balanced as operators and users. Users with low to medium computer experience were chosen (to represent ordinary users), while the operators were chosen with a somewhat higher computer experience and some experience of speech technology, on the assumption that this would make the learning of the operator interface faster. However,

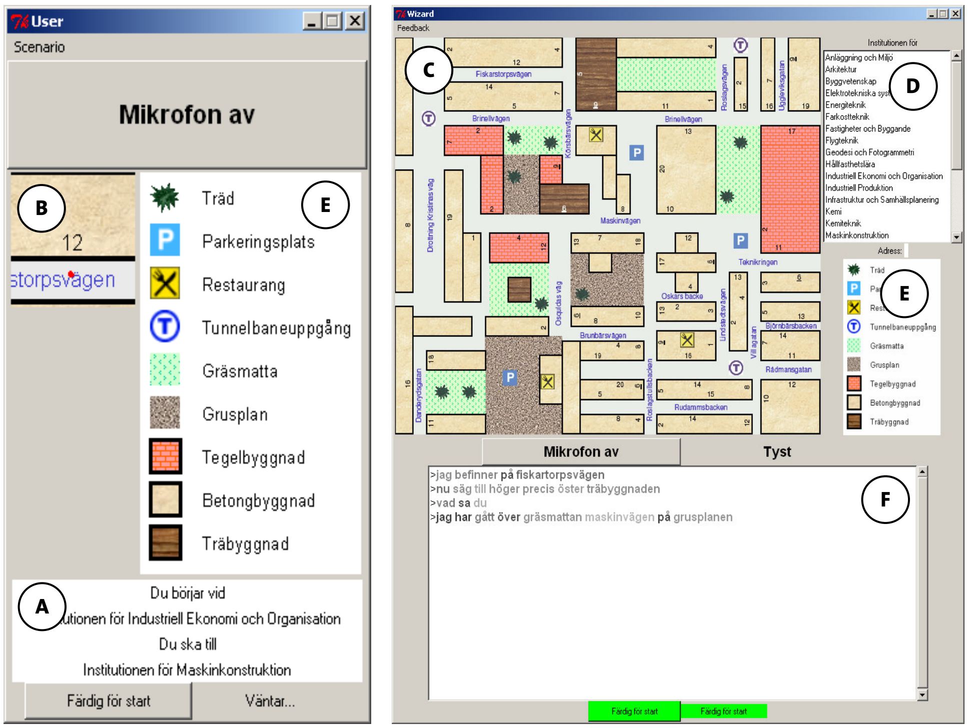
since the purpose of the study was to collect data on "natural" human error handling, people with experience in dialogue system design were not used as operators.

4.2.3.2 Scenarios

The users were given the task to get from one department to another on a simulated campus. The operators' task was to guide the users. The operators had access to a map showing the entire campus to help them with their task. In order to solve the task, the users had to state the goal and continuously describe their current location. When guiding the users, the operators had no direct access to their position, but had to rely on their descriptions of surrounding landmarks. Five different scenarios were given to each pair of subjects, which resulted in 40 dialogues. The order of the scenarios was changed and balanced between pairs, so that general trends after several sessions could be studied independently of scenario.

4.2.3.3 Material

A system for handling the simulation of the campus and for managing the communication between the subjects was built. The user's and the operator's interfaces to the systems are shown in Figure 4.3.



Du börjar vid
Institutionen för Industriell Ekonomi och Organisation
Du ska till
Institutionen för Maskinkonstruktion

Anläggning och Miljö
Arkitektur
Byggvetenskap
Elektrotekniska syst
Energiteknik
Farkostteknik
Fastigheter och Byggande
Flygteknik
Geodesi och Fotogrammetri
Hållfasthetslära
Industriell Ekonomi och Organisation
Industriell Produktion
Infrastruktur och Samhällsplanering
Kemi
Kemiteknik
Maskinkonstruktion

```
>jag befinner på fiskartorpsvägen
>nu säg till höger precis öster träbyggnaden
>vad sa du
>jag har gått över gräsmattan maskinvägen på grusplanen
```

Figure 4.3: The user's interface to the left and the operator's interface to the right.

At the bottom of the user's screen (A), the scenario was presented. Only a small fraction of the map (B) surrounding the current position was shown (seen from above). The user moved around on the campus by using the arrow keys on a keyboard. When the user changed direction, the whole map rotated, so that the user always was facing "up", which made it hard for the subjects to talk about "up", "down", "north" or "south". Instead, they had to use landmarks and relative directions.

The operator's map (C) was identical to the user's, except for some street names that were missing on the user's map. The operator could easily look up where the departments were located (D). The user's position was not shown on the operator's map, so the operator had to rely on the user's descriptions of the environment. On each screen, there was a legend explaining the landmarks, such as trees, parking places and brick buildings (E).

Both the user and the operator were wearing headsets and a push-to-talk mechanism was used. The operator's speech was processed through a vocoder and played back directly to the user. The processed speech was fairly easy to understand, according to post-interviews. However, a lot of prosody was distorted, and the users could not tell whether they listened to a male or female voice.

The user's speech was recognised by a speech recogniser and the recognised string was displayed on the operator's screen (F). An off-the-shelf speech recogniser was used with built-in acoustic models of Swedish. A bigram language model was used, trained on a small corpus of invented dialogues and transcriptions from pilot studies. The vocabulary was of about 350 words. As Fraser & Gilbert (1991) points out, using speech recognition in a Wizard-of-Oz setting might be tough for the operator, since the recognition result may be hard to present and interpret. In this experiment, the words were coloured in greyscale according to each word's confidence score, in order to make them more easily readable, allowing the operator to get an immediate and overall understanding of the confidence scores of the words. Words that were coloured in darker tones had higher confidence scores, while lighter tones reflected lower confidence scores. Since the operator could not hear the user, an indicator on the screen showed whether the user was speaking or not, in order to facilitate turn taking.

4.2.3.4 Procedure

The user and operator were informed separately about the experiment and the setting, and the respective computer interfaces were explained to them. After the instructions, the subjects were placed in different rooms and were not allowed to see each other until the experiment was over. They got no information about each other before the experiment. During the experiment, the conductor of the experiment was sitting behind the operator and could see what the operator was doing and hear what the participants said (using headphones). The conductor also assisted in answering any technical questions about the system from the operator during the experiment. The user was sitting alone. The task was interrupted if the subjects did not complete it within ten minutes. There were no instructions on who should start the conversation, who should take the initiative and "lead" the dialogue, or on possible error handling strategies.

Table 4.1: Example annotation. The columns denote (from left to right): the speaker (Operator or User) and utterance id, the utterance (in case of a user utterance, first the recognised utterance and then the transcription in italics), the understanding, and the dialogue act label. All examples are translated from Swedish. The confidence shading of the words in the speech recognition results have been transferred to the corresponding English words in the translation.

Turn	Utterance	Und.	Dialogue act
O.a1	Take to the right in the crossing and continue. You will have a gravel pitch on your right and then a concrete building and then you will get to a crossing and you can stop there.		AssertRoute
U.a2	**I AM THERE** *(I am there.)*	full	AssertPosition
O.a3	Okay, then continue, eeh let's see, the concrete house that you have on your right side when you have passed the crossing and continued straight forward, well you should pass it and then you should take to the right directly after that house.		Acknowledge AssertRoute
U.a4	**NUMBER FOUR NINETEEN HOUSE NUMBER FIVE** *(number ten in the corner of the house, should I round it?)*	mis	RequestRoute
O.a5	Number nineteen is Machine Construction.		AssertRoute
U.a6	**AT A GRAVEL PITH WITH THIRTEEN** *(At a gravel pitch. Am I right then?)*	partial	AssertPosition RequestRoute
O.a7	Okay, then there is a little problem, I must check, wait a moment.		Acknowledge AssertProblem RequestActWait
U.b1	**HELLO ELEVEN TWENTY ONE TWELVE** *(Yes I am there and where shall I go now?)*	non	AssertPosition RequestRoute
O.b2	Repeat		RequestActRepeat
U.b3	**ELEVEN COME TO A WOODEN BUILDING TWENTY NINETEEN** *(eeh now I have come to the wooden building how should I go then?)*	partial	AssertPosition RequestRoute
O.c1	Do you have a brick building on your right?		RequestPosition
U.c2	**WHAT HEARD** *(Yes I have.)*	non	Yes
O.c3	Do you have a brick building on your right?		RequestPosition (repeat)
U.c4	**WHAT HAS** *(Yes I have.)*	non	Yes
O.c5	What else can you see than the wooden building?		RequestPosition

After each scenario, the subjects filled out a questionnaire about the interaction. The questionnaires consisted of a number of statements, for each of which the subjects stated to what extent they agreed. The set of statements were somewhat different for the operators and the users. Only one of the statements from the users' questionnaire is discussed in this chapter: "we did well in solving the task". There was a choice of seven levels of agreement, ranging from "strongly disagree" to "strongly agree". After the whole experiment, both user and operator were interviewed. For the users, the questions mainly concerned how well they thought that they had been understood and if they had understood the vocoder.

4.2.3.5 Data analysis

All utterances of the users and operators were transcribed and manually annotated. Each utterance was annotated with regard to which dialogue acts it contained and how well it was understood. Some annotation examples are given in Table 4.1. The annotation schemes used are described below and presented in Table 4.2 and Table 4.3.

Table 4.2: The dialogue act categories.

Label	Example
REQUESTGOAL	Where do you want to go?
ASSERTGOAL	I want to go to the department of machine construction.
REQUESTPOSITION	Can you see a wooden building?
ASSERTPOSITION	I have a tree to my left and a concrete building to my right.
REQUESTROUTE	How should I go now?
ASSERTROUTE	After the building, you should take to the left.
SIGNALNONUNDERSTANDING	I do not understand. Please repeat. What did you say?
REQUESTREADY	Are you ready?
ASSERTPROBLEM	There seems to be a problem.
REQUESTACTWAIT	Please wait.
ASSERTACTWAIT	I am waiting.
ACKNOWLEDGE	Okay Yes
NO	No
GREETING	Hello
THANKS	Thank you.

The dialogue act scheme, as presented in Table 4.2 above, was not intended to be general. Instead, it was constructed to cover the most frequent types of dialogue acts in the current experiment. The purpose of annotating dialogue acts was to find relations between these and the understanding of utterances. Unlike more general schemes (such as those discussed in 2.2.1), distinctions were also made between questions and assertions concerning different subtasks in the domain, such as establishing a goal, finding out the users' positions and giving directions. No distinction was made between assertions and answers (which can be seen as a subset of assertions), since this would introduce unnecessary ambiguity, especially as there was a lot of miscommunication. The dialogue acts were encoded based on the spoken, not the recognised, utterances.

Each user utterance was also annotated with regard to how well it was immediately "understood" by the operator. "Understood" here means that the operator continued the dialogue with one interpretation, knowing that it may turn out to be incorrect. It does not necessarily mean that that the operator believed in the interpretation. For example, a clarification question from the operator such as "do you have a tree on your left?" shows that the operator understands that there is a tree on the left from the previous utterance. Based on the user's reaction to this question, the operator may later reject this interpretation. However, it is still annotated as (partially) understood or misunderstood, since this was the immediate interpretation. To estimate the level of understanding, the speech recognition result and the operator's reaction to the utterance were considered. The degree of understanding was classified into four categories, presented in Table 4.3.

Table 4.3: The degrees of understanding each user utterance was annotated with.

Label	**Meaning**
FULL	Full understanding: The full intention of the utterance was understood.
PARTIAL	Partial understanding: Only a fragment or a part of the full intention was understood.
NON	Non-understanding: No part or fragment of the intended message (with the possible exception of a single vague word) was understood.
MIS	Misunderstanding: The operator continued with an interpretation that was not in line with the user's intention.

Of course, the annotator had no direct access to the speaker's intention for each utterance. However, the interpretation was highly constrained by the task context. If there were speech recognition errors that could lead to misunderstanding, these were only marked as misunderstanding if the operator seemed to continue with an interpretation of them. An example of this is utterance Ua.4 in Table 4.1. The utterance Oa.5 suggests that the operator interprets the word "nineteen" as correct, so Ua.4 is classified as a misunderstanding. In contrast, there is nothing in utterance Oa.7 or Ob.2 (or subsequent utterances) that suggests that the previous utterance was misunderstood (in this sense), even though there are many misrecognised words.

The data was transcribed and annotated by one main annotator. To check the reliability of the annotation scheme used, two other persons annotated 1/5 of the dialogues (i.e., full dialogues, randomly selected). The annotators were instructed to annotate according the definitions of understanding, understanding levels, and dialogue acts, as they are described above. For dialogue acts, the main annotator agreed with one of the other annotators in 99.1% of the cases and with both of them in 97.5% of the cases. For the understanding levels, the scores were 94.9% and 89.8%. The most common disagreement was between partial and full understanding. These figures were judged good enough to base the analysis on the main annotation.

4.3 Results

4.3.1 General results

The 40 dialogues resulted in 736 user utterances (18.4 per dialogue on average). The mean utterance length was 6.7 words. 80% of the 40 scenarios were solved within ten minutes. As expected, there were a lot of errors in the recognition results – about 42% word error rate (WER) – due to the users' unrestricted speech and the fact that the bigram language model used was limited: 250 training utterances with a vocabulary of 350 words and 19 classes. 7.3% of the words used were out of vocabulary. There were large individual differences in terms of understanding. The different operators' average understanding is shown in Figure 4.4. In the rightmost bar, the average understanding for all subjects is shown.

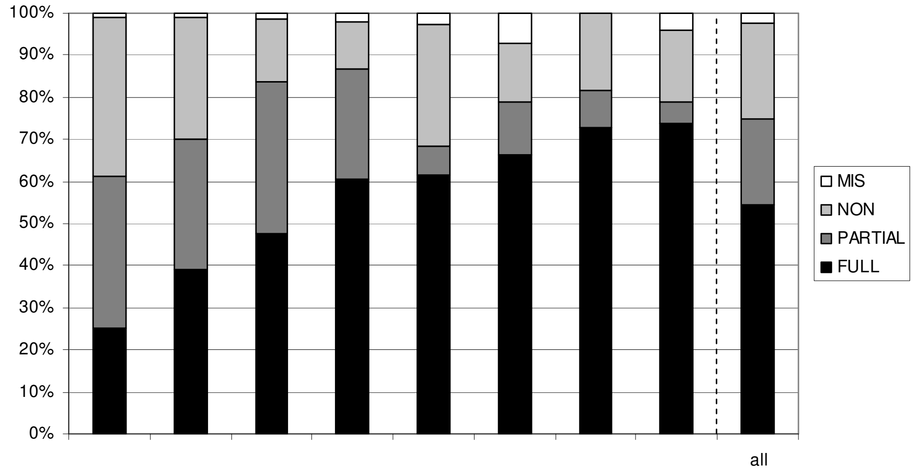


Figure 4.4: The different operators' average understanding of the users' utterances across all five sessions. The rightmost bar shows the distribution for all operators.

As can be seen in the figure, very few of the utterances resulted in misunderstanding. This means that when misrecognitions occurred, the operators were very good at deciding which words were correct and which were not. When there were a lot of misrecognitions, this resulted in partial understanding or non-understanding, instead of misunderstanding. Thus, the operators were very good at error detection. More than 50% full understanding in general may seem to be high compared to the high WER. However all words do not have to be correct for full understanding. Moreover, the WER was not equally distributed between utterances. Some had very low WER and some very high.

In order to find out whether the subjects improved at the task during the five sessions, a trend analysis was tested on several factors. As shown in Figure 4.4, there was a large between-pair variance, which makes it hard to find general trends. However, it turned out that the proportion of non-understanding and the number of user utterances (a measure of the length of the dialogue) changed after subsequent sessions (one-way repeated measures ANOVA; $p<0.05$). An analysis of polynomial contrasts showed that both variables decreased in a linear fashion. The trends are shown in Figure 4.5.

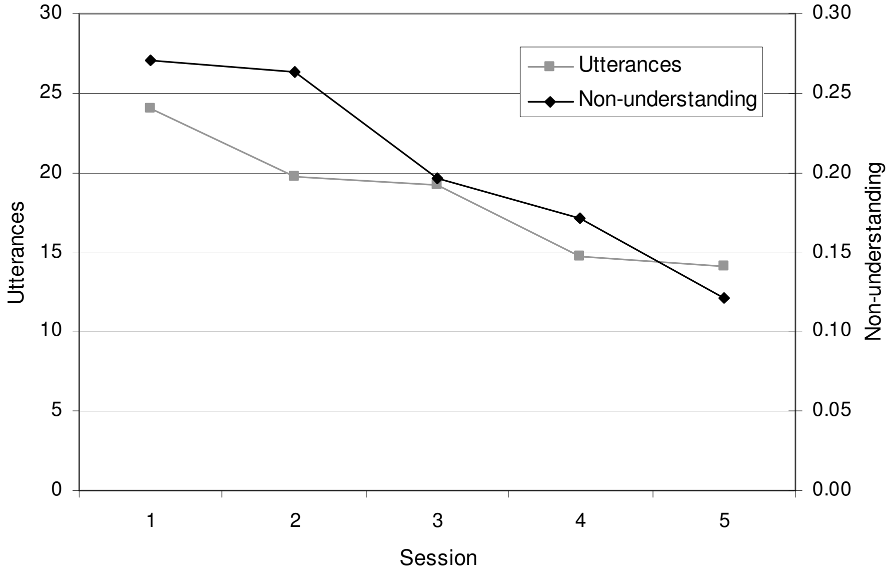


Figure 4.5: The average number of user utterances and proportion of non-understanding in each dialogue, as they decrease after subsequent sessions.

The post interviews revealed that, despite the numerous non-understandings, the users in general experienced that they were almost always understood. It turned out that in many cases, instead of signalling non-understanding – which may seem like the obvious choice – the operators employed other strategies.

4.3.2 Strategies after non-understanding

The operators' strategies after non-understanding were divided into three categories: SIGNALNONUNDERSTANDING, ASSERTROUTE and REQUESTPOSITION. The three groups were of approximately equal size.

4.3.2.1 SIGNALNONUNDERSTANDING

This category includes all reactions to non-understanding where the operator somehow signalled that the utterance was not understood. This includes explicit requests for repetition ("please repeat", "what did you say"), assertions of non-understanding ("I didn't understand"), and repetitions of the same utterance (O.c1-O.c3 in Table 4.1 is an example of a repetition). Consider the following example:

(40) U.d1: **WEST WITH** *(that's right)*
O.d2: Please repeat what you said.
U.d3: **THAT THERE WITH** *(that's right)*

In the example, the problem with the recognition is that the expression "that's right" isn't covered by the language model. Therefore, the repetition doesn't lead to a recovery from the problem. Example U.b1-U.b3 in Table 4.1 is an example where the request for repeat instead triggers the user to rephrase, which leads to a minor recovery (partial understanding).

4.3.2.2 ASSERTROUTE

This category contains all reactions to non-understanding where the operator gave a new route description without any of the signals of non-understanding mentioned above. Here is an example:

(41) O.e1: Continue a little bit forward.
U.e2: **STREET THAT THERE HOUSE** *(past the wooden house?)*
O.e3: Now, walk around the wooden house. Take left and then right.

The only thing that the operator seems to rely on, in this example, is something about a house. Since it is impossible to interpret what the user is trying to say and since the word house does not contribute much (in this domain) to the understanding, this has been classified as a non-understanding. Although it seems like the operator is totally ignoring the user's contribution, the operator utterance implicitly verifies the user's position by referring to a wooden house (something that there are only a few of on the map). If the hypothesis is incorrect and the operator's utterance was out of place, the user has a chance to react so that the recovery process may continue. If it is correct (as in this case), the user will probably perceive the situation as if the utterance was fully understood.

4.3.2.3 REQUESTPOSITION

This category contains all reactions to non-understanding where the operator asked a question about the user's position without any of the signals of non-understanding mentioned above. Here is an example:

(42) O.f1: Do you see a wooden house in front of you?
U.f2: **YES CROSSING ADDRESS NOW** *(I pass the wooden house now.)*
O.f3: Can you see a restaurant sign?

The operator seems unsure of whether the user really can see a wooden house, but instead of asking the user to repeat, another question is asked that is confirming the same hypothesis as the operator wanted to confirm by asking the first question. Another example is Oc.3-O.c5 in Table 4.1. After the non-understanding, the operator asks about a wooden building (which has been mentioned previously in the dialogue). Since the question is task-related (and not related to what has been said), it implicitly confirms the operator's hypothesis about the user's position without signalling non-understanding (just as with ASSERTROUTE).

4.3.3 Error recovery

As discussed in 3.3.2.4, non-understanding may often lead to error spirals, where the user just repeats the non-understood utterance and perhaps starts to hyperarticulate, which may only worsen the recognition performance. A good error recovery strategy should therefore aim at coming to understanding, or at getting "back on track", as quickly as possible after a non-understanding has occurred. In order to evaluate the different strategies based on this criterion, the operator's understanding of the user's utterance following a reaction to a non-understanding was studied. As an example, take the sequence b1-b3 in Table 4.1. After the first non-understanding, the operator selects the strategy SIGNALNONUNDERSTANDING. This strategy leads to a partial understanding of the next utterance. The distribution of the operators' understanding following the different strategies is presented in Figure 4.6. The top bar shows the general distribution for all utterances, also shown in the rightmost bar in Figure 4.4.

Statistical tests showed that there was no deviation from the general distribution after ASSERTROUTE and SIGNALNONUNDERSTANDING, but after REQUESTPOSITION there were significantly less non-understandings and instead significantly more partial understandings (goodness-of-fit test; dF = 3; χ^2 = 12.52; $p < 0.01$). This suggests that REQUESTPOSITION leads to better recovery from the problem.

Why does REQUESTPOSITION lead to less non-understanding? To answer this question, the types of questions that were posed and the reactions to the strategies were analysed further. Approximately 1/3 of the REQUESTPOSITION utterances were wh-questions and 2/3 yes/no-questions. This may suggest that the questions constrain the length of the answers from the user and thereby increase the speech recognition performance. However, yes/no-questions do not always result in simple yes/no answers, as example (42) above illustrates. Table 4.4 shows the utterance length following the different strategies. As seen in the table, the utterances fol-

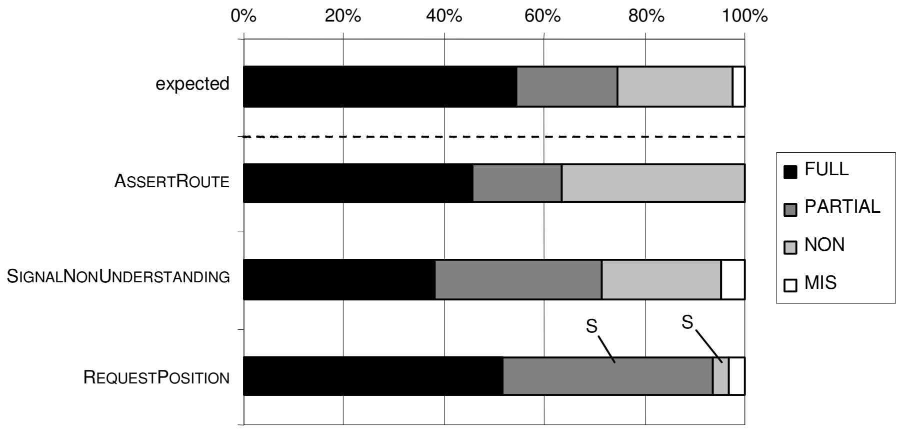


Figure 4.6: The understanding of the user's utterance that follows the operator's reaction to a non-understanding. "S" marks significant deviation from the general distribution.

lowing REQUESTPOSITIONare not shorter, but longer. The better understanding of these utterances is probably explained by the fact that they constrain the vocabulary and syntax of the response to the domain and the language models, which increase speech recognition performance. Moreover, the specific question that precedes the response may also constrain the interpretation of the speech recognition result even if it is bad. This is not true for ASSERTROUTE, and may explain why REQUESTPOSITION works better.

Table 4.4: Mean and median length of the utterances (number of words) following the different strategies.

	Mean utterance length	**Median utterance length**
SIGNALNONUNDERSTANDING	7.4	4
ASSERTROUTE	6.4	5
REQUESTPOSITION	8.6	8

4.3.4 User experience of task success

Fast error recovery can be regarded as a measure of efficiency. But from a user-centred point of view, the experience of using the system should come first, and efficiency should be a means for improving the experience of using the system. Thus, it is interesting to examine how different recovery strategies and other objective measures contribute to the user's experience.

Since there was a large between-pair and within-pair variance (as shown in Figure 4.4 and Figure 4.5), regarding the subjects' performance, it should be possible to correlate the user's experience with objective measures for different pairs and sessions. To investigate this, a multiple regression analysis was used in a way similar to the PARADISE evaluation framework for dialogue systems (Walker et al., 2000a). The idea behind PARADISE is to find out the relation between the subjective measure of the user's satisfaction (which can be collected by using a questionnaire) and a number of objective measures (the task success and dialogue costs, such as number of repetitions, WER, etc.). If this relation can be estimated, it is possible to predict the effect on user satisfaction that the tuning of objective parameters will have (such as improving the WER), without having to run expensive user tests. The method can also be used to give insights into which parameters are important for the user satisfaction of dialogue systems in general and which are not.

The input to the regression analysis is a criterion variable (user satisfaction) and a set of predictor variables (the objective measures). The output is a set of coefficients for the predictor variables that describe the relative contribution of each variable for the variation in the criterion variable. Since the user's task in the current study was given beforehand and was quite artificial, it was hard to get a measure of the "user satisfaction". Instead, the user's experience of task success was used. The question "how well do you think that you did in solving the task?" from the questionnaire was used as the dependent factor, which was a rating from 0 to 6. As predictor variables, factors that were likely to affect the user's experience were selected: time to solve the task, the length of the path that the user took, the mean WER, the number of non-understandings, and the number of uses of the error recovery strategies (SIGNALNONUNDERSTANDING, ASSERTROUTE, REQUESTPOSITION). All 40 dialogues were used as data points.

If several predictor variables correlate, they will explain the same variation in the criterion variable, and the result will depend on which predictor variables are selected. It is therefore important to select the variables systematically. Hinkle et al. (1994) describe three procedures for doing this: backward solution, forward solution and stepwise solution. All three were tested in the current experiment, and they all resulted in a significant correlation between the criterion variable and two of the predictor variables (R^2=0.56; $p < 0.0001$). The contribution of the different variables to the user's experience of task success is shown in Table 4.5. As can be seen in the table, the only factors that contributed were time for task completion and the number of non-understandings that the operator had *signalled* (which both had a negative effect). It is interesting that neither the number of non-understandings nor the WER per se had any effect on the user's experience, but only the cases where the user was made aware of the non-understanding.

Table 4.5: Results from the regression analysis.

Contributing factors	Coeff	SE	T Stat	P-value
Total time	-0,456	0,083	-5,499	< 0,001
# SignalNonUnderstanding's	-0,560	0,262	-2,142	0,039

Non-contributing factors
Total path length
WER
Non-understandings
AssertRoute's
RequestPosition's

4.4 Discussion

The experiment presented in this chapter shows that the operators did not routinely signal non-understanding when faced with incomprehensible speech recognition results. Instead, they tried to ask task-related questions that confirmed their hypothesis about the user's position. This strategy led to fewer non-understandings of the subsequent user utterance, and thus to a faster recovery from the problem. When they did signal non-understanding, this had a negative effect on the user's experience of task success. Despite the numerous non-understandings, users reported that they were almost always understood.

A main point driven by Brown (1995) is that listeners do not primarily strive to arrive at a correct interpretation of utterances. They merely use the utterances as a knowledge source among others to solve the task at hand. This is supported by the findings presented in this chapter. In a problem solving task such as guiding, the goal is established early in the dialogue and the speakers can focus primarily on solving the task by working towards this goal. The results from this study also confirm the argument put forth by Brown (1995), that it may be problematic to study understanding by just analysing ordinary human-human dialogue, since the signals of understanding that the speakers send apparently do not have to reflect their true understanding. As Brown points out, the problem of studying understanding in ordinary conversation analysis is that the analyst has no access to what goes on inside people's heads. The analyst has to rely on the record of the speakers' behaviour, such as grounding and signals of non-understanding. However, it is not certain at all that these signals reflect the true understanding of the speakers. This is a serious problem if the analyst wants to relate the level of understanding to the speaker's behaviour during conversation. In psycholinguistic laboratory experiments, the comprehension of subjects can be studied by carefully controlling the stimuli and measuring the level of understanding after each utterance or fragment. The problem with such experiments is that it is not possible to relate the understanding to an ongoing dialogue that the subject is engaged in. Brown argues that the Map Task method provides a solution to this problem, since the speakers' beliefs about the world are controlled by the experimenter (i.e., what is printed on the maps). Thus, it becomes possible to study miscommunication that arises from the misalignments of the speakers' models of the world. However, the method does not provide information on how people react to *non-understanding*, which the experimental setup used in this study supports.

On average, the speech recognition performance was poor. As mentioned previously, this was partly due to the users' relatively free speech and partly due to the limited training of the language models. This may seem as non-representative for most dialogue systems. However, without the poor performance, it would not have been possible to collect enough data on non-understanding from a reasonable number of dialogues for quantitative analysis. It is also important to stress that the WER varied a lot between utterances and subjects (as can be seen in Figure 4.4), which is often the case in real applications. Some dialogues were smooth and successful, while others were dominated by errors. This also made the experience of task success more varied, which is important for regression analysis. Humans are also probably better at interpreting the bad recognition results than what could be accomplished with a robust interpreter. Thus, the distribution of the levels of understanding may be more representative than the WER.

It would be interesting to find out why the subjects get better at the task after repeated sessions. This is probably due to the fact that they get better at formulating descriptions and route directions. It would also be interesting to find out whether they learn any new error handling strategies. Another question is whether it is mainly the user or the operator that adapts. That is, does a dialogue system have to adapt to the user, or is it enough that the user adapts to the system? No general trends in choice of strategies could be found, probably due to the large inter-subject variance.

In this study, a speech recogniser was used to introduce errors. The drawback with this approach is that there is no direct control over the amount of errors that are introduced, and thus it becomes hard to study how the effects of different error handling strategies depend on the amount of errors. An interesting alternative is presented in Stuttle et al. (2004) where a trained typist transcribes what is said. The typist does not (intentionally) introduce any errors. Instead, the transcribed utterances are processed by a system that simulates a certain rate of speech recognition errors. The system uses a phonetic confusion model and a language model, which results in errors similar to those that a speech recogniser would make.

4.4.1 Comparison to other findings

The finding that signalling non-understanding decreases the understanding of subsequence utterances is in line with other findings, as discussed in 3.3.2.4. However, it is interesting that there are alternative strategies that not only increase the recovery rate, but lead to better user satisfaction.

Bohus & Rudnicky (2005b) describes an experiment with a spoken dialog system that handles conference room reservations (called the RoomLine system). The system randomly chose between 10 different non-understanding recovery strategies, such as AskRepeat (system asks the user to repeat the non-understood utterance), Yield (system remains silent, and thus implicitly notifies the user that a non-understanding has occurred), and Reprompt (system repeats the previous prompt). Among these options was also the strategy MoveOn, defined as “system advances the task by moving on to a different question”, which is very similar to the more successful strategies employed by the operators in the experiment presented here. It turned out that this strategy had the highest performance on error recovery. The authors make the following note:

> The high performance of the MoveOn strategy is consistent with prior evidence from a wizard-of-oz study of error handling strategies […][6] 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 traveled in spoken dialog system design: when non-understandings happen, instead of trying to repair the current problem, use an alternative dialog plan to advance the task.

Schlangen & Fernández (2007a, 2007b) used a setting similar to the one presented here. In one experiment, pairs of subjects were given the task of dictating to each other. In another experiment, pairs of subjects were given a more problem-solving oriented task. In both cases, noise was inserted at random places in the speech to induce perception problems. A comparison between the collected dialogues showed that clarification requests were much more common in the dictation task, although the amount of inserted noise was the same. Thus, if the task is not fully dependent on the precise wordings, people rely on other strategies than clarifying what is said.

Yet other support for the findings presented in this chapter comes from an informal analysis of the Spoken Dutch Corpus, containing transcriptions of human-human dialogues (Boves, 2004):

> The transcripts of the spontaneous conversations […] contain a substantial number of ‘xxx’ codes, which stand for speech that the transcriber could not understand. Although a formal analysis of these situations remains to be performed, the results of the work on multiword expressions suggests that only a small proportion of these unintelligible intervals elicits ‘say that again, please’ replies from the interlocutor. This can mean two things: either the speakers, who are familiar with each other, have much less difficulty understanding each other than a third person, or the fact that one does not understand the interlocutor completely does not always affect the communication to such an extent that a repair meta-dialog is called for.

4.4.2 Application to other domains

One important question is if these results can be applied to domains that are not about navigation. In tasks where a single slot has to be filled by using specific words, there may not be any other option than to signal non-understanding and thereby encourage repetition. In certain other, more complex domains, strategies similar to REQUESTPOSITION, are likely to be appli-

[6] The study reported in this chapter.

cable. To illustrate the possible applications, some examples from different domains will be given. The most obvious are dialogues where the operator is diagnosing a problem. If the system does not understand the answer to one question, it might be better to ask another one instead of signalling non-understanding, given that there are several ways to pinpoint the problem. A similar strategy may also be useful if speech technology is to be used in games, where non-understanding may be frustrating for the user and task-related questions may be used to guide the conversation along certain paths. It should also be possible to ask task-related questions after non-understanding in information-browsing domains. As an example, take the apartment broker domain, which is the domain for the ADAPT spoken dialogue system (Gustafson et al., 2000). The following (invented) dialogue illustrates:

(43)	U.g1:	Tell me about the bathroom. [full understanding]
	S.g2:	*It is a tiled bathroom and it has a bathtub.*
	U.g3:	Is there anything else you can tell me about the apartment? [non-understanding]
	S.g4:	*Do you want me to tell you more about some specific part of the apartment?*
	U.g5:	Yes, tell me about the kitchen.

In this case, the system's response after the non-understanding (S.g4) happens to be appropriate and does not signal non-understanding. Just like REQUESTPOSITION, it is a task-related question that may constrain the interpretation of the user's next utterance. The results from this study suggest that it may have a greater potential for recovering from the error than an explicit signal of non-understanding would have. If it had not been in place, the user would still have a chance to correct the system. Furthermore, the user may not always have a fixed idea of what she wants to know and may experience a question such as S.g4 as helpful. In a multimodal system (such as ADAPT), it is also often possible to switch modality and let the user provide the information in an alternative way. Oviatt & VanGent (1996) have shown that modality switching is a successful method for recovering from error. The results from the study presented in this chapter suggest that it may be better to do this without signalling non-understanding:

(44)	S.h1:	*Which area are you interested in?*
	U.h2:	I would like to live near the water. [non-understanding]
	S.h3:	*Can you mark exactly on the map.*

Utterance S.h3 could start with "Sorry, I couldn't understand", but leaving this out may improve the user experience of task success. The operators in this study sometimes used some word in the poor recognition results when formulating their requests after non-understandings. This may increase the probability of posing a question that seems relevant.

Of course, these are just invented examples of what could possibly be done in other domains to recover from non-understanding without signalling non-understanding. It would be interesting to perform experiments similar to this in other domains to find out if humans benefit from similar strategies, or if they have to signal non-understanding.

4.5 Summary

In this chapter, a method for collecting data on human error handling strategies has been presented. An experiment was conducted based on this method, in which pairs of subjects were given the task of guiding each other in a virtual campus by talking to each other. The person giving directions (the "operator") could not hear what the other speaker (the "user") said. Instead, the user's speech was recognised by a speech recogniser and the operator could read the results on a screen. Due to limited language models, the speech recognition performance was poor and there were many cases of non-understanding. Despite the numerous non-understandings, users reported that they were almost always understood. Unlike most dialogue systems, the operators did not often signal non-understanding. If they did signal non-understanding, this had a negative effect on the user's experience of task success.

An alternative reaction to non-understanding was to ask task-related questions that were confirming the operator's hypothesis about the user's position. This strategy led to fewer non-understandings of the subsequent user utterance, and thus to a faster recovery from the problem.

For the design of spoken dialogue systems in similar domains, the results suggest that when non-understandings occur, a good domain model and robust parsing techniques should be used to pose relevant questions to the user, instead of signalling non-understanding, so that errors can be efficiently resolved without the user experiencing the dialogue as problematic and dominated by explicit error handling.

CHAPTER 5

Early error detection on word level

In the experiment reported in Chapter 4, the high WER caused many non-understandings, but only a few misunderstandings. This means that humans have an impressive capability of early error detection, meaning that they are to a large extent aware of which hypotheses are reasonable, and which are not. An important question is what this awareness is based on. In other words, if we were to build a dialogue system with such capabilities, which knowledge sources would contribute to the detection of errors?

In this chapter, two studies are presented, based on the data collected for Chapter 4. In Study I, machine learning is used with different sets of features. A main issue for machine learning is which factors (knowledge sources) the learning can and should be based on, and how to operationalise these factors into extractable features. Some factors, such as dialogue history, may seem useful for error detection, but are hard to operationalise, especially for longer contexts. Finding whether a factor contributes to the performance of a human subject doing the error detection task may provide some guidance as to its value to the machine learning task. In Study II, humans were given the task of detecting errors with different combinations of knowledge sources.

As described earlier in 3.3.1, previous studies on early error detection have to a large extent focussed on full utterances. More precisely, the task has been to decide whether the word error rate (WER) and/or concept error rate is greater than zero. This is useful for systems where short utterances are expected and their complexity limited. However, when long and complex utterances are expected and an n-gram language model is used for the ASR, many utterances can be expected to contain some errors. Long utterances may also contain more than one concept, rendering an all-or-nothing distinction too blunt. If some content words are intact, the recognition may still prove useful.

The task in the studies presented in this chapter can be described as *binary word-level early error detection*, in other words, to classify each word in the speech recognition result as correct or incorrect. While it would perhaps be more useful to classify concepts in the semantic interpretation of the speech recognition result, the results from this study are not dependent on the semantic model or interpretation technique used.

5.1 Study I: Machine learning

In this study, machine learning was used for the error detection task. Two learners were trained on several different sets of features in order to measure the contribution of different factors to machine learning of early error detection.

5.1.1 Algorithms used

Two machine learning algorithms were tested and compared: *transformation-based learning* and *memory-based learning*. These algorithms were chosen because they represent different machine learning paradigms and they were familiar to the author.

5.1.1.1 Transformation-based learning

In transformation-based learning, the algorithm learns a set of transformation rules that are applied after each other. It was invented by Eric Brill for use in part of speech tagging (Brill, 1995), but has been used for many other tasks as well, such as dialogue act tagging (Lager, 1999). All instances are initially tagged with the most common class. A set of rule templates has to be written specifically for the task. During training, the algorithm finds the instantiation of a template that creates the rule that most efficiently transforms the classes in the material in a positive direction. Rules learned early in the process may include very drastic general transformations that also have negative effects. However, these negative transformations may be recovered later by more specific rules. In the current study, μ-TBL (Lager, 1999) was used for transformation-based learning. μ-TBL supports the definition of clauses written in Prolog, which makes the use of features more flexible, for example when handling numeric features. However, unlike other rule learning algorithms, such as RIPPER (Cohen, 1995), μ-TBL cannot automatically find thresholds for numeric features.

5.1.1.2 Memory-based learning

In memory-based learning (also called *instance-based learning*), the training set is just stored as examples for later evaluation (Mitchell, 1997). The computation is postponed to classification (so-called “lazy” learning), when the instance to be classified is compared to all examples to find the (set of) nearest neighbour(s). The number of nearest neighbours that are compared can be tuned for the task (the algorithm is sometimes called *k-nearest neighbour*, where *k* is the number of nearest neighbours used). To measure the distance between two instances, the vectors of features for the instances are compared. In this study, TiMBL (Daelemans et al., 2003) was used for memory-based learning. TiMBL supports different ways of comparing features.

The most simple is just an overlap measure, where each feature gets one score if the values of the instances are equal. The features are typically weighted using “gain ratio weighting”, a measure that is computed using information theory. This is done at training time by analysing all examples to compute how much each feature contributes to the task. TiMBL also supports other ways of comparing the value of two features. Using “modified value difference”, the examples can be analysed to form a matrix of distances between the values of the feature. If the feature is numeric, it is also possible to use the numerical difference between features as a direct distance metric.

Memory-based learning has the advantage that learning is extremely fast (just storing examples) and that very little preparation has to be done (for example, no templates have to be written). It may also find so-called “islands of exceptions” more easily, without having to discover very specific exception rules. The disadvantage is that classifying new instances may be slow. It is therefore crucial that the algorithm has efficient methods for indexing the examples. Another disadvantage, compared to transformation-based learning, is that it is hard to study what is actually learnt. It is not possible to study any rules that might give insights into systematic properties of the data.

5.1.2 Data and features

The classification task in this experiment was to determine whether a given recognised word was present at the corresponding location in the transcription of the spoken utterance (TRUE) or not (FALSE). For this study, the recognition results from the corpus presented in Chapter 4 were aligned to the transcriptions (using minimum edit distance) in order to determine for each word if it was correct or not. 73.2% of the words turned out to be correct, which gives us a majority-class baseline to compare the machine learning performance with. Of the 4470 words, 4/5 were used as training data and 1/5 as test data.

In Table 5.1, the features that were used for each word are classified into four groups: confidence, lexical, contextual and discourse. For dialogue act tagging, a simple set was constructed specifically for the domain. The content/non-content split was also made with the domain in mind. Content words were mainly nouns, adjectives and verbs.

5.1.3 Results

In order to investigate how the performance varied depending on which features that were used, different combinations of feature set groups were used. The results are shown in Table 5.2. TiMBL seemed to perform best with the IB1 algorithm, gain ratio weighting and overlap as distance metric (except for confidence, for which a numeric distance metric was used). Depending on feature set, different values for *k* were best. Since µ-TBL cannot automatically find thresholds for numeric values, a set of ten (equally sized) intervals were defined for the confidence score.

Table 5.1: Features used for error detection.

Group	Feature	Explanation
Confidence	CONFIDENCE	ASR word confidence score
Lexical	WORD	The word
	POS	The part-of-speech for the word
	LENGTH	The number of syllables in the word
	CONTENT	Is it a content word?
Contextual	PREVPOS	The part-of-speech for the previous word
	NEXTPOS	The part-of-speech for the next word
	PREVWORD	The previous word
Discourse	PREVDIALOGUEACT	The dialogue act of the previous operator utterance (according to Table 4.2)
	MENTIONED	Is it a content word that has been mentioned previously by the operator in the discourse?

Table 5.2: Performance of the machine learning algorithms depending on feature set.

Feature set	μ-TBL	TiMBL
Confidence	77.3%	76.0% (*k*=5)
Lexical	77.5%	78.0% (*k*=1)
Lexical + Contextual	81.4%	82.8% (*k*=1)
Lexical + Confidence	81.3%	81.0% (*k*=5)
Lexical + Confidence + Contextual	83.9%	83.2% (*k*=1)
Lexical + Confidence + Contextual + Discourse	85.1%	84.1% (*k*=1)

As the table shows, each group seems to add (more or less) to the performance. μ-TBL seems to perform a bit better (although the difference has not been tested for significance). With the richest feature set, μ-TBL performs 11.9% better than baseline.

The performance of the two machine learners seems to be very similar. In order to investigate whether they made the same mistakes, the result of the classifications were compared. In 69 cases, both learners made the same mistake, in 137 cases they disagreed. Thus, if a perfect ensemble method would be used that could choose the right classifier, the resulting performance would be 92.3%.

Since many interpretation modules in dialogue systems are mainly dependent on content words, the performance of these are important for detection. There were 285 content words in the test material of which 199 were correctly recognised. This gives a baseline of 69.8%. For these words, the best scores for the classifiers were 87.7% (μ-TBL) and 87.0% (TiMBL). Thus, the best classifier μ-TBL performs 17.9% better than baseline for content words. (A perfect ensemble method would score 94.4%.)

The top rules that were learned by μ-TBL are shown in Table 5.3. The first rule states that all content words with confidence less than 0.5 should be tagged as FALSE. The rest of the rules mainly concern different confidence thresholds depending on type of word (often represented with part-of-speech and word length). There are also some interesting discourse rules, such as the sixth: all two-syllable content nouns with a confidence score high enough that have been mentioned previously by the operator should be tagged as correct.

Table 5.3: The top rules learned by μ-TBL.

Transformation	**Rule**
TRUE → FALSE	CONFIDENCE < 0.5 & CONTENT = TRUE
TRUE → FALSE	CONFIDENCE < 0.6 & POS = Verb & LENGTH = 2
TRUE → FALSE	CONFIDENCE < 0.4 & POS = Adverb & LENGTH = 1
TRUE → FALSE	CONFIDENCE < 0.5 & POS = Adverb & LENGTH = 2
TRUE → FALSE	CONFIDENCE < 0.4 & POS = Verb & LENGTH = 1
FALSE → TRUE	CONFIDENCE > 0.4 & MENTIONED = TRUE & POS = Noun & LENGTH = 2

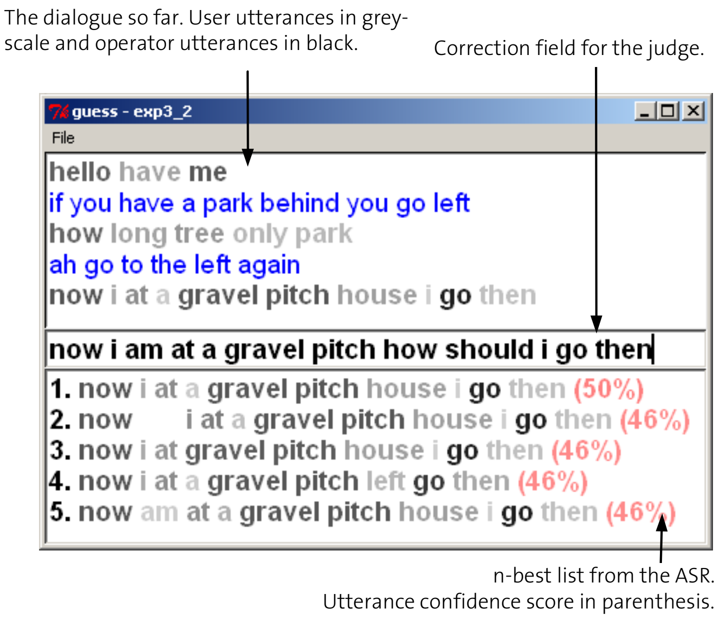
5.2 Study II: Human error detection

The features used in the machine learning study were chosen because they could intuitively contribute to error detection and they were easy to operationalise. However, it should be interesting to examine which factors humans could benefit from in performing the task, especially factors that are hard to operationalise. Finding whether a factor contributes to the performance of a human subject doing the error detection task may provide some guidance as to its value to the machine learning task. In the second study, an experiment was conducted where human subjects (henceforth referred to as judges) were asked to detect errors in ASR results. In order to investigate whether dialogue context, ASR confidence measures, and ASR n-best lists provide help when detecting errors, the judges' access to these factors was varied systematically.

5.2.1 Method

The corpus presented in Chapter 4 was also used for this study. Four dialogues with higher average WER than the corpus as a whole were chosen. The first 15 exchanges of these dialogues were used for the experiment, resulting in a subset of the corpus containing 60 exchanges. 50% of the words in the subset were correctly recognised, which gives the baseline for the task, by either deleting all words or leaving the entire string unaltered.

Eight judges with some limited experience in speech technology were asked to delete words in the ASR output that they believed to be wrong, using a custom-made tool. Figure 5.1 shows the tool in English translation.


The dialogue so far. User utterances in grey-scale and operator utterances in black.
Correction field for the judge.
guess - exp3_2
File
n-best list from the ASR.
Utterance confidence score in parenthesis.

hello have me
if you have a park behind you go left
how long tree only park
ah go to the left again
now i at a gravel pitch house i go then

now i am at a gravel pitch how should i go then

1. now i at a gravel pitch house i go then (50%)
2. now i at a gravel pitch house i go then (46%)
3. now i at gravel pitch house i go then (46%)
4. now i at a gravel pitch left go then (46%)
5. now am at a gravel pitch house i go then (46%)

Figure 5.1: The judges' interface with an example translated into English.

Each judge assessed all four dialogues, with a different amount of visible context for each dialogue. The four levels of context are shown in Table 5.4.

Table 5.4: Context levels.

Label	Description
NoContext	No context. ASR output only, utterances in random order.
PreviousContext	Previous utterance from the operator visible. Utterance pairs in random order.
FullContext	Full dialogue. The operator utterances and the ASR output are given incrementally and stay visible throughout the dialogue.
MapContext	As FullContext, with the addition of the map that was used by the interlocutors.

Furthermore, each ASR result was repeated three times with an increasing degree of information from the ASR attached, and the judge had to reassess the recognition each time. The ASR information levels are listed in order of appearance in Table 5.5. The order of the dialogues and context levels were systematically varied for each judge.

Table 5.5: ASR information levels.

Label	Description
NoConfidence	Recognised string only.
Confidence	Recognised string, colour coded for word confidence (grey scale: dark for high confidence, light for low).
NBestList	As Confidence, but the 5-best ASR result was provided.

5.2.2 Data analysis

The data consists of three versions of each recognised utterance: the transcription, the ASR result, and the judge's correction, which were all aligned to measure the judges' performance. An example is shown in Table 5.6. For each word in the recognition result that was misrecognised, the judge received one error detection point if the word was removed or changed. Since this was an error detection task and not an error correction task, the point was received regardless of whether the judge changed the erroneous word to the correct word or not (see the first word in the example). For each word that was correctly recognised, the judge received one point if the word was not removed or changed. The total number of points in each recognition result was then divided by the total number of words in the result to yield an error detection score between 0.0 and 1.0. The example in Table 5.6 yields an error detection score of 0.6. A score of 1.0 indicates that all incorrectly recognised words (insertions and substitutions) were detected and no correctly recognised words were judged as errors. A score of 0.0 indicates the opposite: all correctly recognised words were judged as errors and all errors were judged as correct.

Table 5.6: Made-up example calculation of error detection score (sub=substitution, ins=insertion).

Transcription	*the*			*correct*	*words*
ASR result	*our*	*system*	*thought*	*correct*	*words*
ASR error	sub	ins	ins	-	-
Judge's correction	*users*		*thought*	*correct*	*text*
Detection point	1	1	0	1	0

5.2.3 Results

The left column of Figure 5.2 shows mean error detection scores for the different ASR information and context levels. PreviousContext, FullContext and MapContext turned out to hold no significant differences and are thus combined into Context.

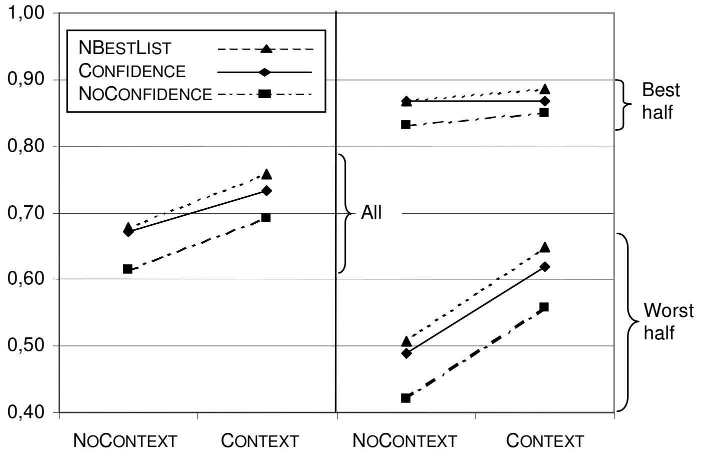


Figure 5.2: Mean error detection scores for the human judges, depending on the availability of the features. The result for all utterances is shown to the left, and the result for the best and worst half are shown to the right.

There were main effects of both ASR information level and context level (two-way repeated measures ANOVA; p < 0.05). Post tests revealed that NBESTLIST was better than CONFIDENCE, which in turn was better than NOCONFIDENCE. PREVIOUSCONTEXT was better than NOCONTEXT (p < 0.05), but there was no difference between PREVIOUSCONTEXT, FULLCONTEXT and MAPCONTEXT. There were no interaction effects between variables. Overall, the judges performed significantly better than the baseline detection score of 0.5.

To investigate what effect average WER had on the judges' results, the figures were recalculated over two subsets of the corpus: one subset containing the 30 utterances with the highest WER, and another subset containing the 30 utterances with the lowest WER. Detection scores for the subsets are shown in the right column of Figure 5.2. The effects for the worst utterances were the same as the effects in general. For the best utterances, the differences between different recognition information levels persisted. However, there were no significant differences between different context levels.

5.3 Discussion

Both studies show that word confidence scores are useful for early error detection, and that other features can be used to improve performance. Utterance context and lexical information improve the machine learning performance. The errors that are found with these features probably reflect constant errors in the language and acoustic models and should be corrected there, if possible. This is not always an easy task, however. Apart from using these methods for improving the performance of a specific application without collecting more data for models, a rule-learning algorithm such as μ-TBL can be used to pinpoint the specific problems. For example, if the algorithm finds that a number of specific words should be classed as incorrect, these may be over-represented in the training material for the ASR language models.

It may be surprising that access to the n-best list improved the judges' performance. When simply detecting errors (and not correcting them), the information contained in the n-best list should be reflected in the word confidence scores; if a word changes in the n-best list, it is a sign that it may be incorrect, but such words usually also get a low confidence score. However, for a human subject, the fact that a word changes in the list may be easier to make use of than the grey scale of the words. Thus, the additional performance that n-best lists give could possibly be achieved by a machine learner by just looking at the confidence scores. If the n-best list would in fact be useful for a machine learner, the question is how it should be operationalised, so that it could be used in the feature set.

The discourse context of the utterance is potentially the most interesting feature, since it is not considered by the ASR. The machine learners improved only slightly from the discourse context, but the results from the second study suggests that the immediate discourse context of the utterance (i.e., the previous operator/system utterance) is the most important to humans for detection. For good recognitions, there was no effect from the discourse context, which indicates that the intact parts of a good recognition may provide sufficient context in themselves. For poorer recognitions, it seems that there is sufficient information in the previous utterance together with the judges' knowledge about the domain, and that further context is redundant. Thus, further work on operationalising context for machine learning should focus on the previous utterance. It could be argued that even though a long dialogue context does not improve the performance of humans, a machine may still be able to use it. Humans, however, generally seem to outperform machines when it comes to utilising context in spoken language.

In the studies presented in this chapter, the task was a binary decision between correct and incorrect. As discussed in 3.3.1, it could sometimes be more useful to derive a continuous probabilistic confidence score as a result of the early error detection. This may be possible to derive from a memory-based learner, either by looking at the entropy of the class distribution or the density of the nearest neighbour set (i.e., the distribution of distances in the different k's; if there are a lot of close competing nearest neighbours, confidence should be low).

Since the classifiers disagree in so many cases, it would also be interesting to test whether it would be possible to use an ensemble method that could pick the right classifier.

5.3.1 Comparison to other findings

As the overview of the research on early error detection in 3.3.1 showed, related studies have also shown that ASR confidence scores are useful for early error detection, but that other features can be used to improve the performance. This study shows that this is equally true for word-level error detection. The other studies in the review did not use features from a larger context and this study confirms that larger context may not contribute much.

As was also shown in the review, other studies of early error detection have benefited from the use of prosody. It would be interesting to see if prosody could also help to detect errors on the word level, either by looking at utterance-level prosodic features or at local features. Local prosodic features may for example help to find world-level errors that arise due to disfluencies.

5.4 Summary

In this chapter, two studies were presented in which the early detection of speech recognition errors on word level was explored. In the first study, memory-based and transformation-based machine learning was used for the task, using confidence, lexical, contextual and discourse features. In the second study, factors humans benefit from when detecting errors were investigated. Information from the speech recogniser (i.e., word confidence scores and 5-best lists) and contextual information were the factors investigated. The results show that word confidence scores are useful, and that lexical and contextual (both from the utterance and from the discourse) features further improve performance, especially for content words. In the case of poor recognitions, human judges seem to benefit from using the dialogue context. However, larger context than the previous utterance does not seem to improve performance for human judges.

PART III

The Higgins Spoken Dialogue System

HIGGINS: I've taught her to speak properly; and she has strict orders as to her behaviour. She's to keep to two subjects: the weather and everybody's health – Fine day and How do you do, you know – and not to let herself go on things in general. That will be safe.
MRS HIGGINS: Safe! To talk about our health! about our insides! perhaps about our outsides! How could you be so silly, Henry?
HIGGINS: Well, she must talk about something.

Pygmalion by *George Bernard Shaw*

CHAPTER 6

Concept-level error handling in Higgins

In Part II, it was shown that, in a conversational dialogue setting, it is indeed possible to detect errors and extract meaning from recognition hypotheses containing a lot of errors – at least for humans. In this part, we will investigate how this and other issues may be handled in a complete spoken dialogue system.

As stated previously, speech recognition output in conversational dialogue systems is often only partially correct. Therefore, error handling in such systems should be done on the concept level, not on the utterance-level. In this chapter, we will present a model for how the *grounding status* of individual concepts may be tracked. Instead of modelling how this grounding status gets updated by a special set of "grounding acts", we will show how all utterances, even those that are mainly task-related, may contribute to the grounding process by updating the grounding status. The grounding status includes the history of when and how the concept is grounded by the participants, and the system's confidence in this. Since the grounding status is modelled on the concept level, the choice of surface realisation will affect the system's model of what has been grounded.

As part of the work for this thesis, the HIGGINS[7] spoken dialogue system has been developed and evaluated (Edlund et al., 2004; Skantze et al., 2006). The system has served as a test-bed for developing and evaluating methods and models for concept-level error handling, such as robust interpretation, modelling grounding status in the discourse, displaying understanding, posing clarification requests, and late error detection. This chapter will describe the domain, semantics and components of this system, and how concept-level error handling is done in all parts of the system. In the next chapter, an evaluation of the system will be presented.

[7] The system, as well as its components, bear Pygmalion-related names. This includes the Greek myth *Pygmalion & Galatea*, the Bernard Shaw play *Pygmalion*, and the musical *My Fair Lady*.

6.1 The Higgins navigation domain

The initial domain chosen for HIGGINS is the same as the one used in Chapter 4: pedestrian city navigation and guiding. As noted previously, this domain is similar to the now classic Map Task domain (Anderson et al., 1991), as well as to a number of guide systems, such as REAL (Baus et al., 2002). In HIGGINS, the user tells the system where she wants to go and the system guides the user by giving verbal instructions. The system does not have access to the user's position by any positioning device. Instead, it has to figure out the position by engaging in a dialogue in which the user describes the 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. Unlike the experimental setting in Chapter 4, a 3D model of a virtual city is used for simulation, see Figure 6.1.

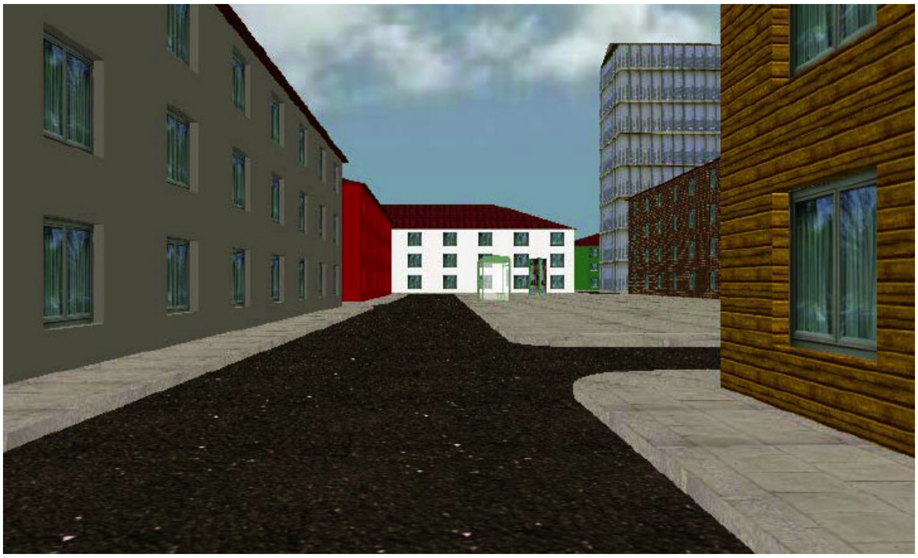


Figure 6.1: The 3D simulation that is used for user tests.

It should be noted that this application would perhaps not be directly useful (as is) for pedestrian navigation in a real city, partly because it would be infeasible to model the city completely and partly because a real application would utilise a GPS. However, the domain is challenging from an error handling perspective; it is a conversational dialogue setting where complex semantic relations and referring expressions need to be expressed.

To illustrate how error handling is done in the HIGGINS system, a made-up example scenario will be used, which is shown in Table 6.1. In this scenario, the ASR results are shown as well, revealing a fair amount of speech recognition errors. The HIGGINS system is currently built for Swedish, but most examples in this thesis have been translated into English.

Table 6.1: 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 eeh a large concrete building on my left and a bus stop in front of me.	**I HAVE TREE A LARGE CONCRETE BUILDING ON MY LEFT AND BUS STOP ON MY RIGHT**
S.5	*What colour is the concrete building?*	
U.6	Red.	**RED**
S.7	*Ok, can you see a wooden building?*	
U.8	I can see a brown building.	**I CAN SEE A BLUE BUILDING**
S.9	*Blue?*	
U.10	No, brown.	**NO BROWN**
S.11	*Do you really have a bus stop on your right?*	
U.12	No, in front of me.	**NO IN FRONT OF SEE**
S.13	*Ok, I think I know where you are. Walk a little bit forward and take left after the red building.*	

The system starts by requesting the user's goal (S.1) and the user replies (U.2). Based on the confidence score of the stated goal, the system decides to generate a *separate display utterance* ("an ATM") as part of the next turn, and continues with a general request about the user's position (S.3), which the user answers to by describing surrounding landmarks (U.4). This utterance is poorly recognised by the ASR, which results in a misunderstanding: the system now believes that the user has a bus stop on his right (which is really in front of him). However, the system stores all confidence scores and information about what has been grounded, so that such errors may be identified later on. To constrain the user's position, the system asks a question about the concrete building (S.5). When doing this, it uses a definite description to refer to the building, which is a way of simultaneously *displaying its understanding*, which the user does not object to in the next utterance (U.6). This way, the system's uncertainty about the concrete building has been reduced. To further constrain the user's position, the system now asks a more specific question (S.7). The user does not directly answer the question, but provides a description that nevertheless helps to constrain the position (U.8). The colour of the building that the user describes is erroneously recognised, but since it gets a low confidence score, the system makes a *fragmentary clarification request* (S.9), and the user corrects the system (U.10). Due to the misunderstanding in U.4, the system now finds out that there is no place that the user can be. To solve this error, the system employs *late error detection* and searches the discourse history and finds out that there is one concept with a relatively low confidence score that has not been grounded: the belief that the user has a bus stop on his right. The system makes a *misunderstanding repair* – it checks the belief with the user (S.11) and the user corrects the system (U.12). The system has now constrained the user's position and may start to give route directions (S.13).

Before describing the details of these error handling mechanisms, we will introduce the semantic representations and architecture used in the HIGGINS spoken dialogue system.

6.2 Semantic representations

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 – not just simple feature-value lists. Semantic descriptions are consistently represented as rooted unordered trees of semantic concepts. Nodes in the tree may represent objects, relations and properties. 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 Kilpeläinen (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 with XML, using a schema that is specific for the domain. Figure 6.2 shows an example: an abstract representation of a wooden building. Figure 6.3 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 (about 60 000 elements) containing all landmarks and their properties, as well as possible user positions and how they relate to the landmarks. All objects in the database have id's. The XML in Figure 6.2 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`, that is, a list of the id's of all the wooden buildings in the database.

The semantic representations may be enhanced with "meta-information", for example about confidence scores, communicative acts, and if information is new or given. Figure 6.4 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 the structure of 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 6.3, in order to find possible referents to the object. The meta-information is easily removed if placed in a special namespace.

```
<object id="$id4">
  <properties>
    <type>
      <value>building</value>
    </type>
    <material>
      <value>wood</value>
    </material>
  </properties>
</object>
```

Figure 6.2: An abstract semantic representation of a wooden building in XML.

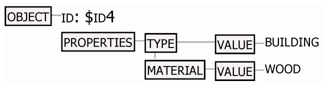


Figure 6.3: The same structure as in Figure 6.2, visualised graphically.

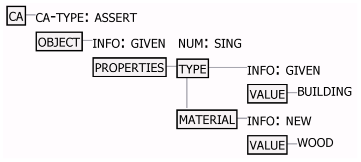


Figure 6.4: The semantic representation of the utterance “the building is made of wood”.

Table 6.2: An example of unification using a template.

Template	**Unification**	
T =	*S1* =	*S2* =
`<template>`	`<object>`	`<colour>`
`  <object>`	`  <properties>`	`  <value>`
`    <properties count="1">`	`    <type>`	`    red`
`      <type count="1">`	`      <value>`	`  </value>`
`        <value count="1"/>`	`        building`	`</colour>`
`      </type>`	`      </value>`	
`      <colour count="*">`	`    </type>`	
`        <value count="1"/>`	`  </properties>`	
`      </colour>`	`</object>`	
`      <size count="*">`		
`        <value count="1"/>`	*Unify*(*S1*, *S2*, *T*) =	
`      </size>`	`<object>`	
`    </properties>`	`  <properties>`	
`  </object>`	`    <type>`	
`</template>`	`      <value>building</value>`	
	`    </type>`	
	`    <colour>`	
	`      <value>red</value>`	
	`    </colour>`	
	`  </properties>`	
	`</object>`	

Tree structures may also be unified, as shown in Table 6.2. A semantic template is used to specify how the nodes may be structured, to guide the unification. As can be seen, the nodes in the template may be marked with how many times it may occur at that location, by using the attribute `count`. The template also makes it possible to unify structures starting at different levels in the tree, as is the case for *S1* and *S2*.

The use of a template for unification makes it possible to easily represent the semantics of fragments (such as verbs, relations, properties, etc.) and combine them into full propositions. Such fragments may be ambiguous, that is, they may fit into different parts of the template. When the fragments are unified and they start at different levels, the unification algorithm tries to combine them with the shortest distance possible. In other words, they get disambiguated. For example, if the semantic concept PRICE (as part of the question "what does it cost") gets unified with the semantic structure VALUE:100 (a representation of the answer "100"), this may result in the structure PRICE:VALUE:100, provided that the template allows such a structure. In itself, VALUE:100 is ambiguous, and may fit into different structures.

6.3 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 6.5 shows the most important modules and messages in HIGGINS, when run in the navigation domain.

From the ASR, the top hypothesis with word confidence scores (2) is sent to a natural language understanding module, called PICKERING. PICKERING makes a robust interpretation of this hypothesis and creates context-independent semantic representations of communicative acts (CA's). In HIGGINS, dialogue management is not implemented as a single module. Instead, this processing is divided into a discourse modeller (called GALATEA) and a set of action managers. GALATEA may be regarded as a further interpretation step, which takes the context into account. Based on incoming CA's, GALATEA builds a discourse model. This discourse model is then consulted by a set of action managers, which initiate systems actions. The purpose of this separation between discourse modelling and action selection is to make the discourse modelling more generic, while the action selection may be highly domain specific. This separation is similar to the approaches taken in Allen et al. (2001b) and Pfleger et al. (2003).

CA's from the user (3) are sent from PICKERING to GALATEA, which adds them to a discourse model. The discourse model (4) is then sent to a *grounding action manager* (GAM) which initiates grounding actions (such as making clarification requests). If the turn is not yielded to the user, the discourse model (5) is passed on to the *navigation action manager* (NAM), which initiates navigation actions (such as requesting the user's position or giving route directions). To do this, the NAM has access to the *domain database*. The NAM may also make modifications to the discourse model, for example if an error is detected, and send it back (6) to GALATEA.

System initiated communicative acts from the action managers (7,8) are sent to a natural language generator (OVIDIUS) which generates a surface string (with some prosodic markup). This string (9) is sent to a TTS which synthesises the spoken output (10). But the communicative acts from the system are also sent back to GALATEA (11), which treats them in the same way as the communicative acts sent from PICKERING (3). Thus, GALATEA models communicative acts both from the user and the system; ellipsis, anaphora and grounding status is handled and modelled in the same way for all communicative acts.

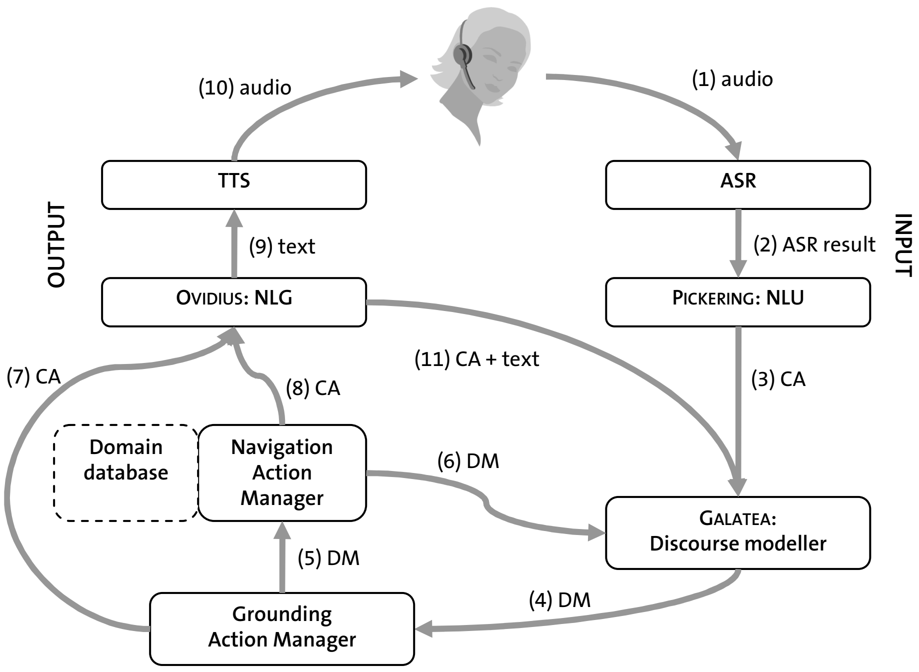


Figure 6.5: The most important modules and messages in the HIGGINS navigation domain. CA stands for communicative act. DM stands for discourse model.

All modules in HIGGINS are fairly generic – the resources that are needed for the specific application are all encoded in XML. The NAM, on the other hand, is written specifically for the domain. However, much of the work that a typical dialogue manager has to do is already handled by the GAM and GALATEA.

All modules operate asynchronously, which means that, for example, the ASR may be recognising a user utterance or the TTS rendering a system utterance, while an action manager is generating a new action.

6.4 PICKERING: Natural language understanding

PICKERING is a robust interpreter, designed to work with continuous incremental input from a speech recogniser with n-gram language models in a conversational dialogue system. The grammar used to parse recognition results is based on context-free grammars (CFG), with some modifications. To add robustness, PICKERING may automatically allow deviations from these grammars, such as allowing partial results, insertions and non-agreement.

Although the combination of features included in PICKERING is (to our best knowledge) unique, much work in the literature has been focussed on achieving robustness in parsing and semantic interpretation beyond keyword spotting. Examples include Mellish (1989), which deals with insertions in chart parsing, and Kasper et al. (1999), in which partial results are combined.

6.4.1 Grammar

The PICKERING grammar rules are enhanced with semantic rules for generating the kind of semantic trees described in 6.2 above. The CFG consists of a rule-set, a collection of lexical entries, and an optional morphology, all of which may carry semantics. Figure 6.6 shows a simple grammar which covers the Swedish phrase “den röda byggnaden” (“the red building”).

In this example, there are three lexical entries and one rule. Both entries and rules have an associated list of features (in the f-namespace), a `<match>` part that specifies what they match, and a `<sem>` part which specifies the resulting semantics (in the s-namespace). Entries may match words, and rules may match words, entries or other rules. In Swedish, words in noun phrases must be congruent: they have to agree on gender, number and definiteness. Features that should agree are specified in the `<agreement>` element. The attribute `propagate="true"` also tells that the agreeing features should be propagated to the matching rule.

Grammar rules also contain instructions for combining semantics from matching entries and rules. A common instruction, as seen in the grammar example, is `<unify>`, which is used to unify semantics. `<ref>` is used to refer to the semantics of the matching parts. `<add>` states that the resulting features should be copied into the semantics, according to the template.

The parse result of the phrase “den röda byggnaden”, using the grammar in Figure 6.6, is shown in Figure 6.7.

```
<grammar>
  <lexicon>
    <entry f:name="det" f:info="given" f:gen="utr" f:num="sing">
      <match>den</match>
      <sem>
        <s:object/>
      </sem>
    </entry>
    <entry f:name="attr" f:info="given">
      <match>röda</match>
      <sem>
        <s:colour><s:value>red</s:value></s:colour>
      </sem>
    </entry>
    <entry f:name="nom" f:info="given" f:gen="utr" f:num="sing">
      <match>byggnaden</match>
      <sem>
        <s:object>
          <s:type><s:value>building</s:value></s:type>
        </s:object>
      </sem>
    </entry>
  </lexicon>
  <rules>
    <rule f:name="object" top="true">
      <agreement features="f:info f:gen f:num" propagate="true"/>
      <match>
        <entry f:name="det"/>
        <entry f:name="attr" link="attrlink"/>
        <entry f:name="nom" link="nomlink"/>
      </match>
      <sem>
        <b:unify>
          <b:ref link="nomlink">
            <b:add f:info="$info" f:num="$num" to="template"/>
          </b:ref>
          <b:ref link="attrlink">
            <b:add f:info="$info" to="template"/>
          </b:ref>
        </b:unify>
      </sem>
    </rule>
  </rules>
</grammar>
```

Figure 6.6: A Pickering grammar fragment.

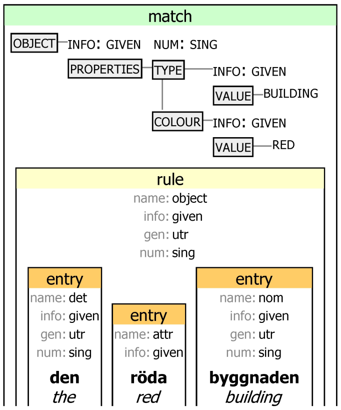

match
OBJECT INFO: GIVEN NUM: SING
PROPERTIES TYPE INFO: GIVEN
VALUE BUILDING
COLOUR INFO: GIVEN
VALUE RED
rule
name: object
info: given
gen: utr
num: sing
entry name: det info: given gen: utr num: sing
entry name: attr info: given
entry name: nom info: given gen: utr num: sing
den the
röda red
byggnaden building

Figure 6.7: Parsing of the phrase "den röda byggnaden" (the red building) using the grammar in Figure 6.6.

6.4.2 Robust interpretation

To add robustness, the interpreter applies a number of additional techniques to the standard CFG parsing algorithm. To illustrate these techniques, an example interpretation of U.4 in Table 6.1 is shown in Figure 6.8. In the figure, the corresponding Swedish phrase is also shown in order to highlight non-agreement in the noun phrase "a large concrete building", as explained below.

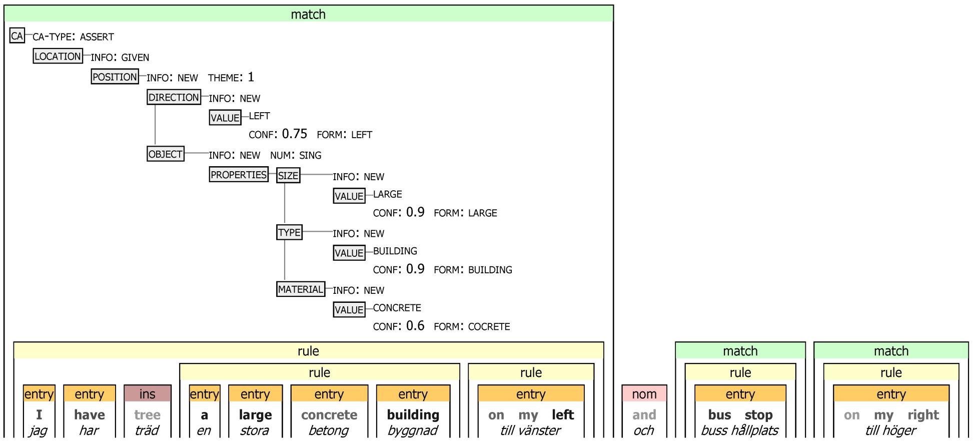


Figure 6.8: An example interpretation of the erroneously recognised utterance U.4 in Table 6.1. A corresponding Swedish translation is shown below containing an additional morphological error. The semantic results of the last two phrases are not shown.

6.4.2.1 Insertions

Disfluencies not modelled by the n-gram language models may easily give rise to unexpected words in the middle of phrases. An example of this is the third word "tree" in Figure 6.8. PICKERING allows insertions of unexpected words anywhere inside a phrase. A parameter can be set that constrains the number of subsequent insertions that are allowed. A simple keyword-spotter would probably have included this content-word in the semantic result, but thanks to the grammatical analysis, PICKERING can treat this as an error and ignore it.

6.4.2.2 Non-agreement

In a complex domain such as pedestrian navigation, morphological distinctions are meaningful in order to distinguish both number and definiteness, which may signal whether objects and properties are given or new. Such morphological distinctions may be more important in some languages than others (which is the case for Swedish compared to English). However, speech recognisers with n-gram language models may often fail to produce the right morphological inflections. Moreover, speakers may also make morphological mistakes in conversational language. This may give rise to non-agreement among the constituents of a phrase. PICKERING deals with this by allowing non-agreement when features are combined according to the `<agreement>` element. If the features do not agree, the majority class is selected (a random choice is used if there is a tie). An example of this is the phrase "a large concrete building" in Figure 6.8. The corresponding Swedish translation shown below contains a morphological error: it is erroneously recognised as "en stora betong byggnad". In Swedish, the correct morphological inflection would be "den stora betong byggnaden" (INFO:GIVEN) or "en stor betong byggnad" (INFO:NEW). Since the latter interpretation is more consistent with the input, it is selected by PICKERING in the robust interpretation. While non-agreement is allowed, it is considered when different solutions are ranked.

6.4.2.3 Fragment spotting

PICKERING does not have to find a rule that covers the complete input string. Instead, it tries to choose the smallest number of matching phrases which covers the largest number of words. Between these phrases, non-matching words are allowed. In Figure 6.8, the best fit is three phrases with a non-matching word ("and") in-between. Incomplete phrases may then be combined by the discourse modeller GALATEA, which will be described in 6.5.2.

6.4.2.4 Concept confidence scores

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 (similar to Gabsdil & Bos, 2003). These confidence scores are then transferred to the semantic result according to the template. Figure 6.8 shows how the concepts are marked with such scores (by the attribute `conf`). Insertions, such as "tree" 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. These concept-level confidence scores may then be used for concept-level error handling, which will be described later on.

6.4.2.5 Surface form

Within a certain domain, a given semantic concept may have several surface forms. For example the forms "building" and "house" both correspond to the concept TYPE:BUILDING in the HIGGINS navigation domain. To keep track of the form used, the semantic template may be marked with slots for surface form. These slots are filled with the forms of the lexical entries that were involved in the production of the semantic concepts. Examples of this are shown in the semantic result in Figure 6.8, by the attribute `form`. These forms may later be used to for example pose fragmentary clarification requests in a correct way, as described later on.

6.4.3 Implementation

PICKERING is a modified chart parser (Jurafsky & Martin, 2000) implemented in Oz[8]. There are some general challenges with robustness in an interpreter. First, there is a risk that the interpreter will find too many interpretations covering different parts of the input without being semantically distinct. PICKERING utilises the semantic results to filter out solutions that are semantically equivalent or are a subset of another solution. Another potential problem is that the interpreter may find erroneous interpretations based on errors in the input. This is a serious problem for keyword-spotters, since virtually every erroneous content word will result in errors in the interpretation. For a very strict parser this is not much of a problem – errors are likely to make parsing of the input impossible – but correct partial solutions are lost. PICKERING deals with this problem by searching for the best set of partial solutions. Finally, robustness can be inefficient if every interpretation is to be considered. The algorithm used in PICKERING is a kind of generate-and-filter technique that ensures that all interpretations are found. This can be costly, but the cost is balanced by the incremental processing – utterances are processed while the user is still speaking. For a more detailed description of the implementation of PICKERING, see Skantze & Edlund (2004).

6.5 GALATEA: Discourse modelling

The discourse modeller in HIGGINS is called GALATEA and is also implemented in Oz. It is designed to be generic – the required configurations and resources are all encoded in XML. As seen in Figure 6.5, the task of GALATEA is to take the communicative acts from both the user (as identified by PICKERING) and the system (as produced by an action manager), and build a discourse model – a model of what has been said during the discourse and which entities are referred to. The discourse model (encoded in XML) consists of two lists:

[8] A multi-paradigm programming language supporting open distributed computing, constraints and logical inference and concurrent object-orientation. See http://www.mozart-oz.org/.

- *CA-list*: A list of past communicative acts in chronological order, with the most recent act first.
- *Entity list*: A list of entities mentioned in the discourse, with the most recently mentioned entity first.

As a new CA is added to GALATEA, the following things are done:

1. SPEAKER and CAID attributes are added to the CA. These attributes contain information about which speaker made the contribution and an id for the CA (an automatically incremented number).
2. Grounding information is added to concepts in the CA, i.e., information about who added the concept to the model, in which turn, and how confident the system is in the concept.
3. Transformations of the CA are made, based on past CA's in the CA-list and a set of transformation rules. This way, ellipses may be resolved.
4. Discourse entities are identified in the CA and are assigned entity id's.
5. The identified entities are extracted from the CA and integrated into the entity list. If an anaphora is identified, the entities are unified.
6. The resulting CA is added to the CA-list.

After the discourse model has been updated, it may be consulted by an action manager that decides what to do next.

6.5.1 Grounding status

The grounding status that is added to concepts in the CA contains information about who added the concept to the model, in which turn, and how confident the system is in the concept. The grounding status is represented as a list, where each item represents an occurrence of the concept in the discourse. Each item in the list contains the following information:

- Who contributed the concept (SPEAKER)
- When was the contribution made (CAID)
- How confident is the system that the contribution was made (if not contributed by the system)? (CONF)
- How was the contribution realised (if not contributed by the system)? (FORM)

The CONF and FORM attributes are taken from the PICKERING results (concept confidence scores and surface form) and placed under a GROUNDING element, together with SPEAKER and CAID attributes, which have been assigned to the CA. In the semantic template used for unification, places where grounding information should be added are marked. Figure 6.9 shows how grounding status has been added to the CA from the parse result in Figure 6.8.

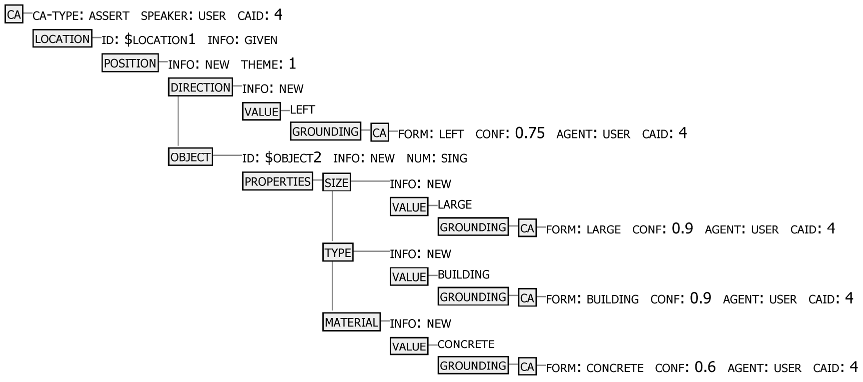

CA CA-TYPE: ASSERT SPEAKER: USER CAID: 4
LOCATION ID: $LOCATION1 INFO: GIVEN
POSITION INFO: NEW THEME: 1
DIRECTION INFO: NEW
VALUE LEFT
GROUNDING CA FORM: LEFT CONF: 0.75 AGENT: USER CAID: 4
OBJECT ID: $OBJECT2 INFO: NEW NUM: SING
PROPERTIES SIZE INFO: NEW
VALUE LARGE
GROUNDING CA FORM: LARGE CONF: 0.9 AGENT: USER CAID: 4
TYPE INFO: NEW
VALUE BUILDING
GROUNDING CA FORM: BUILDING CONF: 0.9 AGENT: USER CAID: 4
MATERIAL INFO: NEW
VALUE CONCRETE
GROUNDING CA FORM: CONCRETE CONF: 0.6 AGENT: USER CAID: 4

Figure 6.9: The semantic interpretation of the first CA in Figure 6.8, after grounding status has been added and entities have been identified.

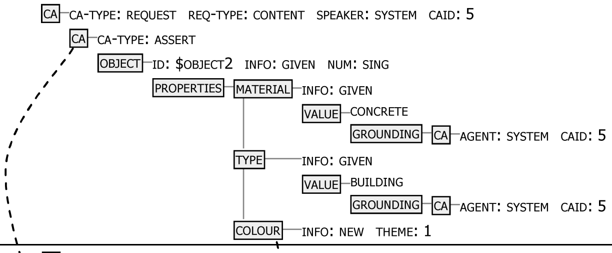
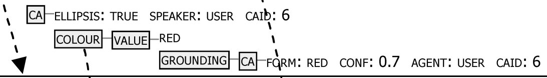
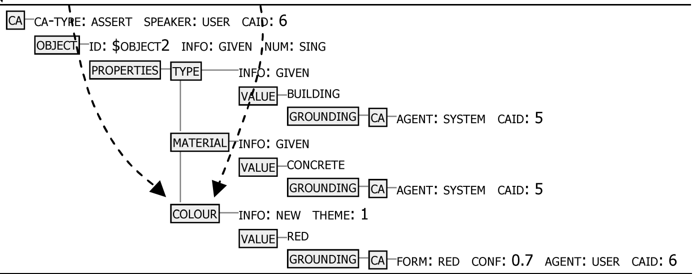
The grounding status can be compared with the "contextual functions" used in Heisterkamp & McGlashan (1996), and the "discourse pegs" used in McTear et al. (2005), that are used to model the grounding status (as discussed in 3.3.3.2).

6.5.2 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 CA's, and a transformation description. The preconditions are formulated as semantic pattern trees that are matched against the target CA's. 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 6.3 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 are 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. The example also shows that the grounding status is added before resolving ellipses. This ensures that only concepts that were part of the original utterance are grounded, not those that are added in the ellipsis resolution.

Table 6.3: Example transformation of an ellipsis into full proposition.

Context: *S.5: What colour is the concrete building?*	 CA CA-TYPE: REQUEST REQ-TYPE: CONTENT SPEAKER: SYSTEM CAID: 5 CA CA-TYPE: ASSERT OBJECT ID: $OBJECT2 INFO: GIVEN NUM: SING PROPERTIES MATERIAL INFO: GIVEN VALUE CONCRETE GROUNDING CA AGENT: SYSTEM CAID: 5 TYPE INFO: GIVEN VALUE BUILDING GROUNDING CA AGENT: SYSTEM CAID: 5 COLOUR INFO: NEW THEME: 1
Ellipsis: U.6: Red.	 CA ELLIPSIS: TRUE SPEAKER: USER CAID: 6 COLOUR VALUE RED GROUNDING CA FORM: RED CONF: 0.7 AGENT: USER CAID: 6
Transformation: U.6: The concrete building is red.	 CA CA-TYPE: ASSERT SPEAKER: USER CAID: 6 OBJECT ID: $OBJECT2 INFO: GIVEN NUM: SING PROPERTIES TYPE INFO: GIVEN VALUE BUILDING GROUNDING CA AGENT: SYSTEM CAID: 5 MATERIAL INFO: GIVEN VALUE CONCRETE GROUNDING CA AGENT: SYSTEM CAID: 5 COLOUR INFO: NEW THEME: 1 VALUE RED GROUNDING CA FORM: RED CONF: 0.7 AGENT: USER CAID: 6

Transformation rules may also be used for robustness to interpret utterances where PICKERING may have identified some fragments. An example of this was shown in Figure 6.8. The second and third phrases are identified as elliptical by PICKERING. In the context of the second phrase ("bus stop"), GALATEA will transform the third phrase ("on my right") into "I have a bus stop on my right". It is, of course, also possible to transform non-elliptical CA's 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.

6.5.3 Anaphora resolution

GALATEA has no access to the domain database. Thus, it does not map discourse entities to "real" objects in the database. Instead, it keeps a list of entities that are mentioned (e. g., "a large building") in the discourse and assigns variable id's 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, that is, bindings to the entity id variables. What counts as an entity in a specific application must be specified so that GALATEA can recognise entities, and it is up to the dialogue system designer to define this. In the HIGGINS domain, entities that are modelled are landmarks, user locations and user goals. Table 6.4 shows a list of the entities modelled during the discourse in Table 6.1, and the variable id's that are assigned to the entities.

Table 6.4: The entities modelled during the discourse in Table 6.1.

Entity	Occurs in turn	Variable id
user goal	S.1, U.2	$goal1
ATM	U.2	$object1
user location	S.3, U.4, S.7, U.8, S.9, U.10, S.11, U.12	$location1
large red concrete building	U.4, S.5, U.6, S.13	$object2
bus stop	U.4, S.11, U.12	$object3
brown building	U.8, S.9, U.10	$object4

As shown previously, 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 information about a given object (see Figure 6.4 for an example). After a CA has been transformed, entities are identified and assigned entity id's according the following principles:

- If the entity has somehow already been assigned an id (for example by the action manager generating it), it is not affected.
- If the entity is marked as new, a new id is generated.
- If the entity is marked as given, 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 targets, and a pattern match is performed.
    - If an antecedent is found, its id is used for the entity to be added.
    - Otherwise, a new id is generated.

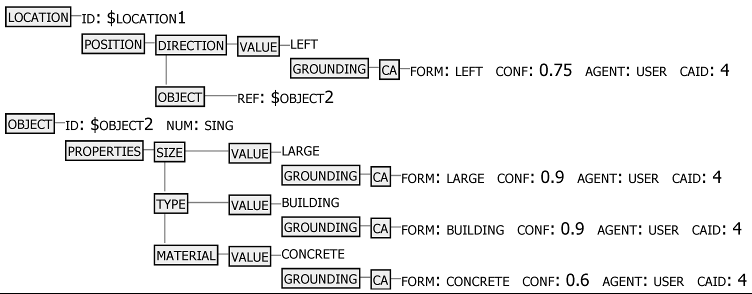
The identified entities are then added to the entity list according the following principles:

- If the id of the entity to be added is the same as for an entity in the entity list, these entities are unified and moved to the first position in the list.
- Otherwise, the entity is simply placed first on the entity list.

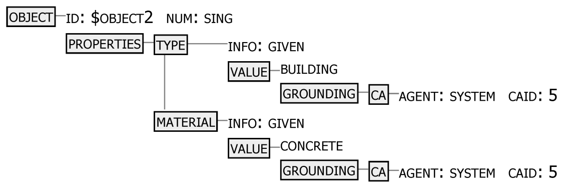
The entity list represents unified asserted information about entities. Therefore, information concerning the structure of the utterances they were extracted from is removed. This includes THEME and INFO attributes, as well as concepts that are only requested, such as COLOUR:RED in the request "is the building red?" Some early error detection is also done on the concept level – concepts with low confidence are filtered out. These concepts may be added later on if they are clarified, which is described in 6.7.2 below. 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. Some examples of extracted entities are shown in Table 6.5.

Table 6.5: Examples of entities extracted from CA's.

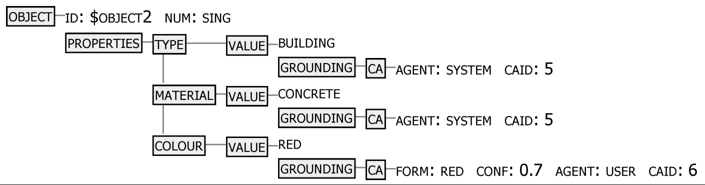
U.4: I have a large concrete building on my left ...


LOCATION ID: $LOCATION1
POSITION
DIRECTION
VALUE LEFT
GROUNDING CA FORM: LEFT CONF: 0.75 AGENT: USER CAID: 4
OBJECT REF: $OBJECT2
OBJECT ID: $OBJECT2 NUM: SING
PROPERTIES
SIZE
VALUE LARGE
GROUNDING CA FORM: LARGE CONF: 0.9 AGENT: USER CAID: 4
TYPE
VALUE BUILDING
GROUNDING CA FORM: BUILDING CONF: 0.9 AGENT: USER CAID: 4
MATERIAL
VALUE CONCRETE
GROUNDING CA FORM: CONCRETE CONF: 0.6 AGENT: USER CAID: 4

S.5: What colour is the concrete building?


OBJECT ID: $OBJECT2 NUM: SING
PROPERTIES
TYPE INFO: GIVEN
VALUE BUILDING
GROUNDING CA AGENT: SYSTEM CAID: 5
MATERIAL INFO: GIVEN
VALUE CONCRETE
GROUNDING CA AGENT: SYSTEM CAID: 5

U.6: Red


OBJECT ID: $OBJECT2 NUM: SING
PROPERTIES
TYPE
VALUE BUILDING
GROUNDING CA AGENT: SYSTEM CAID: 5
MATERIAL
VALUE CONCRETE
GROUNDING CA AGENT: SYSTEM CAID: 5
COLOUR
VALUE RED
GROUNDING CA FORM: RED CONF: 0.7 AGENT: USER CAID: 6

As the entities are unified in the entity list, the grounding status gets updated. Figure 6.10 shows an example of how the instances of $object2 extracted during U.4-U.6 in Table 6.5 are unified into one entity.

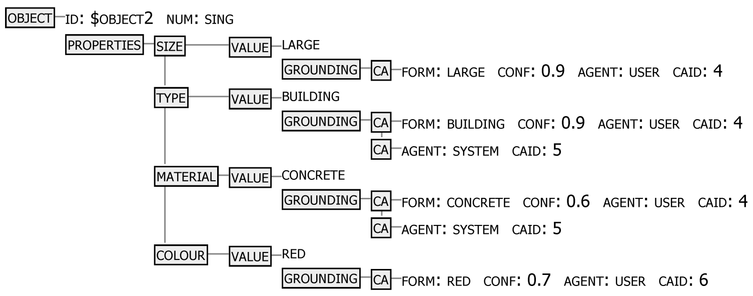

OBJECT ID: $OBJECT2 NUM: SING
PROPERTIES
SIZE
VALUE LARGE
GROUNDING CA FORM: LARGE CONF: 0.9 AGENT: USER CAID: 4
TYPE
VALUE BUILDING
GROUNDING CA FORM: BUILDING CONF: 0.9 AGENT: USER CAID: 4
CA AGENT: SYSTEM CAID: 5
MATERIAL
VALUE CONCRETE
GROUNDING CA FORM: CONCRETE CONF: 0.6 AGENT: USER CAID: 4
CA AGENT: SYSTEM CAID: 5
COLOUR
VALUE RED
GROUNDING CA FORM: RED CONF: 0.7 AGENT: USER CAID: 6

Figure 6.10: How the grounding status for entity $object2 in the entity list has been updated after U.6.

Since assertions about entities are unified in the entity list, it is possible to refer to an entity using a description that have not been used before to refer to that entity. For example, there is a reference in utterance S.13, in Table 6.1, 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.

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 6.1. If the entity is on top of the list, a simple pronoun may be used (unless the entity needs more grounding, which is described in 6.7.4 below). If there are other entities above it, the system may use a more elaborate definite noun phrase.

6.6 NAM: Navigation action manager

While GALATEA keeps the state of the discourse, the action manager(s) may keep the state of the system's intensions and its model of how the discourse entities map to objects in the database. This approach is different from the one taken in for example TrindiKit and the *information state approach* (Larsson & Traum, 2000), where all contextual information is kept in the same store. The purpose of this modularisation is to make the discourse modeller reusable, while the action manager may be highly domain dependent, implemented in any programming language, and limited in its tasks.

In the HIGGINS navigation domain, the navigation action manager (NAM) is the only module that has access to the map database and it is this module that performs the task-related decisions concerning the system's behaviour. Each time the discourse model gets updated, the NAM uses the entity list as a search pattern to find possible referents in the database, as described in 6.2. Table 6.6 below shows an example during turn U.4-U.6. This example shows how the value of one entity variable ($location1) may be constrained as more information is added about another entity ($object2), since the discourse model keeps information about the relations between these.

Table 6.6: How the possible bindings of the variable id's are constrained as more information is added.

		$location1	**$object2**
U.4	I have a large concrete building on my left [...]	loc734, loc82, loc293, loc83, loc94	obj25, obj04, obj73, obj94
S.5	*What colour is the concrete building?*		
U.6	Red	loc734, loc82	obj4, obj73

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. The decision algorithm for the NAM is shown in Table 6.7.

Table 6.7: The decision algorithm for the navigation action manager (NAM).

	Decision	Yes	No
1	Is the latest user utterance a request about the route?	Answer the request. End the turn.	Continue with 2.
2	Has the user stated the goal?	Continue with 3.	Request the goal. End the turn.
3	Is there any place that matches the goal description?	Continue with 4.	Tell the user that there is no such place. End the turn.
4	Has the user given any description of his location?	Continue with 5.	Request the user's position. End the turn.
5	Is there any location the user can be?	Continue with 6.	Perform late error detection and repair (described in 6.7.6).
6	Is the user's location exactly determined?	Continue with 11.	Continue with 7.
7	Is the user's location roughly determined?	Tell the user to position himself between two known objects in the vicinity. End the turn.	Continue with 8.
8	Are there a large number of possible user locations?	Ask the user to describe something more. End the turn.	Continue with 9.
9	Is there any entity in the entity list that may have several instances in the database and lacks description of properties?	Request more information about properties. End the turn.	Continue with 10.
10	Is it useful to ask a y/n-question about a specific object in a specific direction?	Ask the most optimal question. End the turn.	Ask the user to describe something else. End the turn.
11	Is the user at the goal?	Tell the user that he has arrived at the goal. End the turn.	Calculate the shortest path to the goal. Give a route direction to the next waypoint. End the turn.

For example, after U.2 in Table 6.1, the grounding action manager will first decide to display understanding of "an ATM" (as explained in 6.7.1 below). The NAM will then check the discourse model for the user's goal, and find that it is known. The next item on the check list is the user's position, and since there is no information about that in the discourse model, the NAM 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 turn S.7-U.8 from Table 6.1, where the system needs more information about the user's position:

(45) S.7: *Ok, can you see a wooden building?*
U.8: I can see a brown building.

In this example, the user does not directly answer the question. However, using the decision algorithm presented above, 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.

6.7 Error handling actions

By using the grounding status in the discourse model, the action manager(s) may perform various error handling actions, as described in this section.

6.7.1 GAM: Grounding action manager

As seen in the system architecture in Figure 6.5, the grounding action manager (GAM) is located before the navigation action manager (NAM) in the pipeline. The task of the GAM is to produce actions that are not dependent on the domain database. The GAM may do one of the following:

- Produce turn-yielding actions (such as clarification requests) and end the turn.
- Produce turn-keeping actions (such as acknowledgements) and pass the discourse model to the navigation action manager for more actions.
- Do nothing and simply pass the discourse model to the NAM to take actions.

This separation of action selection between the two action managers serves two proposes. First, since the GAM does not have to consult the database, it can typically act faster so that the system may be more responsive. Since the modules operate asynchronously, it may quickly produce actions (such as acknowledgements) that are performed while the NAM is processing. Second, since the GAM only reacts to the content in the discourse model (and does not consult any external knowledge sources), it is fairly generic. It is simply configured with a set of transformation rules written in XML, similar to the ones used in GALATEA for resolving ellipses. Whereas the transformation rules in GALATEA reinterpret new CA's based on past CA's, the transformation rules in the GAM produce new system CA's based on past CA's. The GAM decision algorithm presently used in the HIGGINS navigation domain is presented in Table 6.8.

Table 6.8: The decision algorithm for the grounding action manager (GAM).

	Decision	Yes	No
1	Was the latest user CA a request for repetition?	Repeat the last system CA. End the turn.	Continue with 2.
2	Was the latest user CA a request to wait?	Acknowledge. End the turn.	Continue with 3.
3	Did the user's last CA contain a value (or values) with a low grounding status?	Request clarification on the value or a whole object. End the turn.	Continue with 4.
4	Was the last user CA an assertion?	Acknowledge. Continue with 5.	Continue with 5.
5	Did the user's last CA contain a value with a medium grounding status?	Display understanding. Continue with 6.	Continue with 6.
6	Was the last user CA an expression of greeting or thanks?	Express greeting or thanks. Continue with 7.	Continue with 7.
7	Was the last user CA a fragmentary direction?	Ask the user what he can see in the direction. End the turn.	Continue with 8.
8	Was the last user CA a fragmentary object?	Ask the user if he can see the object. End the turn.	Send the discourse model to the NAM.

Currently, a very simple distinction is made between different levels of grounding status. If the grounding status contains a mention of the concept from the system, it is considered to be high. If the concept is only mentioned by the user, the highest confidence score is compared against a set of pre-defined thresholds. This simplistic model is refined later on in this thesis.

6.7.2 Fragmentary clarification

The following turns from Table 6.1 exemplify the use of fragmentary clarification:

(46) U.8: **I CAN SEE A BLUE BUILDING**
S.9: *Blue?*
U.10a: **NO**
U.10b: **BROWN**

This use of fragmentary clarification requests in spoken dialogue systems have not been studied to a great extent. As discussed in 3.3.2.5, if correctly handled, such requests may increase both the naturalness and efficiency of the dialogue. If the hypothesis is incorrect, the user should be able to efficiently correct the system, as in the example. To handle the turns in the example correctly, a system should be able to do the following things:

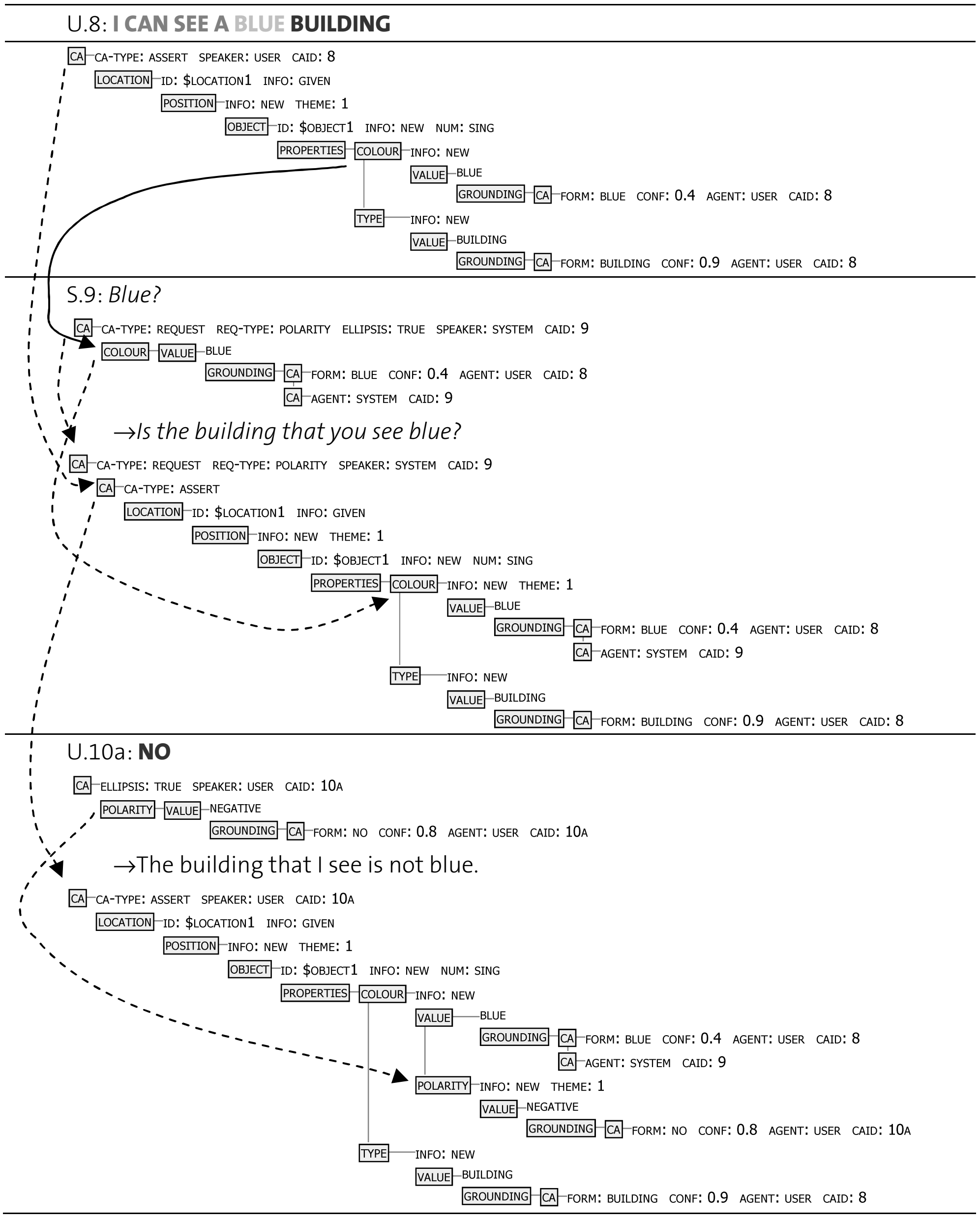
- Identify the problematic concept(s) (in U.8).
- Produce the request (S.9) accurately.
- Interpret the negation (U.10a) correctly. Notice that the user simply negates the proposed colour of the building – the fact that the user can see a building is still accepted.
- Interpret the correction (U.10b) correctly; to understand that the user can see a brown building.
- Understand that only the COLOUR concept has been grounded, not the entire contribution U.8.

We will now show how these requirements are handled in HIGGINS. As seen in Table 6.8, if a concept or a tree of concepts with low grounding status is detected in decision 3, the GAM may pose a fragmentary clarification request and end the turn. In HIGGINS, such utterances are not treated as a special kind of grounding or feedback utterance. Instead, they are resolved just like other ellipses into a full proposition. However, since grounding is modelled for all utterances, the clarification request will help to boost the weak grounding status. The clarification request is very simple to produce – the GAM simply has to embed the concepts in a CA of type REQUEST and send it to OVIDIUS (the natural language generator).

OVIDIUS will make a surface realisation of a fragmentary clarification request (with prosodic markup) and send it to the TTS (which is described in 6.7.7 below). When GALATEA receives this elliptical CA, it is 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. An example of how this is done for U.8-U-10a is shown in Table 6.9. As can be seen in the example, negations are represented with POLARITY nodes that are attached to concepts. This makes it easy to represent and integrate "no" answers, as well as adverbial negations.

Figure 6.11 shows the resulting entity in the entity list after the dialogue. 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. 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 another participant negates it. Thus, POLARITY nodes can be used to model the "acceptance" level discussed in 3.1.1. This also means that POLARITY nodes themselves may have a low grounding status, for example if the "no" in Table 6.9 would get a low confidence score, and need further grounding.

Table 6.9: How a fragmentary clarification request is constructed and interpreted. Dotted lines are part of ellipsis resolution in GALATEA. Solid lines are part of action construction in the GAM.


U.8: I CAN SEE A BLUE BUILDING
CA CA-TYPE: ASSERT SPEAKER: USER CAID: 8
LOCATION ID: $LOCATION1 INFO: GIVEN
POSITION INFO: NEW THEME: 1
OBJECT ID: $OBJECT1 INFO: NEW NUM: SING
PROPERTIES COLOUR INFO: NEW
VALUE BLUE
GROUNDING CA FORM: BLUE CONF: 0.4 AGENT: USER CAID: 8
TYPE INFO: NEW
VALUE BUILDING
GROUNDING CA FORM: BUILDING CONF: 0.9 AGENT: USER CAID: 8
S.9: Blue?
CA CA-TYPE: REQUEST REQ-TYPE: POLARITY ELLIPSIS: TRUE SPEAKER: SYSTEM CAID: 9
COLOUR VALUE BLUE
GROUNDING CA FORM: BLUE CONF: 0.4 AGENT: USER CAID: 8
CA AGENT: SYSTEM CAID: 9
→Is the building that you see blue?
CA CA-TYPE: REQUEST REQ-TYPE: POLARITY SPEAKER: SYSTEM CAID: 9
CA CA-TYPE: ASSERT
LOCATION ID: $LOCATION1 INFO: GIVEN
POSITION INFO: NEW THEME: 1
OBJECT ID: $OBJECT1 INFO: NEW NUM: SING
PROPERTIES COLOUR INFO: NEW THEME: 1
VALUE BLUE
GROUNDING CA FORM: BLUE CONF: 0.4 AGENT: USER CAID: 8
CA AGENT: SYSTEM CAID: 9
TYPE INFO: NEW
VALUE BUILDING
GROUNDING CA FORM: BUILDING CONF: 0.9 AGENT: USER CAID: 8
U.10a: NO
CA ELLIPSIS: TRUE SPEAKER: USER CAID: 10A
POLARITY VALUE NEGATIVE
GROUNDING CA FORM: NO CONF: 0.8 AGENT: USER CAID: 10A
→The building that I see is not blue.
CA CA-TYPE: ASSERT SPEAKER: USER CAID: 10A
LOCATION ID: $LOCATION1 INFO: GIVEN
POSITION INFO: NEW THEME: 1
OBJECT ID: $OBJECT1 INFO: NEW NUM: SING
PROPERTIES COLOUR INFO: NEW
VALUE BLUE
GROUNDING CA FORM: BLUE CONF: 0.4 AGENT: USER CAID: 8
CA AGENT: SYSTEM CAID: 9
POLARITY INFO: NEW THEME: 1
VALUE NEGATIVE
GROUNDING CA FORM: NO CONF: 0.8 AGENT: USER CAID: 10A
TYPE INFO: NEW
VALUE BUILDING
GROUNDING CA FORM: BUILDING CONF: 0.9 AGENT: USER CAID: 8

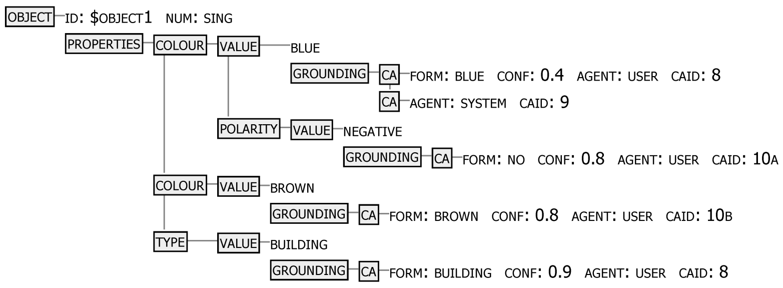


Figure 6.11: How the grounding status for entity $object1 in the entity list has been updated after U.3.

Like all requests, clarification requests do not need to be answered. The concepts which are to be clarified are not transferred to the entity list, since they have low grounding status. Thus, if the clarification request would not have been answered, there would be no information about the concept BLUE for this entity. This is also true if the user would have answered just “brown”, in which case the entity would have the concept BROWN, but no information on the concept BLUE. If the user reactions have low confidence scores, this will trigger new clarification requests. Of course, reactions in the form of full propositions are also possible, such as “I can see a brown building”.

The fragmentary clarification requests discussed above express request for confirmation for concepts that the system lacks confidence in. However, as discussed in 3.1.4, clarification requests may also be caused by (partial) lack of hypotheses and express request for repetition. The following example (taken from a real dialogue presented in the next chapter) illustrates such a request for partial repetition:

(47) S: *Can you see a brick building on your left?*
U: **NOW ON MY RIGHT** (No, on my right.)
S: *What do you see on your right?*

In this example, the system misrecognises the first part of the user correction (“no”) and GALATEA thereby fails to interpret the elliptical utterance “on my right”. However, this is recognised as an unresolved fragment containing a direction, and Decision 7 in Table 6.8 will lead the GAM to pose a clarification request for the missing object.

6.7.3 Separate display utterances

As seen in Table 6.8, decision 5 may lead the GAM to produce a display utterance, presumably after it has triggered on an assertion and produced an acknowledgement (decision 4). The NAM may then continue and produce the next task-related utterance. This is exemplified in Table 6.10.

Table 6.10: How a display utterance is selected and interpreted by GALATEA.

Turn	Decision	Before ellipsis resolution	After ellipsis resolution
S.1	NAM: 2-no	*Where do you want to go?*	*Where do you want to go?*
U.2		TO AN **ATM**	I want to go to an ATM.
S.3a	GAM: 4-yes	*Ok*	*Ok*
S.3b	GAM: 5-yes	*an ATM*	*you want to go to an ATM.*
S.3c	NAM: 4-no	*Can you describe where you are now?*	*Can you describe where you are now?*

The utterances “Ok” and “an ATM” are synthesised and played back while the NAM is asynchronously deciding on the next act. In this example, it takes a very short time to produce S.3c, and the utterance is simply queued up in the TTS. However, if the NAM had needed more time for database searches, this would have helped the system to act more responsively.

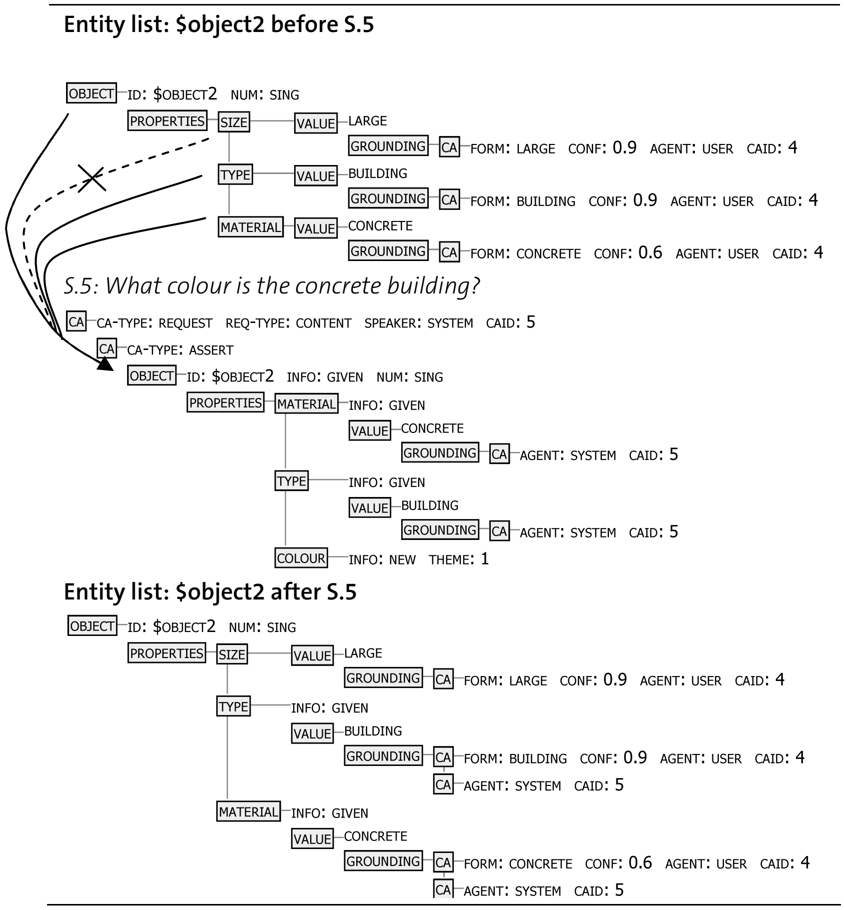
Display utterances are handled in a way very similar to fragmentary clarification requests. However, while these ellipses are resolved as requests, display utterances are resolved as assertions (“you want to go to an ATM”), which the user may object to. Since the concepts that are displayed have a higher confidence score, they are directly transferred to the entity list. Therefore, the user does not have to (but may, if he wish) confirm the displayed concept. If the user objects, a negative POLARITY node is attached.

It is also possible that the user might object to a displayed misunderstanding by other means, such as those listed in example (39) on page 57. Such objections could be handled by either representing them as a special type of negation (which are only treated as negations by GALATEA after a display of understanding), or let one of the action managers remove the erroneous concepts in the discourse model if such an objection is detected (see late error detection below).

6.7.4 Integrated display of understanding

Speakers do not only display their understanding using separate display utterances. They also do this to various extents while performing task-related CA’s (i.e., integrated display of understanding). As example (35) on page 54 shows, one way of doing this is to choose how to refer to entities. In HIGGINS, the NAM makes such decisions. Every time the NAM refers to a given entity, appropriate integrated display of understanding is automatically done. To construct a referring expression to an entity that is in the entity list, the NAM 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 6.11. 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 often needed for a valid referring expression – otherwise it would say "what colour is the concrete". Since GALATEA also models the system's actions, those concepts will then get a high grounding status.

Table 6.11: How the system creates a referring expression to $object2 and how this affects the grounding status of $object2 in the entity list.



```
Entity list: $object2 before S.5

OBJECT─ID: $OBJECT2  NUM: SING
       PROPERTIES─SIZE───────VALUE─LARGE
                   │               GROUNDING─CA─FORM: LARGE  CONF: 0.9  AGENT: USER  CAID: 4
                  TYPE───────VALUE─BUILDING
                   │               GROUNDING─CA─FORM: BUILDING  CONF: 0.9  AGENT: USER  CAID: 4
                  MATERIAL───VALUE─CONCRETE
                                   GROUNDING─CA─FORM: CONCRETE  CONF: 0.6  AGENT: USER  CAID: 4

S.5: What colour is the concrete building?

CA─CA-TYPE: REQUEST  REQ-TYPE: CONTENT  SPEAKER: SYSTEM  CAID: 5
  CA─CA-TYPE: ASSERT
     OBJECT─ID: $OBJECT2  INFO: GIVEN  NUM: SING
           PROPERTIES─MATERIAL─INFO: GIVEN
                       │       VALUE─CONCRETE
                       │             GROUNDING─CA─AGENT: SYSTEM  CAID: 5
                      TYPE─────INFO: GIVEN
                       │       VALUE─BUILDING
                       │             GROUNDING─CA─AGENT: SYSTEM  CAID: 5
                      COLOUR───INFO: NEW  THEME: 1

Entity list: $object2 after S.5

OBJECT─ID: $OBJECT2  NUM: SING
       PROPERTIES─SIZE───────VALUE─LARGE
                   │               GROUNDING─CA─FORM: LARGE  CONF: 0.9  AGENT: USER  CAID: 4
                  TYPE───────INFO: GIVEN
                   │         VALUE─BUILDING
                   │               GROUNDING─CA─FORM: BUILDING  CONF: 0.9  AGENT: USER  CAID: 4
                   │                         CA─AGENT: SYSTEM  CAID: 5
                  MATERIAL───INFO: GIVEN
                             VALUE─CONCRETE
                                   GROUNDING─CA─FORM: CONCRETE  CONF: 0.6  AGENT: USER  CAID: 4
                                             CA─AGENT: SYSTEM  CAID: 5
```

6.7.5 Non-understanding recovery

As discussed in 3.3.2.4 and as the results of the experiment in Chapter 4 suggest, it may not be optimal to signal non-understanding when the system does not understand what the user is saying, even if there is a complete non-understanding. Instead, humans tend to ask new task-related questions. This behaviour is implemented in HIGGINS in the following way. Every time the NAM selects an action according to Table 6.7, it stores this as the last utterance. If the next user utterance is not understood by the system, the decision algorithm interpreter is programmed to not produce the same action again. For many decisions, there are several possible outcomes. For example, 10-yes may result in different questions. The most optimal question is selected first, but after a non-understanding, the system will be hindered to ask this question again, and will thus select the second most optimal question. Here is an example:

(48) S: *Can you see a tree in front of you?*
U: [non-understanding]
S: *Can you see a bus stop on your left?*
U: yes

6.7.6 Late error detection and repair

Due to the misrecognition in U.4 (in Table 6.1), 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. The decision 5-no in Table 6.7 tells the NAM that it should now do late error detection and repair. To do this, it looks through the discourse model to find concepts with low grounding status. The only concept with a relatively low grounding status is shown in Figure 6.12.

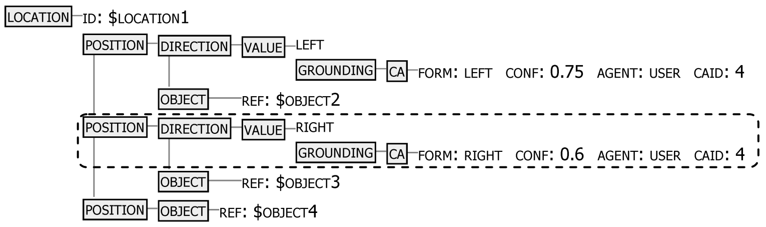


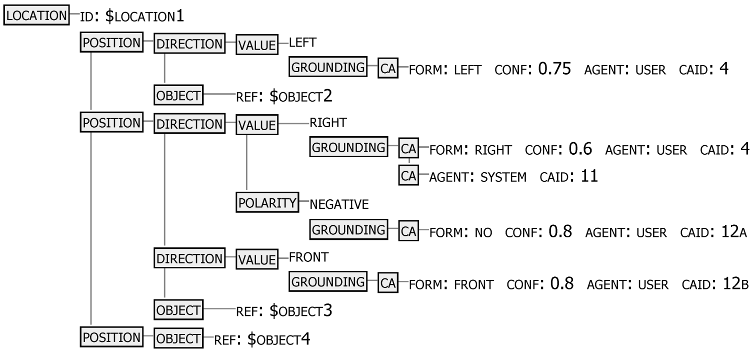
Figure 6.12: A potential error is detected in the user location.

One way of repairing the potentially erroneous concept is to remove it from the discourse model and search for possible user locations again. Note that this only would be done on the concept level; the system would still keep the information that the user can see a tree somewhere.

In this example, the system instead chooses to make a late clarification request. Table 6.12 shows how this clarification is interpreted. Late clarification requests are realised as full propositions and not as fragments, since this would generally not be suitable (the context needed for their resolution is too distant). Figure 6.13 shows the resulting user location entity after this clarification.

Table 6.12: The interpretation of the late clarification request.

Turn	CA	Resolved by GALATEA
S.11	*Do you really have a bus stop on your right?*	
U.12a	**NO**	I do not have a bus stop on my right.
U.12b	**IN FRONT**	I have a bus stop in front of me.



LOCATION ID: $LOCATION1
POSITION
DIRECTION
VALUE LEFT
GROUNDING CA FORM: LEFT CONF: 0.75 AGENT: USER CAID: 4
OBJECT REF: $OBJECT2
POSITION
DIRECTION
VALUE RIGHT
GROUNDING CA FORM: RIGHT CONF: 0.6 AGENT: USER CAID: 4
CA AGENT: SYSTEM CAID: 11
POLARITY NEGATIVE
GROUNDING CA FORM: NO CONF: 0.8 AGENT: USER CAID: 12A
DIRECTION
VALUE FRONT
GROUNDING CA FORM: FRONT CONF: 0.8 AGENT: USER CAID: 12B
OBJECT REF: $OBJECT3
POSITION
OBJECT REF: $OBJECT4

Figure 6.13: The user location after the late clarification request.

It is also possible to remove information that is associated with a specific turn, by looking at the CAID attribute in the grounding status, for example after the user objects to a display of understanding. Since the model also contains information about what the user has grounded, it is also 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 6.1, and this can be detected in the discourse model.

When the system discovers that there is no place where the user can be, there will sometimes be no information with low grounding status in the discourse model that may be removed. In such cases, information about the user's location will simply be cleared in the discourse model and the system will say "sorry, I have lost you". Decision 4 in Table 6.7 will then lead the NAM to produce something like "can you describe where you are".

6.7.7 Utterance generation

As can be seen in the system architecture in Figure 6.5, the textual representation of system utterances are realised by a module called OVIDIUS, also implemented in Oz. OVIDIUS takes a system CA as input and generates a text that is to be synthesised by a TTS. The text may also have some prosodic markup.

OVIDIUS uses a set of template rules, working much like inverted PICKERING grammar rules – they match on semantic structures and produce text strings. The processing is similar to XSLT (Clark, 1999): one template rule may call another template rule to generate sub-parts of the semantics (such as nominal phrases). Much of the vocabulary expected from the user is also used in the system output. To make the input and output vocabularies coherent and to reduce double work, OVIDIUS may use the same lexicon that is used in PICKERING to map semantic structures to surface strings.

As discussed in 2.3.4, humans engaging in dialogue tend to coordinate their linguistic behaviour with each other. Users of spoken dialogue systems tend to adjust the vocabulary and expressions to the system and may expect the system to do the same. When a speaker deliberately chooses another wording than their interlocutor, the other speaker may feel obliged to adapt to this. This phenomenon is especially important to understand when making clarification requests. Consider the following example:

(49) U.1: I have <u>a violet house</u> on my left.
S.2: *<u>A purple building</u>?*

In S.2, the system makes a surface realisation of the semantic representation of the fragment to be clarified. The template rules and lexicon allows OVIDIUS to generate different forms from the same semantic representation – the underlined phrases have exactly the same semantic representations (in this specific domain). Thus, it would make perfect sense to the system to produce S.2 in order to clarify the referring expression. However, this may be confusing for the user, who may get the impression that the system is trying to correct her or enforce a certain language use. In HIGGINS, this is avoided by tracking the surface form in the grounding status, as mentioned previously. As can be seen in Table 6.9, when a clarification ellipsis is generated in the GAM, the previous grounding status of the concepts to be clarified is not removed. Figure 6.14 shows what the message from the GAM to OVIDIUS looks like when utterance S.2 in example (49) above is to be generated. The FORM attribute in the grounding status helps OVIDIUS to select the same lexical entry that was used when parsing the original user utterance.

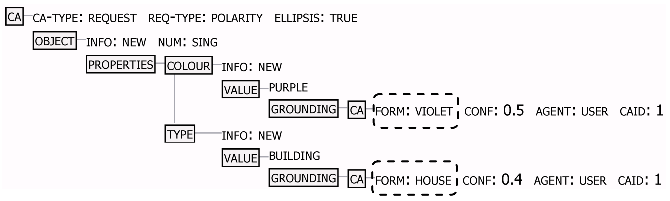

CA CA-TYPE: REQUEST REQ-TYPE: POLARITY ELLIPSIS: TRUE
OBJECT INFO: NEW NUM: SING
PROPERTIES COLOUR INFO: NEW
VALUE PURPLE
GROUNDING CA FORM: VIOLET CONF: 0.5 AGENT: USER CAID: 1
TYPE INFO: NEW
VALUE BUILDING
GROUNDING CA FORM: HOUSE CONF: 0.4 AGENT: USER CAID: 1

Figure 6.14: The semantic representation of S.2 in example (49), to be transformed into surface form by OVIDIUS.

Fragmentary clarification requests and display utterances have the same textual form. The interpretation of these utterances is therefore dependent on prosody to a large extent. For speech synthesis, the KTH rule-based speech synthesis system (Carlson et al., 1982) was used, together with a diphone Swedish male MBROLA voice (Dutoit et al., 1996). The only available parameters for this TTS were pitch range, speaking rate and pitch base. Since fragmentary grounding utterances are dynamically generated, it is not possible to hand-craft the pitch curve for each utterance. To approximate the commonly described tonal characteristic for questions – overall higher pitch (Hirst & Cristo, 1998) – the pitch range was initially simply increased for fragmentary clarification requests, in order to distinguish them from display utterances. In Chapter 9, we will explore the relationship between the prosodic realisation of fragmentary grounding utterances and the interpretation of them in more depth.

6.8 Discussion

This chapter has focussed on how to model the way all utterances may provide evidence of understanding, not just special "grounding acts". Thus, as the system gives the user a route direction (a mainly task-related act), it may simultaneously display its understanding by the way it refers to landmarks. Another implication is that for example clarification requests are treated in the same way as other requests and not as a special type of utterance. The tracking of groundings status ensures that the evidence that is simultaneously provided is accounted for. It has also been shown how the grounding status is tracked while performing ellipsis and anaphora resolution. This means that the choice of referring and elliptical expressions affects the system's model of what has been grounded.

The notion of high and low grounding status is very simplistic. A more advanced model, where the grounding status could be evaluated as a continuous (possibly probabilistic) score – similar to the "belief updating" approach in Bohus & Rudnicky (2005a) – would be interesting to explore. To improve such classification, the grounding status could be enriched with more information on how the concept was grounded, such as prosodic information. The confidence scores used for the grounding status are taken directly from the speech recogniser. An interesting improvement would be to include the methods for early error detection explored in Chapter 5.

Currently, the thresholds used for choosing error handling strategies have been manually tuned. Also, the choice of strategy is not dependent on the current task. For example, when the user asserts her goal, as in U.2 in Table 6.1, the system might navigate the user to the wrong place and have to start all over again. Thus, the system should have a relatively high threshold for accepting such hypotheses, since the cost of task failure is higher. When the user asserts her position, as in U.4, an error would probably be detected rather early, when the system finds out that there is no place the user can be, and the threshold should be lower. In Chapter 8, we will show how a decision-theoretic framework might be applied to make such choices.

In this chapter, we have only described how evidence may be given on the perception level (with the reading "did you say X?"). Since the domain encourages the use of ellipses and anaphora, it would be useful to also give evidence on the understanding level, as in the following example:

(50) U: Now I can see the building.
S: *The red building?*

In this example, the system does not make a clarification request due to low ASR confidence score, but rather because the reference resolution is ambiguous. To handle this, GALATEA should be endowed with the capability of detecting that a referring expression is ambiguous and, if this is the case, not resolve the expression, but to let the GAM pose a clarification request.

Neither does the system currently handle fragmentary clarification requests from the user, since this would require prosodic analysis. This might be a problem if the users adapt to the system and start to make such requests (which they are likely to do).

An important question is to what extent the methods and models described in this chapter may apply to other domains. The HIGGINS components have been used in Connector, a dialogue system acting as an automatic switchboard and secretary (Edlund & Hjalmarsson, 2005). Connector is part of the EU-funded CHIL-project, 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.

6.9 Summary

In this chapter, we have described how speech recognition errors are handled in the HIGGINS spoken dialogue system. It has been shown how all utterances may operate on the domain level, while simultaneously providing evidence of understanding. The discourse modeller GALATEA keeps track of this by modelling the grounding status of concepts while resolving ellipses and anaphora. The grounding status includes information such as concept confidence scores and surface realisation, which are extracted by the robust interpreter PICKERING. The grounding status may be used by a set of action managers to perform concept-level error handling, such as display of understanding, clarification requests and misunderstanding repair.

CHAPTER 7

Higgins evaluation

In the previous chapter, the HIGGINS system was presented with a focus on concept-level error handling in the robust interpreter PICKERING and the discourse modeller GALATEA. In this chapter, two evaluations of these HIGGINS components are presented based on different data sets. The first is an evaluation of PICKERING, performed before the complete system was put together, and it is intended to explore how some of the robustness features implemented in PICKERING contribute to the performance. In the second evaluation, naive users were allowed to interact with the complete HIGGINS system. This evaluation is focussed on the performance of GALATEA, but also on users' behaviour when faced with fragmentary clarification requests.

7.1 PICKERING evaluation

A robust interpreter may be too allowing and, as a result, find incorrect interpretations. Hence, it cannot be taken for granted that the techniques PICKERING employs (as described in 6.4.2) increase the performance of the interpreter under error conditions. To check for this, PICKERING was evaluated with respect to how it performs under different error conditions, and how its performance depends on the robustness techniques. The evaluation included data with varying degrees of errors, to ensure that the interpreter works well under perfect conditions and degrades gracefully in the presence of errors.

7.1.1 Method and data

PICKERING was evaluated before the complete system was built. In order to collect data, eight subjects, all native speakers of Swedish, were given the task of moving around in the virtual 3D-city while they described their positions relative to objects in their surroundings. This resulted in 340 utterances similar to those used during the positioning phases in the pedestrian navigation domain. The utterances were transcribed and a PICKERING grammar was written to cover the syntax and semantics that were deemed to be relevant to the domain and the objects and properties contained in the database.

For evaluation, 16 new subjects were recorded and their utterances transcribed using the same procedure. Due to time limitations, a test set of 100 of their utterances were manually annotated with semantics to create a gold standard for the evaluation. The average utterance length of the utterances was 11 words. The utterances were recognised using the KTH LVCSR speech recogniser (Seward, 2003) with a trigram language model. In order to study the effect of varying levels of errors in the ASR results on the PICKERING performance, the ASR was run repeated times on the material with the beam pruning level[9] set at different values. 18 pruning levels where used; the lowest pruning yielded an average WER rate of 34.2% and the highest 95.3%.

Next, the transcriptions and the ASR results were processed by PICKERING and compared to the gold standard in order to study the effects of ASR errors. The following interpreter parameters were systematically varied and combined:

- Agreement:
    - Weak: Agreement inside phrases (mostly congruence in nominal phrases) was preferred, but not required
    - Strong: Agreement inside phrases was required
- Permitted insertions inside phrases:
    - 0, 1, 2 or unlimited.

To simplify comparison of the results with the gold standard, an approximation was made by flattening the semantic trees to form lists of minimal concepts (i.e., nodes in the trees). On average, there were 32 such concepts per utterance. The lists were compared to the gold standard using standard WER calculation to get a concept error rate (CER). The percentage of concept insertions and deletions were calculated separately. Substitutions were counted as a combination of a deletion and an insertion, and were added to both groups.

The lexicon of the interpreter was also used to identify keywords in the utterances (i.e., words carrying semantics). This way, it was possible to estimate performance of PICKERING compared to that of a simple keyword spotter.

7.1.2 Results

For transcribed utterances (i.e., WER=0), the CER was 20%, with 12.6% deletions and 9.9% insertions (using the default setting, arbitrarily chosen when PICKERING was implemented: 2

[9] The beam pruning level determines how much of the ASR HMM space is explored for the optimal solution. A high pruning level means that less space is explored, which increases the speed of the ASR but reduces the accuracy.

allowed insertions, weak agreement). Figure 7.1 shows how the performance for insertions and deletions (Y-axis) varies as WER increases. The X-axis represents the mean WER for the different beam pruning settings. The figure also includes the ASR insertion/deletion performance for keywords.

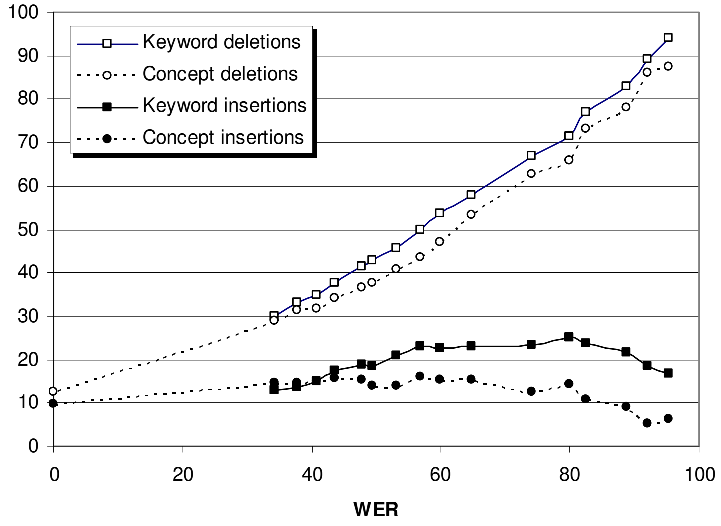


Figure 7.1: The percentage of insertions and deletions for keywords and concepts, depending on mean WER.

The figure shows that deletions rise steadily both for concepts and keywords, while insertions rise to a certain peak for keywords, but not for concepts. This is probably due to the fact that PICKERING may ignore erroneous content words in the input that do not fit into any syntax, as described in 6.4.2.1. To test the significance of the differences between keywords and concepts, the areas between the lowest and the highest WER's under the curves shown in the figure were compared for all utterances. There were significant differences between keywords and concepts for both insertions and deletions (two-tailed paired t-tests; dF=106; $p<0.05$).

To compare the effects of different parameter settings on the general robustness of PICKERING, a mean relative CER score (CER minus WER) was calculated for each parameter setting, which is shown in Figure 7.2. In order for this score to reflect the general performance over the whole WER span (34-95%), the average of all values of CER (relative to WER) along this span was calculated.

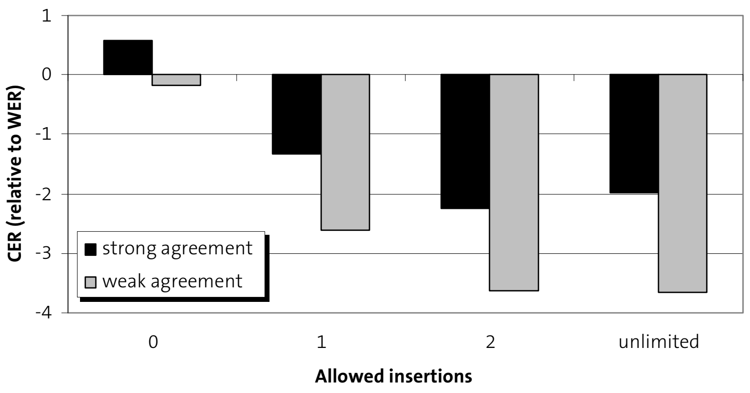


Figure 7.2: Mean CER (relative to WER) depending on parameter settings.

There were main effects for both strength of agreement and number of allowed insertions (two-way repeated measures ANOVA; dF=106; p<0.05). Post tests revealed that there were significant differences between 0, 1 and 2 insertions (p<0.05), but the performance did neither decrease nor increase when more than two insertions were allowed.

The results indicate that PICKERING generalises well when applied to unseen utterances, within the limited domain, produced by new speakers. As the WER of the ASR output increases, the set of robustness techniques utilised leads to graceful degradation of the interpretation results. The two techniques used to relax the CFG-constraints that were tested – allowing non-agreement and insertions – both improved performance. Since allowing insertions increases the size of the chart during parsing and thus slows down the computation, there seems to be no reason for allowing more than two insertions (given the circumstances).

7.2 GALATEA 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 error handling techniques; the analysis of the results will focus on robustness, ellipsis resolution and fragmentary clarification.

7.2.1 Method

7.2.1.1 Subjects

16 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.

7.2.1.2 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.

The subject was placed in a sound-proofed room in front of a computer screen, where the 3D model of the virtual city was displayed from a first-person perspective, as seen in Figure 6.1. The subject spoke to the system through a headset and used a mouse to control the movement in the virtual city. 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. 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.

7.2.1.3 Data, ASR and TTS

The system was trained and configured mainly based on two different sets of data: the data presented in Chapter 4 and the data used for the evaluation of PICKERING. 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 language models. Since the ASR used for the PICKERING evaluation presented above did not support word confidence scores, an off-the-shelf ASR with such support was used. A trigram class-based language model was used, trained on the 1500 utterances that had been collected. Words that only occurred once in the training material were pruned, resulting in a vocabulary size of approximately 600 words.

As described in 6.7.7, a diphone Swedish male MBROLA voice was used for TTS and a very simple model for generating fragmentary clarification requests was used.

7.2.2 Results

7.2.2.1 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 – 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 CA's were not detected at all by the ASR, some were split over several ASR results, some ASR results contained several CA's. There were a total of 1894 ASR results and 2007 CA's, 1565 of the ASR results contained only one unsplit CA.

Both ASR results and CA's were annotated based on how well they were understood by the system. In the separate evaluation of PICKERING presented above, the commonly used measure of concept error rate (CER) was used. In this evaluation, however, the result that is to be assessed is an updated discourse model, including identified referents and enriched fragments. It would be too time-consuming to hand-craft a target discourse model to compare with for each utterance, at least if the whole data set is to be evaluated. To make the analysis more straightforward and the results easier to understand, five different levels of understanding were defined, similar to those used in Chapter 4. These levels are described in Table 7.1.

Table 7.1: The definitions of the understanding levels used in the annotation of ASR results and CA's.

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; fragmentary utterances have to be correctly transformed into full propositions, and anaphoric expressions correctly resolved. This includes cases where the system asks for clarification on some of the concepts that it lacks confidence in.
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 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.)

The CA's were also annotated based on the form and function of the utterance, but 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 7.2.

Table 7.2: A dialogue excerpt from the evaluation data, translated from Swedish. The third column shows ASR results in which word confidence scores are represented with different shades of grey, where dark means high confidence. 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 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

7.2.2.2 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 3.8 words per CA. This average figure is weighted down by the frequent use of short acknowledgements (like "yes" and "ok") during the route description phases of the dialogues, as well as the high number of fragmentary utterances. For assertions, the average number of words per CA were 7.3. 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 7.3 shows the distribution of understanding levels for ASR results and CA's, as well as for the ASR results containing just one CA, and for the subset of these that had a WER of 0%.

Table 7.3: The number of instances and distribution of understanding for ASR results and CA's. 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 7.1.

	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 %		

The rightmost column in Table 7.3 reflects the performance of PICKERING and GALATEA, that is, how well the rules written to handle the training data generalised to new data, given that there are no ASR errors. Considering the limited "training data", 92.9% FULLUND should be regarded as a promising performance. This also confirms that the ASR is the major source of errors.

The third column (*CA's*) shows that 8.0% of the CA's were not detected by the system at all. This is partly explained by the fact that there were a lot of feedback utterances from the user (such as "mhm") after separate display utterances from the system, which did not have enough intensity to trigger the voice activity 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 7.2, where the utterance ends while S.9 is spoken.

7.2.2.3 Robustness and early error detection

To study the robustness of PICKERING and GALATEA against ASR errors, the ASR results were divided in WER intervals. The distribution of understanding for these spans is shown in Figure 7.3.

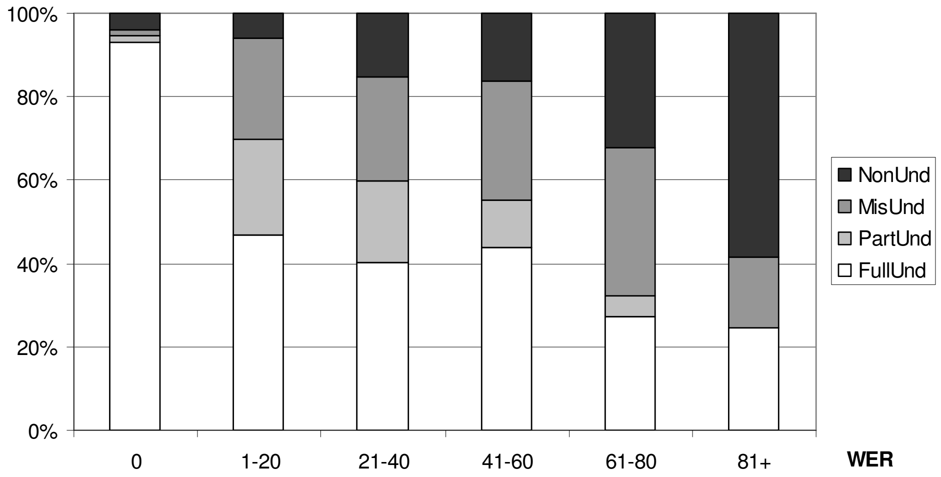


Figure 7.3: The distribution of understanding depending on the WER of the ASR result (rounded up).

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.

7.2.2.4 Ellipsis resolution

To find out how well GALATEA manages to resolve ellipsis, correctly recognised task-related complete CA's were grouped based on their form. Table 7.4 shows the distribution of understanding for some relevant forms. In this context, the form "fragment" includes adjectives, nouns, nominal phrases, propositional phrases, etc. As the table 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 7.2 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.

Table 7.4: The distribution of understanding for recognised task-related complete CA's 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 %

7.2.2.5 Fragmentary clarification

There were a total of 94 fragmentary clarification requests in the data. Of these, 68.1% followed upon a correct hypothesis of the user's utterance and 31.9 % followed upon an incorrect hypothesis (i.e., 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 7.5 shows the distribution of these types.

Table 7.5: Immediate user reaction to fragmentary clarification requests. The second column shows the distribution for cases where the request followed upon a correct hypothesis, and the third column cases where the request followed upon an incorrect hypothesis.

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 %

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 may be explained partly by the fact that users may not expect such human-like behaviour from dialogue systems. Another explanation is that the clarification requests were sometimes used in contexts where a human would not have used them. A third explanation is that the prosodic model used to realise them (as described in 6.7.7) was very simplistic and not tested. This issue is explored in more depth in Chapter 9.

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 request "red?" after a misrecognised "I want to go to a bus stop" may be perceived as inadequate.

There were many cases where the users ignored the clarification request after a correct recognition. However, it is possible that these 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 concepts that are clarified must be confirmed to be considered as being 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 "subdialogue", but rather as a signal from the system that it lacks understanding, in the form of a request, which 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 doubted the correctness of the user's description.

As discussed in Schlangen & Fernández (2007a), the frequent use of elaboration as a reaction to clarification requests may suggest that the users interpret them as concerning the understanding level and not the perception level (as discussed in 3.1.4), that is, they may have interpreted them as "do you really mean X", instead of "did you say X". If that was the case, it is possible that these interpretations were caused by the simplistic prosodic model used (which is explored further in Chapter 9).

Table 7.6 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 CA's in general (compare with Table 7.3). However, the performance is poorer for responses to requests based on misunderstandings, reflecting the fact that these were less predictable.

Table 7.6: 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 %
NONHEAR	7.8 %	6.7 %

7.2.2.6 Late error detection

The log files from the action manager showed that there were a total of 78 cases where there was not any 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, as it is beyond the scope of this thesis.

7.3 Summary

An evaluation of the robust interpreter PICKERING indicates that it generalises well when applied to unseen utterances, within the limited domain, produced by new speakers. As the WER of the ASR output increases, the set of robustness techniques utilised leads to graceful degradation of the interpretation results. The two techniques used to relax the CFG-constraints that were tested – allowing non-agreement and insertions – both improved performance. Allowing an unlimited number of insertions into syntactical structures caused neither decline nor increase in accuracy.

An evaluation of the complete HIGGINS system showed that the performance of GALATEA and the rest of the system looks promising, not only when utterances are correctly recognised, but also when ASR errors are introduced. There have previously not been many studies on the use of fragmentary clarification requests in spoken dialogue systems interacting with real users. The results from this evaluation show that users may have difficulties understanding these requests, especially after incorrect speech recognition hypotheses, and that a more elaborate model of when to use them, how to realise them, and how to understand the user's reaction to them is needed.

PART IV

Deciding and Realising Grounding Actions

DAVE: Open the pod bay doors, HAL.
HAL: I'm sorry Dave, I'm afraid I can't do that.
DAVE: What's the problem?
HAL: I think you know what the problem is just as well as I do.
DAVE: What are you talking about, HAL?
HAL: This mission is too important for me to allow you to jeopardize it.

2001 – A Space Odyssey
screenplay by *Stanley Kubrick* and *Arthur C. Clark*

CHANCE: I like to watch.
EVE: To watch...? To watch me...?
CHANCE: Yes. I like to watch.

Being There
screenplay by *Jerzy Kosinski*

CHAPTER 8

Making grounding decisions

Given a speech recognition hypothesis, a dialogue system has the choice of accepting or rejecting this hypothesis, but can also choose to provide evidence of understanding, such as a clarification request, or display its understanding. In 3.3.2.7, this choice was referred to as the *grounding decision problem*. In the previous chapters, a static model with hand-crafted thresholds was used. In this chapter, we will use a data-driven decision-theoretic model for the grounding decision problem. Based on task analysis of the HIGGINS navigation domain, dialogue cost functions will be derived, which take dialogue efficiency, consequence of task failure and information gain into account. The dialogue data presented in the previous chapter will then be used to estimate parameters for these cost functions, so that the grounding decision may be based on both confidence and dialogue context.

8.1 The grounding decision problem

The approach to grounding decisions used in the previous chapters (and which is used in many other dialogue systems) is to simply accept a speech recognition hypothesis when the confidence score is high, display understanding for middle-high scores, make a clarification request for middle-low scores and reject the hypothesis for low scores. The problem is that the confidence thresholds for these decisions are most often (as in the previous chapters) based on intuition and not on any theoretically sound and empirically based principle.

In 3.1.2 three important factors for making this decision were discussed, which we may summarise as follows:

1. The result of the early error detection: how confident the system is in its understanding.
2. Task consequences: the cost of falsely accepting an hypothesis (i.e., a misunderstanding), as well as the cost of a false rejection (i.e., a non-understanding).
3. The cost of realising the grounding action and possible reactions to it.

In the simplest case, the choice is between accept and reject, and only Factor 1 above (confidence of understanding) is considered, by comparing the confidence score against a static confidence threshold. This threshold may be optimised to minimise the sum of false acceptances and false rejections, as described in 3.3.1.2, assuming that these errors have the same costs associated with them.

In order to take Factor 2 (task-related costs and utility) into account, Bohus & Rudnicky (2001) use a data-driven technique to derive actual costs in data from the CMU Communicator system, which showed that false acceptances were more costly than false rejections.

Another aspect is that the task costs often vary depending on dialogue state. To incorporate this aspect, Bohus & Rudnicky (2005c) present a method where binary logistic regression is used to determine the costs (in terms of task success) of various types of understanding errors involved in the rejection trade-off. Different regressions may then be calculated in different dialogue states, resulting in dynamic thresholds. Surprisingly, for many dialogue states, the optimal threshold was 0 (i.e., accept everything).

However, none of these methods consider other grounding options than accept and reject and Factor 3 above (cost of grounding actions) is not considered. In some machine-learning approaches to early error detection or n-best list reordering, the machine-learner has been trained to not only consider accept and reject, but also grounding acts such as clarification (Gabsdil & Lemon, 2004; Jonson, 2006). The problem here is how to annotate the training material. When should the system ideally make a clarification? If we know that the hypothesis is correct in the training material, the desired action would be to accept, and if it is incorrect, the desired action would be to reject. Gabsdil & Lemon (2004) suggest that the system should reject when the WER falls below 50% and clarify above that threshold. However, no theoretical motivation for this is provided.

8.1.1 A decision-theoretic approach

Paek & Horvitz (2003) present a decision theoretic approach to the grounding decision problem, based on the framework of *decision making under uncertainty*. According to this proposal, the optimal grounding action *GA* should satisfy the Principle of Maximum Expected Utility (MEU), which can be defined as follows: *Choose an action a, so that the expected utility EU(a) is maximised.* When making this decision, the world may be in one of the states h_1, h_2, $h_3 \ldots h_n$, and this state may have an impact on the effect of the action taken. This effect can be described by the function *Utility(a,h$_i$)*, which is the utility for action *a* under state *h$_i$*. Thus, for each action *a*, the probability for each possible state and the utility for taking action *a*, given that state, should be summed up:

(51) $$GA = \arg\max_a EU(a) = \arg\max_a \sum_{i=1}^{n} P(h_i) \times Utility(a, h_i)$$

In Paek & Horvitz (2003), the utilities used in the model were estimated directly by the dialogue designer. In this chapter, we will move one step further and show how this may be estimated from data. We will also show how the model may account for both task-related costs and grounding-related costs, thus accounting for all decision factors discussed above. Before presenting the model, we give a brief overview of the research done on data-driven action selection.

8.1.2 Data-driven action selection

As noted in 2.3.3.3, a lot of recent effort has been invested in making action selection in spoken dialogue systems data-driven, and the grounding decision problem is clearly an instance of action selection.

One approach to data-driven action selection is supervised learning, where a dialogue corpus is used to learn a mapping between the current dialogue state and the action to be taken. The main problem with this approach is how to collect the large amount of data that is needed. The data should contain human-computer dialogues that are representative for the system that is to be built. One possibility could perhaps be to use a complete spoken dialogue system for the target domain interacting with users to collect the data, but this is normally not available (otherwise one would not want to build the system). Also, the machine learner would probably just learn the strategies already utilised by the system. Another solution is to use a human operator acting as a dialogue manager in a Wizard-of-Oz setting. However, since the amount of data that is needed typically is very large, this may be costly to perform. Such an approach also rests on the assumption that the Wizard's behaviour is an optimal model for dialogue system behaviour. The approach is perhaps more suitable for learning general policies for very specific choices. Bohus & Rudnicky (2005b) is an example of this.

Another data-driven approach is to model action selection as a Markov Decision Process (MDP). MDP's consist of a state space with transition probabilities and cost assignments. Unlike supervised learning, an MDP chooses actions that maximise a long-term cumulative sum of rewards (such as user satisfaction). Thus, it can be said to perform planning. Such a model is trained by reinforcement learning, so that the long term reward may be propagated to the different decisions that led to the outcome. An obvious problem is that reinforcement learning may need even more data than supervised learning. To solve this problem, Levin et al. (2000) present an approach in which they estimate a *user model* (MDP parameters that quantify the users' behaviour) by training a supervised learner on a smaller amount of dialogue data. Reinforcement learning is then used to estimate optimal policies by interacting with the simulated user. Levin et al. (2000) show how their system may learn policies for collecting information from the user that seem to be intuitively sound.

A shortcoming with MDP's is that the current state is supposed to be known. This is problematic, since a fundamental problem for spoken dialogue systems is to deal with uncertainty. Williams & Young (2007) proposes the use of an even more advanced stochastic model for action selection: a Partially Observable Markov Decision Process (POMDP). The strengths of POMDP models are that they combine the techniques of automated planning with parallel dialogue state hypotheses and the use of confidence scores, into one statistical framework that admits global optimisation. However, as pointed out by Williams & Young (2007), it is computationally challenging to scale POMDP models to real-world problems, and it is yet unclear whether they will be applicable to more complex domains. Also, a more general concern for data-driven methods relying on user models is how representative such models are of real users.

8.2 The proposed model

In this chapter, we will show how the utilities in the decision-theoretic model discussed in 8.1.1 above may be estimated directly from a small amount of collected dialogue data, based on task-analysis and boot-strapping. To do this, the problem will be described as that of minimising costs, and a general cost measure will be defined. If we want to consider costs instead of utilities, the principle of MEU can be transformed into the Principle of Minimum Expected Cost (EC), where cost should be understood as negative utility:

(52)
$$GA = \arg\min_{a} EC(a) = \arg\min_{a} \sum_{i=1}^{n} P(h_i) \times Cost(a, h_i)$$

Now, this can be applied to the grounding decision problem in the following way: *Choose a grounding action a, so that the sum of all task-related costs and grounding costs is minimised, considering the probability that the recognition hypothesis is correct.* Thus, the world may be in two states (*correct* and *incorrect* recognition), and a probability measure for these states is needed, as well as a cost function for calculating the costs of the different grounding actions, given these states. The problem is expressed in the following equation (where *P(incorrect)* equals 1-*P(correct)*):

(53)
$$GA = \arg\min_{a} \begin{pmatrix} P(correct) \times Cost(a, correct) + \\ P(incorrect) \times Cost(a, incorrect) \end{pmatrix}$$

In this chapter, these cost functions will be defined by analysing the consequences of different grounding actions. A unified cost measure (accounting for both task-related and grounding costs) will be defined, and the cost functions will use parameters that can be estimated from data.

The proposed model rests on some simplifying assumptions, which will be discussed in more detailed later on:

- Only one concept in the recognised utterance will be considered as being correct or incorrect.
- The possibility that an incorrect recognition hypothesis may be a substitution for a similar concept is not considered. Alternative hypotheses are not considered.
- The costs and probabilities are not dependent on the dialogue history. For example, the utility of grounding actions do not change when they are repeated subsequently.

To select the optimal grounding action according to equation (53) above, a probability measure of the state *correct* is needed, as well as a cost function for calculating the costs of the different grounding actions, given these states.

8.2.1 P(correct)

The most obvious candidate for an estimation of *P(correct)* is the speech recognition confidence score. Although this score should generally not be used directly as a measure of probability (as discussed in 3.3.1.1), it should be possible to approximate such a probabilistic score by using a phoneme recogniser, filler models, or deduce it from the word graph, as argued by Wessel et al. (2001).

Another possibility is to deduce a probabilistic score given a specific application and data collected within it. We will here analyse the confidence scores obtained in the data collection presented in the previous chapter. In this collection, an off-the-shelf ASR was used, and we did not have access to the exact workings of the ASR confidence scoring. In HIGGINS, the word confidence scores from the ASR are averaged into concept confidence scores, as described in 6.4.2.4. To analyse the relation between these confidence scores and *P(correct)*, all recognised concepts in the data were divided into ten interval groups, depending on their confidence scores, that is, scores around 0.1, 0.2, 0.3, etc. Figure 8.1 (left) shows the total number of instances in each such interval (black bars), as well as the number of correct instances (grey bars). Figure 8.1 (right) shows the proportions of correct instances in each interval (diamonds), with a second order polynomial trendline (dotted). The trendline fits the data nicely (R^2 = 0.999), indicating that the confidence scores actually do reflect the probability of correctness, although not with a one-to-one mapping.

The rest of this chapter will continue on the assumption that *P(correct)* can be calculated. If it cannot be directly estimated in the ASR, it may be deduced by a regression analysis on collected data. It should be noted, however, that most scores are centred on the median in these data, and are thus not contributing with much information.

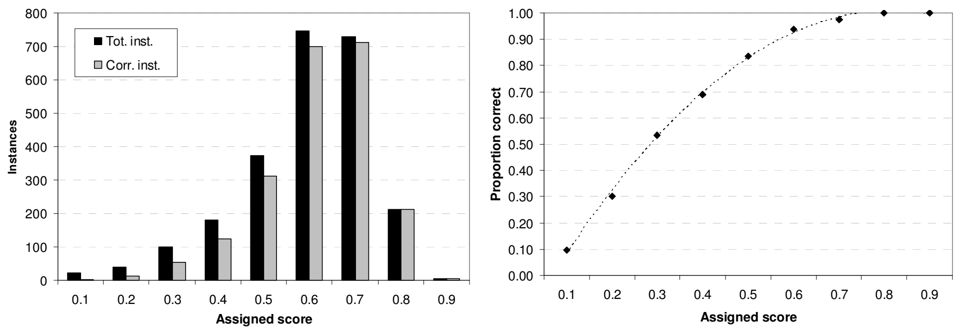


Figure 8.1: The left figure shows the total number of concepts in each confidence interval and the number of concepts that were correctly recognised. The right figure shows the proportion of correct words for each interval, with a second order polynomial trendline.

8.2.2 Cost measure

The model presented in this chapter relies on a unified measure of cost, which may be used for estimating both the task-related costs and the cost of grounding actions. The ultimate measure of cost would be the reduction of user satisfaction. However, user satisfaction is practically only obtainable on the dialogue level, and we need a much more detailed analysis. A cost measure that is relevant for both grounding actions and the task, and that is obtainable on all levels of analysis, is *efficiency*. This is reflected in Clark's principle of least effort (mentioned in 3.1.2.2): "All things being equal, agents try to minimize their effort in doing what they intend to do" (Clark, 1996). Thus, efficiency and user satisfaction should correlate to some degree, at least in a task-oriented dialogue setting as the one used in this chapter. Efficiency may be measured in different ways: by the time spent or the number of utterances, words or syllables used.

To see if these measures had an impact on user satisfaction, the users' estimation of their satisfaction after the dialogues in the collected data were correlated against all these measures of efficiency. As a measure of user satisfaction, the subject's agreement to the statement "I was satisfied with the system" on a scale ranging from 0 to 6 was used. It turned out that all measures of efficiency correlated fairly well with user satisfaction. The one that correlated best was the *total number of syllables* uttered (from both the user and the system). This non-linear regression is shown in Figure 8.2 (logarithmic regression; y = -2.19Ln(x) + 16.54; R2 = 0.622).

It should be noted that correlation with user satisfaction is problematic, partly because it is an ordinal scale. Thus, it is hard to tell whether the non-linear relationship is due to a possible non-linearity of the user satisfaction scale, or to the possibility that user satisfaction reduction decreases as the dialogues get longer. This analysis is only meant to serve as a rough indicator that efficiency is a relevant measure. The correlation is not perfect, and there are of course other factors that are important as well. However, longer dialogues often reflect that a lot of grounding actions (such as clarifications) have been needed, or that misunderstandings have

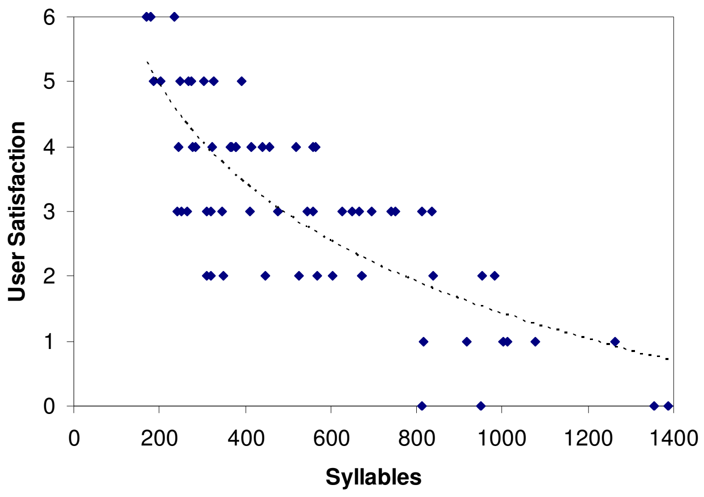


Figure 8.2: Correlation between user satisfaction and total number of syllables per dialogue.

occurred, so that the user has to start all over again. The impact of efficiency on user satisfaction in task-oriented dialogue has also been reported in other studies, such as Bouwman & Hulstijn (1998).

8.2.3 Cost functions

Using efficiency as a cost measure, we will analyse the consequences of different actions, given the correctness of the recognition hypothesis. The actions that will be considered are shown in the following alternative system responses:

(54) U: I can see a red building.
S (ACCEPT): *Ok, can you see a tree in front of you?*
S (DISPLAY): *Ok, a red building, can you see a tree in front of you?*
S (CLARIFY): *A red building?*
S (REJECT): *What did you say? [or just continue]*

We will here analyse the costs (in terms of syllables) for these different grounding actions, given the correctness of the recognition hypothesis. These costs will be based on a set of parameters that are deemed to be important for explaining the costs involved. The parameters are all average estimations over a set of dialogues.

Cost(ACCEPT, correct)

Accepting a correct concept has no cost.

Cost(ACCEPT, incorrect)

Accepting an incorrect concept will lead to a misunderstanding. This error will in many cases somehow slow down the dialogue. Either the user and system have to repair the error, or they might have to start the task all over again. The number of extra syllables the misunderstanding adds to the dialogue will be referred to as *SylMis*.

Cost(REJECT, correct)

If the system rejects a correct concept, the system and user must spend syllables on retrieving a new concept of the same value, either by the system requesting the user to repeat, or by continuing the dialogue and retrieving another concept. The number of syllables it takes to receive new information of the same value as the rejected concept will be referred to as *SylRec.*

Cost(REJECT, incorrect)

Rejecting an incorrect concept has no cost.

Cost(DISPLAY, correct)

Displaying a correct hypothesis will slow down the dialogue by the number of syllables spent on the display utterance and the possible reaction from the user. This will be referred to as *SylDispCor*. Since a concept that is displayed is treated as correct unless the user initiates a repair, it does not matter if the user confirms the display or ignores it.

Cost(DISPLAY, incorrect)

Displaying an incorrect hypothesis will also slow down the dialogue by the number of syllables spent on the display utterance and the possible reaction from the user. This will be referred to as *SylDispInc*. However, since a concept that is displayed is treated as correct unless the user initiates a repair, the user must object to the display. Otherwise, we may say that the grounding has *failed* and a misunderstanding has been introduced (which will prolong the dialogue by *SylMis* number of syllables, as described above). The probability that the user does not correct the system and the grounding fails will be referred to as *P(Fail|Disp,Inc)*. Thus, the expected cost for displaying an incorrect hypothesis is:
SylDispInc + P(Fail|Disp,Inc) x SylMis

Cost(CLARIFY, correct)

Clarifying a correct hypothesis will slow down the dialogue by the number of syllables spent on the clarification request and the possible reaction from the user. This will be referred to as *SylClarCor*. A concept that is clarified is not treated as correct unless the user confirms it. Thus, the clarification of a correct hypothesis will *fail* if the user does not confirm it. The probability

that this happens will be referred to as *P(Fail|Clar,Cor)*. If this happens, the concept is lost and the system and user must spend syllables on retrieving a new concept of the same value (*SylRec*). Thus, the expected cost for clarifying a correct hypothesis is:
SylClarCor + P(Fail|Clar,Cor) x SylRec

Cost(CLARIFY, incorrect)

Clarifying an incorrect hypothesis will slow down the dialogue by the number of syllables spent on the clarification request and the possible reaction from the user. This will be referred to as *SylClarInc*. Since a concept that is clarified is not treated as correct unless the user confirms it, it does not matter if the user disconfirms or ignores the clarification request.

The analysis given above is schematised in Figure 8.3. Together with equation (53), this analysis may then be used to derive cost functions for the different actions, which are shown in Table 8.1.

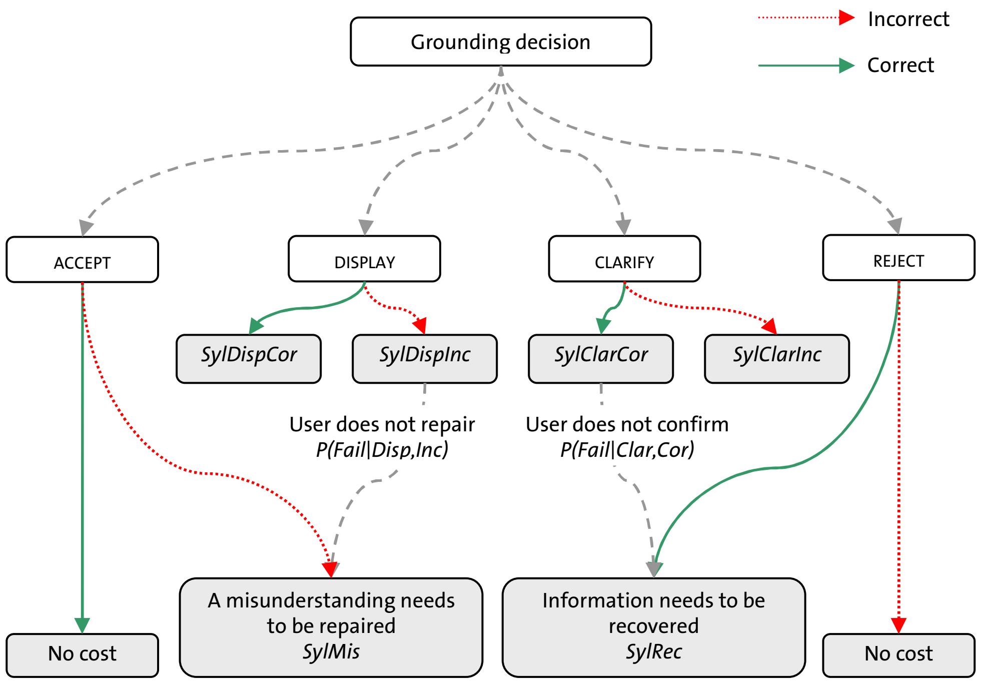


Figure 8.3: Costs involved in taking different grounding actions.

Table 8.1: Cost functions for different grounding actions.

Action	Expected cost
ACCEPT	*P(incorrect)* x *SylMis*
DISPLAY	*P(correct)* x *SylDispCor* + *P(incorrect)* x *(SylDispInc* + *P(Fail|Disp,Inc)* x *SylMis)*
CLARIFY	*P(correct)* x *(SylClarCor* + *P(Fail|Clar,Cor)* x *SylRec)* + *P(incorrect)* x *SylClarInc*
REJECT	*P(correct)* x *SylRec*

8.3 Application to the Higgins navigation domain

The cost functions derived above should be applicable to many dialogue systems, regardless of domain. However, the estimation of the parameters *SylRec* and *SylMis* is highly dependent on the domain. To show how these parameters may be estimated from data, we will make a task analysis specific for the HIGGINS navigation domain used in the previous chapters. In this domain, it is possible to distinguish three different sub-tasks which have different costs associated with them: *positioning the user*, *establishing the goal*, and *guiding the user*. We will here analyse the first two of these to show how different the task-related costs may be.

8.3.1 Positioning the user

We will start with the positioning task, when the user describes her position, as in the following example:

(55) U: I can see a red building.
S: *Red?*

8.3.1.1 SylRec

The parameter *SylRec* describes the number of syllables it will take to get the same amount of information after a concept has been rejected. This parameter is highly context dependent – it depends on how much information the hypothesised concept provides (its *information gain*), compared to the average concept. This proportion will be referred to as *ConValueH*. The system and the user spent on average 15.0 syllables per important concept[10] accepted by the system. We will refer to this as *SylCon*. Based on these two parameters, *SylRec* can be calculated as follows:

(56) $SylRec = SylCon \times ConValueH$

[10] By important concept, we mean concepts that contribute in the current task. In this example, RED is important, but not BUILDING, since there are buildings everywhere.

How can *ConValueH* be estimated for an individual concept in the positioning phase? The purpose of the positioning phase is to cut down the number of possible user locations. Thus, the value of a concept can be described as the proportion of the set of possible user locations that are cut down after accepting it, compared to the average concept. The proportion of possible locations that are reduced on average after a single concept is accepted can be estimated from data (on average 0.34, which we will refer to as *CutDownA*). The dialogue system can then use the domain database to calculate the proportion of possible locations that would be cut down if the hypothesised concept would be accepted (*CutDownH*). By accepting *ConValueH* number of average concepts, each leaving a proportion of 1 - *CutDownA* possible locations, a proportion of 1 - *CutDownH* locations should be left. This is expressed in the following formula:

(57) $$(1 - CutDownA)^{ConValueH} = (1 - CutDownH)$$

For example, if half of all possible positions are cut down on average for each concept (*CutDownA* = 0.5), and the hypothesised concept reduces ¾ of the possible positions (*CutDownH* = 0.75), it will take two average concepts to achieve the same effect (*ConValueH* = 2): $(1 - 0.5)^2 = (1 - 0.75)$.

By combining equations (56) and (57), *SylRec* can be calculated with the following formula:

(58) $$SylRec = SylCon \times \frac{\log(1 - CutDownH)}{\log(1 - CutDownA)}$$

8.3.1.2 SylMis

We will now turn to the parameter *SylMis,* which describes the number of extra syllables a misunderstanding adds to the dialogue. The risk of accepting an incorrect concept during the positioning phase is that the set of possible user positions may be erroneously constrained. If this happens, the positioning often has to start all over again. Thus, *SylMis* should reflect the number of syllables a complete positioning takes (on average 97.0, which we will refer to as *SylPos*). However, the set of possible user locations does not *need* to be erroneously constrained when accepting an incorrect concept – the user may actually see a red building, even if this was not what she said. The probability that the correct position actually is lost can be described by the parameter *CutDownH* defined above, which describes the proportion of possible locations that is reduced if the hypothesised concept is accepted. Thus, *SylMis* can be calculated as follows:

(59) $$SylMis = SylPos \times CutDownH$$

8.3.1.3 Grounding parameters

The rest of the parameters can be calculated from the data by counting the number of syllables spent on the grounding subdialogues and the number of times they failed. These parameters are shown in Table 8.2. *SylGA* is the number of syllables involved in the grounding act (in the case of DISPLAY or CLARIFY).

Table 8.2: Initial estimation of parameters for example (6).

Parameter	Value
SylClarCor	*SylGA* + 1.4
SylClarInc	*SylGA* + 2.1
SylDispCor	*SylGA* + 0.1
SylDispInc	*SylGA* + 1.2
P(Fail|Clar,Cor)	0.33
P(Fail|Disp,Inc)	0.82

As discussed in the previous chapter, the high value of *P(Fail|Clar,Cor)*, and especially *P(Fail|Disp,Inc)*, might be explained by the fact that the system did not use an elaborate prosodic model for the realisation of fragmentary DISPLAY and CLARIFY acts, that they were sometimes used in inadequate situations, and that the use of such fragments is still very uncommon in dialogue systems.

8.3.1.4 Examples

We will now consider two examples where the concept information gain differs a lot (the concepts under question are underlined):

(60) I can see a <u>mailbox</u>. (*CutDownH* = 0.782; *SylGA* = 2)
(61) I can see a <u>two</u> storey building. (*CutDownH* = 0.118; *SylGA* = 1)

Figure 8.4 shows the difference in number of possible user positions after accepting these two different utterances. Using these parameters, the cost function for the different grounding actions, depending on *P(correct)*, can be calculated to find out which action has the lowest cost for each value of *P(correct)* and thus derive confidence thresholds, as shown in Figure 8.5 and Figure 8.6. In these figures, the costs for the different actions are plotted as functions of *P(correct)*. For each value of *P(correct)*, the action with the lowest cost can be determined. The thresholds at which the optimal action shifts are marked with vertical lines. As the figures show, example (60) has a much higher information gain and thus a wide confidence interval where a clarification request is optimal, whereas example (61) has less information gain and is optimally either accepted or rejected, but never clarified.

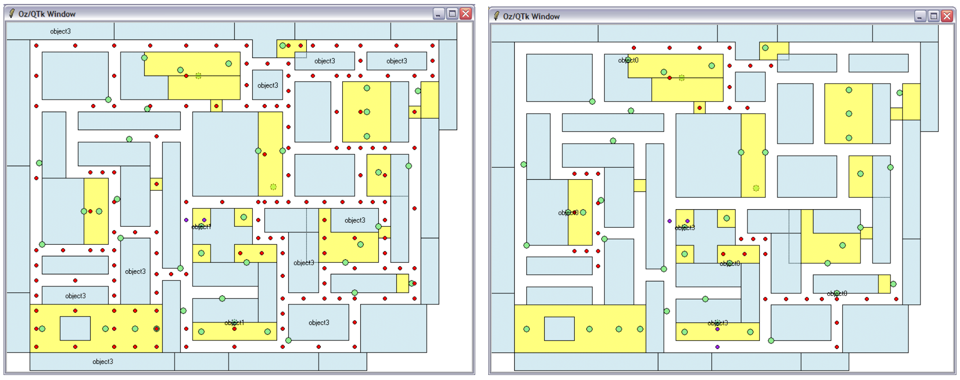


Figure 8.4: Possible user positions (red dots) in the virtual city after accepting the utterance “I can see a two storey building” (left) versus “I can see a mailbox” (right).

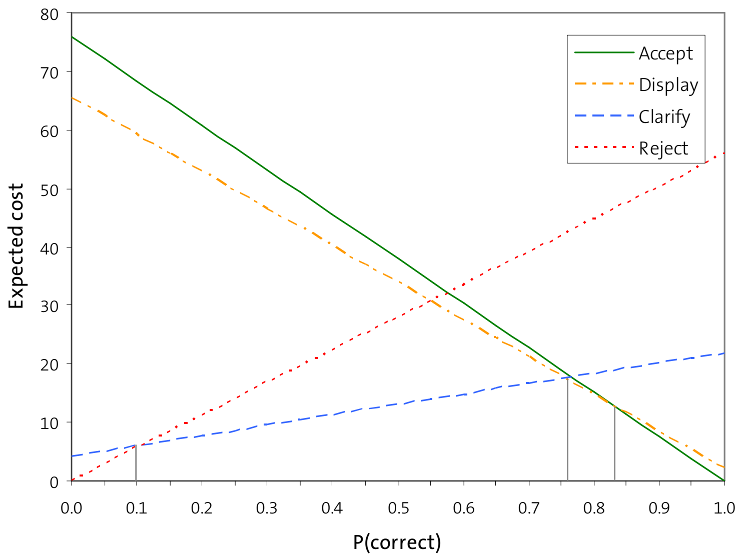


Figure 8.5: Cost functions and confidence thresholds for grounding the concept MAILBOX after “I can see a mailbox”.

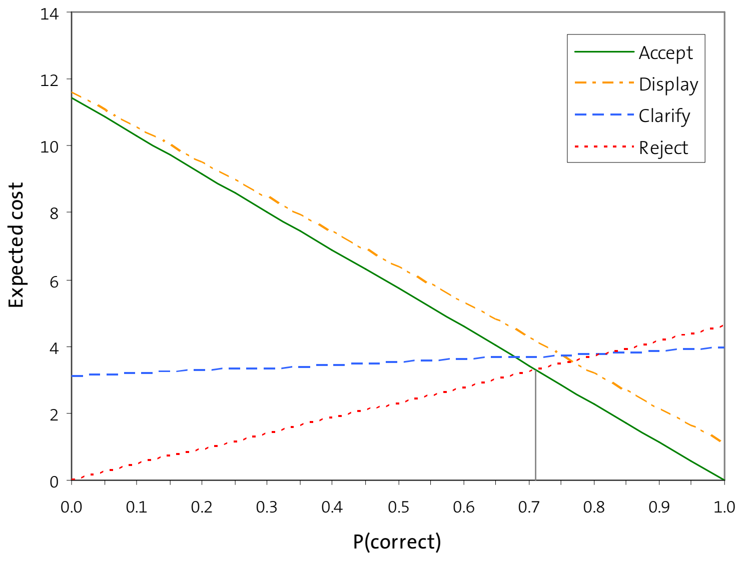


Figure 8.6: Cost functions and confidence thresholds for grounding the concept TWO after "I can see a two storey building".

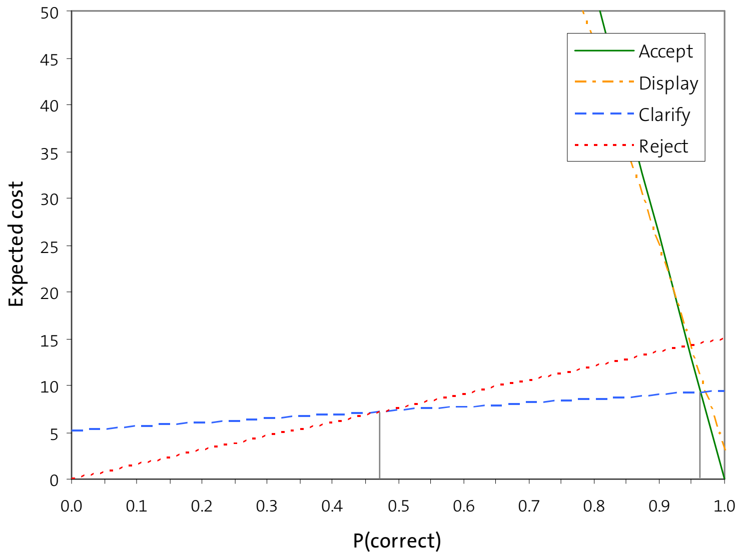


Figure 8.7: Cost functions and confidence thresholds for grounding the concept ATM after "I want to go to an ATM".

The graphs presented above, and the calculation of thresholds, are of course only useful for illustrative purposes. A dialogue system would just calculate the most optimal action, given the value of *P(correct)*. It should be noted that these estimations are based on the data collected with hand-crafted confidence thresholds. If the derived model would be applied to the system, the parameters values would change, thus affecting the parameters in the model. This means that the presented model should be derived iteratively, using bootstrapping, and the parameter values presented here are just the first step in such an iteration. To estimate the parameters, transcription of the dialogues and some annotation is needed. However, given that the logging is adapted for this, we believe that this can be done rather efficiently.

8.3.2 Establishing the goal

In the previous examples, we considered the positioning of the user. However, there is another important task, the establishing of the goal:

(62) U: I want to go to an <u>ATM</u>. (*SylGA*=3)

If this hypothesis would constitute a misunderstanding, it would lead to much higher costs than a misunderstood positioning statement. The misunderstanding might not be identified until the system has actually navigated the user to an ATM, which may take some time. Thus, we can define *SylMis* as the number of syllables it takes on average until the user has reached the (incorrect) goal or restated the goal, which can be estimated to 261.6 from the data. We will assume that *SylRec* is equal to *SylCon* (15.0), and that the other parameters are the same as in the positioning phase. The cost functions and thresholds for grounding "ATM" in the example above are shown in Figure 8.7. Due to the high cost of misunderstandings, a simple accept requires a very high confidence, and goal assertions will therefore most often be clarified.

8.4 Possible extensions

The model presented above may be extended to incorporate other aspects and address some of the simplifying assumptions behind it. We will here briefly discuss such extensions.

8.4.1 Substitutions

In the proposed model, there was no cost associated with rejecting an incorrect concept. This may seem wrong, since the incorrect concept may often be a substitution for a correct concept which is lost and must be recovered. But, if we would add this cost, it should also be added when accepting an incorrect concept, in which case the incorrect concept must be repaired *and* the lost concept must be recovered. If we add the same cost to all actions, the model will not be affected.

However, in the case of DISPLAY or CLARIFY, a substituted concept may be more efficiently recovered, as in the following example:

(63) U: I can see a green building? [RED]
S: *Red?*
U: No, green [GREEN]

Thus, the cost of these grounding actions has been somewhat overestimated. To compensate for this, we must first calculate the probability that an incorrect concept is a substitution for a similar concept, *P(Subst),* and multiply this with *SylRec*. This cost should be added to ACCEPT and REJECT, in the case of an incorrect hypothesis. This cost should also be added to CLARIFY and DISPLAY, if they do not succeed in recovering the substituted concept. The probabilities of this can be described as *P(Fail|Clar,Subst)* and *P(Fail|Disp,Subst),* respectively. Table 8.3 shows the updated costs in case of incorrect hypotheses.

Table 8.3: Costs in case of an incorrect hypothesis that incorporates the possibility of a substitution.

Action	**Cost(a, incorrect)**
ACCEPT	*SylRep + P(Subst) * SylRec*
REJECT	*P(Subst) * SylRec*
DISPLAY	*SylDispU + SylDispInc + P(Fail|Disp,Inc) * SylRep + P(Fail|Disp,Subst) * P(Subst) * SylRec*
CLARIFY	*SylClarU + SylClarInc + P(Fail|Clar,Subst) * P(Subst) * SylRec*

8.4.2 Concept-level grounding

In the proposed model, only one concept in the hypothesis was considered. The model also accounts for utterance-level grounding, where the whole utterance is considered as being correct or incorrect. However, the model could also be extended to cope with several concepts in an utterance, of which some may be correct and some not, as in the following example (with confidence scores in parenthesis):

(64) U: I can see a red building to the left. [RED (0.7) LEFT (0.2)]

In this case, we should consider 4 possible states instead of 2, as shown in Table 8.4. The combination of grounding actions for each concept may lead to different realisations of grounding moves. Some examples of such actions are shown in Table 8.5.

Table 8.4: Example states when two concepts are considered.

RED	LEFT	P
CORRECT	CORECT	0.14
CORRECT	INCORRECT	0.56
INCORRECT	CORECT	0.06
INCORRECT	INCORRECT	0.24

Table 8.5: Example actions when two concepts are considered.

RED	LEFT	Utterance
CLARIFY	ACCEPT	*Red?*
DISPLAY	CLARIFY	*Do you have the red building on your left?*
CLARIFY	CLARIFY	*A red building on your left?*

8.4.3 Temporal modelling

Another assumption behind the proposed model was that the costs and probabilities were not dependent on the dialogue history, in other words, there is no temporal aspect. However, as Paek & Horvitz (2003) points out, for example repeated requests for repetitions or clarification requests may decrease the utility of such actions.

It would of course be possible to increase the number of parameters and introduce a temporal aspect. However, in the data collected here, there are very few instances of, for example, repeated clarification requests.

Another temporal aspect is that not only the utility, but also the probability of a certain hypothesis should be affected by a history of repeated clarifications. This should ideally be considered in the early error detection, where both the ASR confidence score and the dialogue history could be combined into a more elaborate model of *P(correct)*.

8.4.4 Utterance generation

The model presented in this chapter encourages efficient system utterances by parameters such as *SylClarCor*. However, the model also accounts for the users' recognition of them by parameters such as *P(Fail|Clar,Cor)*. For example, in the data studied here, the system used efficient elliptical clarification requests and display utterances, but this had the negative consequence that they often failed.

The proposed model should therefore be usable for testing the benefits of different utterance realisations. For example, the fragmentary clarification requests used here could be compared with sentential clarification requests (such as "did you say red?"). *SylClarCor* and *SylClarInc* would be higher, but *P(Fail|Clar,Cor)* would perhaps be lower.

8.4.5 User adaptation

Many of the parameters vary considerably between different users; see for example parameter *SylPos* in Figure 8.8. Thus, if the parameters could be tuned for the specific user, the system could adapt its behaviour accordingly. For example, some users may not respond well to clarification requests. This would be reflected in some of the parameters, and the system could avoid making clarifications.

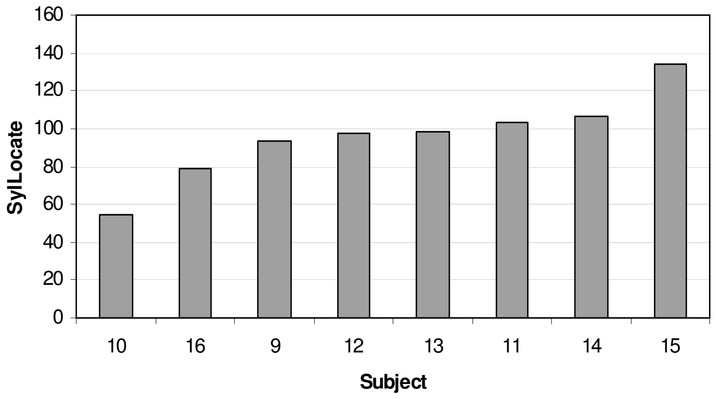


Figure 8.8: How parameter *SylPos* varies between different subjects.

8.5 Discussion

There are still several aspects that are not considered in the proposed model. For example, it is not possible to choose actions that maximise a long-term cumulative sum of rewards (i.e., perform planning). Another limitation is that it only considers one hypothesis from the ASR and cannot hold parallel hypotheses. As discussed in 8.1.2, a more complex model that may account for these factors is Partially Observable Markov Decision Processes (POMDP). The model proposed in this chapter is much simpler and more knowledge-driven (since it is based on task analysis). Thus, it is based on more assumptions and includes more bias, but at the same time it requires less resources and should be easier to apply. As Williams & Young (2007) point out, it is also computationally challenging to scale POMDP models to more complex applications.

Efficiency does not cover all costs involved in dialogue, even in a task-oriented domain such as navigation. For example, the results presented in Chapter 4 indicated that the frustration that the signalling of understanding gives rise to may decrease user satisfaction *per se*, that is, not just by the number of syllables added to the dialogue. It would be interesting to use a more elaborate cost model, for example by applying regression analysis of user satisfaction, as in the PARADISE evaluation framework (Walker et al., 2000a).

The proposed model cannot be directly implemented in the HIGGINS architecture as described in Chapter 6. In this architecture, the grounding action manager (GAM), which only considers the discourse history, is separated from the navigation action manager (NAM), which also looks into the domain database. This distinction was made in order for the grounding actions to be realised as quickly as possible, while the NAM made more complex decisions. However, the model proposed in this chapter relies on the possibility of looking into the domain database for making grounding decisions as well. This shows that it might be unfeasible to maintain the separation of action managers if the system is to take more complex grounding decisions.

As stated above, the cost functions presented in Table 8.1 should be applicable to other domains as well. The two parameters that are task-dependent are *SylRec* and *SylMis*. In this chapter, it was shown how these may be estimated for the HIGGINS navigation domain. For a much simpler domain, such as a standard slot-filling travel booking domain, these parameters could possibly be estimated in a more straightforward manner. In the navigation domain, there is not a fixed set of slots that are to be filled. Thus, each concept may contribute with a different amount of information (the concept information gain). In a domain where a fixed set of slots needs to be filled, this notion is not relevant. Instead, *SylRec* could possibly be mapped directly to *SylCon* (the number of syllables it takes on average to receive a new concept). If there is a final confirmation dialogue at the end of the slot-filling, *SylMis* could possibly be estimated as the number of syllables it takes on average to reach the final confirmation and make the repair.

The presented model also remains to be evaluated, for example by comparing the performance of a system using this model with a system based on handcrafted thresholds, or a more complex model, such as POMDP.

8.6 Summary

This chapter has presented a data-driven decision-theoretic approach to making grounding decisions in spoken dialogue systems, that is, to decide which recognition hypotheses to consider as correct and whether to make a clarification request or display understanding. This model accounts for the uncertainty of the speech recognition hypothesis, as well as the costs involved in taking grounding actions and the task-related costs that a misunderstanding or a rejection would have. Based on a task analysis of the HIGGINS navigation domain, cost functions were derived. It was argued that efficiency– the number of syllables uttered by the user and system – was useful as a cost measure for the navigation domain. Dialogue data was then used to estimate parameters for these cost functions, so that the grounding decision may be based on both confidence and dialogue context. For example, it was shown how concepts with high information gain should more often be clarified than concepts with low information gain, which are either simply rejected or accepted. To silently accept a concept which is associated with a very high cost of misunderstanding, a very high confidence in this concept is required.

CHAPTER 9

Prosody in fragmentary grounding

The evaluation of the HIGGINS system presented in Chapter 7 showed that fragmentary grounding utterances often failed, in the sense that the users often did not seem to understand them and act as expected. As already noted, this may be explained partly by the fact that users do not expect such human-like behaviour from dialogue systems, partly because they were used in contexts where a human would not have used them, and partly because the prosodic model was very simplistic and not tested.

In this chapter, the effects of prosodic features on the interpretation of synthesised fragmentary grounding utterances in Swedish dialogue are studied. First, the users' interpretation of such utterances, depending on their prosodic realisation, will be explored in a perception experiment. In a second experiment, we will test the hypothesis that users of spoken dialogue systems not only perceive the differences in prosody of synthesized fragmentary grounding utterances, and their associated pragmatic meaning, but that they also change their behaviour accordingly in a human-computer dialogue setting.

The following scenario, taken from the pedestrian navigation domain used in previous chapters, was used in the first experiment presented in this chapter:

(65) U.1: Further ahead on the right I see a red building.
S.2: *Red (?)*

As discussed in 3.1.4, the evidence of understanding that the system provides in S.2 in this example may have different readings, depending whether we interpret it is as positive or negative evidence, and depending on what level of action the evidence concerns. Three possible readings of S.2 are shown in Table 9.1.

Table 9.1: Different readings of the fragmentary grounding utterance S.2 in example (65).

Reading	Paraphrase	Evidence of understanding
ACCEPT	*Ok, red*	Display of understanding. Positive on all levels.
CLARIFYUND	*Do you really mean red?*	Clarification request. Positive perception, negative/uncertain understanding.
CLARIFYPERC	*Did you say red?*	Clarification request. Positive contact, uncertain perception.

The reading "positive understanding, negative acceptance" (as discussed in 3.1.4) has not been included here. The reason for this is that it is hard to find examples which may be applied to spoken dialogue systems (at least in the studied domain) where reprise fragments may have such a reading.

9.1 Prosody in grounding and requests

Considerable research has been devoted to the study of question intonation in human-human dialogue. However, there has not been much study on the use of different types of interrogative intonation patterns in spoken dialogue systems. Not only does question intonation vary in different languages, but also different types of questions (e.g., wh and yes/no) can result in different intonation patterns (Ladd, 1996).

In very general terms, the most commonly described tonal characteristic for questions is high final pitch and overall higher pitch (Hirst & Cristo, 1998). In many languages, yes/no questions are reported to have a final rise, while wh-questions typically are associated with a final low. In Dutch, for example, van Heuven et al. (1999) have documented a relationship between incidence of final rise and question type, in which wh-questions, yes/no questions and declarative questions obtain an increasing number of final rises, in that order. Wh-questions can, moreover, often be associated with a large number of various contours. Bolinger (1989), for example, presents various contours and combinations of contours which he relates to different meanings in wh-questions in English. One of the meanings most relevant to the present study is what he terms the "reclamatory" question. This is often a wh-question in which the listener has not quite understood the utterance and asks for a repetition or an elaboration. This corresponds to the paraphrase, "What did you mean by red?"

In Swedish, interrogative mode is most often signalled by word order with the finite verb preceding the subject (yes/no questions) or by lexical means (e.g., wh-questions). Question intonation can also be used to convey interrogative mode when the question has declarative word order. This type of echo question is relatively common in Swedish especially in casual questions (Gårding, 1998). Question intonation of this type has been studied in scripted elicited questions and has been primarily described as marked by a raised topline and a widened F_0 range on the focal accent (Gårding, 1998).

In recent perception studies, however, House (2003) demonstrated that a raised fundamental frequency (F_0) combined with a rightwards focal peak displacement is an effective means of signalling question intonation in Swedish echo questions (declarative word order) when the focal accent is in final position. Furthermore, there was a trading relationship between peak height and peak displacement so that a raised F_0 had the same perceptual effect as a peak delay of 50 to 75 ms.

In a study of a corpus of German task-oriented human-human dialogue, Rodriguez & Schlangen (2004) found that the use of intonation seemed to disambiguate clarification types with rising boundary tones used more often to clarify acoustic problems than to clarify reference resolution.

9.2 Experiment I: Interpretations

In Experiment I, subjects were asked to listen to short dialogue fragments in Swedish, similar to example (65) above, where the computer is saying a fragmentary grounding utterance after a user turn, and to judge what was actually intended by the computer, based on prosodic features of the utterance.

9.2.1 Method

9.2.1.1 Stimuli

Three test words comprising the three colours: blue, red and yellow (*blå, röd, gul*) were synthesized using an experimental version of LUKAS (Filipsson & Bruce, 1997) diphone Swedish male MBROLA voice (Dutoit et al., 1996) implemented as a plug-in to the WaveSurfer speech tool (Sjölander & Beskow, 2000).

For each of the three test words, the intonational contour (i.e., the F_0 curve) was manipulated by changing the following parameters: 1) F_0 peak POSITION, 2) F_0 peak HEIGHT, and 3) Vowel DURATION. Three peak positions were obtained by time-shifting the focal accent peaks in intervals of 100 ms comprising *early*, *mid* and *late* peaks. A *low* peak and a *high* peak set of stimuli were obtained by setting the accent peak at 130 Hz and 160 Hz respectively. Two sets of stimuli durations (*normal* and *long*) were obtained by lengthening the default vowel length by 100 ms. All combinations of three test words and the three parameters gave a total of 36 different stimuli. Six additional stimuli, making a total of 42, were created by using both the early and late peaks in the long duration stimuli which created a double peaked stimulus. A possible late-mid peak was not used in the long duration set since a late rise and fall in the vowel did not sound natural. The stimuli are presented schematically for the word "yellow" in Figure 9.1.

The first turn of the dialogue fragment in example (65) above was recorded for each colour word and concatenated with the synthesized test words, resulting in 42 different dialogue fragments similar to example (65).

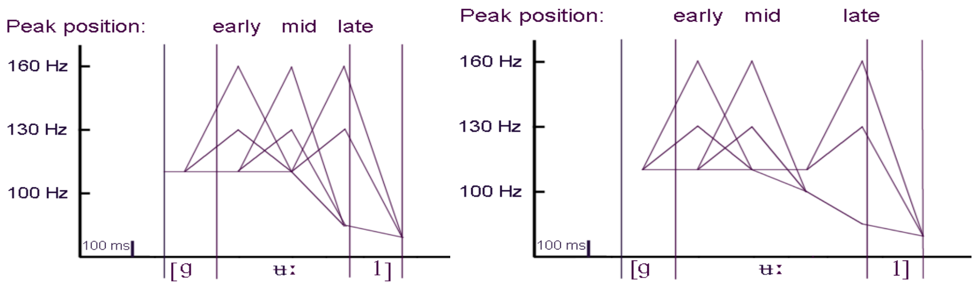


Figure 9.1: Stylized representations of the stimuli "gul" ("yellow"), showing the F_0 peak position. The left panel shows normal duration, the right lengthened duration.

9.2.1.2 Experimental design and procedure

The subjects were 8 Swedish speakers in their 20s and 30s (2 women and 6 men, 2 second language speakers and 6 native speakers). All of the subjects had some knowledge of speech technology, although none of them worked with the issues addressed in the experiment.

The subjects were placed in front of a computer monitor in a quiet room. In order to give a sense of the kind of domain envisaged in the experiment, the subjects were shown a video demonstrating a typical dialogue between the HIGGINS spoken dialogue system and a user. The subjects were told that they would listen to 42 similar dialogue fragments containing a user utterance and a system utterance each, and that their task was to judge the meaning of the system utterance by choosing one of three alternatives and to rate their own confidence in that choice. They were also informed that they could only listen to each dialogue fragment once. After the instructions, the test was started and the subjects were left alone for the duration of the experiment.

During the experiment, the subjects were played each of the 42 stimuli once, in random order, on a loudspeaker. After each stimulus, they used the GUI shown in Figure 9.2 to pick a paraphrase for the system utterance and to judge their own confidence in that choice. The different paraphrases corresponded to the ones shown in Table 9.1 above. The subjects could not listen to the stimulus more than once, nor could they skip any stimuli. The total test time was around five to ten minutes per subject.

9.2.2 Results

There were no significant differences in the distribution of votes between the different colours ("red", "blue", and "yellow") (χ^2=3.65, dF=4, p>0.05). There were not any significant differences for any of the eight subjects (χ^2=19.00, dF=14, p>0.05), nor had the DURATION parameter any significant effect on the distribution of votes (χ^2=5.72, dF=2, p>0.05).

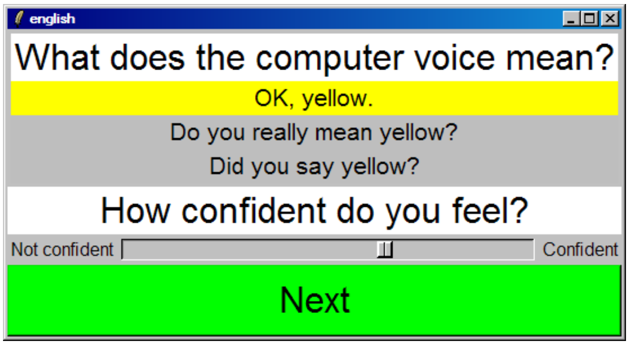

english
What does the computer voice mean?
OK, yellow.
Do you really mean yellow?
Did you say yellow?
How confident do you feel?
Not confident
Confident
Next

Figure 9.2: The test GUI (translated from Swedish).

Both POSITION and HEIGHT had significant effects on the distribution of votes, which is shown in Table 9.2 (χ^2=70.22, dF=4, p<0.001 resp. χ^2=59.40, dF=2, p<0.001). The interaction of the parameters POSITION and HEIGHT also gave rise to significant effects (χ^2=121.12, dF=10, p<0.001), as shown in the bottom of Table 9.2. Figure 9.3 shows the distribution of votes for the three interpretations as a function of position for both high and low HEIGHT.

Table 9.2: Interpretations that were significantly overrepresented, given the values of the parameters POSITION and HEIGHT, and their interactions. The standardized residuals from the χ^2-test are also shown.

POSITION	**Interpretation**	**Std. resid.**
early	ACCEPT	3.1
mid	CLARIFYUND	4.6
late	CLARIFYPERC	3.6

HEIGHT	**Interpretation**	**Std. resid.**
high	CLARIFYUND	3.2
low	ACCEPT	4.0

POSITION* HEIGHT	**Interpretation**	**Std. resid.**
early*low	ACCEPT	3.4
mid*low	ACCEPT	3.4
mid*high	CLARIFYUND	5.6
late*high	CLARIFYPERC	4.4

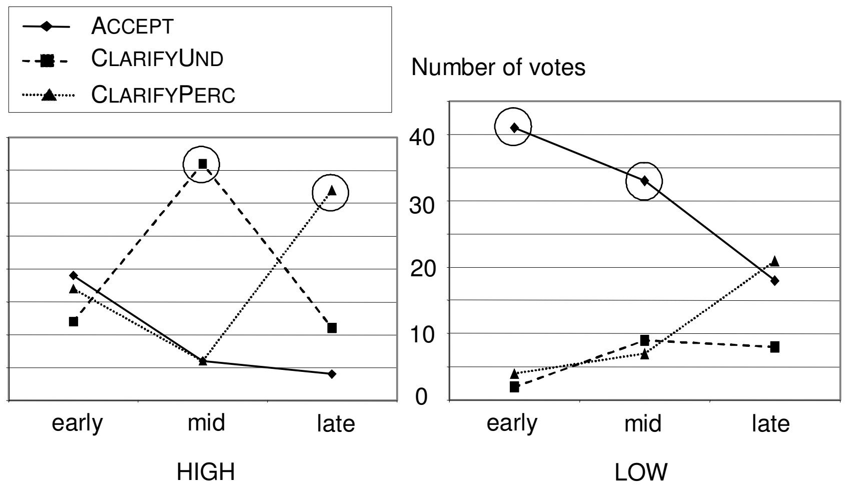


Figure 9.3: The distribution of votes for the three interpretations as a function of position: where HEIGHT is high on the left, and low on the right. The circles mark distributions that are significantly overrepresented.

Weighting the votes with the subjects' own confidence scores only seemed to strengthen the results, so they were not used for further analysis. Results from the double-peak stimuli were generally more complex and are not presented here.

In summary, this first experiment shows that three prototypical intonation patterns can be distinguished, corresponding to the different readings of the fragmentary grounding utterance: an early low F_0 peak corresponds to ACCEPT ("ok, red"), a mid high F_0 peak corresponds to CLARIFYUND ("do you really mean red?"), and a late high F_0 peak corresponds to CLARIFYPERC ("did you say red?").

9.3 Experiment II: User responses

In Experiment II, we wanted to test the hypothesis that users of spoken dialogue systems not only perceive the differences in prosody of synthesized fragmentary grounding utterances, and their associated pragmatic meaning, but that they also change their behaviour accordingly in a human-computer dialogue setting.

9.3.1 Method

To test our hypothesis, an experiment was designed in which subjects were given the task of classifying colours in a dialogue with a computer. They were told that the computer needed the subject's assistance to build a coherent model of the subject's perception of colours, and that this was done by having the subject choose among pairs of the colours green, red, blue and yellow when shown various nuances of colours in-between (e.g., purple, turquoise, orange and chartreuse). An example classification task is shown in Figure 9.4.

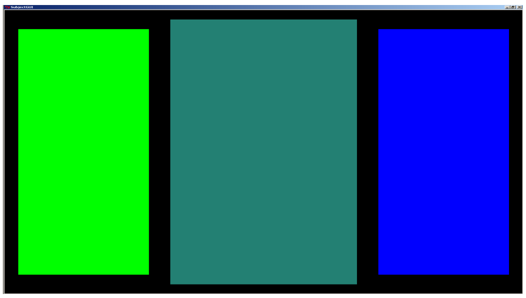


Figure 9.4: An example colour classification task. The computer asked the subject which of the two colours on the flanks was most similar to the one in the middle.

The subjects were also told that the computer may sometimes be confused by the chosen colour or disagree. The test configuration consisted of a computer monitor, loudspeakers, and an open microphone in a quiet room. An extra close-talking microphone was fitted to the subject's collar. An experiment conductor sat behind the subjects during the experiment, facing a different direction. The total test time was around ten minutes per subject.

The experiment used a Wizard-of-Oz set-up: a person sitting in another room – the Wizard – listened to the audio from the close talking microphone (a radio microphone). The Wizard fed the system the correct colours spoken by the subjects, as well as giving a go-ahead signal to the system whenever a system response was appropriate. The subjects were informed about the Wizard setup immediately after the experiment, but not before. Here is an example of a typical dialogue fragment (translated from Swedish):

(66)
S.1: *[presents turquoise flanked by green and blue]*
which colour is closest to the one in the middle?
U.2: green
S.3: *green*
U.4: mm
S.5: *okay*
[presents orange flanked by red and yellow]
and this?
U.6: yellow perhaps

The Wizard had no control over what utterance the system would present next. Instead, this was chosen by the system depending on the context, just as it would be in a system without a Wizard. The grounding fragments (such as S.3 above) came in four flavours: a repetition of the colour with one of the three prototype intonations found in Experiment I (ACCEPT, CLARIFYUND or CLARIFYPERC) or a simple acknowledgement consisting of a synthesized /m/

or /a/ (ACKNOWLEDGE) (Wallers et al., 2006). The system picked these at random so that for every eight colours, each grounding fragment appeared twice.

All system utterances were synthesized using the same voice as the experiment stimuli used in Experiment I. The prosody of each utterance was hand-tuned before synthesis in order to raise the subjects' expectations of the computer's conversational capabilities as much as possible. As seen in the dialogue example above, the computer made heavy use of conversational phenomena such as backchannels and ellipses. There was also a rather high degree of variability in the exact rendition of the system responses. Each of the non-stimuli responses was available in a number of varieties, and the system picked from these at random. Due to the simplicity of the task and the Wizard-of-Oz setup, the system was very responsive, with virtually no delays caused by processing.

The subjects were 10 Swedish speakers between 20 and 65 years old (7 women and 3 men, 1 second language speaker and 9 native speakers). One of the subjects had some knowledge of speech technology, although he did not work with the issues addressed in the experiment.

9.3.2 Results

The recorded conversations were automatically segmented into utterances based on the logged timings of the system utterances. User utterances were then defined as the recorded audio segments in-between these. Out of ten subjects, two did not respond at all to any of the grounding utterances (i.e., didn't say anything similar to U.4 in the example above). For the other eight, responses were given in 243 out of 294 possible places. Since the object of our analysis was the subjects' responses, two subjects in their entirety and 51 silent responses distributed over the remaining eight subjects were automatically excluded from analysis.

In almost all cases, subjects simply acknowledged the system's grounding utterance with a brief "yes" or "mm" as the utterance U.4 in the example above. However, when listening to the dialogues, we got the impression that the response time differed. For example, the response time after a grounding fragment with the meaning "do you really mean red?" seemed to be longer than after a fragment meaning "did you say red?".

To test whether the response times were in fact affected by the type of preceding fragment, the time between the end of each system grounding fragment and the user response (in the cases there was a user response) was automatically determined using /nailon/, a software package for extraction of prosodic and other features from speech (Edlund & Heldner, 2006). Silence/speech detection in /nailon/ is based on a fairly simplistic threshold algorithm, and for our purposes, a preset threshold based on the average background noise in the room where the experiment took place was deemed sufficient. The results are shown in Table 9.3. The table shows that, just in line with our intuitions, ACCEPT fragments are followed by the shortest response times, CLARIFYUND the longest, and CLARIFYPERC between these. The differences are statistically significant (one-way within-subjects ANOVA; F=7.558; dF=2; $p<0.05$).

These response time differences are consistent with a cognitive load perspective that could be applied to the fragment meanings ACCEPT, CLARIFYPERC and CLARIFYUND. To simply acknowledge an acceptance should be the easiest, and it should be nearly as easy, but not quite, for users to confirm what they have actually said. It should take more time to re-evaluate a decision and insist on the truth value of the utterance after CLARIFYUND. This relationship is nicely reflected in the data.

Table 9.3. Average of subjects' mean response times after grounding fragments.

Grounding fragment	**Response time**
ACCEPT	591 ms
CLARIFYUND	976 ms
CLARIFYPERC	634 ms

9.4 Discussion

The results of these studies can be seen in terms of a tentative model for the intonation of fragmentary grounding utterances in Swedish. A low-early peak would function as an ACCEPT statement, a mid-high F_0 peak as a CLARIFYUND question, and a late high peak as a CLARIFYPERC question. This would hold for single-syllable accent I words. Accent II words and multi-word fragments are likely to be more complex.

For these single-word grounding utterances, the general division between statement (early, low peak) and question (late, high peak) is consistent with the results obtained for Swedish echo questions (House, 2003) and for German clarification requests (Rodriguez & Schlangen, 2004). However, the further clear division between the interrogative categories CLARIFYUND and CLARIFYPERC is especially noteworthy. This division is related to the timing of the high peak. The high peak is a prerequisite for perceived interrogative intonation in this study, and when the peak is late, resulting in a final rise in the vowel, the pattern signals CLARIFYPERC. This can also be seen as a yes/no question and is consistent with the observation that yes/no questions generally more often have final rising intonation than other types of questions. The high peak in mid position is also perceived as interrogative, but in this case it is the category CLARIFYUND which dominates as is clearly seen in the left panel of Figure 9.3. This category can also been seen as a type of wh-question similar to the "reclamatory" question discussed in Bolinger (1989). For example, the question "do you really mean red?" is similar to (and may have the same effect as) "what do you mean by red?"

Another interesting result is the evidence of an interaction between the parameters peak height and peak position when the peak position is mid. Here, the high-mid peak is perceived as the CLARIFYUND question, while the low-mid peak is perceived as the ACCEPT statement. A similar type of interaction is the trading relationship between peak height and peak displacement in House (2003), where a higher earlier peak has the same perceptual status as a lower later peak.

It is somewhat surprising that the longer duration was not perceived as more interrogative, as this was expected to be interpreted as hesitation and uncertainty. The fact that the majority of the stimuli ended in a very low F_0 may have precluded this interpretation.

Although we have not quantified other prosodic differences in the users' responses in Experiment II, we also got the impression that there were subtle differences in, for example, pitch range and intensity. These differences may function as signals of certainty following CLARIFYPERC and signals of insistence or uncertainty following CLARIFYUND. More neutral, unmarked prosody seemed to follow ACCEPT.

9.4.1 Future Work

When listening to the resulting dialogs from Experiment II as a whole, the impression is that of a natural dialogue flow with appropriate timing of responses, feedback and turn-taking. To be able to create spoken dialogue systems capable of this kind of dialogue flow, we must be able to both produce and recognise fragmentary grounding utterances and their responses. Further work using more complex fragments and more work on analysing the prosody of user responses is needed.

As grounding fragments become more complex, the interaction between focus and level of action must also be understood. Consider the following examples:

(67) U: I can see a blue brick building.
S: *A **red** brick building?*
U: No, blue

(68) U: I can see a red concrete building.
S: *A red **brick** building?*
U: No, concrete

By using different prosodic realisations in these examples, the system may signal more precisely where the uncertainty is located. This should in turn affect how a potential negation by the user should be integrated in the resulting semantic structure. However, it is also possible that the uncertainty may be associated with the different levels of action dealt with in this chapter. To understand the prosodic interplay between level of action and focus is an interesting challenge.

9.5 Summary

In this chapter, two experiments have been presented. In the first experiment, subjects were given the task of listening to short dialogue fragments containing synthesised fragmentary grounding utterances, and choosing the most likely paraphrase. The prosody of these utterances was systematically varied in order to study how the prosodic realisation affects the interpretation of them; whether they signalled acceptance or were interpreted as a clarification request, and which level of action was concerned. The results show that a low early F_0 peak is interpreted as acceptance; a mid high F_0 peak is interpreted as a clarification of understanding; and a late high F_0 peak is interpreted as a clarification of perception.

The second experiment show that users of spoken dialogue systems not only perceive the differences in prosody of synthesized fragmentary grounding utterances, and their associated pragmatic meaning, but that they also change their behaviour accordingly in a human-computer dialogue setting. The results show that the subjects' response times differed significantly, depending on the prosodic features of the grounding fragment spoken by the system.

PART V

Conclusion

CHAPTER 10

Summary and discussion

10.1 Thesis summary

In spoken dialogue systems, uncertainty and errors are inevitable, mostly due to the error-prone speech recognition process. Even as speech recognition technology improves, users and developers of dialogue systems will likely try to make the interaction more efficient by taking risks and introducing more ambiguity and uncertainty, at least in systems targeted towards more human-like conversational dialogue. A dialogue system must therefore be aware of and react appropriately to these errors. It must have models and methods for detecting potential errors in its hypotheses of what the user is saying, for deciding what to do depending on its uncertainty of these hypotheses and the costs of different outcomes, for displaying its understanding to the user, for making clarification requests, and for detecting errors in propositions it has already accepted.

This thesis has presented experiments on these issues and suggested methods and models for handling them. We will here make a brief summary of these contributions.

10.1.1 Methods for exploring human error handling

As stated in the introduction, apparently satisfactory communication may often take place between humans without the listener arriving at a full interpretation of the words used. The question is how this seemingly smooth handling of uncertainty and miscommunication in human-human dialogue can be transferred to human-computer dialogue In Part II of this thesis, two experimental setups were presented, exploring how humans might deal with errors caused by imperfect models in the speech recognition process.

In a first experiment, pairs of subjects were given the task of guiding each other on a virtual campus by talking to each other. The person giving directions (the "operator") could not hear what the other speaker (the "user") said. Instead, the user's speech was recognised by a speech recogniser and the operator could read the results on a screen. This way, their reactions to speech recognition errors in a real dialogue setting could be studied.

In a second experiment, human judges were presented with data collected from the first experiment. Their task was to study the erroneous speech recognition results and try to determine which words were correct and which were not. By varying the amount of information given to them (such as dialogue context, ASR confidence scores, n-best lists), it was possible to study which factors the operators might have relied upon when detecting errors in the first experiment.

10.1.2 Non-understanding recovery

The first of the two experiments described above revealed that the operators did not routinely signal non-understanding, such as uttering "what did you say?", when faced with incomprehensible speech recognition results. Instead, they tried to ask task-related questions that confirmed their hypothesis about the user's position. This strategy led to fewer non-understandings of the subsequent user utterance, and thus to a faster recovery from the problem. When they did signal non-understanding, this had a negative effect on the user's experience of task success. Despite the numerous non-understandings, users reported that they were almost always understood.

This is very different from the behaviour of most current dialogue systems. When a system is faced with a non-understanding, it is often assumed that there is nothing left to do but to signal non-understanding and thereby encourage repetition. There are three possible reasons why this strategy failed more often than others in the experiment, and why they often fail in spoken dialogue systems. First, speakers tend to hyperarticulate when repeating themselves, and hyperarticulated speech is something that many speech recognisers do not have in their acoustic models. Second, non-understandings often occur because the utterance to recognise is poorly covered by the speech recognition models (it may even be out-of-vocabulary). If the models did not cover the utterance the first time, chances are that they will not do it a second time either. Third, signalling non-understanding may be perceived as frustrating by the users, who may experience the dialogue as dominated by explicit error handling. Thus, for the design of spoken dialogue systems, the results suggest that when non-understandings occur, a good domain model and robust parsing techniques should be used to, if possible, pose relevant task-related questions to the user, instead of signalling non-understanding.

10.1.3 Early error detection

In, the second experiment on human error handling, early error detection on word level was explored, in other words, the immediate detection of erroneous words in the speech recognition results. The results show that, in doing this task, humans benefit from both word confidence scores and 5-best lists delivered by the speech recogniser. Immediate dialogue context (the previous operator/system utterance) was helpful (as long as the recognitions were not too poor), but longer context had no effect.

A machine learning experiment using two different learners for the task (memory-based and transformation-based) showed that word confidence scores were useful for automatic classification, and that other factors may contribute as well. Both lexical and contextual (from the utterance and from the discourse) features further improve performance, especially for content words.

10.1.4 Concept- and word-level error handling

A common approach to the decision between accepting and rejecting a hypothesis of what the user has said is to simply use the confidence score and compare it against a threshold. The results presented above, showing that other factors may contribute, is in line with previous research done in the area. However, most previous studies on early error detection have focussed on the detection of errors on the utterance level – the task has been to decide whether a hypothesis of a complete user utterance is correct or not. Such utterance-level error handling is often feasible in command-based dialogue systems where utterances are relatively short and predictable. However, in conversational dialogue systems, utterance-level error handling is often too simplistic. Humans engaging in conversation may often focus on parts of utterances, for example by posing fragmentary clarification requests, and thereby increase the efficiency of the dialogue. In dialogue systems that are targeted towards more human-like conversation, speech recognition results and the semantic interpretations of them may often be partly correct. This calls for error handling on a "sub-utterance" level.

As part of the work for this thesis, the HIGGINS spoken dialogue system has been developed and evaluated. The initial domain for this system has been pedestrian navigation. The system has served as a test-bed for developing and evaluating techniques and models for concept-level error handling, such as robust interpretation, modelling grounding status in the discourse, displaying understanding, posing clarification requests, and late error detection.

10.1.5 Robust interpretation

The module for natural language understanding developed for HIGGINS, called PICKERING, is a robust interpreter, designed to parse results from a speech recogniser with n-gram language models in a conversational dialogue system. A context-free grammar (CFG) is used for parsing the input, but to add robustness, the interpreter applies a number of additional techniques to the standard CFG parsing algorithm. It allows unexpected words between and inside phrases, allows non-agreement in phrases, and computes concept-confidence scores.

An evaluation of PICKERING indicates that it generalises well when applied to unseen utterances, within the limited domain, produced by new speakers. As the WER of the ASR output increases, the set of robustness techniques utilised leads to graceful degradation of the interpretation results. The two techniques used to relax the CFG-constraints that were tested –

allowing non-agreement and insertions – both improved performance. Allowing an unlimited number of insertions into syntactical structures caused neither decline nor increase in accuracy.

10.1.6 Modelling grounding status

In HIGGINS, the dialogue management is divided into discourse modelling and action selection. The discourse modeller, called GALATEA, can be regarded as a final step in the interpretation processing chain, in which the dialogue context is considered. The task of GALATEA is to resolve ellipses and anaphora, but also to model the grounding status of concepts. This grounding status contains information about who has mentioned a given concept, when it has been mentioned, the surface realisation of it, and the system's confidence in it. As the same concept is mentioned several times, the grounding information gets unified and the grounding status is boosted. This way, the system may identify concepts in which it has low confidence, and then decide to display its understanding to the user or make a clarification request. However, since the confidence is represented in the model, the system may also postpone error handling and identify misunderstandings at a later stage, so-called late error detection. GALATEA does not only model utterances from the user, but also from the system. The system may therefore track how its own actions affect the grounding status of concepts. Since GALATEA also resolves ellipses and anaphora, the choice of utterance realisation (for example choice of referring expression) will affect how the grounding status gets updated.

An evaluation of the complete HIGGINS system showed that the performance of GALATEA and the rest of the system looks promising, not only when utterances are correctly recognised, but also when ASR errors are introduced.

10.1.7 Making grounding decisions

In the initial implementation of HIGGINS, the grounding decisions – that is, decisions about which recognition hypotheses to consider as correct and which grounding actions to take – were, as in most dialogue systems, based on hand-crafted confidence thresholds. In such an approach, a low confidence leads to rejection, a mid confidence to a clarification or display of understanding, and a high confidence to a silent acceptance. The problem with this approach is that the thresholds used are static, and not based on any empirical material or any theoretically sound model.

Based on the data collected for evaluating HIGGINS, a decision-theoretic approach was used to build a data-driven model for making grounding decisions. Ideally, the grounding decision should take into account the uncertainty of the hypothesis, the costs involved in taking the different grounding actions, and the costs of rejecting a correct hypothesis or falsely accepting an incorrect hypothesis. Based on task analysis of the HIGGINS navigation domain, cost functions that take these factors into account were derived. It was argued that efficiency– the number of syllables uttered by the user and system – is useful as a cost measure for the navigation domain. Dialogue data was then used to estimate parameters for these cost functions, so that the grounding decision may be based on both confidence and dialogue context.

For example, it was shown how concepts with high information gain should more often be clarified than concepts with low information gain, which are either simply rejected or accepted. To silently accept a concept which is associated with a very high cost of misunderstanding, a very high confidence in this concept is required.

10.1.8 Fragmentary grounding

Fragmentary utterances, like S.2 in the following example, are commonly used as evidence of understanding in human-human dialogue:

(69) U.1: I can see a red building.
S.2: *Red?*

By using fragmentary grounding, speakers may not only improve the efficiency of the dialogue, they may also pinpoint the source of the problem (in the example, the word "red").

In this thesis, the use of fragmentary grounding for concept-level error handling in spoken dialogue systems has been explored. It has been shown how such utterances are produced and interpreted in the HIGGINS system, that is, how the system may choose the right lexical realisation and how the grounding status gets updated after the user has responded. There have previously not been many studies on the use of fragmentary clarification requests in spoken dialogue systems interacting with real users. The results from the HIGGINS evaluation showed that users of spoken dialogue systems may have difficulties understanding fragmentary clarification, especially after incorrect speech recognition hypotheses, and that further research on when to use them, how to realise them, and how to understand the user's reaction to them is needed.

One of the difficulties with fragmentary grounding utterances is that they provide little syntactic guidance, and that their pragmatic meaning therefore is dependent on context and prosody to a large extent. In two experiments, the relation between prosodic realisation and interpretation of such utterances was explored. In the first experiment, subjects listened to synthesised fragmentary grounding utterances in which prosodic features were varied systematically, and the subjects were given the task of choosing between different paraphrases. The results showed that the position and height of the F_0 peak not only affects whether the grounding utterance is interpreted as positive or negative evidence, but also which level of action is concerned in the case of negative evidence. In a second experiment, a Wizard-of-Oz setting was used to show that users of spoken dialogue systems not only perceive the differences in prosody and the pragmatic meaning of grounding utterances, but that they also change their behaviour accordingly in a human-computer dialogue setting.

10.2 Discussion and future work

10.2.1 Integration and improvements

As stated in the introduction, all aspects of error handling cannot be covered in the scope of this thesis, but the ambition has been to study how error handling may be performed in different parts of a complete spoken dialogue system. It should be noted, though, that not all of the models and methods explored in this thesis are currently integrated in the HIGGINS system. There are also many ways in which they may be improved and explored further. We will continue with a brief discussion on some of these issues.

The machine-learning methods for early error detection explored in Chapter 5 are not yet used by the HIGGINS system to sort out incorrect words. For these methods to be really useful in HIGGINS, methods for assigning probabilistic scores instead of binary decisions should be explored. A further improvement would be to do this confidence estimation on the concept-level (i.e., on the PICKERING results), resulting in concept confidence scores that may be used for modelling grounding status in GALATEA.

The action selection in HIGGINS was divided into a navigation action manager (NAM) and a grounding action manager (GAM), where only the NAM consults the domain database. The purpose of this division was to make the GAM more generic and responsive. However, if the grounding decision model proposed in Chapter 8 was to be implemented in HIGGINS, this division may not be viable. In order to make dynamic grounding decisions that take concept information gain into account, the grounding decision maker has to consult the domain database.

Much more work is needed to understand the relation between prosody and the pragmatic effects of fragmentary grounding utterances, before it results in a full model that may be used in a spoken dialogue system. However, the explorations in Chapter 9 may be regarded as a step towards such a model for Swedish.

In Chapter 6, it was shown how clarification on the perception level is handled in HIGGINS. An important next step is to model the clarification of ambiguous referring expressions or ellipses. As the system starts to make fragmentary clarification requests, users are likely to do this as well. Thus, the system must also be capable of recognising such requests.

It was also shown how the modelling of grounding status and confidence over time made it possible for the system to postpone error handling and detect errors later on (late error detection). However, while this information is stored for later use, a very simplistic model for detecting errors based on this information was used. Here, an empirical approach (such as machine-learning) would be interesting to explore, as discussed in 6.8. Another aspect of late error detection is to understand the users' reactions after display of understanding and identify the errors that they may reflect. As was evident in Chapter 8, display of understanding after incorrect recognitions very often failed, in the sense that the users did not react to them in a way that allowed the system to repair the misunderstanding.

10.2.2 Generalisation

Throughout this thesis, the guiding and navigation domain has been used in experiments and implementations. This has allowed us to reuse data and experiences between the different studies. As pointed out in previous chapters, the domain is similar to the Map Task setting used in many linguistic studies on human-human dialogue. There are several reasons why this domain is so frequently studied. Since the maps used are provided by the experimenter, it gives her control over the task and what the subjects will talk about. If we want to study conversational dialogue systems, this predictability is very useful, since the vocabulary may be restricted. The Map Task also allows the experimenter to make the maps different, in order to study how speakers deal with such problems. This has not been done in the work presented here, since the focus has been on problems that arise from speech recognition errors. Another feature of this domain is the frequent use of referring expressions and anaphora, which is interesting from a grounding perspective. As pointed out in 2.3.3.1, anaphora is not very often addressed in dialogue systems. One explanation for this is that anaphora is not typically needed in the extensively studied travel booking domain.

It remains to be investigated to what extent the results obtained in this thesis apply to other types of dialogue systems in other domains. The possibility to generalise the results on human non-understanding recovery presented in Chapter 4 was discussed in 4.4.2. The methods for early error detection investigated in Chapter 5 should be applicable to other systems and domains. However, the set of features that is found to be useful will likely vary. The generic modules in the HIGGINS dialogue system presented in Chapter 6 have been used to implement a few other domains, as mentioned in 6.8. The general model and parameters for making grounding decisions presented in Chapter 8 should be applicable to other domains. However, as discussed, the more specific estimation of the task-dependent parameters must be adapted for the domain. The tentative prosodic model presented in Chapter 9 should not be domain dependent.

As stated in the introduction, two general themes in this thesis have been to draw lessons from human error handling and to explore concept-level error handling. These are issues that may be more important for conversational dialogue systems than for command-based systems, in which more human-like error handling strategies may seem confusing and in which a typical utterance may not contain too many concepts. However, early error detection and grounding decisions are important in such systems as well, and the methods proposed here for these issues should be applicable. It is possible that other error handling issues than the ones addressed in this thesis may be more important in command-based systems. For example, the "speech graffiti" approach discussed in 2.1 may be used to provide the user with a pre-defined set of error correction commands. A much larger vocabulary and shorter utterances may also increase the usefulness of n-best lists, as discussed in 3.3.1.4.

References

Ainsworth, W. A., & Pratt, S. R. (1992). Feedback strategies for error correction in speech recognition systems. *International Journal of Man-Machine Studies, 36*, 833-842.

Allen, J. F., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L., & Stent, A. (2001a). Towards conversational human-computer interaction. *AI Magazine, 22*(4), 27-37.

Allen, J. F., & Core, M. (1997). *Draft of DAMSL: Dialog act markup in several layers*. Unpublished manuscript.

Allen, J. F., & Ferguson, G. (1994). Events and actions in interval temporal logic. *Journal of Logic and Computation, 4*(5), 531-579.

Allen, J. F., Ferguson, G., & Stent, A. (2001b). An architecture for more realistic conversational systems. In *Proceedings of the 6th international conference on Intelligent user interfaces* (pp. 1-8).

Allen, J. F., Miller, B. W., Ringger, E. K., & Sikorski, T. (1996). Robust understanding in a dialogue system. In *Proceedings of ACL 1996* (pp. 62-70).

Allen, J. F., & Perrault, C. R. (1980). Analyzing intention in utterances. *Artificial Intelligence, 15*(3), 143-178.

Allwood, J., Nivre, J., & Ahlsen, E. (1992). On the semantics and pragmatics of linguistic feedback. *Journal of Semantics, 9*(1), 1-26.

Amalberti, R., Carbonell, N., & Falzon, P. (1993). User representations of computer systems in human-computer speech interaction. *International Journal of Man-Machine Studies, 38*(4), 547-566.

Anderson, A., Bader, M., Bard, E., Boyle, E., Doherty, G., Garrod, S., Isard, S., Kowtko, J., McAllister, J., Miller, J., Sotillo, C., Thompson, H., & Weinert, R. (1991). The HCRC Map Task corpus. *Language and Speech, 34*(4), 351-366.

Austin, J. L. (1962). *How to do things with words.* Cambridge, MA: Harvard University Press.

Balentine, B., Morgan, D. P., & Meisel, W. S. (2001). *How to build a speech recognition application: a style guide for telephony dialogues.* San Ramon CA: Enterprise Integration Group.

Bangalore, S., Hakkani-Tür, D., & Tur, G. (2006). Introduction to the special issue on spoken language understanding in conversational systems. *Speech Communication, 48*(3-4), 233-462.

Baus, J., Kray, C., Krüger, A., & Wahlster, V. (2002). A resource-adaptive mobile navigation system. In *Proceedings of the 7th international conference on Intelligent user interfaces* (pp. 15-22).

Bell, L., Boye, J., & Gustafson, J. (2001). Real-time handling of fragmented utterances. In *Proceedings of NAACL 2001 Workshop: Adaptation in Dialogue Systems*. Pittsburgh, PA.

Bell, L., & Gustafson, J. (1999). Repetition and its phonetic realizations: Investigating a Swedish database of spontaneous computer directed speech. In *Proceedings of ICPhS-99* (pp. 1221-1224).

Beskow, J. (2003). *Talking heads - Models and applications for multimodal speech synthesis*. Doctoral dissertation, KTH, Department of Speech, Music and Hearing, KTH, Stockholm.

Beskow, J., Edlund, J., & Nordstrand, M. (2005). A model for multi-modal dialogue system output applied to an animated talking head. In Minker, W., Bühler, D., & Dybkjaer, L. (Eds.), *Spoken Multimodal Human-Computer Dialogue in Mobile Environments, Text, Speech and Language Technology* (pp. 93-113). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Black, A., & Lenzo, K. (2000). Limited domain synthesis. In *Proceedings of ICSLP* (pp. 410-415). Beijing, China.

Bohus, D. (2007). *Error awareness and recovery in conversational spoken language interfaces*. Doctoral dissertation, Carnegie Mellon University, Pittsburgh, PA.

Bohus, D., & Rudnicky, A. (2001). Modeling the cost of misunderstandings in the CMU Communicator dialog system. In *Proceedings of ASRU*. Madonna di Campiglio, Italy.

Bohus, D., & Rudnicky, A. (2002). *Integrating multiple knowledge sources for utterance-level confidence annotation in the CMU Communicator spoken dialog system*. Technical Report CS-190, Carnegie Mellon University, Pittsburgh, PA.

Bohus, D., & Rudnicky, A. (2005a). Constructing accurate beliefs in spoken dialog systems. In *Proceedings of ASRU 2005*.

Bohus, D., & Rudnicky, A. (2005b). Sorry, I didn't catch that! - An investigation of non-understanding errors and recovery strategies. In *Proceedings of SigDial* (pp. 128-143). Lisbon, Portugal.

Bohus, D., & Rudnicky, A. (2005c). A principled approach for rejection threshold optimization in spoken dialog systems. In *Proceedings of Interspeech* (pp. 2781-2784). Lisbon, Portugal.

Bohus, D., & Rudnicky, A. (2007). Implicitly-supervised learning in spoken language interfaces: an application to the confidence annotation problem. In *Proceedings of SigDial* (pp. 256–264). Antwerp, Belgium.

Bolinger, D. (1989). *Intonation and its uses: Melody in grammar and discourse.* London: Edward Arnold.

Bos, J., & Oka, T. (2002). An inference-based approach to dialogue system design. In *Proceedings of the 19th international conference on Computational linguistics* (pp. 1-7). Taipei, Taiwan.

Bousquet-Vernhettes, C., Privat, R., & Vigouroux, N. (2003). Error handling in spoken dialogue systems: toward corrective dialogue. In *Proceedings of Workshop on Error Handling in Spoken Dialogue Systems* (pp. 41-45). Chateau d'Oex-Vaud, Switzerland.

Bouwman, G., & Hulstijn, J. (1998). Dialogue strategy redesign with reliability measures. In *Proceedings of the First International Conference on Language Resources and Evaluation* (pp. 191-198). Granada, Spain.

Bouwman, G., Sturm, J., & Boves, L. (1999). Incorporating confidence measures in the Dutch train timetable information system developed in the Arise project. In *Proceedings of ICASSP'99* (pp. 493-496).

Boves, L. (2004). Robust conversational system design. In *Proceedings of the ITRW on Robustness Issues In Conversational Interaction*. Norwich, UK.

Boye, J., Gustafson, J., & Wirén, M. (2006). Robust spoken language understanding in a computer game. *Speech Communication, 48*(3-4), 335-353.

Boye, J., Wiren, M., & Gustafson, J. (2004). Contextual reasoning in multimodal dialogue systems: two case studies. In *Proceedings of The 8th Workshop on the Semantics and Pragmatics of Dialogue Catalogue'04* (pp. 19-21). Barcelona.

Brennan, S. E. (1996). Lexical entrainment in spontaneous dialog. In *Proceedings of ISSD* (pp. 41-44).

Brill, E. (1995). Transformation-based error-driven learning and natural langue processing: a case study in part of speech tagging. *Computational Linguistics, 21*(4), 543-565.

Brill, E., Florian, R., Henderson, J., & Mangu, L. (1998). Beyond n-grams: Can linguistic sophistication improve language modeling?. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics* (pp. 186-190). San Francisco, California.

Brown, G. (1995). *Speakers, listeners and communication.* Cambridge: Cambridge University Press.

Brown, G., & Yule, G. (2004). *Discourse analysis.* Cambridge: Cambridge University Press.

Burnett, C., Walker, M., & Hunt, A. (2004). *Speech synthesis markup language (SSML): version 1.0.* http://www.w3.org/TR/speech-synthesis/.

Campbell, N. (2007). On the use of non-verbal speech sounds in human communication. In *Proceeedings of ParaLing'07* (pp. 23-28). Saarbrücken, Germany.

Carbonell, J. G. (1983). Discourse pragmatics and ellipsis resolution in task-oriented natural language interfaces. In *Proceedings ACL 1983* (pp. 164-168).

Carletta, J., & Mellish, C. S. (1996). Risk-taking and recovery in task-oriented dialogue. *Journal of Pragmatics, 26*(1), 71-107.

Carlson, R., Granström, B., & Hunnicutt, S. (1982). A multi-language text-to-speech module. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing,* (pp. 1604-1607). Paris.

Chapanis, A. (1951). Theory and method for analyzing errors in man-machine systems. *Annals of the New York Academy of Science, 51*, 1179-1203.

Chotimongkol, A., & Rudnicky, A. (2001). N-best speech hypotheses reordering using linear regression. In *Proceedings of Eurospeech* (pp. 1829–1832).

Clark, H. H., & Schaefer, E. F. (1989). Contributing to discourse. *Cognitive Science, 13*(2), 259-294.

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

Clark, J. (1999). *XSL Transformations (XSLT) Version 1.0.* http://www.w3.org/TR/xslt.

Cohen, W. (1995). Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning.*

Daelemans, W., Zavrel, J., van der Sloot, K., & van den Bosch, A. (2003). *TiMBL: Tilburg memory based learner, version 5.0, reference guide*. Technical Report 03-10, Tilburg University.

Dahlbäck, N., Jönsson, A., & Ahrenberg, L. (1993). Wizard of Oz studies – why and how. In *Proceedings from the 1993 International Workshop on Intelligent User Interfaces* (pp. 193-200).

Dascal, M. (1999). Introduction: Some questions about misunderstanding. *Journal of Pragmatics, 31*(6), 753-762.

Dutoit, T., Pagel, V., Pierret, N., Bataille, F., & Vreken, O. v. d. (1996). The MBROLA project: Towards a set of high-quality speech synthesizers free of use for non-commercial purposes. In *Proceedings of ICSLIP '96* (pp. 1393-1396).

Edlund, J., & Heldner, M. (2005). Exploring prosody in interaction control. *Phonetica, 62*(2-4), 215-226.

Edlund, J., & Heldner, M. (2006). /nailon/ - software for online analysis of prosody. In *Proc of Interspeech 2006 ICSLP.* Pittsburgh PA, USA.

Edlund, J., Heldner, M., & Gustafson, J. (2006). Two faces of spoken dialogue systems. In *Interspeech 2006 - ICSLP Satellite Workshop Dialogue on Dialogues: Multidisciplinary Evaluation of Advanced Speech-based Interactive Systems.* Pittsburgh PA, USA.

Edlund, J., & Hjalmarsson, A. (2005). Applications of distributed dialogue systems: the KTH Connector. In *Proceedings of ISCA Tutorial and Research Workshop on Applied Spoken Language Interaction in Distributed Environments (ASIDE 2005)*. Aalborg, Denmark.

Edlund, J., Skantze, G., & Carlson, R. (2004). Higgins - a spoken dialogue system for investigating error handling techniques. In *Proceedings of the International Conference on Spoken Language Processing, ICSLP 04* (pp. 229-231). Jeju, Korea.

Ericsson, S. (2005). *Information enriched constituents in dialogue*. Doctoral dissertation, Göteborg University.

Esteve, Y., Raymond, C., Bechet, F., & De Mori, R. (2003). Conceptual decoding for spoken dialog systems. In *Proceedings of Eurospeech* (pp. 617–620).

Ferrer, L., Shriberg, E., & Stolcke, A. (2002). Is the speaker done yet? Faster and more accurate end-of utterance detection using prosody. In *Proceedings of ICSLP* (pp. 2061-2064).

Filipsson, M., & Bruce, G. (1997). *LUKAS - a preliminary report on a new Swedish speech synthesis.* Department of Linguistics and Phonetics, Lund University.

Flycht-Eriksson, A. (2001). *Domain knowledge management in information-providing dialogue systems*. Licentiate dissertation, Linköping University.

Fraser, N. M., & Gilbert, G. N. (1991). Simulating speech systems. *Computer Speech and Language, 5*(1), 81-99.

Gabsdil, M. (2003). Clarification in spoken dialogue systems. In *Natural Language Generation in Spoken and Written Dialogue: Papers from the 2003 Spring Symposium* (pp. 28-35).

Gabsdil, M., & Bos, J. (2003). Combining acoustic confidence scores with deep semantic analysis for clarification dialogues. In *Proceedings of IWCS-5* (pp. 137-150).

Gabsdil, M., & Lemon, O. (2004). Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems. In *Proceedings of ACL 2004* (pp. 343-350).

Garrod, S., & Anderson, A. (1987). Saying what you mean in dialogue: A study in conceptual and semantic co-ordination. *Cognition, 27*(2), 181-218.

Ginzburg, J., & Cooper, R. (2001). Resolving ellipsis in clarification. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics* (pp. 236-243).

Ginzburg, J., Gregory, H., & Lappin, S. (2001). SHARDS: Fragment resolution in dialogue. In *Proceedings of the 4th International conference on Computational Semantics*. Tilburg.

Gorin, A. L., Riccardi, G., & Wright, J. H. (1997). How may I help you?. *Speech Communication, 23*, 113-127.

Grosz, B. J., Joshi, A. K., & Weinstein, S. (1995). Centering: a framework for modeling the local coherence of discourse. *Computational Linguistics, 21*(2), 203-225.

Gustafson, J., Bell, L., Beskow, J., Boye, J., Carlson, R., Edlund, J., Granström, B., House, D., & Wirén, M. (2000). AdApt - a multimodal conversational dialogue system in an apartment domain. In *Proc. of ICSLP 2000, 6th Intl Conf on Spoken Language Processing* (pp. 134-137). Beijing.

Gustafson, J., Larsson, A., Carlson, R., & Hellman, K. (1997). How do system questions influence lexical choices in user answers?. In *Proc of Eurospeech '97, 5th European Conference on Speech Communication and Technology* (pp. 2275-2278). Rhodes, Greece.

Gårding, E. (1998). Intonation in Swedish. In Hirst, D., & Di Cristo, A. (Eds.), *Intonation Systems* (pp. 112-130). Cambridge: Cambridge University Press.

Harris, T. K., & Rosenfeld, R. (2004). A universal speech interface for appliances. In *Proceedings of ICSLP.*

Hazen, T., Seneff, S., & Polifroni, J. (2002). Recognition confidence scoring and its use in speech understanding systems. *Computer Speech and Language, 16*(1), 49-67.

He, Y., & Young, S. (2003). A data-driven spoken language understanding system. In *Proceedings of IEEE Automatic Speech* (pp. 583–588).

Heisterkamp, P., & McGlashan, S. (1996). Units of dialogue management: an example. In *Proceedings of ICSLP 1996* (pp. 200-203).

Hinkle, D. E., Wiersma, W., & Jurs, S. G. (1994). *Applied statistics for the behavioral sciences.* Boston: Houghton Mifflin Company.

Hirst, D., & Cristo, A. D. (1998). A survey of intonation systems. In Hirst, D., & Di Cristo, A. (Eds.), *Intonation Systems* (pp. 1-45). Cambridge: Cambridge University Press.

Hirst, G., McRoy, S., Heeman, P., Edmonds, P., & Horton, D. (1994). Repairing conversational misunderstandings and non-understandings. *Speech Communication, 15*, 213-230.

Hjalmarsson, A., Wik, P., & Brusk, J. (2007). Dealing with DEAL: a dialogue system for conversation training. In *Proceedings of SigDial* (pp. 132-135). Antwerp, Belgium.

House, D. (2003). Perceiving question intonation: the role of pre-focal pause and delayed focal peak. In *Proc of ICPhS, XV Intl Conference of Phonetic Sciences* (pp. 755-758). Barcelona, Spain.

Huang, X., Acero, A., & Hon, H-W. (2001). *Spoken language processing: a guide to theory, algorithm and system development.* Prentice Hall.

Jackson, E., Appelt, D., Bear, J., Moore, R., & Podlozny, A. (1991). A template matcher for robust NL interpretation. In *Proceedings of DARPA Speech and Natural Language Workshop.*

Jiang, H. (2005). Confidence measures for speech recognition: A survey. *Speech Communication, 45*(4), 455-470.

Jonson, R. (2006). Dialogue context-based re-ranking of ASR hypotheses. In *Proceedings of Spoken Language Technology Workshop, IEEE* (pp. 174-177).

Jurafsky, D., & Martin, J. (2000). *Speech and language processing.* Englewood, NJ, US: Prentice Hall, Inc.

Jönsson, A. (1997). A model for habitable and efficient dialogue management for natural language interaction. *Natural Language Engineering, 3*, 103-122.

Kasper, W., Kiefer, B., Krieger, H., Rupp, C., & Worm, K. (1999). Charting the depths of robust speech parsing. In *Proceedings of ACL.*

Katagiri, Y., & Shimojima, A. (2000). Display acts in grounding negotiations. In *Proceedings of Gotalog.*

Kilpeläinen, P. (1992). *Tree matching problems with applications to structured text databases.* Doctoral dissertation, Department of Computer Science, University of Helsinki.

Klahr, D., Langley, P., & Neches, R. T. (1987). *Production system models of learning and development.* MIT Press.

Knight, S., Gorrell, G., Rayner, M., Milward, D., Koeling, R., & Lewin, I. (2001). Comparing grammar-based and robust approaches to speech understanding: a case study. In *Proceedings of Eurospeech 2001* (pp. 1779-1882).

Krahmer, E., Swerts, M., Theune, M., & Weegels, M. (2001). Error detection in spoken human-machine interaction. *International Journal of Speech Technology, 4*(1), 19-29.

Ladd, D. R. (1996). *Intonation phonology.* Cambridge: Cambridge University Press.

Lager, T. (1999). The µ-TBL system: logic programming tools for transformation-based learning. In *Proceedings of the Third International Workshop on Computational Natural Language Learning.*

Larsson, S. (2002). *Issue-based dialogue management.* Doctoral dissertation, Goteborg University.

Larsson, S. (2003). Interactive communication management in an issue-based dialogue system. In Kruijff-Korbayova, I., & Kosny, C. (Eds.), *Proceedings of the 7th Workshop on the Semantics and Pragmatics of Dialogue (DiaBruck)* (pp. 75-82). Saarbrücken, Germany.

Larsson, S., & 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, 6*(3-4), 323-340.

Levin, E., Pieraccini, R., & 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.

Levinson, S. C. (1983). *Pragmatics.* Cambridge: Cambridge University press.

Levow, G. (1998). Characterizing and recognizing spoken corrections in human-computer dialogue. In *Proceedings of COLING/ACL.*

Lippmann, R. P. (1997). Speech recognition by machines and humans. *Speech Communication, 22*, 1-15.

Litman, D., Hirschberg, J., & Swerts, M. (2000). Predicting automatic speech recognition performance using prosodic cues. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics* (pp. 218-225). Seattle, WA.

Litman, D., Swerts, M., & Hirschberg, J. (2006). Characterizing and predicting corrections in spoken dialogue systems. *Computational Linguistics, 32*(3), 417-438.

McGlashan, S., Burnett, D. C., Carter, J., Danielsen, P., Ferrans, J., Hunt, A., Lucas, B., Porter, B., Rehor, K., & Tryphonas, S. (2004). *Voice extensible markup language (VoiceXML): version 2.0.* http://www.w3.org/TR/voicexml20/.

McRoy, S. W., & Hirst, G. (1995). The repair of speech act misunderstandings by abductive inference. *Computational Linguistics, 21*(4), 435-478.

McTear, M., O'Neill, I., Hanna, P., & Liu, X. (2005). Handling errors and determining confirmation strategies - An object-based approach. *Speech Communication, 45*(3), 249-269.

Mellish, C. (1989). Some chart-based techniques for parsing ill-formed input. In *Proceedings of ACL.*

Mitchell, T. M. (1997). *Machine learning.* Singapore: McGraw-Hill Book Co.

Oviatt, S., Levow, G., MacEachern, M., & Kuhn, K. (1996). Modeling hyperarticulate speech during human-computer error resolution. In *Proceedings of ICSLP.*

Paek, T. (2001). Empirical methods for evaluating dialog systems. In *ACL 2001 Workshop on Evaluation Methodologies for Language and Dialogue Systems.*

Paek, T., & Horvitz, E. (2003). On the utility of decision-theoretic hidden subdialog. In *ISCA Workshop on Error Handling in Spoken Dialogue Systems* (pp. 95-100).

Pfleger, N., Engel, R., & Alexandersson, J. (2003). Robust multimodal discourse processing. In *Proceedings of the 7th Workshop on the Semantics and Pragmatics of Dialogue (DiaBruck).*

Pieraccini, R., & Huerta, J. (2005). Where do we go from here? Research and commercial spoken dialogue systems. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue* (pp. 1-10). Lisbon, Portugal.

Purver, M. (2004). *The theory and use of clarification requests in dialogue*. Doctoral dissertation, University of London.

Purver, M., Ginzburg, J., & Healey, P. (2001). On the means for clarification in dialogue. In *Proceedings of SIGdial 2001* (pp. 235-255).

Rayner, M., Carter, D., Digalakis, V., & Price, P. (1994). Combining knowledge sources to reorder n-best speech hypothesis lists. In *Proceedings of the 1994 ARPA Workshop on Human Language Technology.*

Reason, J. (1990). *Human error.* Cambridge University Press.

Reiter, E., & Dale, R. (2000). *Building natural language generation systems.* Cambridge University Press.

Rieser, V. (2004). *Fragmentary clarifications on several levels for robust dialogue systems*. Master's thesis, University of Edinburgh.

Ringger, E. K., & Allen, J. F. (1997). Robust error correction of continuous speech recognition. In *Proceedings of the ESCA-NATO Robust Workshop*.

Rodriguez, K. J., & Schlangen, D. (2004). Form, intonation and function of clarification requests in German task oriented spoken dialogues. In *Proceedings of Catalog '04*. Barcelona, Spain.

Satta, G., & Stock, O. (1994). Bidirectional context-free grammar parsing for natural language processing. *Artificial Intelligence, 69*(1-2), 123-164.

Schegloff, E. (1982). Discourse as an interactional achievement: Some uses of 'uh huh' and other things that come between sentences. In Tannen, D. (Ed.), *Analyzing Discourse: Text and Talk* (pp. 71-93). Washington, D.C., USA: Georgetown University Press.

Schegloff, E. (1992). Repair after next turn: the last structurally provided defense of intersubjectivity in conversation. *American Journal of Sociology, 97*(5), 1295-1345.

Schegloff, E., & Sacks, H. (1973). Opening up closings. *Semiotica, 8*, 289-327.

Schlangen, D. (2003). *A coherence-based approach to the interpretation of non-sentential utterances in dialogue*. Doctoral dissertation, School of Informatics, University of Edinburgh.

Schlangen, D. (2004). Causes and strategies for requesting clarification in dialogue. In *Proceedings of SIGdial 2004* (pp. 136-143).

Schlangen, D., & Fernández, R. (2007a). Beyond repair? Testing the limits of the conversational repair system. In *Proceedings of SigDial.* Antwerp, Belgium.

Schlangen, D., & Fernández, R. (2007b). Speaking through a noisy channel: Experiments on inducing clarification behaviour in human-human dialogue. In *Proceedings of Interspeech*. Antwerp, Belgium.

Searle, J. S. (1979). *Expression and meaning: studies in the theory of speech acts.* Cambridge University Press.

Segarra, E., Sanchis, E., Garcia, F., & Hurtado, L. F. (2002). Extracting semantic information through automatic learning techniques. *International Journal of Pattern Recognition, 16*(3), 301–307.

Seward, A. (2003). *Efficient methods for automatic speech recognition*. Doctoral dissertation, Department of Speech, Music and Hearing, KTH, Stockholm, Sweden.

Shriberg, E. (1994). *Preliminaries to a theory of speech disfluencies*. Doctoral dissertation, University of California.

Sjölander, K., & Beskow, J. (2000). WaveSurfer - an open source speech tool. In Yuan, B., Huang, T., & Tang, X. (Eds.), *Proceedings of ICSLP 2000, 6th Intl Conf on Spoken Language Processing* (pp. 464-467). Beijing.

Skantze, G. (2002). Coordination of referring expressions in multimodal human-computer dialogue. In *Proceedings of ICSLP 2002* (pp. 553-556). Denver, Colorado, USA.

Skantze, G., & Edlund, J. (2004). Robust interpretation in the Higgins spoken dialogue system. In *ISCA Tutorial and Research Workshop (ITRW) on Robustness Issues in Conversational Interaction*. Norwich, UK.

Skantze, G., Edlund, J., & Carlson, R. (2006). Talking with Higgins: Research challenges in a spoken dialogue system. In André, E., Dybkjaer, L., Minker, W., Neumann, H., & Weber, M. (Eds.), *Proceedings of Perception and Interactive Technologies* (pp. 193-196). Springer.

Stuttle, M., Williams, J. D., & Yound, S. (2004). A framework for dialogue data collection with a simulated ASR-channel. In *Proceedings of ICSLP*. Jeju, South Korea.

Traum, D. (1994). *A computational theory of grounding in natural language conversation*. Doctoral dissertation, University of Rochester.

Turunen, M. (2004). *Jaspis - a spoken dialogue architecture and its applications*. Doctoral dissertation, University of Tampere, Department of Computer Sciences.

van den Bosch, A., & Daelemans, W. (1993). Data-oriented methods for grapheme-to-phoneme conversion. In *Proceedings of European Chapter of ACL* (pp. 45-53).

van Heuven, V. J., Hann, J., & Kirsner, R. S. (1999). Phonetic correlates of sentence type in Dutch: Statement, question and command. In *Proceedings of ESCA International Workshop on Dialogue and Prosody* (pp. 35-40).

van Noord, G., Bouma, G., Koeling, R., & Nederhof, M-J. (1999). Robust grammatical analysis for spoken dialogue systems. *Natural Language Engineering, 5*(1), 45-93.

Walker, M., Kamm, C., & Litman, D. (2000a). Towards developing general models of usability with PARADISE. *Natural Language Engineering, 6*, 363-377.

Walker, M., Wright, J., & Langkilde, I. (2000b). Using natural language processing and discourse features to identify understanding errors in a spoken dialogue system. In *Proceedings of the Seventeenth International Conference on Machine Learning*.

Walker, M. A., Langkilde, I., Wright, J., Gorin, A., & Litman, D. J. (2000c). Learning to predict problematic situations in a spoken dialogue system: experiments with How may I help you?. In *Proceedings of North American Meeting of the Association of Computational Linguistics*.

Wallers, Å., Edlund, J., & Skantze, G. (2006). The effects of prosodic features on the interpretation of synthesised backchannels. In André, E., Dybkjaer, L., Minker, W., Neumann, H., & Weber, M. (Eds.), *Proceedings of Perception and Interactive Technologies* (pp. 183-187). Springer.

Ward, N. (2004). Pragmatic functions of prosodic features in non-lexical utterances. In *Proceedings of Speech Prosody* (pp. 325-328).

Ward, W. (1989). Understanding spontaneous speech. In *Proceedings of the workshop on Speech and Natural Language Understanding* (pp. 137-141). Philadelphia, Pennsylvania.

Weegels, M. (2000). User's conceptions of voice-operated information services. *International Journal of Speech Technology, 3*(2), 75-82.

Weigard, E. (1999). Misunderstanding: the standard case. *Journal of Pragmatics, 31*(6), 763-785.

Wessel, F., Schluter, R., Macherey, K., & Ney, H. (2001). Confidence measures for large vocabulary continuous speech recognition. *IEEE Trans. Speech Audio Process, 9*(3).

Williams, J. D., & Young, S. (2007). Partially observable markov decision processes for spoken dialog systems. *Computer Speech and Language, 21*(2), 393-422.

Yngve, V. (1970). On getting a word in edgewise. In *Papers from the sixth regional meeting of the Chicago Linguistic Society* (pp. 567-578). Chicago.

Young, S. (2002). Talking to machines (statistically speaking). In *Proceedings of ICSLP* (pp. 9-12). Denver, CO.