Data-driven Methods for Spoken Dialogue Systems

Applications in Language Understanding, Turn-taking, Error Detection, and Knowledge Acquisition

RAVEESH MEENA

Doctoral Thesis
Stockholm, Sweden 2016

KTH Royal Institute of Technology
School of Computer Science and Communication
Department of Speech, Music and Hearing
100 44 Stockholm, Sweden

ISBN 978-91-7595-866-8
ISSN 1653-5723
ISRN KTH/CSC/A-16/03-SE
TRITA-CSC-A-2016:03

Akademisk avhandling som med tillstånd av Kungliga Tekniska Högskolan framlägges till offentlig granskning för avläggande av filosofie doktorsexamen fredagen den 18 mars klockan 10.00 i sal F3, Kungliga Tekniska Högskolan, Lindstedtsvägen 26, Stockholm.

Printed by Universitetsservice US-AB

Abstract

This thesis explores data-driven methodologies for development of spoken dialogue systems. The motivation for this is that dialogue systems have to deal with the ubiquity of *variability*—in user behavior, in the performance of the various speech and language processing sub-components, and in the dynamics of the task domain. However, as the predominant methodology for dialogue system development is to handcraft the sub-components, these systems typically lack robustness in dealing with the challenges arising from variability. Data-driven methods, on the other hand, have been shown to offer robustness to variability in various domains of computer science and artificial intelligence and have steadily become popular in spoken dialogue systems research.

This thesis makes four novel contributions to the data-driven approaches to spoken dialogue systems development. First, a data-driven approach for spoken language understanding (SLU) is presented. The presented approach caters to two important requirements of SLU: robust parsing to deal with "ungrammatical" user input (due to spontaneous nature of speech and errors in speech recognition), and the preservation of structural relations among concepts in the semantic representation. Existing approaches to SLU address only one of these two challenges, but not both.

The second contribution is a data-driven approach for timing system responses. The proposed model exploits prosody, syntax, semantics, and dialogue context, for determining when in a user's speech it is appropriate for the system to give a response.

The third contribution is a data-driven approach for error detection. While previous works have focused on error detection for online adaptation and have only been tested on single domains, the presented approach contributes models for offline error analysis, and is trained and evaluated across three different dialogue system corpora.

Finally, an implicitly supervised learning approach for knowledge acquisition is presented. The approach leverages the interactive setting of spoken dialogue and the power of crowdsourcing and demonstrates a novel system that can acquire street-level geographic details by interacting with users. The system is initiated with minimal knowledge and exploits the interactive setting to solicit information and seek verifications of the knowledge acquired over the interactions.

The general approach taken in this thesis is to model dialogue system tasks as a *classification* problem and investigate features from the discourse context that can be used to train a classifier. During the course of this work, various lexical, syntactic, semantic, prosodic, and contextual features have been investigated. A general discussion on the contributions of these features in modelling the aforementioned tasks constitutes one of the main contributions of this work. Yet another key focus of this thesis is on training models for online use in dialogue systems. Therefore only automatically extractable features are used (requiring no manual labelling) to train the models. In addition, the model performances have been assessed on both ASR results and transcriptions, in order to investigate the robustness of the proposed methods.

The central hypothesis of this thesis is that the models of language understanding, turn-taking, and error detection trained using the features proposed here perform better than their corresponding baseline models. The empirical validity of this claim has been assessed through both quantitative and qualitative evaluations, using both objective and subjective measures.

Sammanfattning

Den här avhandlingen utforskar datadrivna metoder för utveckling av talande dialogsystem. Motivet bakom sådana metoder är att dialogsystem måste kunna hantera en stor variation, i såväl användarnas beteende, som i prestandan hos olika tal- och språkteknologiska delkomponenter. Traditionella tillvägagångssätt, som baseras på handskrivna komponenter i dialogsystem, har ofta svårt att uppvisa robusthet i hanteringen av sådan variation. Datadrivna metoder har visat sig vara robusta mot variation i olika problem inom datavetenskap och artificiell intelligens, och har på senare tid blivit populära även inom forskning kring talande dialogsystem.

Den här avhandlingen presenterar fyra nya bidrag till datadrivna metoder för utveckling av talande dialogsystem. Det första bidraget är en datadriven metod för semantisk tolkning av talspråk. Den föreslagna metoden har två viktiga egenskaper: robust hantering av ”ogrammatisk” indata (på grund av talets spontana natur samt fel i taligenkänning), samt bevarande av strukturella relationer mellan koncept i den semantiska representationen. Tidigare metoder för semantisk tolkning av talspråk har typiskt sett endast hanterat den ena av dessa två utmaningar.

Det andra bidraget i avhandlingen är en datadriven metod för turtagning i dialogsystem. Den föreslagna modellen utnyttjar prosodi, syntax, semantik samt dialogkontext för att avgöra när i användarens tal som det är lämpligt för systemet att ge respons.

Det tredje bidraget är en data-driven metod för detektering av fel och missförstånd i dialogsystem. Där tidigare arbeten har fokuserat på detektering av fel on-line och endast testats i enskilda domäner, presenterats här modeller för analys av fel såväl off-line som on-line, och som tränats samt utvärderats på tre skilda dialogsystemkorpusar.

Slutligen presenteras en metod för hur dialogsystem ska kunna tillägna sig ny kunskap genom interaktion med användaren. Metoden är utvärderad i ett scenario där systemet ska bygga upp en kunskapsbas i en geografisk domän genom så kallad "crowdsourcing". Systemet börjar med minimal kunskap och använder den talade dialogen för att både samla ny information och verifiera den kunskap som inhämtats.

Den generella ansatsen i den här avhandlingen är att modellera olika uppgifter för dialogsystem som *klassificeringsproblem*, och undersöka särdrag i diskursens kontext som kan användas för att träna klassificerare. Under arbetets gång har olika slags lexikala, syntaktiska, prosodiska samt kontextuella särdrag undersökts. En allmän diskussion om dessa särdrags

bidrag till modellering av ovannämnda uppgifter utgör ett av avhandlingens huvudsakliga bidrag. En annan central del i avhandlingen är att träna modeller som kan användas direkt i dialogsystem, varför endast automatiskt extraherbara särdrag (som inte kräver manuell uppmärkning) används för att träna modellerna. Vidare utvärderas modellernas prestanda på såväl taligenkänningsresultat som transkriptioner för att undersöka hur robusta de föreslagna metoderna är.

Den centrala hypotesen i denna avhandling är att modeller som tränas med de föreslagna kontextuella särdragen presterar bättre än en referensmodell. Giltigheten hos denna hypotes har bedömts med såväl kvalitativa som kvantitativa utvärderingar, som nyttjar både objektiva och subjektiva mått.

Acknowledgements

This thesis is a culmination of five years of research endeavors, of me and my collaborators, in the area of spoken dialogue systems. It has been a great learning experience and I have thoroughly enjoyed the pursuit of the scientific process. I would like to thank everyone to whom I owe this.

First, I want to thank my main supervisor Gabriel Skantze for teaching me how to conduct scientific inquiry, enabling me in working on diverse research topics, showing me how to write and present research work, being available for discussions of all sorts, always giving me positive feedback and encouragement. Thank you also for helping me with writing the thesis and reviewing the many draft versions.

I would like to also thank my second supervisor Joakim Gustafson, who has always been there for discussions and shared his insights in the research area of speech communication.

I collaborated with Johan Boye on developing the system for crowdsourcing, presented here in Chapter 6. Although it was a short project, it has been a rewarding experience and has enabled me to think about machine learning through spoken interaction. Thank you also for reviewing this thesis and giving useful comments.

I want to extend my thanks to José Lopes for all the discussions, questions and ideas that came up during our joint work on error detection, presented here in Chapter 5.

I want to thank Gabriel, Joakim, Johan and others, who have worked hard on obtaining funding for the research work presented here. I want to extend my thanks to the various funding agencies for supporting this work and covering the travel expenses.

Next, I want to thank my officemate, Martin Johansson, who has always lent an ear to my questions and excitements. Thank you for listening, helping, and being patient.

I have enjoyed being at the department from day one. I have admired the warm traditions of Fredags fika and the culture of celebration. Thank you all for your warmness, and above all for your inspiring research works and discussions. In particular, thank you Kalin and Giampi for your inputs on machine learning topics, Catha, Sofia, Niklas for the many discussions at the lunch table and your company.

I would like to also thank my former supervisors, Ivana Kruijff-Korbayová and Geert-Jan Kruijff, who gave me the opportunity to work as a research assistant in their group at DFKI Language Technology Lab

in Saarbrücken. This is where I gained hands on experience in spoken dialogue systems and developed stronger interests, which later led me to pursue a Ph.D. in this area.

I have learned a lot from discussions with fellow researcher Pierre Lison. Thank you for always encouraging me in my pursuits and for reading and commenting on parts of this thesis.

I want to thank Kristiina Jokinen for giving me the opportunity to work on the WikiTalk project (at eNTERFACE'12 workshop), which enabled me to learn about non-verbal modalities of communication.

I have found tremendous inspiration from interacting with fellow researchers at the Young Researchers' Roundtable on Spoken Dialogue Systems (YRRSDS). Thank you everyone in the community.

I also want to thank all the participants of the various experiments presented in this thesis.

I want to thank Verena, Kanishk, and Els for proof-reading parts of the thesis. All remaining errors and mistakes are solely due to me.

Thank you all my dearest friends for being ever-supportive and bringing joy to my life.

Lastly, I want to thank my family for their unconditional love and support: my sisters Smita and Priya, and my parents Motiyan and Gopal.

Preamble

In the fall of year 2006, I was visiting Europe and this was the first time I had stepped outside India. Among the many impressive things about this part of the world, I was most amazed by the use of vending machines for dispensing train and bus tickets (and even sausages, in The Netherlands). I could buy a ticket in no matter of time, my astonishment had no end. In comparison[1], back then in India, I would spend quarters of hours jostling with others in crowded queues at the ticket counters. I think my personal record stands somewhere above an hour. Every time I had passed by a vending machine in Europe, I had wished for something like that back home. While that would have solved problems for some of us, I reckoned that for most people (given that only about 65% of the masses were literate), interactions with these seemingly effective machines—using the graphical user interface and the arrays of push-buttons—would still be an intimidating experience.

In the fall of 2007, I came to Europe again, and it was only during an introductory course in Language Technology, in my master's studies that I first came to learn about spoken language interfaces. It was an equally astonishing moment—I reckoned that if those ticket dispensing machines could be equipped with spoken language interfaces and skills in collaborative problem solving, they could potentially make the user experience less intimidating and help to cut the queues short.

In subsequent years, I learned more about spoken dialogue systems and their applications. I have grown fond of the research area, due to the sheer potential of dialogue systems in making technological advances accessible to common people, the interdisciplinary nature of research required to develop them, and the interactions with the vastly talented community of fellow researchers engaged in pushing the frontiers forward. This thesis is my contribution to the community—a tiny drop in the ocean—but I have some sense of fulfilment.

[1] This is just my personal opinion and should not be seen as the East vs. West debate.

Publications and contributors

A major part of this thesis is based on work previously presented in conferences, workshops, and journal articles. Some of the works have been done in collaboration with others. Below, we present the publications this thesis is based on, and the contributions of the author and the collaborators:

Chapter 3

The chapter is based on the following two publications:

Meena, R., Skantze, G., & Gustafson, J. (2012). A Data-driven Approach to Understanding Spoken Route Directions in Human–Robot Dialogue. *In INTERSPEECH-2012,* (pp. 226–229). Portland, OR, USA.

Meena, R., Skantze, G., & Gustafson, J. (2013). Human Evaluation of Conceptual Route Graphs for Interpreting Spoken Route Descriptions. *In Proceedings of IWCS 2013 Workshop on Computational Models of Spatial Language Interpretation and Generation (CoSLI-3),* (pp. 13–18). Potsdam, Germany

The chapter presents a *Chunking parser* based approach for semantic interpretation of spoken utterances. The framework was first developed by Gabriel Skantze and evaluated on transcribed (Swedish) utterances in Johansson et al. (2011). The author further developed the framework, applied it to the interpretation of spoken utterances in English, conducted quantitative and qualitative evaluations, and presented the research work, under the supervision of Gabriel.

Chapter 4

This chapter presents a data-driven model for timing system responses. Data collected from a previous system developed by Gabriel Skantze (Skantze, 2012), was used by the author to train more sophisticated models, with inputs from Gabriel Skantze and Joakim Gustafson. The author conducted the quantitative and qualitative evaluations of the proposed method. The models and the findings were presented in the following conference paper and journal article (on which this chapter is based). They were written with supervision and inputs from Gabriel.

Meena, R., Skantze, G., & Gustafson, J. (2013). A Data-driven Model for Timing Feedback in a Map Task Dialogue System. *In Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL),* (pp. 375–383). Metz, France. (*)

(*) *Nominated for Best Paper Award at SIGDIAL 2013*

Meena, R., Skantze, G., & Gustafson, J. (2014). Data-driven Models for timing feedback responses in a Map Task dialogue system. *Computer Speech and Language*, 28(4), pp. 903–922.

Chapter 5

This chapter presents methods for error detection that were earlier published in:

Meena, R., Lopes, J., Skantze, G., & Gustafson, J. (2015). Automatic Detection of Miscommunication in Spoken Dialogue Systems. *In Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL),* (pp. 354–363). Prague, Czech Republic

The paper is an outcome of the joint work with José Lopes. We conducted independent experiments—I worked with the Cambridge dataset while José explored the Let's Go and the SweCC datasets. Following this, we prepared a list of features that were common to the three datasets. Using this feature set, the author ran all the experiments again for presenting the models and the findings in the paper. All through this work, Gabriel and Joakim provided their expert inputs.

A scheme for error analysis was developed by the author with inputs from Gabriel, following this publication. An analysis of error causes, using this scheme is presented in this chapter.

Chapter 6

The chapter is based on the following paper:

Meena, R., Boye, J., Skantze, G., & Gustafson, J. (2014). Crowdsourcing Street-level Geographic Information Using a Spoken Dialogue System. *In Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL),* (pp. 2–11). Philadelphia, PA, USA.

The paper presents a dialogue system for crowdsourcing geographic information. The requirement analysis for the system was jointly done by

the four co-authors. The author implemented the system, encoded a scheme for implicit-learning, and conducted the in-lab crowdsourcing experiment. On the crowd-sourced data, Johan Boye conducted the analysis using the CityCrowdSource Trust software package and the author performed the analysis of the learning rate. The author wrote the paper with inputs from Johan and Gabriel. All through this work Gabriel, Johan, and Joakim gave valuable ideas for data analysis.

Complete list of publications

§ *indicates papers related to this thesis work*
% *indicates papers related to other projects during the Ph.D. studies*

Articles

Chapter 4 Meena, R., Skantze, G., & Gustafson, J. (2014). Data-driven Models for timing feedback responses in a Map Task dialogue system. *Computer Speech and Language*, 28(4), pp. 903–922.

% Lison, P., & Meena, R. (2014). Spoken Dialogue Systems: The new frontier in human–computer interaction. *In XRDS Crossroads The ACM Magazine for Students*, (pp. 46–51). US: ACM.

International conferences and workshops

Chapter 5 Meena, R., Lopes, J., Skantze, G., & Gustafson, J. (2015). Automatic Detection of Miscommunication in Spoken Dialogue Systems. *In Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, (pp. 354–363). Prague, Czech.

§ Lopes, J., Salvi, G., Skantze, G., Abad, A., Gustafson, J., Batista, F., Meena, R., & Trancoso, I. (2015). Detecting Repetitions in Spoken Dialogue Systems Using Phonetic Distances. *In INTERSPEECH-2015*, (pp. 1805–1809). Dresden, Germany.

Chapter 6 Meena, R., Boye, J., Skantze, G., & Gustafson, J. (2014). Crowdsourcing Street-level Geographic Information Using a Spoken Dialogue System. *In Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, (pp. 2–11). Philadelphia, PA, USA.

§ Meena, R., Skantze, G., & Gustafson, J. (2013). A Data-driven Model for Timing Feedback in a Map Task Dialogue System. *In Proceedings of the 14th Annual Meeting of the Special*

Interest Group on Discourse and Dialogue (SIGDIAL), (pp. 375–383). Metz, France. (*)

(*) ***Nominated for Best Paper Award at SIGDIAL 2013***

§ Meena, R., Skantze, G., & Gustafson, J. (2013). The Map Task Dialogue System: A Test-bed for Modelling Human-like Dialogue. *In Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, (pp. 366–368). Metz, France.

Chapter 3 Meena, R., Skantze, G., & Gustafson, J. (2013). Human Evaluation of Conceptual Route Graphs for Interpreting Spoken Route Descriptions. *In Proceedings of the IWCS 2013 Workshop on Computational Models of Spatial Language Interpretation and Generation (CoSLI-3)*, (pp. 13–18). Potsdam, Germany

% Meena, R., Jokinen, K., & Wilcock, G. (2012). Integration of Gestures and Speech in Human–Robot Interaction. *In Proceedings of the IEEE 3rd International Conference on Cognitive Infocommunications (CogInfoCom 2012)*, (pp. 673–678). Kosice, Slovakia.

% Csapo, A., Gilmartin, E., Grizou, J., Han, J., Meena, R., Anastasiou, D., Jokinen, K., & Wilcock, G. (2012). Multimodal Conversational Interaction with a Humanoid Robot. *In Proceedings of the IEEE 3rd International Conference on Cognitive Infocommunications (CogInfoCom 2012)*, (pp. 667–672). Kosice, Slovakia.

% Csapo, A., Gilmartin, E., Grizou, J., Han, J., Meena, R., Anastasiou, D., Jokinen, K., & Wilcock, G. (2012). Open-Domain Conversation with a NAO Robot. *In Proceedings of the IEEE 3rd International Conference on Cognitive Infocommunications (CogInfoCom 2012)*. Kosice, Slovakia.

Chapter 3 Meena, R., Skantze, G., & Gustafson, J. (2012). A Data-driven Approach to Understanding Spoken Route Directions in Human–Robot Dialogue. *In INTERSPEECH-2012*, (pp. 226–229). Portland, OR, USA.

Swedish conferences and workshops

§ Meena, R., Boye, J., Skantze, G., & Gustafson, J. (2014). Using a

Spoken Dialogue System for Crowdsourcing Street-level Geographic Information. *In Proceedings of the 2nd Workshop on Action, Perception and Language, SLTC 2014*. Uppsala, Sweden.

% Meena, R., Dabbaghchian, S., & Stefanov, K. (2014). A data-driven approach to detection of interruptions in human–human conversations. In Heldner, M. (Ed.), *In Proceedings of FONETIK 2014*, (pp. 29–32). Stockholm, Sweden.

§ Meena, R., Skantze, G., & Gustafson, J. (2012). A Chunking Parser for Semantic Interpretation of Spoken Route Directions in Human–Robot Dialogue. *In Proceedings of the 4th Swedish Language Technology Conference, SLTC 2012,* (pp. 55–56). Lund, Sweden.

Contents

Chapter 1

Introduction

1.1 Spoken human–computer interaction

The conventional medium for human–computer interaction is the graphical user interface. Users provide input to computers by typing in text, navigating through graphical menus, and clicking menu-items, buttons, and icons. The computers respond by rendering changes to the graphical elements on the interface and visual presentation of information. The advent of speech enabled personal assistants, such as Apple's Siri, Google Now, and Microsoft's Cortana, brings a hitherto unfamiliar[2] medium of interaction to many users: *speech*. This allows users to give input to computers through spoken utterances and for computers to complement their actions with spoken responses.

This comes as a result of the longstanding (and ongoing) pursuit of researchers in the speech technology and artificial intelligence communities to endow computers with the skills to understand and use natural languages, such as English or Swedish. The objective is to enable human users to interact with computers in a human-like manner, i.e., just as humans do when conversing with each other. The motivation for this is that speech communication comes naturally to humans; it is intuitive, efficient, and has robust mechanisms to deal with breakdowns in communication. Thus, enabling *spoken human–computer interaction* has the potential of offering natural and effortless interactive experiences with computers. Various scientific and industrial endeavors towards this goal have led to the emergence of *spoken dialogue systems*, which can be defined as software interfaces that enable spoken human–computer interaction.

The arrival of speech enabled personal assistants shows the significant progress made in the development of industrial spoken dialogue systems.

[2] While applications for speech enabled human–computer interaction have been developed and used in academic research for over four decades now, it is only with the arrival of speech enabled personal assistants that the masses have become familiar with speech based interaction and its use in everyday life.

However, the type of spoken interactions supported by these systems can at best be described as command-like and not very natural and human-like. In a recent study, Larsson (2015) observed that interactions with Apple's Siri, Google Now, and Microsoft's Cortana exhibit very limited capabilities with regard to handling variability in user requests. The user is constrained to answer precisely what the system has requested; otherwise, the interaction runs into problems and at times breaks down completely. The lack of robustness in handling variations in user requests and behavior is generally also an issue for advanced dialogue system prototypes developed in academic research. For example, in an analysis of human–computer interaction data from a dialogue system for room reservations, Bohus & Rudnicky (2005a) reported that while failure to recognize user utterances was the single biggest cause of problems in communication, the system's lack of knowledge required to correctly interpreting user requests was the second biggest factor. One definite cause for this general lack of robustness in handling user requests, in industrial systems and academic prototypes, is the traditional practice of handcrafting dialogue system components.

Spoken dialogue systems are inherently complex and comprise of various speech and language processing components that perform automatic speech recognition, natural language understanding, dialogue management, natural language generation, and speech synthesis. A typical approach to develop dialogue systems is to use handcrafted rules. Usually the designer or programmer encodes her knowledge of human language use and conversational behavior as rules for tasks such as understanding user utterances, mapping the user requests to system actions, and generating system responses. However, as the system is developed further to support complex tasks, maintaining hand-crafted rule based systems becomes challenging. For this reason, dialogue systems are designed to deal with tasks from a single domain (e.g., weather forecast) and equipped with limited pre-specified domain knowledge required to perform the task. Such systems offer reliable performance, but only for dialogue events that have been anticipated by the designer and encoded in the rules and the knowledge bases. However, due to the large variability of factors, such as the performances of the speech and language processing components, the nature of the task, and the user's behavior, handcrafted rules are typically not very robust (Lemon & Pietquin, 2007).

For these reasons, data-driven methods for machine learning have increasingly gained popularity in spoken dialogue systems research. Data-driven methods allow systems to automatically learn generalizable rules

(or models) from a large dataset of examples. The trained models offer robustness to errors, broader task coverage, and can be quickly trained for a new domain. The central aim of this thesis is to explore the data-driven methodology for development of spoken dialogue systems. As mentioned above, dialogue systems comprise of various components that perform different tasks. Covering all of them in the course of a thesis work would be infeasible. Therefore, we set out with the ambition to explore some breadth on topics, but at the same time gain clear insights and advance the state-of-the-art in these areas. In the scope of this work, we will investigate the application of data-driven methodology for four tasks that are of key importance to dialogue systems: (a) robust semantic interpretation of spoken utterances, (b) managing smooth exchanges of speaking turns in an interaction, (c) detecting problems in communication, and (d) exploiting the interactive setting of spoken dialogue to acquiring new knowledge.

1.2 Motivation

In recent years, data-driven approaches have been applied to development of almost every sub-component of dialogue systems, i.e., speech recognition (Huang et al., 2001), language understanding (De Mori et al., 2008), dialogue management (Frampton & Lemon, 2009), language generation (Rieser & Lemon, 2012), and speech synthesis (Zen et al., 2009). Thanks to machine learning, the adaptation of data-driven methods has led to significant progress, particularly in speech recognition and speech synthesis. However, the other higher-level components, such as language understanding and dialogue management, have not yet benefited from these approaches to the same extent[3]. This can be attributed to the fact that the design choices for these two sub-components depend on the complexity of the task and the application domain. For example, in dialogue system interfaces for information systems, such as the ones for providing information on bus schedule, simply spotting the key concepts (e.g., 'bus number', 'departure/arrival station names' and 'time of departure/arrival'), in the user utterances can be sufficient for language understanding. On the other hand, dialogue systems for problem solving, such as a system for assisting customers in assembling pieces of furniture,

[3] While the advances in speech recognition and speech synthesis due to the use of data-driven methods have been widely adopted by the industry, for language understanding and dialogue management, the adoption of the methodology is still limited to academic research.

would generally require methods to map the complex user utterances into some system specific plan templates, through which the system could assess the state of the world, i.e., the status of the assembled pieces.

Similarly, the task of dialogue management—deciding what the system should say following a user request—in the aforementioned systems can also vary in complexity. For example, in a system for information on bus schedule, the purpose of the dialogue is to mainly fill a form for querying the backend database. The slots in the form more or less define the scope of the dialogue, the type of utterances that users could possibly use at a particular stage in the dialogue, and the range of possible questions and clarifications a system could ask. In a system for problem solving, dialogue management is very different from the one in the information system because the state of the world (the status of the assembled furniture) cannot be simply assessed using some list of slots. Even though the task of assembling the furniture can be decomposed into many sub-tasks for putting the pieces together in stages, accomplishing this may still require an elaborate dialogue at each stage. The range of user utterances and system responses would vary according to which sub-task the user is trying to accomplish. In fact, a large part of the dialogue may consist of making the sub-tasks clear to the user and assessing if the user has accomplished them.

As a consequence, approaches to designing solutions for language understanding and dialogue management are generally based on the practical considerations of the specific task. However, the design choices are not only governed by the task complexity, there is also a question of system objective—what type of interaction experience for the user does the dialogue system want to create? Should the users perceive the dialogue system as a mere substitute for the existing graphical interface to the information system or should the dialogue system be perceived as a human-like artificial agent that can engage with the users in a manner the human counterpart would—like the personnel at the information desk at a bus station or a human customer care agent at the furniture store. These considerations have a great impact on the design requirements of the various sub-components of dialogue systems.

If the objective is to build systems that interact with users in a human-like manner, then the design choices have to be based upon some considerations of human-like dialogue. For example, humans converse using natural language and not some command-like instructions. This suggests that dialogue systems should enable the users to speak in a natural way and be able to understand them. In conversation, human

participants take turns to speak without actually disrupting the flow of the dialogue and to show continued attention to the speaker by using short feedback utterances such as "*mh-hmm*", "*okay*". It is desirable for dialogue systems to be able to know when it is appropriate to take the turn to speak and how to show continued attention to the user. Human conversational partners actively engage in detecting and resolving any confusion or breakdowns in communication (i.e., errors) that may occur during the conversation. This enables them to successfully exchange information and knowledge. In order to complete its task(s), the dialogue system must know if there has been any misunderstanding in the dialogue and take appropriate actions to resolve them. In addition to resolving this, human participants generally take the conversation as an opportunity to learn pieces of information that is unknown to them. Spoken conversation is therefore a means for humans to gain new knowledge. Following this we may also want to build dialogue systems that are able to detect whether they lack some knowledge and use the resourcefulness of the human users to acquire it.

There are many other aspects of human conversations that could be desirable for modelling in dialogue systems, listing all of them is not the intention here. But the motivation is to enable more human-like interaction experiences for users of dialogue systems. In the scope of this thesis work, we have set the ambitions to deal with the four aspects of human conversation that we have just mentioned—*language understanding*, *turn-taking*, *error detection*, and *knowledge acquisition*. An elaborate discussion on the methodology to investigate aspects of human-like behavior that could be useful in dialogue systems can be found in Edlund (2011). In the rest of this section, we will elaborate on the exact problems that we have set out to deal with in this thesis. Following this we will list our contributions to these problems which will be discussed in detail in the four chapters that make up the core of this thesis.

1.2.1 Problem 1– language understanding

Enabling spoken human–computer interaction inevitably requires the dialogue system to interpret the meaning of user utterances, i.e., the intentions of the user. This task is known as spoken language understanding (SLU) and involves extracting a semantic representation of the user's utterance (as recognized by the speech recognizer) that can be used by the system to decide the next system response. For example, in a dialogue system for bus schedule, the semantic representation for user

request, such as "*I want to take a bus to Stockholm center*.", could be a set of slot-value pairs, where slots refer to the concepts in a bus information database system and for which values have been observed in the utterance. Thus, the slot-value pairs (also called a *frame*), such as (1), could be sufficient.

(1) { DESTINATION=StockholmC,
TASK=getBusNr }

Here, semantic interpretation can be seen as simple mapping of concepts in user speech to the concepts in bus information domain.

However, mappings based on such slot-value representations may not be sufficient to capture the semantics of utterances that contain structured information. Imagine a service robot that is capable of maneuvering its way in public spaces, asking human passers-by for directions to the nearest post-office. To interpret a route direction instruction, such as "*take the second left after the church*", the slot-value pairs, as in (2), is not sufficient because the interpretation of the route instruction also requires capturing the knowledge of relationships among the slots (the domain concepts).

(2) { ACTION=take,
COUNT=2,
DIRECTION=left,
ROUTER=after,
LANDMARK=church }

To explain this further, consider another route instruction, "*take the left after the second church*", which has an identical slot-value semantic representation as the previous instruction, but has a completely different meaning. In order to obtain the correct interpretation of these two route instructions, the semantic representation has to also capture the relation among the concepts. Figure 1.1 illustrates a graphical semantic representation that is able to capture the structural relations (for the former route instruction). For route instructions that comprise of many turns, the graphical representation will often be deeply structured. Thus, capturing the knowledge about relations among concepts in the semantic representation is often a critical requirement for meaningful interpretation of user utterances, e.g., in interpreting route instructions.

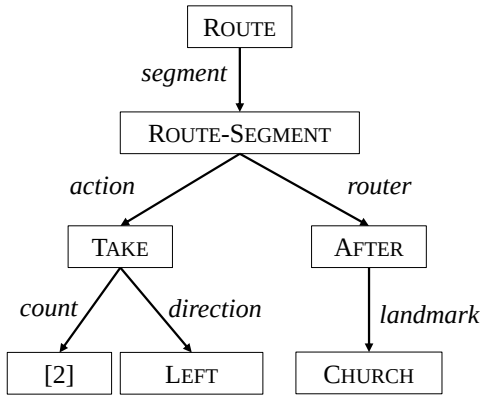


Figure 1.1: A semantic representation that preserves the structural relations among the concepts in the utterance "*take the second left after the church*."

Another major requirement of any approach for SLU is dealing with "ungrammatical" user utterances. This is due to the spontaneous nature of conversational speech—it contains phenomena such as hesitations, restarts, repetitions, and self-corrections. As a result, the user utterance, as recognized by the speech recognizer can be partly "ungrammatical" (i.e., deviates from the grammatical conventions of written sentences). Furthermore, speech recognition is known to be error prone—because of the variations in users' speaking style, gender, age, demography, and the background noise in the environment of use (in cars, on streets, in the living rooms). Errors in speech recognition could result in deletion of words in recognized user utterances, or insertion of new words not uttered by the speaker, or substitution of spoken words with similar sounding words or phrases. Due to these errors, the generated user utterance can contain recognition errors, in addition to being "ungrammatical".

The deeper structures in a semantic representation can be captured with grammar based approaches for semantic analysis, but such models are typically not very robust in handling "ungrammatical" and erroneous user utterances, as recognized by the speech recognizer. To deal with this issue, data-driven approaches for robust SLU have been presented in the literature; however, they have predominately focused on only identifying key concepts (slot-value pairs), and have not addressed the issue of extracting deeper structures in the semantic representation.

In this thesis, we will present an approach that aims to address both the requirements—of robust interpretation of spoken utterances and extraction of deeper structures in the semantic representation.

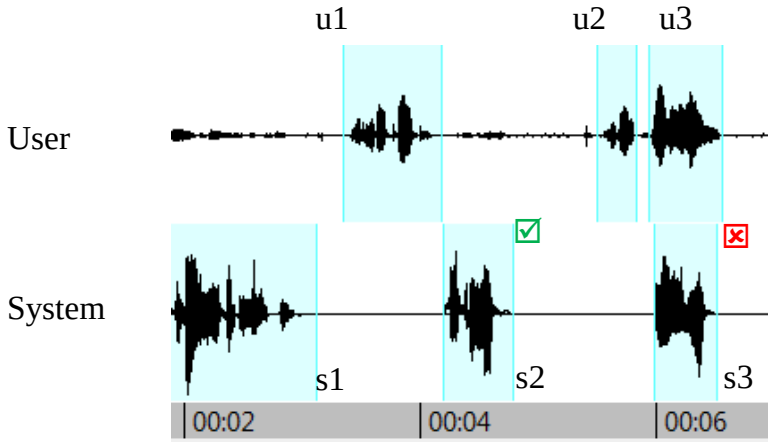


Figure 1.2: A turn-taking scenario in human–computer dialogue. Signs ☑ and ☒ indicate the appropriateness and inappropriateness of the timing of system response during the user speech, respectively.

1.2.2 Problem 2 – Turn-taking

In spoken dialogue, participants take turns to speak. Responsiveness of the listeners in the conversation, i.e., speaking at the appropriate time, is critical for managing a smooth flow of turn exchanges in the dialogue. Human conversational partners are typically able to figure out, without much trouble, when it is their turn to speak. It is also desirable in human–computer interaction that the dialogue systems know when it is appropriate to take the turn. For example, systems should be able to produce responses when expected by the user; otherwise it may give the user the impression that they have not been understood. This may cause them to either restart the dialogue from the beginning or respond (with repeat/rephrasing of the initial request or seek clarifications) or silently wait for the system to respond. Such situations in interaction lead to inefficiency in human–computer dialogue and are likely to adversely affect the user's experience. This necessitates models for efficient management of turn-taking in dialogue systems.

A typical approach in dialogue systems is to use silence thresholds for predicting end of user speech and use that as a cue for producing system responses. However, such a naïve model is sub-optimal, since users may briefly pause while they think about what to say without inviting a response. This can cause disruptions in the dialogue as the system might end up interrupting the speaker too often or speak at the same time as the systems. Figure 1.2 illustrates this issue during an exchange of turns

between a dialogue system and a user. Since the dialogue system uses a simple silence threshold based model to predict end of user speech, it can occasionally produce a turn-taking behavior that is appropriate, such as, the system segment s2 following the user segment u1 (in Figure 1.2). However, it can also produce inappropriate turn-taking behavior, as in system segment s3 following the user segment u2. We can see in Figure 1.2 that due to this misjudgment, s3 has overlapped with u3. That is, the system is talking at the same time the user is speaking. This can cause not only annoyance for the user, but also make it difficult for the system to recognize what the user said in u3. Here, the pause in user speech was not an appropriate cue for predicting end of user turn. For a dialogue system it would be desirable to be able to have smooth turn-exchanges, as seen in u1 and s2, in Figure 1.2.

How to build a more sophisticated model of turn-taking in dialogue systems is the second problem we will deal with in this thesis work. We investigate how a human-like turn-taking behavior can be modelled for dialogue system. For this, we will take inspiration from the various theoretical works that have analyzed the turn-taking mechanism in human–human conversations and observed that listeners take cues from speaker's turns for identifying turn-taking relevant places. The cues are conveyed by various modalities, such as the content of speaker utterances, the prosody, pauses, gestures and the context of what was recently said.

1.2.3 Problem 3 – Error detection

Miscommunication is a common phenomenon in human–human and human–computer interactions. While humans are very good at detecting and recovering from errors, state-of-the-art dialogue systems have problems with this. Take for example the interaction in Table 1.1. In the interaction, the user is asking the system for restaurant recommendation (this example is adapted for illustration purpose from the CamInfo Evaluation Dialogue corpus (Jurcicek et al., 2012) that comprise of actual human–computer dialogue logs). The interaction shows two common problems in communication (a) *non-understanding*: the system fails to understand what the user said (see system turn 3) and (b) *misunderstanding*: the system misunderstands the user's intentions (see system response in turn 5 to user request in turn 4). The detection of such errors and their resolution is essential for successful exchange of information and task completion. For this reason, almost all the approaches to error detection presented in the literature focuses on online detection and recovery.

Table 1.1: A sample human–computer dialogue. S and U represent system and user turns respectively. User turns are manually transcribed. System responses are based on actual speech recognition results of user turns (not shown here).

1.	S:	*How may I help you?*
2.	U:	Can you recommend a Turkish restaurant in downtown area?
3.	S:	*Could you please rephrase that?*
4.	U:	A Turkish restaurant in downtown.
5.	S:	*Clowns, which serves Italian food, is a great restaurant in downtown area.*
6.	U:	I am looking for a Turkish restaurant

However, awareness about errors in the dialogues can also be used by the dialogue designers to identify potential flaws in system design. For example, knowledge about non-understanding errors can be used to see if the system lacks knowledge to recognize and understand certain concepts. In the example interaction in Table 1.1, one could speculate whether the system can recognize the word "Turkish" at all, and if it then needs to be added to the speech recognizer's grammar. The knowledge about errors can also be used to assess the effectiveness of certain dialogue strategies that are encoded in the dialogue design. For example, in turn 3 (in the interaction in Table 1.1) the system decided to ask the user to rephrase, whereas in turn 5, it went ahead with the assumption that the user is looking for an "Italian" restaurant which was proven to be wrong, as evident from the user response in turn 6. Furthermore, the effectiveness of the strategy (of assumption) used in turn 5 would depend on the system's ability to accept any corrective suggestions that the user would present in response to it (as it happened in turn 6). If the system does not have much success in doing so then the system would likely need changes in design, e.g., better methods for correction detection and better policy for choosing among the different dialogue strategies.

Unfortunately, this kind of analysis is typically done manually, in both industrial and academic research on dialogue systems. This can be a laborious and expensive task. Automation of offline error analysis is likely to add efficiency to the dialogue systems development life cycle. In this thesis, we will present our contributions towards methods for automatic detection and analysis of errors in dialogue system interaction logs. The proposed models could be used for both online error detection in dialogue systems and offline error analysis by the designers.

1.2.4 Problem 4 – Knowledge acquisition

Typically, dialogue systems rely on pre-specified knowledge bases for performing their tasks. Knowledge bases generally have a limited coverage which creates gaps in the system's knowledge. These gaps could lead to problems in dialogue. For example, in the previous section we speculated that the dialogue system for restaurant recommendation may not know the word "Turkish". This lack of knowledge could be the cause of the two errors we saw in the human–computer dialogue in Table 1.1. If it is actually the case that the word "Turkish" is not known to the system, then it is not possible for the system to understand the user requests. In such an event, depending on the ambition of the system, a dialogue can be initiated to learn this unknown piece of information.

The ability of a dialogue system to acquire new knowledge would add to the system's skill set in learning from interactions and gain some autonomy. Thus, dialogue system designers may not have to provide the system with all the possible knowledge, but the skills to acquire knowledge. This idea of using the interactive settings of spoken dialogue for knowledge acquisition has gained increased attention in the field of human–robot interactions (Kruijff et al., 2007; Johnson-Roberson et al., 2011; Pappu, 2014). This is because robots share the space with their human interlocutors and cannot be imparted with all the knowledge in advance.

In this thesis, we investigate whether the interactive setting of spoken dialogue and the crowdsourcing scheme for gathering knowledge, can be leveraged by a dialogue system to extract information from its users and over the interactions acquire knowledge that is reliable. We develop and test our idea using a fully automated dialogue system that could extract information from users to improve the coverage of a geographical database, which can be used for generating route instructions for pedestrian navigation.

1.3 Contributions

The contributions of this thesis are the following:

1. a data-driven method for semantic interpretation of spoken route instructions (Chapter 3)
2. a data-driven method for turn-taking control in human–computer dialogue (Chapter 4)

3. three data-driven models for miscommunication detection in dialogue system interactions (Chapter 5)
4. a system and method for acquiring geographical information from users (Chapter 6)

In developing these methods, the main focus is to explore the data-driven methodology for dialogue system development. The general approach taken in this thesis is to model dialogue system tasks as a *classification* problem (using supervised learning) and investigate features from the various knowledge sources (at the disposal of the dialogue system) that can be used to train classifiers on a corpus of interaction data. An exception is the method for acquiring geographical information from users, which can be described as *implicitly-supervised learning* because the system does not require any kind of offline supervision.

During the course of this work various lexical, syntactic, semantic, prosodic, and contextual features have been investigated. A general discussion on the contributions of these features in modeling the aforementioned tasks forms one of the main contributions of this work. Yet another key focus of this thesis is on training models for online use in dialogue systems, i.e., for use in actual human–computer dialogue. Therefore, only corpora of human–computer interactions have been used as a basis, and all models have been trained with automatically extractable features (requiring no manual labelling). In addition, the model performances have been assessed on both speech recognition results and transcriptions, in order to investigate the robustness of the proposed methods.

The central hypothesis of this thesis is that models for dialogue systems tasks trained using the proposed contextual features perform better than their corresponding baseline models. The empirical validity of this claim has been assessed through both quantitative and qualitative evaluations, using both objective and subjective measures.

1.4 Outline of the thesis

The remaining of the thesis is organized as follows. In **Chapter 2**, we present an overview of spoken dialogue systems and the data-driven development methodologies. We will briefly discuss each of the major components of typical dialogue systems. We will describe the theoretical background and the design methodologies used to develop the sub-components, with a focus on data-driven development. During the course

of this chapter we will identify the problem areas where this thesis makes contributions.

The four major works reported in this thesis are covered in **Chapter 3** – Spoken Language Understanding, **Chapter 4** – Turn- taking, **Chapter 5** – Error Detection, and **Chapter 6** – Knowledge Acquisition.

Chapter 3 starts with an outline of the task of interpreting verbally given route instructions. In this chapter, we discuss our approach of using the *Chunking parser* (a syntactic analyzer) for robust semantic interpretation of verbally given route instructions. We will present two evaluations of the proposed method.

The issue of when a system should take the turn to speak is discussed in **Chapter 4**. Here again, we begin with the state-of-the-art in modeling turn-taking in dialogue systems and proceed to the details of our data-driven method. We present the methodologies for data-collection, annotation, training the models, and report both quantitative and qualitative evaluations of the models.

In **Chapter 5**, we report our method for detection of problematic system behavior in interactions. We will report three models for error detection and discuss various discourse features in order to train them. We will evaluate the utility of these models on three separate corpora. Lastly, we briefly discuss a scheme for automatically identifying dialogue situations which lead to problems in communication.

Chapter 6 presents our in-lab experiment on using a dialogue system for acquiring knowledge through interactions with multiple users. We discuss the design of the dialogue system we have developed for crowdsourcing information from users and describe two simple methods to learn reliable information from this data. An in-lab experimental study is presented to evaluate the proposed scheme.

In **Chapter 7**, we reflect on the contributions of this thesis to the field of spoken dialogue systems and speculate on how the methods presented here could generalize to other tasks in dialogue systems.

Chapter 2

Spoken Dialogue Systems

2.1 Introduction

The idea of being able to converse with artificial creatures of human-like abilities has fascinated mankind for millennia: Mary Shelly's *Frankenstein* (1823), Arthur C. Clarke's sentient computer *HAL 9000* (1968), Issac Asimov's humanoid robot in *The Bicentennial Man* (1976), and more recently, the operating system *OS1* in Spike Jonze's movie *Her* (2013). However, this fascination with artificial talking creatures is not just limited to science fiction. The idea of having effortless spontaneous conversation with computers has captivated researchers in the speech technology and artificial intelligence communities for decades. Scientific and industrial endeavors in this direction have led to the emergence of spoken language technologies, which include speech recognition and speech synthesis, language understanding and generation, dialogue systems, speech translation, spoken information retrieval, and speaker recognition.

Some of the early and well-known applications of spoken language technology are dictation systems and voice-control interfaces (e.g., in-car voice-control systems). Dictation systems primarily make use of speech recognition technology to automatically transcribe user speech in real-time. An example is the Dragon NaturallySpeaking software package that is currently developed by Nuance Communications[4]. Voice-control applications enable users to give input to computer-based systems via speech instead of having to use a keyboard. One popular application of voice-control technology is in-car infotainment systems, such as BMW's iDrive and Ford's Sync. Such systems take voice-commands as input to perform tasks for which the user would generally give input using the conventional button-and-knobs interface, e.g., for selecting the travel destination or music to play, from a list of options.

However, spoken interactions with such systems should perhaps not be termed conversations, and are not very natural. Spoken dialogue systems

[4] http://www.nuance.com/dragon/index.htm

offer an interface for humans and computers to interact using speech in a manner similar to human-human conversation. Dialogue systems can be viewed as advanced applications of spoken language technologies, including speech recognition, language processing, dialogue management, and speech synthesis. One of the most basic examples of dialogue systems are the Interactive Voice Response (IVR) system commonly used in customer care centers for handling high-volume call volumes. IVR technology allows computers to interact with humans using voice and DTMF (Dual-Tone Multiple-Frequency) tone inputs. In recent times, voice enabled assistants have been introduced in smartphones, such as Apple's Siri and Mircosoft's Cortana. These assistants provide a natural language interface to the user for performing tasks, such as adding calendar entries and simple question answering. A well-known system developed in the research community is the Let's Go system (Raux et al., 2005), which provides bus schedule information for Pittsburgh's Port Authority buses during off-peak hours.

Dialogue systems come in variety and there is no easy and clear way to categorize them. Within the research community it is common to refer to systems as either simple or complex. However, as Skantze (2007) observes, the distinction between simple and complex is not as clear. The general tendency is to make this distinction based on the difficulty of the task performed by the systems. Thus a system performing task of giving simple information, such as train schedules, would be called *simple*, whereas a dialogue system for problem solving, such as giving instructions on assembling pieces of furniture together, would generally be referred to as *complex*. With this comes the propensity to refer to industrial systems (e.g., IVR systems) as simple, whereas the prototypical systems in academic research that support spontaneous speech are termed complex. But this distinction is inaccurate as industrial systems are used by real users, and supporting real-time interactions may require technologically complex solutions, whereas academic prototypes are generally tested in in-lab settings (Pieraccini & Huerta, 2005).

A less common yet familiar scheme is to refer to dialogue systems based on how dialogue control is modeled (McTear, 2002). Thus systems can be distinguished as *finite-state-based*, *frame-based*, and *plan-based*. In more recent times dialogue modelling is done using stochastic methods (Frampton & Lemon, 2009). Following this, another distinction can be made with regard to whether the dialogue modelling is *rule-based* or *stochastic*. We will discuss more about dialogue modelling later in this chapter.

Yet another popular scheme is to distinguish systems based on the naturalness of the interactions with the users. Thus, systems that expect users to give only specific commands as inputs are referred to as *command-based* systems, while those that support spontaneous and unrestricted speech are termed *conversational system*. However, these are prototypical categories and all dialogue systems do not fit neatly into one of them (Skantze, 2007). Perhaps these two distinctions can be best viewed through the two metaphors the system can exploit—the *interface* metaphor or the *human* metaphor (Edlund, 2011). In the interface metaphor, the system is projected as a "voice interface" to some existing system and uses speech to complete tasks that would otherwise be done through keyboard and touch-tones. In the case of human metaphor, the system is projected to be a (human-like) conversational partner.

One of the challenges faced by both command-based and conversational system is that the user does not know what to expect of the system. A user may speak in a command-like manner with a conversational system, and vice versa, and this mismatch could make the interaction difficult. As Edlund (2011) argues, neither of the two metaphors is necessarily better than the other, however, it is crucial that the knowledge about which of two metaphors the system assumes is revealed to the user early on in the interaction and that the system behavior is consistent with the metaphor throughout.

If neither of the two metaphors is better than the other, then why build conversational systems, particularly when modelling human-like language skills is a really complex and challenging task? There are two arguments in favor of conversational systems. First, it is not the goal of a conversational dialogue system to model all aspects of human language use. In fact, as Allen et al. (2001) argues, full natural-language understanding by a machine is a complex task which may not be feasible. Second, and more importantly, dialogue systems generally do not require complete expertise on language use as the scope of the interaction is limited by the task domain. This constrains the user's expectations and behavior, which generally cuts down the complex task of modeling human language use into something computationally feasible. This applies to command-based systems as well. So building conversational systems is complex, but not infeasible.

The second argument in favor of conversational systems is that the user's familiarity with the human metaphor makes the system easy-to-use. This is because humans have already learned how to communicate with other humans. Moreover, human conversation is flexible, responsive and

fast, and has mechanisms for resilient error handling and grounding. These qualities of human conversation make the interactions efficient, effective, and enjoyable. One of the paramount objectives of dialogue systems is to enable an efficient and effective interaction. Thus developing systems as human metaphor, i.e., as conversational systems, holds the promise of closing-in on these objectives.

In the course of this thesis work, we have worked with both conversational and command-based systems. Our general observation is that a system offering a conversational setting for interaction can benefit from the conversational cues that are exhibited by the users due to the very nature of the conversation. We will show that these cues can be used towards training sophisticated models for decision making in dialogue systems, e.g., using prosody in user utterances for modeling turn-taking, using user repetitions and corrections as cues for error detection, and leveraging the conversational setting to engage in open-ended dialogue to acquire unknown information from the users. In the rest of this chapter, we will discuss the key components of spoken dialogue systems and data-driven development methodologies. In doing so, we will highlight the problem areas where this thesis makes novel contributions.

2.2 Dialogue system architectures

Spoken dialogue systems are complex systems and comprise of various components for tasks such as speech recognition, language understanding, dialogue management, language generation, and speech production. In a simplest configuration these sub-systems are connected in pipeline architecture, as shown in Figure 2.1. The general workflow in this system architecture is as follows: The user speech signal is first mapped into a sequence of words by the Automatic Speech Recognition (ASR) system. The sequence of words is then processed by the Spoken language understanding (SLU) module to obtain a meaningful interpretation of the user's intentions. Following this, the dialogue management (DM) module decides the communicative action the system should take. The natural language generation (NLG) module produces a sentence to expresses the system's intention in the best possible way. This sentence is then produced as an audio signal by the text-to-speech (TTS) module. A dialogue system generally relies on some knowledge base for performing its functions, such as giving the user information about bus schedule. The knowledge base can be a simple information database (e.g., database containing information of bus schedule) or a more complex knowledge

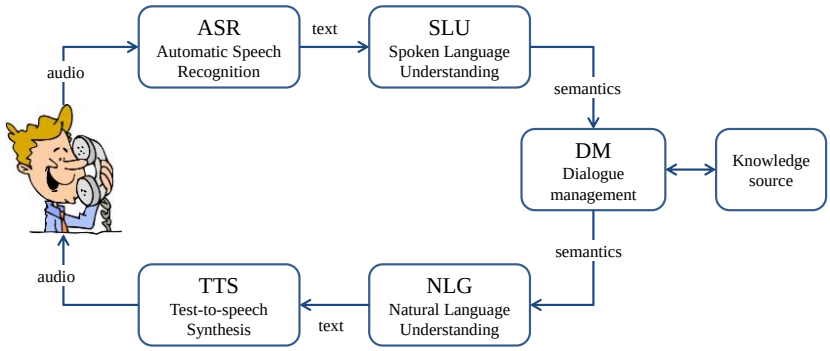


Figure 2.1: A typical workflow in a spoken dialogue system

source containing various types of domain knowledge (e.g., knowledge about assembling pieces of furniture).

A strict pipeline architecture has some limitations as there is no possibility for a component to give feedback to the components providing its input. For example, the SLU component is unable to request the ASR component to provide an alternative hypothesis when it fails to parse a given recognized hypothesis. As a result, the failure of SLU to parse the input is passed on to the DM as a parse failure error. Thus, in the pipeline architecture errors propagate across the components. An alternative is to send the n-best hypotheses. This generally requires the receiving component to have mechanisms for handling multiple hypotheses.

An advanced alternative to the pipeline architecture is the *blackboard architecture* where the components of the system share information by writing and reading to a shared space (Turunen, 2004). Figure 2.3 illustrates the architecture where the various components of in a dialogue system read and write to the shared information store.

In typical dialogue systems, information is passed from one module to another or written to the shared information store only when it is "complete." For example, the ASR system will wait until it has obtained a more or less complete hypothesis before it is passed on to the SLU. Similarly, the SLU will only write something to the shared store once the complete parse has been obtained. This scheme of information processing by a dialogue system is contrary to the fact that human process information incrementally. Support for incremental processing by humans has been found in psycholinguistic studies (Tanenhaus & Brown-Schmidt, 2008; Levelt, 1989). This puts constraints on the responsiveness of dialogue systems which in turn affect the users' experience, as the system

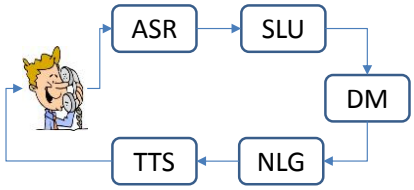


Figure 2.2: A pipeline architecture

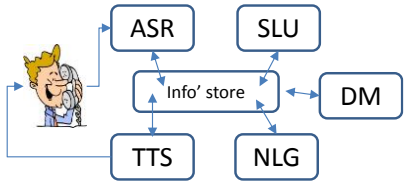


Figure 2.3: Blackboard architecture

may be perceived unnecessarily slow and inefficient. More recently, incremental processing architectures have been proposed (Schlangen & Skantze, 2009). In the incremental architecture, the ASR component recognizes the user utterances word by word and the SLU component simultaneously parses them. As the input is incrementally updated, the components revise their initial hypothesis and inform the other components. Such a framework enhances the responsive generation in dialogue systems. The incremental framework has been tested and shown to improve the responsiveness of dialogue systems (Skantze & Schlangen, 2009; Skantze & Hjalmarsson, 2010).

2.3 Automatic Speech Recognition

The task of an Automatic speech recognition (ASR) system is to map spoken utterances (a speech signal) into sequences of words. For this purpose the incoming speech signal is first segmented into regions containing speech and silence, so-called voice activity detection (VAD). A simple approach to voice activity detection is to treat regions with energy level lower than some set threshold as silence and the rest as speech. This is not a trivial task as silence is not an accurate measure of whether a speaker has finished speaking. Also, background noise in the environment of system use can further make it hard to tell apart speech from silence. More sophisticated approaches can also take spectral features into account to discriminate between speech and noise (Huang et al., 2001). Once a speech segment is obtained a corresponding sequence of words is searched. This involves many complex steps.

Development of speech recognition system comprises of two phases: training and recognition. Simply speaking, in the training phase, one or more acoustic patterns (or templates) of linguistics units such as the words (or phonemes) in the recognition vocabulary are derived. In the recognition process, the task is to match an incoming speech with the stored acoustic patterns. The stored template pattern with the lowest

distance measure from the input pattern is the recognized word. Early speech (or word) recognition systems used Dynamic Time Wraping (DTW) algorithm for identifying the best match between two given sequences.

In modern system, however, the pattern-matching is modelled as a probabilistic process to account for *temporal* and *acoustic variability* in the input speech signal. The temporal variability in the speech signal arises due to differences in speaking rate (within speaker and across speakers) and the spontaneous nature of speech. The acoustic variability, on the other hand, arises due to linguistic phenomena (e.g., co-articulation), speaker characteristics (age, gender, physical attributes), and channel factors (e.g., background noise, transmission network or microphones). Training a model that is able to account for all these variations is a complex process indeed.

In the probabilistic framework, the speech recognition process is represented with the equation (3).

$$(3)\quad W^* = \text{argmax}_W P(O|W)\, P(W)$$

Here, W* represents the word sequence that has the maximum *a posteriori* probability, and O represents the observation that is derived from the speech segment. To estimate the maximum a posteriori probability, the model relies on two probabilities:

- P(O|W) : the probability that a sequence of words W would produce the observation O. This is referred to as the *acoustic model* and the probabilities (*likelihood*) are derived from a corpus of recorded audio and corresponding transcriptions.

- P(W): the probability of a sequence of words W in a language. This is referred to as the *language model* and the probabilities (*prior*) are derived from the corpus of a language.

The observation O is extracted from the speech signal and comprises of a series of vectors representing the acoustic feature of speech. To obtain O, the speech signal is first converted into a digital format and split into short frames of about 10 ms (generally a sliding frame is used). From each frame a set of acoustic features are extracted using digital signal processing techniques. One of the goals during feature extraction is to

give importance to perceptually important speaker-independent features and discard the redundant features (Huang et al., 2001).

In ASR systems for large vocabulary continuous word recognition the *acoustic models* are of some units of sound such as phonemes or other linguistic units that make up speech. Larger sequence of phones, such as *triphones* are more useful as they model a sound in the context in which it is used, i.e., in terms of the sounds preceding it and sounds succeeding it. The process of mapping the relation between the continuous speech signal and the discrete sounds of the words is known as ***acoustic modelling***. In modern general-purpose speech recognition systems acoustic modelling is done using hidden Markov model (HMM), in which states are represented by the phones while the observations corresponds to the acoustic features.

From an application point, the most important element of the ASR systems is the ***language model***: P(W). The language model captures the context in which words are more likely to occur in a language. Here again, for small vocabulary systems the knowledge can be encoded using simple grammar rules (e.g. a context-free grammar). From these rules a finite state network model can be derived that covers all the possible word sequences in the language. For large vocabulary continuous speech recognition, N-gram model are common in use. In N-grams models, probability of a sequence of words is computed as a product of the probabilities of each word. This assumes that the occurrence of each word is determined by the preceding $N - 1$ words. However, in practice estimating the probability of a word given a large number of preceding words can be computationally expensive (and may lead to overtraining) and therefore N-grams are usually limited to bigram (N=2) or trigram (N=3).

Evaluation. A measure of goodness of a language models is *perplexity*, which can be thought of as the branching factor in the *word lattice* representation of the sentences used to train the language model. The word lattice is a directed acyclic graph with a single start node and edges labelled with word and weight. A path from the start node to the end represents a sentence. This representation enables storing of exponential number of sequences in polynomial space. Perplexity can also be seen as the average number of words that might follow a given word in the word lattice. Low perplexity implies lower branching factor, i.e., a reduced search space for matching. This in turn offers higher recognition accuracy.

The performance of an ASR system is measured in terms of *word accuracy* or more commonly as *word error rate* (WER) and is derived from the Levenshtein distance or number of edits required on the *reference* sequence of words to obtain the corresponding sequence of words output by the ASR system (Jurafsky & Martin, 2000). WER is calculated using equation (4).

(4) $WER = (S + D + I) / N$

In equation (4), S, D and I are the number of word substitutions, deletions, and insertions in the reference sequence, respectively. N is the total number of words in the reference string. *Word accuracy* is than obtained as: 1 - WER.

Speech recognition systems can be configured to return more than one hypothesis—sequence of words—for each observation. Each alternative is given a *confidence score*, which indicates how well the acoustic and language models match the specific hypothesis. Some systems for recognition can also be configured to provide *word level confidence score* i.e., a score for each word in the recognized hypothesis. Systems that are configured to provide only one hypothesis return the one with the highest confidence score. Usually there are only small differences (one or two words) between the hypotheses and therefore the alternatives can be more compactly represented as a word lattice. The selection of most likely sequence can be made by other dialogue systems component. For example, the language understanding component can use the word lattice to produce a semantic hypothesis or a lattice of semantic hypotheses (along with their confidence scores).

Speech recognition systems can also be set to provide *incremental* output, i.e., instead of returning a complete sequence of word as output the system could incremental sequences of words (possibly with revisions on the earlier sequences). This allows for other components in the dialogue systems to build incremental models for processing, e.g. incremental language understanding and incremental language generation.

Challenges. Training and deploying ASR systems for a command-base dialogue systems is a more tractable task than a system for conversational interactions. Systems for supporting conversational speech would need to rely on N-gram based language models for the extended coverage. This leads to two issues, first, a larger vocabulary requires tremendous amount of data for robustness in coverage, but a large vocabulary would also

imply a language model with high perplexity, which may affect the recognition accuracy.

In addition to vocabulary size, ASR systems have to deal with other factors such as: *speaker independence* (training a speaker independent acoustic model), *continuous speech* (since there is no physical separation in continuous time speech signal it is difficult to determine word boundaries); *spontaneous conversational speech* (the speech to dialogue systems is spontaneous and unplanned and contain disfluencies, such as hesitations, fillers, restarts and revisions). Depending on the end goal of the dialogue system, the ASR systems may require training for detection of these linguistic phenomena, which can be vital cues for modelling user state and discourse flow.

2.4 Spoken Language Understanding

Spoken language understanding (SLU) deals with the interpretation of the meaning contained in the speech signal, i.e., the user's utterance. The meaning of the utterance is generally represented using a conceptual representation that is useful for the dialogue system application. The task of SLU is similar to that of Natural language understanding (NLU) in the sense that they both aim at making computers understand the meaning of natural language sentences. However, while NLU deals specifically with the analysis of well-formed (grammatically correct) written sentences, SLU deals with the processing of *spontaneous* spoken utterances. This poses additional requirements on the task of language understanding. For example, spoken utterances contain phenomena such as hesitations, restarts, revisions and repetitions (an exception is read-out or rehearsed speeches). As a result, spoken utterances are not always grammatically correct and complete. Thus, one requirement on the approaches to SLU is dealing with the "ungrammatical" and incomplete utterances.

In comparison to written sentences, speech is a much richer medium of expression and transcends beyond linguistic expressions. It contains paralanguage (such as rate of speaking, loudness, voice quality) and prosodic variations (such as rhythm, intonation and stress), which convey details about the speaker's mental and emotional state. Information from these modalities, such as prosody, can be useful in obtaining contextually appropriate interpretation of user utterances. However, in conventional approaches to SLU, these modalities are ignored and typically the result(s) from the ASR component is used as input for processing (see Figure 2.2). Thus, SLU in its narrow sense is seen as mapping from

sequence of words, as recognized by ASR component, to some conceptual representation of meaning.

However, deletion, insertion or substitutions of words during ASR can produce an erroneous representation of spoken utterances. As discussed in the last section, such errors may arise from the background noise in the environment of usage (in-car, in public spaces), but also due to the spontaneous nature of speech that is unpredictable and hard to model. Thus, in addition to the issue of "ungrammatical" nature of spoken utterances (discussed above), SLU has to also deal with the errors introduced by the ASR system.

In the NLU community, the goal has been to obtain complete and correct syntactic and semantic analysis of the well-formed texts. To achieve this, grammars for parsing, such as Context-free grammar (CFG) and Head-Driven Phrase Structure Grammar (HPSG) are used (Charniak, 1997; Uszkoreit, 2000). The syntactic analysis decomposes a sequence of words into constituent part-of-speech and produces a *parse tree* that shows the syntactic relations among the constituents. During the syntactic analysis semantic labels (such as predicate and arguments) can also be assigned to the components of the parse tree. The semantic analysis explains how the constituent concepts combine together to form meaning, e.g., who is the actor and what action is being taken.

Since the input to SLU can be ungrammatical and erroneous, the grammar based parsing approaches are not suitable for interpretation. If one were to use these methods then the grammar rules must be relaxed in order to allow ungrammatical and incomplete utterances to be analyzed. Another issue with grammar based approaches is that they may have to be defined separately for each dialogue system application as the language use may vary across tasks. For these reasons approaches to SLU in dialogue systems have taken a pragmatic approach and the complexity of the methods depends on the nature of dialogue system tasks.

For example, De Mori et al. (2008) discuss three levels of complexities for SLU in dialogue system application. At the first level, a mapping from words to basic conceptual (semantic) constituents is achieved. This is sufficient for applications such as call routing or utterance classification (with a mapping to disjoint categories). At the second level, semantic composition of basic constituents is needed to achieve a fine-grained understanding. This is useful for tasks related to question/answering, and inquiry qualification. At the third level, a broad context can be taken into account for context-sensitive interpretation of an utterance. An utterance

can be viewed as set of sub-utterances and the interpretation of one sub-utterance is context sensitive to the other sub-utterances.

Robust parsing. Approaches to SLU inherit some properties from those of robust parsing used in NLP for processing of news-paper texts, which tend to be different from the well-formed written sentences. The central idea behind robust parsing is that of partial parsing, i.e., the syntactic analysis need not account for every element that is seen in the text, but recover chunks that can be used to extract the essential components of meaning in the text (Abney, 1997).

Such methods often lay more emphasis on the semantic analysis than on syntactic analysis. For this reason the level of details in the analysis is motivated by the dialogue system task, and each dialogue system may define its own conceptual representation of semantics. This has the disadvantage that the same model cannot be directly applied to different task domain, which means that the model has to be re-built from scratch. A survey of various approaches to spoken language understanding for dialogue systems is reported in De Mori et al. (2008).

In Chapter 2, we will present our approach to SLU for interpretation of verbally given route directions. The semantics of route descriptions involves structural relations among the concepts in the utterance. For example, correct semantic interpretation of the utterance "*take the second left after the church.*", requires not only detection of concepts such as LEFT (the direction of re-orientation), LANDMARK (at which to change orientation), but also the relations that the action of turning has to be taken AFTER the church, and it is precisely the SECOND turn and not any other. Preserving these relations is crucial and requires a semantic representation for representing structured information, e.g., a graphical structure as shown in Figure 1.1.

The deeper structures can be captured with semantic grammars (Charniak, 1997; Uszkoreit, 2000), but as argued earlier these models are typically not very robust in handling spoken utterances. Data-driven approaches for robust SLU has been proposed in the literature, however, they do not capture the deep structural relations. In Meza-Ruiz et al. (2008), a method for SLU using Markov Logic Networks is presented, but the resulting semantic representations are limited to a set of slot-value pairs (i.e., they are not structured). Another approach is presented by Wong & Mooney (2007), where a context-free grammar (CFG) augmented with semantic instructions (based on first-order logic) is learned. However, the approach assumes that the input may be captured

with a CFG, which, as we discussed above, may not be the case for speech recognition results. He & Young (2006) presents a Hidden Vector State model (an extended HMM) which may produce deeply structured semantic interpretations. It is shown to have good performance in a travel-booking domain. However, it is not clear whether it may utilize an ontology and back off to more general concepts in order to learn generalizations, which could be useful in domains where the semantic representation shares structures of the domain ontology. The approach proposed in Chapter 2, focuses on how to extract the semantic relationships between concepts in the semantic representation.

Integrated ASR and SLU. Typically, language understanding is performed using only one recognition hypothesis at a time. However, it is possible to integrate the ASR and SLU components (McTear, 2002). The key motivation here is to provide the components with the additional knowledge that is accessible to the individual component. For example, the knowledge about the constituents and compositional structures accessible to the SLU can be used by ASR is selecting the best hypothesis. In one simple scheme the same grammar for SLU is used for training the ASR language model. However, this has the disadvantage that it put constraints on the language usage. On the other hand the approach can be effective in parsing specific type of information, for instance, time related expressions (e.g., in a query to dialogue systems for train-timetable). Another alternative is to give all the ASR word hypotheses (n-best or lattice) to SLU and let it identify the best grammatical utterance. This would require dealing with the computational complexity for a large number of word hypotheses. Another alternative way of integrating ASR and SLU is to re-rank the ASR hypotheses by the SLU component (Rayner et al., 1994).

Both ASR and SLU can be benefited from contextual awareness of the dialogue. For example, Lison (2010) presents approaches to context-sensitive speech recognition and robust parsing, in the context of human–robot interaction. Here contextual information from the physical environment (e.g., the visually salient objects) and concepts from dialogue history are used to dynamically update the language model, and for performing robust incremental parsing on ASR produced word lattices.

Evaluation. There is not standard evaluation metric for comparing performance of SLU components. However, the concept error rate, a

notion similar to that of word error rate in ASR, is commonly used in SLU. Obtaining this requires a reference SLU parse to measure the edit distance with. Most SLU methods also return a confidence score in the individual parse hypotheses. This is usually derived from the ASR confidence score and the quality of resulting parse structures. For example, well-connected parse trees would give more confidence than sparsely connected concepts.

While obtaining good ASR and SLU hypotheses is crucial for robustness in dialogue, the simple fact that dialogue can be used to correct mistakes in understanding or recognition suggest that dialogue systems should also be equipped with robust mechanism for detection and recovery from such errors. In Chapter 5, we will discuss approaches to error detection in dialogue systems and propose new models for error detection and analysis.

2.5 Dialogue Management

At the heart of a dialogue system is the dialogue manager, which is responsible for controlling the flow of the dialogue. Simply speaking, the dialogue manager's task is to find an appropriate system response to a user utterance. However, this may be more complex than directly mapping concept(s) in user's speech to some system action. For example, obtaining correct interpretation of a user's intentions may require an analysis in the context of what has been said so far. Also, the dialogue manager's task is not limited to deciding *what* to say, but also identifying *when* to say something, i.e., taking the turn at appropriate places without disrupting the flow of the dialogue, by either being overly interruptive or irresponsive. Besides these tasks, a dialogue manager often needs to interact with some external knowledge source. For dialogue system interfaces to information systems, this knowledge source is the database where the information is stored. Furthermore, the dialogue manager may have to interact with some other knowledge bases, such as a domain model to process domain specific concepts and relations, and general world knowledge to resolve expressions such as "*... on next Friday*" to the correct date (Flycht-Eriksson, 2001).

On a more principled level, having successful dialogues requires the conversational partners to collaborate cooperatively, establishing what Clark (1996) refers to as the *common ground*, which is in effect the sum of their mutual, common, or join knowledge, beliefs, and suppositions. This process is known as *grounding*, and is an essentially element of

human–human dialogue. In a human–computer dialogue, the dialogue manager must also model this process in order to have a successful communication with the user. Since the dialogue may run into problems due to miscommunication, which could be caused by the errors in speech recognition or/and language understanding, the dialogue manager should also be able to detect miscommunication and take appropriate steps for recovering from problematic situations.

In the remaining of this section we will elaborate on the issues elaborated here. Towards the end, we will discuss and compare different frameworks to model dialogue control.

2.5.1 Contextual interpretation

Interactions with a spoken dialogue system, for example a train-time table system, could be thought of as simple exchanges of utterances in which the user requests information, the dialogue manager formulates the database query, and presents the information fetched from the database to the user. However, in many cases, the input from the last user utterance does not in itself carry enough information to present an appropriate answer. An easy alternative for the dialogue system at this stage is to simply inform the user about the problem and request for repeating or rephrasing the utterance. However, this often leads to very tedious dialogues, and it does not reflect the way humans deal with these issues in dialogue. In fact, humans rely to a large extent on contextual information to interpret utterances, and the dialogue manager must thus model the context accurately. To understand how the context is used to form expectations in dialogue, we will here discuss four important concepts that should be taken into account in dialogue management: *speech acts*, *discourse context*, *dialogue structure*, and *user model* (McTear, 2002).

The concept of ***speech act*** was introduced by Austin (1962) and further developed by Searle (1979). A speech act refers to the *function* of an utterance, for example, a request or an assertion. However, the function of an utterance is not always evident from its *form*—the sequence of words in it. For example, in the following two exchanges the responses of B have the same form, but the correct assessment of the underlying intentions requires the interpretation to be done in the context of A's utterances.

(5) A: What time do you want to leave from Paris?
B: Eight o'clock (*a short-answer*)

(6) A: Okay, departing from Paris at ten o'clock.
B: Eight o'clock (*a correction*)

In the exchanges (5) and (6), the syntactic and semantic analysis of B's responses should be the same and suggest that a time value has been mentioned. However, it is only in the context of the preceding speech act, i.e., intentions of speaker A, that the intentions underlying the responses of B can be inferred, which are an answer in (5), but a correction in (6). Thus, the dialogue manager should be aware of which speech acts are being used by the system and user in order to infer the function of the user input.

An utterance can have multiple communicative functions, each of which is referred to as a communicative act (CA) (a notion similar to the speech act). A scheme popular among modern dialogue system is the DAMSL—Dialogue Act Markup in Several Layers (Allen & Core, 1997), where each CA has a forward-looking and backward-looking function. The forward-looking function corresponds to the traditional speech act from Searle, such as statement, information-request, or greeting. The backward-looking function corresponds to the CA's relation with the previous dialogue act, such as agreement, correction, answer. Although, no classification schemes is exhaustive enough to be directly applicable across dialogue domains, some form of categorizations of CA's functions is required to be able to analyze patterns in discourse that emerges during a dialogue.

Another source of information at the dialogue manager's disposal for contextual interpretation of user requests is the ***discourse context***, which is required for the interpretation of items that cannot be interpreted in themselves, such as pronouns (*he, she, it*) and deictic expressions (the *previous one, the next one, that one*). The usage of these expressions refers to items that have been previously discussed in the dialogue or are pointed out by other means. Among the various proposals to resolving such discourse related issues a common idea is to maintain a history list of items mentioned in the discourse and the notion of *centering* (Grosz et al., 1995; Walker, 1989) Central to the notion of centering is the mechanism of identifying which elements in the discourse history are to be in the focus at a given point.

The user inputs may also be incomplete due to ellipsis—the input contains clauses that are syntactically incomplete. Recovering the missing part requires access to the previous context, as illustrated in example (7):

(7) A: I want to take a train to London at eight o'clock.
B: Eight? (as in "Did you say eight o'clock")

In general, resolution of discourse phenomena such as ellipsis and anaphora may require more sophisticated representations of the local context that contains the syntactic and semantic structures of previous clauses.

Dialogue structure is another knowledge source at the dialogue manager's disposal. In analyzing large records of conversations we will find patterns of conversational acts. For example, questions are followed by answers; greetings are followed by greetings; offers are responded with either acceptance or rejections. Schegloff & Sacks (1973) termed this phenomenon as *adjacency pair*. The notion of adjacency pairs could be seen as constraints on what can be said next. If dialogue partners are cooperative, one can use such an expectation model to guide the interpretation of the user utterances. Expectation-based constraints can be used for switching to appropriate grammars for speech recognition and language understanding at any given point in a dialogue (McTear, 2002). For example, a request for departure time is likely to be followed by time information. Therefore at this point in dialogue it may be suitable to give preference to the parse results obtained from a grammar explicitly designed for parsing utterances with time details. The expectation-based processing of inputs has been influential in development of the notion of *attentional state* for describing the dialogue structure (Grosz & Sidner, 1986). The attentional state refers to the focus of attention of the dialogue participants, which could be the concepts that are being talked about, and objects in the physical world in a problem-solving task domain.

Yet another knowledge source for contextual interpretation of user input is the ***user model***. The idea here is to model system behavior based on an estimate of user's beliefs, goals and plan. A user's ill-formed or incomplete utterances could be interpreted by inferring the plan underlying user's query and responding in terms of the plan (McTear, 2002). Achieving this requires the dialogue manager to jointly develop a plan with the users to achieve their goal. Another way in which a user model could be useful is through lexical entrainment—a linguistic phenomenon where conversational partners align their lexical choices (Nenkova et al., 2008). Knowledge about user's experience with the system, style of language usage, and the familiarity with the task could be used by the system to align its behavior with that of the user. This may help in reducing errors in speech recognition (Stoyanchev & Stent, 2009).

2.5.2 Grounding

Communication can essentially be seen as the process by which speakers make their knowledge and beliefs *common*. Clark (1996) argues that in order to establish a mutual understanding on what is being added to the conversation, the dialogue partners have to work jointly, in cooperation. That is, speakers cannot simply say something and expect that it has been accepted by the listeners. The dialogue partners have to send and receive positive and negative evidence of understanding. Thus dialogue participants give constant *feedback* to each other regarding the status of the common ground.

Clark & Schaefer (1989), argue that some kind of positive evidence is required to add a piece of information to the common ground. They identify five different types of positive evidence:

- *Continued attention*: the listener exhibits continued attention to the speaker
- *Relevant next contribution*: the listener produces a relevant response to the speaker's contribution, e.g., an affirmation
- *Acknowledgment*: the listener acknowledges the contribution of the main speaker by uttering backchannels such as "*mh-hmm, uh-hun, yeah, okay*" or head nods.
- *Demonstration*: the listener demonstrates understanding of speaker's contribution, for example, by completing or paraphrasing the interlocutor's utterance.
- *Display*: the listener display understanding by repeating or reusing words from what the speaker just said.

The evidences vary with regard to their strength, that is, while some evidences (such as backchannels) may only suggest that the listener is following the speaker without actually wanting to disrupt the flow other evidences (such as completing the speaker's turn) could be stronger and may suggest that the listener has actually understood the speaker's contributions.

The type (and strength) of evidence a listener should give is governed by the purpose of the task and the current state of dialogue (Clark, 1996). For example, if the hearer is uncertain about what has been just said and resolving that is critical at this stage then a rather strong evidence of uncertainty is required. On the other hand, if the hearer believes or hopes to resolve the uncertainty in near future in the dialogue, giving strong

evidence may not be necessary. Thus Clark argues that in deciding the level of evidence the following three factors play a role: the *level of uncertainty*, the *cost of misunderstanding*, and the cost of *giving strong evidence* (and its reaction).

While positive evidence is required for establishing common ground, negative evidence may be given to indicate problems in grounding something. This may arise as a consequence of something being misrecognized or not recognized or unknown. Thus, the hearer may ask the speaker to repeat as in "*Could you say that again?*" or clarify, as in "*Sorry, but did you say Paris?*" to inform the main speaker the nature of failure.

In human–computer dialogue the dialogue manager will also have to send and receive positive and negative evidence for establishing mutual ground with the user. Achieving this requires the dialogue manager to be aware of the context of the dialogue.

2.5.3 Confirmation and Verification

As suggested earlier, the dialogue manager has the possibility to inform the user about the problems in communication and request for repetition or reformulation of the request. However, this is inefficient as often requests for repetition have been followed with poor speech recognition. This is partly because the user may hyper-articulate in order to be understood. The dialogue manager needs some mechanism to effectively deal with miscommunication. There are different ways of analyzing miscommunication in human–human interactions. In one analysis, a distinction is made between misunderstanding and non-understanding Hirst et al. (1994). *Misunderstanding* occurs when the listener obtains an interpretation that is not in line with the intentions of the speaker. *Non-understanding* occurs when the listener has either not been able to obtain any interpretation, or is not confident enough to choose a certain interpretation. What distinguishes non-understandings from misunderstandings is that non-understandings are quickly noticed by the listener, while misunderstandings may not be identified until a later stage in the dialogue (Skantze, 2007a).

Another way of analyzing miscommunication is at the *action level* in the process of grounding. Both Allwood et al. (1992) and Clark (1996) proposed that speaker and listeners establish mutual understanding at *four levels of action* (also referred to as *levels of coordination* or *understanding* in the literature). This is illustrated with the conceptual example in Table 2.1.

Table 2.1: The four levels of coordination, listed from higher to lower (an adaptation from Paek (2003) and Bohus & Rudnicky (2005a)). Arrows indicate the flow of information in human–computer interaction.

Level	**Speaker**	**Listener**
Conversation	↓ S is proposing *a* (*goal*)	↑ L is considering proposal *a* (*interpretation*)
Intention	↓ S is signaling that *i* (*semantic*)	↑ L is recognizing that *i* (*parsing*)
Signal	↓ S is presenting signal *s* (*lexical*)	↑ L is identifying signal *s* (*recognition*)
Channel	↓ S executes behavior *b* (*acoustic*)	↑ L is attending to behavior *b* (*VAD*)

At the speaker end, high level communicative goal is presented through a spoken utterance. At the listener end, the speaker's goal is reconstructed from the speech. In this process, speaker and listener coordination is required at all of these four levels. Starting with the most basic level, the Channel level, the speaker S opens the communication channel by executing behavior *b* (by speaking something). For the communication to be successful, listener L must also be attending to S, i.e., the speaker must have the listener's attention for coordination at the Channel level. The dialogue partners must achieve grounding at all the levels for the listener to reconstruct the speaker's communicative goal.

The order of the levels is important; in order to succeed at a level, all the levels below it must be completed (grounded). According to Clark the *principle of upward completion* applies, i.e., it is not possible to know what the speaker wants if the listener cannot interpret the words in the context; no interpretation can be obtained without hearing the words; no words can be heard without paying attention to the speaker.

Misunderstandings and non-understandings may occur at all the four levels. For example, a non-understanding may occur because the listener did not recognize the words. A misunderstanding may occur because the listener picked the wrong interpretation of the speaker's intentions. Speakers may give evidence of understanding at each of the action levels. According to Clark (1996), when positive evidence is given at one level, all the levels below it are considered complete. Clark calls this the *principle of downward evidence*. Skantze (2007a) suggests a third distinction: *partial misunderstanding* and *partial non-understanding*

which arise from problems with grounding part of the contents of an utterance.

Negative evidence may be given by the listener to indicate that a problem has occurred and initiate a *repair* process. This initiates a new grounding process which is embedded within the main grounding process. Here again the dialogue partners need to cooperate to establish mutual-understanding about what went wrong. Hence it may be useful for the listener to convey in its presentation the action level where the problem occurred. For example, in (8) below, the different responses of B inform the speaker A about the different action levels (specified within brackets) where the error occurred.

(8) A: I want to travel from Paris to London.
  a. B: silence (Channel)
  b. B: What did you say? (Signal)
  c. B: So you want to buy a ticket? (Intention)
  d. B: From Paris to London, right? (Conversation).

If speaker B accepts the contribution of A and gives some positive evidence, this evidence may also tell speaker A that a misunderstanding has occurred, as in (9) below. Speaker A responds to this with negative evidence and initiates a repair.

(9) A: I want a train ticket from Paris to London.
  B: Okay, a train ticket from London to Paris. (Conversation)
  A: No, from Paris to London.

A request for repair to initiate recovery from miscommunication is called a *clarification request.* If the request is for recovering from lack of a hypothesis (non-understanding), the clarification can be initiated with a *request for repetition*, using a wh-question, as in (8)-b above. If the request is for recovering from lack of confidence in a hypothesis, it can be initiated with a *request for confirmation*, using a yes/no-question, as in (8)-c.

In spoken dialog systems, two well-known techniques for establishing grounding is *explicit* and *implicit verification*. In explicit verification, the system asks a clarification request about some content of the user input. It is typically in the form of a reprise sentence (Purver et al., 2001), as illustrated in exchange (10), where U and S represent the user and system turns, respectively:

(10) U: I want to take a train from Paris to London.
S: You said you want a train to Paris. Is it correct?

In implicit verification the system's clarification request consists of a combination of positive evidence to the previous user input and an additional contribution to the grounding process. The dialogue exchange in (11) illustrates this.

(11) U: I want to take a train from Paris to London.
S: At want time do you want to go to Paris?
U: No, to London, at 9 o'clock.

In (11) the systems response has two components: a request for time of departure, and positive evidence about the destination station mentioned by the user in the previous turn. In this example, the system is unsure about the destination station and made an implicit attempt to clarify it. While explicit verification necessitates the user to give a response, there is no such expectation with implicit verifications. If the user does not notice any problem with the system's hypothesis, it is considered to be correct and grounded. This has led researchers to argue that implicit verification strategies are likely to be more efficient than explicit verification, as it reduces the number of overall turns required to complete the grounding process.

However, using the implicit verification strategy requires the system to be able to handle the various user responses that could follow. In exchange (11), the positive evidence conveyed by the system ("*...you want to go to Paris?*") brought to the user's notice that there has been a misunderstanding. The user initiates repair by using an utterance that contains (i) a marked disconfirmation ("No") of what the system was implicitly trying to verify (the destination station), (ii) a corrected slot value for the destination station, and (iii) information about the time of travel. The example highlights that using implicit verification strategy requires the system to be able to detect misunderstandings, which can be harder to detect than simple yes/no responses expected by the explicit verification. McTear (2002) reports various studies on evaluations of the two strategies and observes that neither of the two strategies has a definite merit over the other, but they offer a technological trade-off. Explicit verifications may not lead to efficient grounding but are effective, as the user response is a yes/no response. Implicit verification offers the

possibility of an efficient dialogue, but demand mechanism for detection of misunderstandings. In the event of misunderstanding a repair and recovery process should be initiated, thereby affecting the efficiency of the strategy.

This brings us to the issue of error detection in dialogue systems. Among the various approaches to error detection reported in the literature, the common aspect is that they aim at online error detection (i.e., detecting errors during an interaction) although they differ in what they model as error. Some of the earliest works have been aimed at detecting whether speech recognition is problematic (Litman et al., 1999; Hirschberg et al., 2004). Another line of work has focused on detecting whether a correct understanding of user intentions has been obtained (Walker et al., 2000; Bohus & Rudnicky, 2002). At another level, failure to successfully complete the task has been modelled as error (Walker et al., 2002). Schmitt et al. (2011) use the notion of interaction quality in a dialogue as an estimate of errors at arbitrary points in a dialogue. Krahmer et al. (2001), Swerts et al. (2000), Bohus (2007) and Skantze (2007a), have modelled misunderstandings on the system's part as errors. Based on these definitions of error in dialogue different features from dialogue discourse have been used to develop methods for automatic error detection.

The various approaches to detections of errors presented in the literature can be broadly classified in two categories—*early error detection* and *late error detection*—based on at what turns in the dialogue the assessments about errors are made (Skantze, 2007b). In early error detection approaches the system makes an assessment of its current hypothesis of what the user just said. This involves detecting whether a non-understanding has occurred due to failure of ASR in providing a hypothesis (Litman et al., 1999; Hirschberg et al., 2004)) or assessing whether the hypothesis obtained from the language understanding component is reliable (Walker et al., 2000; Bohus & Rudnicky, 2002; Skantze, 2007a). In contrast, late error detection aims at finding out whether the system has made false assumptions about user's intentions in previous turns (Krahmer et al., 2001; Swerts et al., 2000; Bohus, 2007; Skantze, 2007a). Achieving this requires assessment of the system's most recent action in light of the most recent user response, which may contain a feedback. In example (9), the response of speaker A to that of B, illustrates the role of interlocutor feedback on assessing misunderstandings made by A.

Most work on error detection has focused on online detection. However, awareness about errors in dialogues has relevance not only for making online decisions, but also for off-line analysis, for example by dialogue system designers. Access to information about in which states the dialogue fails or runs into trouble could enable system designers to identify potential flaws in the dialogue design. Unfortunately, this type of error analysis is typically done manually, which is laborious and time consuming. Automation of this task has high relevance for dialogue system developers. In Chapter 5, we present data-driven models for automatic analysis of dialogue system logs for detection of problematic system behavior. The presented models can be used for both online error detection and offline error analysis.

2.5.4 Timing system responses

The dialogue manager also has to decide when to take the turn to speak. Traditionally, dialogue systems have rested on a very simple model for turn-taking: the system uses a fixed silence threshold to detect the end of the user's utterance, after which the system responds. However, this model does not capture human–human dialogue very accurately. Sometimes a speaker simply hesitates and no turn-change is intended; sometimes the turn changes after barely any silence (Sacks et al., 1974). Therefore, such a simplistic model can result in systems that frequently produce responses at inappropriate occasions, or produce delayed responses or no response at all when expected, thereby causing the system to be perceived as interruptive or unresponsive. Enabling smooth exchange of speaking turns in dialogue systems would require a more sophisticated model of turn-taking.

Two influential theories that have examined the turn-taking mechanism in human conversations are the signal-based mechanism of Duncan (1972) and the rule-based mechanism proposed by Sacks et al. (1974). According to Duncan, "the turn-taking mechanism is mediated through signals composed of clear-cut behavioral cues, considered to be perceived as discrete". Duncan identified six discrete behavioral cues that a speaker may use to signal the intent to yield the turn, involving prosody, syntax and gestures. Speakers may display these behavioral cues either singly or together, and when displayed together the likelihood that the listener attempts to take the turn increases. According to the rule-based mechanism of Sacks et al. (1974) turn-taking is regulated by applying rules (e.g. "one party at a time") at Transition-Relevance Places (TRPs)—possible completion points of basic units of turns, in order to minimize

gaps and overlaps. The basic units of turns (or turn-constructional units) include sentential, clausal, phrasal, and lexical constructions.

Related to the problem of turn-taking is that of *backchannels* (Yngve, 1970). Backchannel feedback—short utterances such as *uh-huh* or *mm-hmm*—are used by human interlocutors to signal continued attention to the speaker, without claiming the conversational floor. Thus backchannels typically only give positive evidence on the channel level (continued attention), but that they do not necessarily (yet) give evidence of understanding or agreement on a higher level. In this sense, they should perhaps not be confused with acknowledgements (a more firm "okay"), which could be regarded as stronger signs of understanding and which might be timed differently.

Similar observations have been made in various other corpora-based studies investigating the turn-taking and back-channeling phenomena in human conversations. Ward (1996) has suggested that a low pitch region is a good cue that backchannel feedback is appropriate. Koiso et al. (1998) have argued that both syntactic and prosodic features make significant contributions in identifying turn-taking and backchannel relevant places. Cathcart et al. (2003) have shown that syntax in combination with pause duration is a strong predictor for backchannel continuers. Gravano & Hirschberg (2011) identified seven turn-yielding and six backchannel-inviting cues spanning over prosodic, acoustic, and lexico-syntactic features that could be used for recognition and generation of turns and backchannels. If a dialogue system is to manage smooth exchange of speaking turns and provide backchannel feedback without being interruptive, it must be able to first identify suitable locations in the user's speech to do so. In order to achieve this, the dialogue manager should exploit these various cues in the user's speech to some extent.

However, there is a general lack of studies on how such models could be used online in dialogue systems and to what extent that would improve the interaction. There are two problems in doing so. First, the data used in the studies mentioned above are from human–human dialogue and it is not obvious to what extent the models derived from such data transfers to human–computer dialogue. Second, many of the features used in the proposed models were manually extracted. This is especially true for the transcription of utterances, but several studies also rely on manually annotated prosodic features. In Chapter 4, we will present a data-driven approach to detection of locations in user speech where it is appropriate for a system to give a feedback response. We will discuss the state-of-the art in managing turn-taking in dialogue systems and present in details our

methodology for developing a data-driven model and experimental evaluations.

2.5.5 Knowledge management

A dialogue manager must generally rely on various knowledge sources to perform its functions. As discussed in Section 2.5.1, knowledge about the dialogue and discourse context is useful for contextual interpretation of user utterances. In dialogue systems for problem solving, such as repairing internet connectivity, the dialogue manager requires access to a domain model which captures how the various goals, actions and states of the task domain are related, e.g., the Ethernet cable must be plugged-in in order for the connection to work. In such task domain, the dialogue manager would have to mediate between the user and the *domain model* in order to assess the current state of the system and ascertain that the user has taken the appropriate actions that would lead to accomplishment of the dialogue goals. In dialogue systems that use *planning systems* for problem solving it is often not straightforward to map user utterances to specific goals. The dialogue manager needs to *infer* the user's goal through interaction. During the course of problem solving, user beliefs, desires and intentions are inferred through dialogue, to negotiate and develop a plan (McTear, 2002).

For typical dialogue system applications like speech interfaces to information systems (e.g., a train-timetable system), the dialogue manager's task is primarily to map entities in user's speech to those of the database entities to formulate a database query. The knowledge about the type of information stored in the database could be useful in speech recognition and language understanding, for example, by constraining the recognition and parse grammar to the content of database. For example, in train-timetable systems the list of all possible station names can be used to enhance speech recognition of the station names. However, this may not always be straightforward and requires additional general knowledge to, say, map expressions such as "to somewhere central in Stockholm" to a specific neighborhood or list of stops in the central Stockholm area. General world knowledge is also required to map expressions such as "*early in the morning*" or "*next Friday*" to the appropriate query-fields of the database system.

Users may have false assumptions about the system which may result in an ill-formed query. For example, a user may request for a specific train connection with preference for on-board Wi-Fi access. If that specific train does not have an on-board Wi-Fi access then the query may

result in no train connection being found. In this case, the appropriate system response should be to convey that the connection exists but the train has no on-board Wi-Fi access. The dialogue manager would need to interact with the database to ascertain the situation and take appropriate actions.

While knowledge-bases are a vital source of information for the dialogue manager, only a finite amount of knowledge can be stored in them. Even if a system is connected to some online knowledge-base, it cannot be guaranteed that all the knowledge is represented there. This causes gaps in dialogue system's knowledge which may contribute to a large fraction of errors during the interaction. For example, the speech recognition component would not be able to correctly recognize a station name that is not available in the database (i.e., the vocabulary). Typically, the recognition system would choose a station name that sounds most similar to the one mentioned. The system may then decide to either verify the station name by requesting an explicit or implicit verification. This may lead to misunderstanding and initiation of repair by the user. Recovering from such errors requires the system to first detect that there are gaps in its knowledge and then use appropriate recovery strategies, e.g., perhaps ask the user for a nearby station name.

Errors in dialogue due to lack of knowledge can amount to a large fraction of non-understanding errors. In a study, Bohus & Rudnicky (2005a) observed that 62% of the non-understanding errors and 72% of the misunderstandings errors are due to speech recognition problems. The remaining 38% of non-understanding errors basically lead to rejections of user requests as they probably contained information that the system could not handle (e.g., recognize). Pappu (2014) argues that instances of such errors provide an opportunity for the system to learn the missing information from the user. While mechanisms for detection and recovery from errors arising from non-understandings and misunderstandings have been extensively studied in the literature, errors arising due to lack of knowledge have been largely ignored by current dialogue systems.

Attempts have been made to learn out-of-vocabulary words, but much of the work is done in standalone or batch setting and do not leverage the interactive setting. One notable exception is Filisko & Seneff (2005), who developed a speak-spell strategy to acquire spellings of city names through interaction. However, using dialogue as a means to acquire new information or confirm existing knowledge has only gained attention in recent years, particularly in dialogue systems for human–robot interaction (Kruijff et al., 2007; Johnson-Roberson et al., 2011; Pappu, 2014).

In Chapter 6, we present a first of its kind study on acquiring reliable information from users in an interactive setting. We present a dialogue system that uses crowdsourcing to seek and confirm knowledge. We have found in one of our experiments that by interacting with as few as 4 different users, the system can obtain information about properties of landmarks with a rather high confidence.

2.5.6 Managing dialogue control

The dialogue manager is responsible for controlling the flow of the dialogue. There are two aspects to this: the extent to which the agents in the dialogue maintain the initiative, and how the system decides on which actions to take. There are three schemes for dialogue initiative: system-initiative, user-initiative, and mixed-initiative. In the case of ***system-initiative***, the system leads the dialogue, i.e., the system has a sequence of questions that it asks the user. The user responses are limited to answering to these specific questions and possible clarification requests. This type of dialogue control can be seen in dialogue system for information systems, where the system has a sequence of slots to fill (departure station, end station, time and day of departure). The flow of the dialogue is more or less defined by the order of these questions.

In ***user-initiative*** dialogue control, the user leads the dialogue, i.e., she has a sequence of questions that she asks to the system in order to obtain some information, or a set of commands that she wants the system to perform. This type of dialogue control can for example be seen in dialogue systems for restaurant recommendations. Here the user may have a preference on the type of cuisine, location in town, and price-range. The user's goal is to obtain the address or phone number details of a suitable restaurant. The system's task here is to mostly answer the user's questions or narrow the search space as the user provides more inputs about the preferences. However, since the user has the initiative the system has no control over the flow, except for initiating clarification dialogue.

In a ***mixed-initiative*** dialogue control, both the user and the system share the initiative. So for example, the user may ask for a restaurant serving Japanese food, but the system can take control and ask questions about the price-range or food-preferences to narrow down the search space. In the follow-up turns the user may take control again to request the phone-number. Since the dialogue control is shared, the system cannot define the flow of the dialogue as a predefined state of sequences. On the contrary, modelling mixed-initiative requires the system to be able to understand all possible user requests in any dialogue state.

Several different models have been proposed for representing these strategies, including finite state machines, frame-based, plan-based, and more recently stochastic methods.

In ***finite state-based*** approaches, a state transition graph is used as a representation of the dialogue structure. The nodes in the graph may represent system utterances, whereas edges between the nodes may represent possible user utterances. Thus, the possible paths through the graph represent all the possible legal dialogues. Typically, in one state one specific question is asked to the user and the system awaits specific responses. This approach makes the interaction less flexible as only certain types of inputs from the user (typically words or short phrases) are accepted. However, Hone & Baber (1995) observed that this approach has the advantage that carefully designed system prompts can elicit intended responses from the user and hence specific grammars can be used for speech recognition and language understanding in each state. In another study, Potjer et al. (1996) observed that the likelihood of errors is greater for a dialogue system that uses a less constrained grammar for recognition. The reason for this is the larger active recognition vocabulary. In another reported study, it was observed that while a mixed-initiative system has fewer turns per state, more additional turns were required in comparison to a system-initiative model because of the greater number of recognition errors in the mixed-initiative model. Thus, finite-state based approaches are ideal for modelling well-structured tasks involving a predefined sequence of actions, with the system retaining the control.

Since the possible legal dialogues have been predefined by the transitions in the network, finite-state approaches may not permit deviations that may arise from user requests not covered by the existing transitions. Thus, provisions for error detection and recovery must be made at each node. This may lead to explosion in state space which makes it difficult to keep the dialogue structure tractable.

A ***frame-based*** approach is used when there is a set of questions to be addressed but in no specific order. An example task where such a framework can be useful is the restaurant recommendation task, where the user has the initiative and may approach the task differently from other users (e.g., ask questions in any particular order). The approach also easily lends itself to model system-initiative dialogue control. Here the system does not ask questions in a specific order, rather questions are asked based on the current state of the dialogue. For example, if the user

has already specified the departure and destination station and the day of journey, the system may ask the question about the time of departure.

The VoiceXML standard for defining dialogue flow uses a related concept: the *form*. A form in VoiceXML consists of *field* items and *control* items. The field items gather information from the user through dialogue. They may contain prompts directing the user to seek the values, the grammar to parse user input, and any event handlers. The control items contain the non-recognition based items, e.g., sequence of statements for prompting the user and performing computation. The *Form Interpretation Algorithm* determines which items in a form to visit based on the status of their guard condition, which indicates whether the item has a value. If it does, then the algorithm skips the item. Forms can be *directed*, as in system-led dialogue control, or *mixed-initiative*, which in combination with a grammar enables the user to input all the required inputs attributes in just one utterance. Other frame-based approaches have used the concept of electronic-form, schemas, task-structure graphs and type hierarchies (McTear, 2002).

In general, the advantage of frame-based approaches is that they give the user more control by permitting natural language inputs. Thus users can specify multiple slots in a single utterance, and may even attempt corrections of earlier mistakes made by the system (in grounding). On the other hand, this requires the system to use robust methods for speech recognition and language understanding. As frame-based systems provide for event-driven dialogue control in comparison to the pre-determined flow of state-based systems they offer greater flexibility over state-based systems. However, both approaches have a common limitation that they are suitable when the task is well defined, such as in retrieving information from databases. For task domains where mapping user intentions to goals are not straightforward approaches from *planning systems* have been used.

In application domain where neither the user nor the system can alone solve the problem, ***plan-based*** approaches to model the dialogue control may be used. Here, dialogue is viewed as a collaborative activity in which participants negotiate and develop a plan to solve problem. Take for example, a dialogue assistant for building objects from Lego-building blocks. The system may have access to a knowledge-base of plans on building all possible shapes, but it does not have direct control over the current state of the world, i.e., the actions of the user in putting the pieces into blocks and blocks into larger objects. The system has to interact with the user to monitor the state of the world, e.g., by asking questions about

the outcome of the recent instructions. At the same time the user may also clarify with the system if something is unclear. A user may have put the blocks wrongly and this may require revisiting the earlier plans of building a block. Achieving this type of dialogue flow through state-based or frame-based approaches is inconceivable.

Various frameworks have been explored to develop dialogue control for problem solving domains. These include ***theorem proving*** paradigm in which task completions is viewed as solving a theorem; *plan-based* approaches where user utterances are treated as actions in planning system, i.e., utterance has a goal to change the state of the world, and the dialogue manager's task is to map user intentions to part of a plan that might achieve a particular goal (Allen & Perrault, 1980). Challenges for these approaches arise from building a theorem or plan dynamically with the user. Thus the advantage of plan-based approaches over the state-based and frame-based approaches is that it allows for collaborative problem solving. However, achieving this requires the dialogue manager to maintain and reason over explicit models of task, current dialogue state, own and agent's beliefs and intentions, its dialogue and discourse obligations (McTear, 2002).

All the three—state-based, frame-based, and plan-based—approaches assume complete observability of the dialogue context. That is, the approaches make a clear decision about user's intentions and beliefs and provide limited account of the uncertainties (often limited to raising clarification requests). Also, the frameworks require the dialogue domain to be hand-crafted (e.g., by defining a finite state-automation for dialogue flow, or collections of rule for deciding the next system action, or using schemata for planning). This makes a strong assumption about the users' behavior and results in rigid dialogue control.

In order to address these two limitations of the rule based dialogue management, ***stochastic*** approaches have been developed over the last decade. These approaches use interaction data to learn optimal dialogue policy: association between each dialogue state with a system action. The data-driven approaches presented in the literature can be categorized into supervised approaches and reinforcement learning.

In ***supervised learning***, interaction data, from human–human or human–computer interaction data is fed to algorithms for learning a dialogue policy that imitate the policy of a human operator or the wizard. Wizard-of-Oz setup is a paradigm used for collecting realistic human–computer interaction data. A human wizard operates "behind the curtain" while the users are made to believe that they are interacting with a real

dialogue system (Dahlbäck et al., 1993). This setup allows for collecting interaction data in the absence of having any working prototype.

In supervised learning, the task of learning a dialogue policy is modelled as a classification problem with dialogue state as input and action as output. The classifier can be estimated using a whole range of discriminative (e.g., Decision Tree, Logistic Regression, Perceptron learner) and generative models (e.g., Naïve Bayes).

A common drawback of the supervised learning approaches is that action selection is viewed as a sequence of isolated decisions problems. This formulation disregards the temporal characteristics of the conversational dialogue, that is, actions of past have an influence in current decision state (Levin et al., 2000). One approach to deal with this is to capture the discourse history by using variables. However, the state space can grow exponentially. This has ramifications for the amount and quality of training data required to train a learner. Essentially, the amount to training data should increase with the number of additional features. However, obtaining useful interaction data is yet another challenge for development of stochastic dialogue systems. Thus a trained model might not have seen all possible user behavior. To mitigate this, development efforts are focused on using generalizable features and abstraction techniques from machine learning.

Reinforcement learning offers another alternative solution to the problem of dialogue policy optimization. In this scheme, an agent interacts with the environment (by taking actions), observes the changes in the environment as a consequence of these actions, and obtains a numeric reward, which informs the positive or negative function of the action. The learning task for the agent is to learn an optimal action for a given state, through trial and error. The optimal policy is the one that maximizes the agent's expected long-term reward. While the mapping of actions and states is learned from interaction experiences, reward functions are generally hand-crafted, e.g., a positive reward is given if the dialogue is completed, a large negative reward if the dialogue fails, and a small negative reward for additional turns (e.g., for clarification requests) introduced by the system. Since this type of learning requires an agent to explore all possible actions in a given state, learning from real users is impractical. Instead approaches have been developed to learn from interaction data from simulated user behavior (Frampton & Lemon, 2009; Rieser & Lemon, 2012). Interaction data from real user or Wizard-of-Oz collection is used to build a model of user behavior and then used to simulate a user in interaction with the dialogue system. The approach

holds promise for testing of a dialogue system application, but the down-side is that simulated data may deviate from the behavior of real users and may not contribute any new knowledge.

Reinforcement learning methods are modeled using Markov Decision Processes (MDP), which make an assumption about the observability of the world events. As a consequence, the model does not offer an account of uncertainty in dialogue state variables. A solution to this limitation has been offered by the Partially Observable Markov Decision Process (POMDP) models. In this scheme the dialogue state is treated as a hidden variable that is indirectly inferred from the observations. The POMDP model achieves this by extending the MDP model with a set of variables representing the observations in the dialogue and the mapping of observations with the dialogue states. The dialogue manager's knowledge at a given point in dialogue is represented by a belief state, which captures the probability distribution over all possible states (Williams & Young, 2007). As with policy learning in MDP models, POMDP models also use simulated users for leaning the optimal policy (Keizer et al., 2010). However, the optimization process is much more complex, as the belief state is a continuous and high-dimensional structure.

While these stochastic models offer a dialogue control framework that is grounded in sound mathematical theory, the models are complex and require significant amount of data for learning the optimal policy for dialogue control. The application domains in which the models have been developed and tested are simple slot-filling dialogues, such as restaurant recommendation system (Jurcicek et al., 2012). A middle ground between purely stochastic and rule-based approaches is proposed by Lison (2013), wherein *probabilistic rules* are used in conjunction with probabilistic frameworks for modelling dialogue management. The basic premise of this work is that some of the knowledge about dialogue policies can be hand-crafted as rules. Thus, some of the conditional relations between dialogue states can be pre-specified, whereas, the weights for these rules can be learned from data. The rules help in confining the dialogue state space, which in turn will cut down on the number of parameters and hence the amount of training data required for learning the dialogue policy.

2.6 Natural Language Generation

The task of natural language generation (NLG) can be seen as the opposite of natural language understanding, i.e., generating natural

language from some meaning representation system, such as knowledge bases or logical forms. Traditionally, natural language generation has been used for presenting information contained in knowledge bases in some human readable format. For this reason the application systems are also known as data-to-text systems. Examples of applications include generating textual weather forecasts, textual summaries of databases, explaining medical information, and generating descriptions of concepts in knowledge bases (Reiter & Dale, 1997).

As with the various approaches for semantic interpretation of written text, many techniques have been proposed for written language generation. Depending on the task it may be sufficient to just select from a list of pre-written text messages or create one by concatenating them. Alternatively, templates with slot-filling can be used to generate standard messages, e.g., to notify customers regarding transactions in their bank account. More complex systems dynamically create text to achieve the intended goal. The basic steps in the NLG task, as outlined in Reiter & Dale (1997) are: content determination, discourse planning, sentence aggregation, lexicalization, referring expression generation, and linguistic realization.

In spoken dialogue systems, a typical approach is to use the template based text generation. For example, certain prompts such as greetings, thanking, and simple request for repetition can be pre-written. However, the task domain and the interaction setting can offer various challenges for language generation in dialogue system. For example, a communicative goal to present results from an information database (e.g., for bus schedule) can be presented in many ways: as a summary, or by comparing items, or by picking one item. Furthermore, the dialogue system would have to adapt the generated output to the aspects of the dialogue context, such as user preference, user knowledge, and limitations from the synthesis component. Rieser & Lemon (2012) present a data-driven methodology in which language generation is modelled as an optimization task which factors the trade-off between length of system utterances, information conveyed and cognitive load on the user for processing the information.

However, spoken language is expressive, which offers the possibility for efficient communication. The expressiveness comes from the prosodic variations and stress marking in the utterances, and the use of fragmentary utterances, backchannels (e.g., "*mh-hmm*", "*mh-hmm?*", and "*ah!*") and acknowledgement ("*okay*", "*yeah*") for grounding in dialogue. Fragmentary utterances are effective and efficient ways to achieve

grounding (Skantze, 2007a), as illustrated by the utterance "*Eight?*" (as in "*Did you say eight?*"), and "*Eight!*" (as in "*I am surprised the beer costs eight Euros*"). However, using such utterances in a spoken dialogue puts new set of requirements on the speech production component (as we will see in the next section). A NLG component has to therefore also bear the limitations of the synthesis component in deciding the forms for some communicative action in planning.

2.7 Text-to-Speech synthesis

Speech synthesis in dialogue systems is performed by the text-to-speech synthesis (TTS) components. The TTS takes as input the text generated by the NLG component and generates the corresponding speech waveform. This involves two steps: in the first step the input text is normalized and in the second step the speech signal is generated. During the normalization step, raw text containing symbols like numbers and abbreviations are replaced with their word representation. Then each word is assigned phonetic transcriptions and the text is marked with prosodic units, like phrases and sentences. This symbolic representation is then sent for synthesis, i.e., conversion to a sound signal.

There are different schemes for synthesis, such as concatenation synthesis, formant synthesis, articulatory synthesis, and statistical parametric synthesis, which uses HMM-based (Hidden Markov Model), and more recently GMM-based (Gaussian Mixture Models) synthesis. The most popular scheme for synthesis has been the concatenation based methods such as unit-selection due to their naturalness; however, in recent times the parametric methods have also been able to match the quality of unit-selection synthesis (Zen et al., 2009).

Unit-selection synthesis concatenates units of pre-recorded speech that are stored in a database. For creating the database, first, utterances are recorded from a single reader (read-out-speech) and then segmented into units such as individual phones, diphones, syllables, morphemes, words, phrases, and sentences. This process may involve human inputs, e.g., in identifying the segment boundaries. The segments are then indexed based on segmentation and acoustic parameters such as pitch, duration, position in the syllable, and neighboring phones. During synthesis, the given utterance (sequence of words) is again automatically segmented and the best candidate of chain units from the database is selected.

The size of the speech units in the database influences the quality (naturalness) of unit-selection based synthesis. For example, larger units

when concatenated would produce more natural speech, as the large units capture the contextual naturalness in the recorded speech. However, using the larger units in the database means that a lot of recording have to be made to cover a range of utterances. This will add to the computational complexity, particularly if the synthesis is to be done on electronic devices, such as a smartphone for speech enabled personal assistants. On the other hand, using smaller units of recordings would reduce the computation complexity; however, since the concatenated utterances would now have many joints, the synthesized utterances generally lose the naturalness.

Generally, unit selection based synthesis result is the most natural sounding synthesis. However, as the naturalness is derived from the recorded units, the scheme does not lend itself well for generating speech with variations (e.g., emotions) that have not been captured in the dataset. Statistical parametric synthesis attempts to provide an alternative to unit-selection speech synthesis and address this issue. Here, instead of using the recorded units directly for synthesis, various parameters (representing vocal tract, voice source, and prosody) are extracted from them and used at runtime to generate speech waveforms for the input text. Thus, parametric approach offers the possibility to generate speech for the same speech with different emotional affects (e.g., disappointment, excitement, happiness, anger).

Speech is an expressive medium of communication. An important element of this is the emotion affect. Synthesis of speech with emotional effect is a topic on ongoing research. Similarly, conversational speech is different from read-out speech. Since there is currently no satisfactory model for synthesizing fragmentary utterances (e.g., backchannels and acknowledgment) with varying intonation, in Chapter 4 and Chapter 6, we have used customized speech. For this, fragmentary utterances, such as "*okay*", "*oaky?*" and "*mh-hmm?*", were produced by the actor with the corresponding pragmatic meaning (e.g., acknowledgement, question, doubt) and recorded. The recorded waveforms were directly used for synthesis of fragmentary system responses. This enables the system to say "*okay"* as in an acknowledgement and "*okay?"* as in expressing hesitation in agreement with what the user said.

2.8 Data-driven methodology

In this section, we present a brief overview of the basic aspects of data-driven methodology for developing models of interaction behavior in

dialogue system. The main objective here is to provide the readers with the necessary background for understanding the approaches followed in this thesis work, but the presented information can also be used by researchers in dialogue systems who may want to use data-driven methodology for system development.

2.8.1 Machine learning

Machine learning gives computers the ability to learn how to perform important tasks by generalizing from data (Domingos, 2012). Thus, computers do not have to be explicitly programmed. For example, a system for spam email filtering can be built by writing a program which looks for certain key phrases, such as 'free installation' and 'stop snoring', to classify emails as spam. The main task for the programmer is to identify an exhaustive list of such phrases. Alternatively, machine learning can be used to train a system that learns the classification task by generalizing from examples of spam and not-spam emails. Machine learning uses algorithms that can learn the models for classification given a set of features—characteristics of the data (e.g., words, phrases, sentences) and sufficient examples of the classes of interest (i.e., samples of spam and not-spam mails). During the training phase the algorithms learns to identify an optimal mapping between the features and the class label. In the test phase the input to the system is a new email and the output is the class label, indicating whether the previously unseen mail is spam or not-spam. Machine learning thus offers a cost-effective (no manual programming required) and adaptive solution (previously unseen examples of spam and not-spam mails can be added to the example set and train a better model).

There are various approaches in the area of machine learning, e.g., supervised learning, unsupervised learning, and reinforcement learning. The example of spam and not-spam classification described above follows the supervised learning approach due to the fact that the algorithm needs supervision, i.e., samples of spam and not-spam emails. The nature of the supervision needed for learning is the key difference among the machine learning approaches: unsupervised learning is used to identify unknown structures in the data, i.e., identify classes that could be of interest; in reinforcement learning the program learns to accomplish its goals by interacting with the environment with lots of trials and errors.

In this thesis we have mostly investigated the ***supervised learning*** approach. More specifically, we have modeled the dialogue system tasks as classification problems, where the outcome is a class label. For this

reason, the discussion on data-driven methodology in this section is presented with focus on supervised learning for classification tasks. One exception is the proposed framework for knowledge acquisition (Chapter 6), which could be better viewed as ***implicitly-supervised learning*** (Bohus & Rudnicky, 2007). The dialogue system learns from data that it collects over the interactions with the users. Moreover, it avoids any offline expert supervision and instead learns and revises its hypotheses over time with interaction with users.

2.8.2 Appropriateness of data

Supervised learning requires a corpus of examples, which are annotated for the target classes for learning. The relevance of the data used for training a learner is critical in order for the trained model to be effective in dealing with classification of unseen samples. In other words, the training examples should be representative of the examples in the testing phase.

One option is to use ***human–human interaction data*** as a basis for modelling dialogue system components (for example, models of turn-taking and backchanneling in Ward (1996); Cathcart et al. (2003); Gravano & Hirschberg (2011)). However, in this thesis, we have argued that this can be problematic. One problem is that humans are likely to behave differently towards a computer as compared to another human. This makes it hard to assess the contributions of a model trained on human–human dialogue data in improving human–computer interaction. Another problem is that if the system produces a slightly different behavior than what was observed in the human–human corpus, this would likely result in a different (previously unobserved) behavior in the user. Yet, another problem is that much of the dialogue behavior is optional and therefore makes the actual behavior seen in human–human interaction hard to use as a gold standard. For these reasons, we have argued that ***human–computer interaction data*** is better representative of the type of interactions a machine would need to deal with. Therefore, in the all the four major works presented in this thesis we have used only human–computer interaction data.

2.8.3 Data collection

Obtaining human–computer interaction data for training a model of interaction leads to the chicken-and-egg situation—we need a working model of interaction to collect the data in the first place. To deal with this situation, a popular approach is to use the ***Wizard-of-Oz*** interaction

settings to collect interaction data (Dahlbäck et al., 1993). In this setup, a user is made to believe that she is interacting with a computer, while the computer is operated by a human operator hidden "behind the curtain". This setup has been argued to approximate real human–computer interaction. In our approach for spoken language understanding, presented in Chapter 3, we will use a corpus of interaction data that was collected in a Wizard-of-Oz study.

The limitation of this technique for data collection is that the dialogue phenomena under investigation may require the wizard (the operator) to behave consistently throughout. Ensuring this may not be straightforward. Attempts to deal with this issues aim at regulating the wizard's behavior, e.g., through a graphical interface containing push-buttons for triggering possible system actions (e.g., responses, questions). Such a setup could be too restrictive and problematic if the issue under investigation is about responsiveness, e.g., timing of turn-taking and backchanneling in spoken dialogue.

In order to avoid any idiosyncratic behavior of the human wizard from adversely affecting the model's generalizability, an ideal setup would be to use a fully automated version of the dialogue systems using a naïve model of the interaction phenomena under investigation. The data collected through user interactions with this version of the system is then used to train a more sophisticated model of interaction in the next iteration. This scheme of system development is known as ***bootstrapping***. In Chapter 4, we will use this scheme to build incremental versions of a model for timing the feedback response of a dialogue system.

2.8.4 Data annotation

Supervised learning requires that the dataset to be labelled with the classes of interest. A general practice is to use the service of two or more **domain experts** for annotating the data (the examples). Following this, inter-annotator agreement measures such as **Kappa statistics**[5] are used to see how much the annotators (experts) agree on the annotations. Usually a sufficiently high agreement between the annotators is used to determine with the labelled data can be used a gold standard on which a model is to be estimated (Landis & Koch, 1977). While using domain experts has the potential of offering high quality (accurate) annotations, the process can

[5] Cohen's Kappa is a statistics that measures inter-annotator agreement for qualitative items. When assessing the agreement between more than two annotators, Fleiss's Kappa is used instead of Cohen's Kappa.

be expensive (both in terms of financial cost and time). As a result, in practice, often a small portion of the training dataset (say 10-20%) is labelled by multiple expert annotators and tested for agreement. Based on the annotations in this smaller set, guidelines are made (or discussed between the annotators) and only one of the expert annotator labels the entire dataset. This offers a middle way for obtaining good quality annotations at the same time saving time and cost. In our work for turn-taking, presented in Chapter 4, we will follow this approach for annotating the data.

An alternative is to recruit non-experts and provide them with guidelines for annotation, in order to ensure consistency. This will help obtain a high Kappa statistics for the non-expert annotators. However, formulating guidelines for annotation can be a non-trivial task and would still require a couple of experts to deliberate, in order to avoid any individual biases. Nevertheless, this scheme is common for annotating data in dialogue systems research.

Yet another scheme for annotation (or transcriptions) that has become popular in the field of speech technology in recent years is ***crowdsourcing*** (Parent & Eskenazi, 2011). In this scheme, information or annotation is solicited from a group of non-experts, which are recruited over the internet (rather than using traditional employees). While crowdsourcing offers a solution that is cheaper and quicker for obtaining annotations in large number, ensuring the quality remains an issue. Since the incentive for the annotators is the money they make per annotation, it is important to keep the annotation task simple (straightforward to understand) and quick to execute. This may help obtain reliable annotations to some extent. In our work on error detection, presented in Chapter 5, we will use crowdsourcing for annotating the corpus of human–computer interaction logs. In Chapter 6, we will explore how a dialogue system could be used as a vehicle for crowdsourcing, in an implicitly supervised learning framework.

2.8.5 Machine learning algorithms

There exist various machine learning algorithms to estimate classifiers from training data. A good description on them can be found in text books such as Mitchell (1997) and Duda et al. (2000). In this thesis, we have mostly used algorithms that perform linear classification and decision tree learning. The fundamental differences in the algorithms arise from how they use the characteristics of the problem (as represented by the features) to make a decision about which class it belongs to. A ***linear classifier***

makes the classification decision based on linear combination of the features. In the earlier spam filter example, the presence of many spam words (characteristics of spam mail) would linearly add up as evidence for the algorithms to classify the email as spam. Linear classifiers have been shown to be useful for classification problems where a large number of features are available, such as document classification (spam filtering can be thought of as a specific type of document classification). In this thesis, we have used the following linear classifiers: Support Vector Machines, Perceptron, and Naïve Bayes classifier.

Decision tree learning follows the notion of entropy (unpredictably) of a given dataset. If in a set of 10 emails, five belong to spam and the rest five belong to not-spam class, then the likelihood that an email randomly picked from this set is spam is only 0.5, i.e., almost chance and highly unpredictable. On the other hand, if all 10 were spam, the likelihood is 1.0, almost a certainty and predictable. Decision tree algorithm learns a classifier by recursively splitting the initial dataset into smaller subsets, until a set containing elements of just one class is obtained. For splitting the dataset, at each step a feature (characteristic of the problem) is assessed with regard to its power in reducing the entropy of produced subsets. For example, an assessment can be "Does the email contain phrase 'free installation'". The notion of information gain is used to choose among the features. During the training, the decision tree learners produce graphical trees where the leaf nodes are the subsets containing elements of just (mostly) one class, and the edges leading to the root node show the feature combination that led to the creation of the subset. The rules can be readily used for classifying previously unseen problems. There are many specific decision-tree learning algorithms, e.g., ID3 and C4.5 (Quinlan, 1993). Another algorithm that can learn rules is the RIPPER algorithm (Cohen, 1995).

In this work, the main purpose of using different algorithms is to ascertain the quality of features investigated. Features that obtain comparable performances using different learners are likely to be strong indicators of the phenomena under investigation. Furthermore, in a data-driven approach to machine learning, the best algorithms and their parameters cannot be predetermined, but are determined from the data for the task at hand. For this reason, we will train various models for the classification tasks in this thesis using different learning algorithms and compare their contributions. Efficient and well tested implementations of machine learning algorithms are nowadays available in forms of various open-source machine learning **toolkits**, namely the WEKA toolkit (Hall et

al., 2009) and the Illinois NLP Tools build on the Learning Based Java (LBJ) framework (Rizzolo & Roth, 2010).

2.8.6 Features and knowledge sources

As mentioned above, in a data-driven approach to machine learning, the best algorithms for a task are estimated from the training data at hand. The most important task for a system designer is to identify attributes that are characteristic of the classes of interest. One possibility is to start creating features based on intuitions about the problem, e.g., the word "free installation" could indicate that the email is a spam. Another approach could be to just treat every single word or all phrase pairs in the emails as an independent feature, as well as the email id of the sender, the subject of the email, and the name of the sender. By observing the performances of the individual features in classifying emails as spam and not-spam, one can identify a sub-set of features that are most appropriate for the task. This is an evidence-based approach to feature creation.

For building data-driven models of human-like interaction behavior in dialogue systems, a range of knowledge sources can be used. This includes the user utterances, the syntactic structure of the utterances, the semantic content, the prosody of spoken utterances, and the discourse context. For training models that can be used and evaluated in real interactions, it is important that the features are automatically extractable, i.e., require no manual labelling. Although model performances can be assessed on manually annotated features, e.g., patterns of the pitch contour or manually transcribed user utterances or semantic concepts, the performances are only indicative of the capabilities of the model when the ground truth is known. However, in online use, the model will need to rely on some automatic methods for feature extraction, e.g., words from speech recognition or concepts detected by the language understanding component. It is therefore important to train and test the models using automatically extractable features to assess its likely contributions in real user interactions.

2.8.7 Evaluations

Validating the actual contributions and utility of the trained model is an integral part of data-driven methodology. The conventional *validation* technique is to split the corpus of annotated examples into two sets: the training dataset and the test dataset. The examples in the *training set* are used to learn the parameters of the classifier. The examples in the *test set* are used to evaluate the performance of the trained classifier (the model).

The parameters of the model can be optimized by developing the model on the training set and repeatedly evaluating it on the test set. However, there is a problem in this approach: while the trained model will be tuned very well on this particular test set, it will not perform well on other, unobserved data. This is known as overfitting—the developed model is too specific to the characteristics of the test dataset and has poor predictive performance. For this reason, the conventional validation scheme is not very useful. A common technique to avoid overfitting is ***cross-validation***. Here the corpus is split into three sets: training set, validation set, and the test set. The training set is used to learn the parameters of the classifier. The *validation set* is used to tune the parameters of the classifier by repeatedly evaluating the trained model on it. Finally, the test set is used to assess the performance of the final model. The error rates of the model on the test set would be indicative of its performance on unseen data. After this, the model should not be tuned any further.

Different types of cross-validation schemes are used in the literature. They mainly differ in how the data is split into training and validation sets. A common scheme is ***n-fold cross-validation***, which we have used for evaluating the models presented in this thesis. In n-fold cross-validation, the corpus is randomly split into *n* equal sized subsets. In each iteration, one of the *n* subsets (a unique set) is used as the validation set and the remaining *n-1* subsets are used for training the model. A total of *n* iterations (*n-fold*) of cross-validation are performed such that each of the *n* subsets is exactly used once as the validation set. The average of the results from each round is taken as the estimate of the model performance. Thus, all the examples in the corpus are used for both training and validation, and each example is used for validation only once.

For measuring the performances of the developed models, some common metrics are accuracy and F-measure. ***Accuracy*** gives an estimate of how close is the performance of the developed model to the true values. It is measured as the fraction of the correct decisions made by the model. In corpora, where the class representation has a skew, accuracy is not a good estimate of the model performance. By simply assigning all the test examples to the majority class, the model can achieve a high accuracy. Instead, ***F-measure*** is considered a better estimate of a model's performance on corpus with skew in class representation. The F-measure combines the recall and precision of the model. *Recall* is measured as the fraction of correct decisions (in the labelled corpus) recovered by the

model. *Precision* is measured as the fraction of decisions made by the system that are correct. A traditional F-measure is the *harmonic mean* of recall and precision. Thus, a balanced weight is given to recall and precision of the model.

Qualitative evaluation of the trained models allows for assessing the contributions of the trained modeled from subjective perspective, e.g., how is the trained model perceived in real human–computer interactions. However, this is not trivial because unlike the quantitative evaluations there are no agreed metrics and no straightforward methods to measure them. For example, user satisfaction is an important metric; however, it is actually unclear how to measure this. Some of the common approach is to ask the users of a system to respond to pre- and post-use questionnaires (Jokinen & Hurtig, 2006) in order to obtain an idea of their expectations and actual experiences, or ask third party to judge the quality of the model (Edlund et al., 2013). Despite the lack of common metrics, it is worthwhile to assess the actual contributions of the proposed models by identifying suitable metrics and detailed analysis of the observations in the real human–computer interactions.

2.9 Summary

In this chapter, we provided an overview of spoken dialogue systems. We started with a discussion on command-like and conversational dialogue systems, and argued that building conversational system is feasible. Following this, we described the various dialogue system sub-components, including their tasks, the traditional rule-based approaches for developing these components, and the more recent data-driven approaches. During these discussions we established the background for the four major works in this thesis. These are spoken language understanding, turn-taking, error detection, and knowledge acquisition. In the last section, we presented a brief overview of the data-driven methodology for the perspective of dialogue system development. We will follow this methodology in this thesis to investigate and develop methods for the aforementioned four problems.

Chapter 3

Spoken Language Understanding

3.1 Introduction

Spoken language understanding (SLU) is the task of interpreting the meaning contained in a speech signal. The problem can be formulated as taking a speech recognition result and producing a semantic representation that can be used by the dialogue manager to decide the next system action. As discussed in Section 2.4, one of the demands on SLU is *robust parsing* of the input, which is the text string obtained from the speech recognizer. The input could be erroneous as automatic speech recognition (ASR) is error-prone. Furthermore, as speech is spontaneous in nature, the input can be "ungrammatical", and often contain phenomena such as hesitations, restarts, revisions, repetitions, and filled pauses. Traditionally, the approaches to SLU have focused on slot-filling tasks where a semantic representation that maps words in the input to a flat list of concepts in the task domain may suffice. However, in task domains where user utterances contain structured information, e.g., route directions, the semantic representations are complex and require relationships among the concepts to be represented. Here, preserving the structural relations among the concepts in user utterance becomes a requirement for the task of SLU.

These structural relations can be captured with grammars for semantic analysis (Charniak, 1997; Uszkoreit, 2000), but as argued earlier in Section 2.4, these models are typically not very robust in interpreting spoken utterances. Data-driven approaches for robust SLU have been proposed in the literature (He & Young, 2006; Meza-Ruiz et al., 2008), however, they do not capture the deep structural relations. In this chapter we present an approach to SLU that aims at meeting these two demands: robust parsing and preserving structural relations. Our approach is a novel application of the *Chunking parser* (Abney, 1991), for the task of spoken language understanding. The approach of parsing by chunks was proposed by Abney, as an alternative to *Chart parsing* for syntactic analysis of natural text. The task of chunking has gained popularity in

language processing as it can be modeled as a classification problem (Sang & Buchholz, 2000). Johansson et al. (2011) presented a novel application of the Chunking parser for automatic interpretation of manual transcription of verbally given route descriptions (in Swedish) collected in a Wizard-of-Oz study. In the work reported here we extend their approach in three ways. First, we evaluate the approach on a larger corpus of route descriptions in English. Second, we try to learn a deeper semantic structure. Third, we evaluate the robustness of the method on speech recognized results of the spoken route instructions.

In the rest of this chapter, we first present (in Section 3.2) the task domain—interpreting route directions—in which we develop our approach to SLU and elaborate on the deep structures we have been speaking about so far. In Section 3.3, we present the intuition behind the Chunking parser and discuss why it is suitable for the task at hand. The semantic representation for route descriptions used in this work will be discussed in Section 3.4. Following this, in Section 3.5, we present the Chunking parser for semantic interpretation of route directions. In Sections 3.6 and 3.7 we will present the quantitative and qualitative evaluations of the proposed approach, respectively.

3.2 Interpreting route directions

Robots are increasingly finding place in our daily lives. This transition from constrained and well-controlled industrial settings into dynamic environments where the objectives and situations may change radically over time makes it infeasible to equip robots with all necessary knowledge a-priori. While robots can learn from experience, understanding something hitherto unknown remains a challenging task for them. Humans, however, are a rich source of information. By engaging in a spoken dialogue with humans, robots can extract this information and gain new knowledge about the environment they find themselves in.

The scenario that we are interested in is that of an urban robot that is capable of navigating its way in public spaces. Urban robots generally rely on some geographic information system to find map details and plan their path for navigation. However, there are some limitations to relying entirely on information systems. For example, the reality (state of the world) may differ from what is stored in the information system, e.g., paths may be blocked due to construction work or an accident. In this eventuality the robot will have to find an alternative way to reach the destination. Another limitation is that it cannot be guaranteed that all the

details about the real world are represented in the databases. A common strategy that humans follow in such situations is to ask passers-by for directions. If urban robots could be endowed with the ability to engage in spoken dialogue with humans and acquire directions to destinations, it would empower the robot with increased autonomy. Achieving this requires the SLU component of the robot's dialogue system to interpret spoken route directions into a semantic formalism that is useful for robot navigation. In addition to this, the approach has to be robust to the noisy conditions in the urban environment.

In a study on human–human dialogue with an error prone speech recognition channel, Skantze (2005) observed that humans that have a very clear goal of the interaction may accurately pick out pieces of information (despite poor accuracy in speech recognition) from the speech recognition results that are relevant to the task, and ask relevant task related questions to recover from the problem. This may suggest that a data-driven approach to *keyword spotting* where specific words in the input are associated with certain concepts or slot-value pairs might server our purpose. However, the semantics of route directions is highly structured and cannot be treated as "bag-of-words/concepts". For example, a bag-of-concept (slot-value pair) representation of the route instruction "*take the second left after the Church*" is shown in (12).

(12) { ACTION=take,
COUNT=2,
DIRECTION=left,
ROUTER=after,
LANDMARK=church }

Here, we can see that the bag-of-concept as semantic representation is not sufficient as it does not preserve the relationship between the concepts. That the concept ACTION (of changing orientation) has a relation with concept DIRECTION (left); the ACTION has a specific detail to it, i.e., COUNT (second left and not any other); and the change in orientation takes place in spatiotemporal relation (ROUTER) with a specific location LANDMARK (the church). The bag-of-concept representation of a different route instruction "*take the left after the second Church*" would in fact be the same as above, and it fails to correctly represent the semantics of the instruction. For longer route instructions that involve sub-routes the relations between the concepts would need to be preserved in deeper

structure. Any approach to automatically interpret this route instruction must to some extent preserve these structural relationships.

Another requirement for understanding freely spoken route directions is that the data-driven approach should be able to generalize to some extent when encountered with unseen concepts, and be able to back off to more general concepts, without breaking the conceptual structure. In an analysis of route instructions to a miniature robot, Bugmann et al. (2001) found that on average one new word is added to the vocabulary for every additional route instruction collected. We therefore need a domain model (ontology) which defines concepts on different levels of specificity and specifies how they may be structured and the data-driven approach should take this domain model into account.

Our focus in this work is on spoken language understanding, however, interpretation of route instruction has also achieved significant interest in the field of human–robot interaction. Both data-driven and grammar based parsing approaches for semantic interpretation of route descriptions have been presented and evaluated with specific focus on *route following* through human participants and/or robots in real and/or virtual environments (Bugmann et al., 2004; Mandel et al., 2006; Kollar et al., 2010; Pappu & Rudnicky, 2012; MacMahon et al., 2006). However, most of these works have focused on interpreting manually transcribed or human written route descriptions. Understanding verbally given route descriptions has not received much attention. In (Bugmann et al., 2004) an ASR system has been used for recognizing verbal route descriptions, but the recognized text was translated to primitive routines using a translation scheme. In this regard, the data-driven approach proposed in this work for semantic interpretation of spoken route descriptions is one of the first.

3.3 Chunking parser

In order to address the requirements of robust parsing and preserving deep structural relations we take inspiration in the chunking parser, which was proposed by Abney (1991). The intuition behind parsing using chunks, as argued by Abney, is that humans process information in chunks rather than word by word. The chunk units are revealed by the prosodic patterns—of stresses interleaved by pauses—that accompany the utterance as a sentence is read. A chunk typically comprise of content words surrounded by a constellation of function words. This allows for chunks to be represented using fixed templates comprising of syntactic

categories. The relationships among chunks on the other hand are mediated by lexical selection property of the *head* chunk rather than by some rigid template.

The *chunking parser* was proposed by Abney for syntactic analysis of sentences. The parser composed of two separate stages: the *chunker* and the *attacher*. The chunker converts a stream of words into a stream of chunks, and the attacher converts the stream of chunks into a stream of sentences. The attacher assembles chunks into a complete parse tree by attaching one chunk to another by adding missing arcs between parse-tree nodes. Since chunks can be represented by templates the chunker can easily rely on a simple context free grammar for parsing them. The attacher on the other hand has to deal with word's selectional properties and therefore requires access to a word's set of subcategorization frames. These frames can be converted into rules of a context free grammar. The example in (13) (replicated from Abney (1991) p. 2) illustrates the intuition behind the chunking parser. The chunker converts the stream of eight words into a stream of three chunks. The attacher adds an arc from the IP chunk to the DP dominating *the bald man*, and another arch from the inner VP node to the PP node. In this manner the chunking parser preserves the relations among the chunks. Furthermore, in the chunking parser terminal nodes can be excluded from a chunk if their inclusion leads to disconnected sub-graphs for a chunk. This aspect can be useful for parsing speech recognized utterances with errors.

(13) Sentence: *the bald man was sitting on his suitcase*
Chunks:
$[_{DP}$ the $[_{NP}$ bald man]]
$[_{CP}[_{IP}[_{VP}$ was $[_{VP}$ sitting]]]]
$[_{PP}$ on $[_{DP}$ his $[_{NP}$ suitcase]]]

Attacher:
$[_{CP}$ $[_{IP}[_{DP}$ the bald man]]
$[_{VP}$ was sitting $[_{PP}$ on his suitcase]]]

Abney (1991) proposed a non-deterministic version of the Left-to-Right parser for both the chunker and the attacher. The approach of parsing using chunks has since gained popularity for various tasks of language processing and has been framed as a classification problem. In the CoNLL-2000 shared task on chunking various rule and data-driven methods for chunking has been proposed (Sang & Buchholz, 2000).

3.4 Representing navigational knowledge

Navigation has been vastly studied in cognitive psychology, linguistics, biology, and robotics. One of the strategies for navigation is route-based navigation, in which an agent moves from one location to another by following a particular route. A common way of representing route-based navigational knowledge is the *route graph* (Werner et al., 2000). A route graph is a directed graph that represents one of several possible ways of reaching a goal from a starting point. Figure 3.1 shows a route graph for reaching to the post-office from the starting point. Depending on the type of information needed for navigation the representation of route graph varies. For example, a *topological route graph* is a directed graph where nodes represent intersections on the route and edges straight paths which connect these intersections (as in Figure 3.1). If metric coordinates are assigned to the nodes and edges (e.g. distances and angles) a *metric route graph* is constructed, which a robot may use to follow the path. Such graphs are useful for representing a route from a survey perspective and to guide the robot's locomotion. However, they are not representative of how humans describe routes. Instead, a *conceptual route graph* (CRG) is needed that can be used to represent human route descriptions semantically.

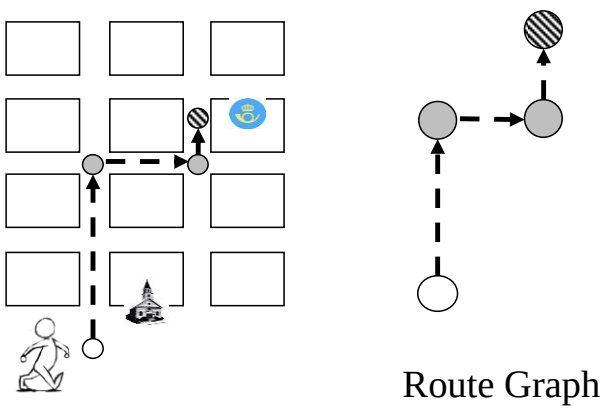


Figure 3.1: Route graph for representing navigational knowledge

Various studies have analyzed how human give route descriptions (Daniel & Denis, 1998; Tversky & Lee, 1998). It has been observed that route descriptions can be segmented into pieces mainly containing

information belonging to four categories: starting-point, reorientation, progression, and goal. In addition to these, route descriptions contain additional information such as extra landmarks, cardinal directions, and the shape of path between landmarks. According to Tversky & Lee (1998), this information is not essential but it can be important for keeping the route follower on track. Based on these studies, Müller et al. (2000) proposed a scheme for representing the semantics of route description that can be useful for robot navigation. In this scheme, conceptual route graphs are similar to topological graphs in that nodes represent places where a change in direction takes place and edges connect these places. The route graph may be divided into *route segments* where each segment consists of an edge and an ending node where an action to change direction takes place. For example, the route graph in Figure 3.1 is composed of three route segments. Conceptually, each segment consists of the following components:

- **Controllers**: A set of descriptors that ensures that the traversal along the edge is maintained. These are typically indicators of the distance to travel, and important landmarks to keep track of. For example, "*continue for 100 meters*", "*go up the street*", "*pass by the church on your right*"
- **Routers**: A set of place descriptors that helps to identify the ending node, i.e., the end of a route segment.
- **Action**: The action to take at the ending node in order to change directions.

It is not mandatory for a route segment to have all the components, but it must contain at least one of these components. Mandel et al. (2006) developed this representation further and applied to a corpus of route descriptions (indoor robot navigation) involving 27 subjects. They observed that the framework had good coverage for the various types of descriptions in the corpus.

A domain model for route descriptions

Ontologies provide the means of describing the structural relations among concepts in a certain domain. An ontology consists of a hierarchy of concepts and relations between these concepts, describing their various features and attributes. For the task of interpreting verbally given route instructions we have defined a domain model of route graphs based on previous research in the field (Krieg-Brückner et al., 2005). Various

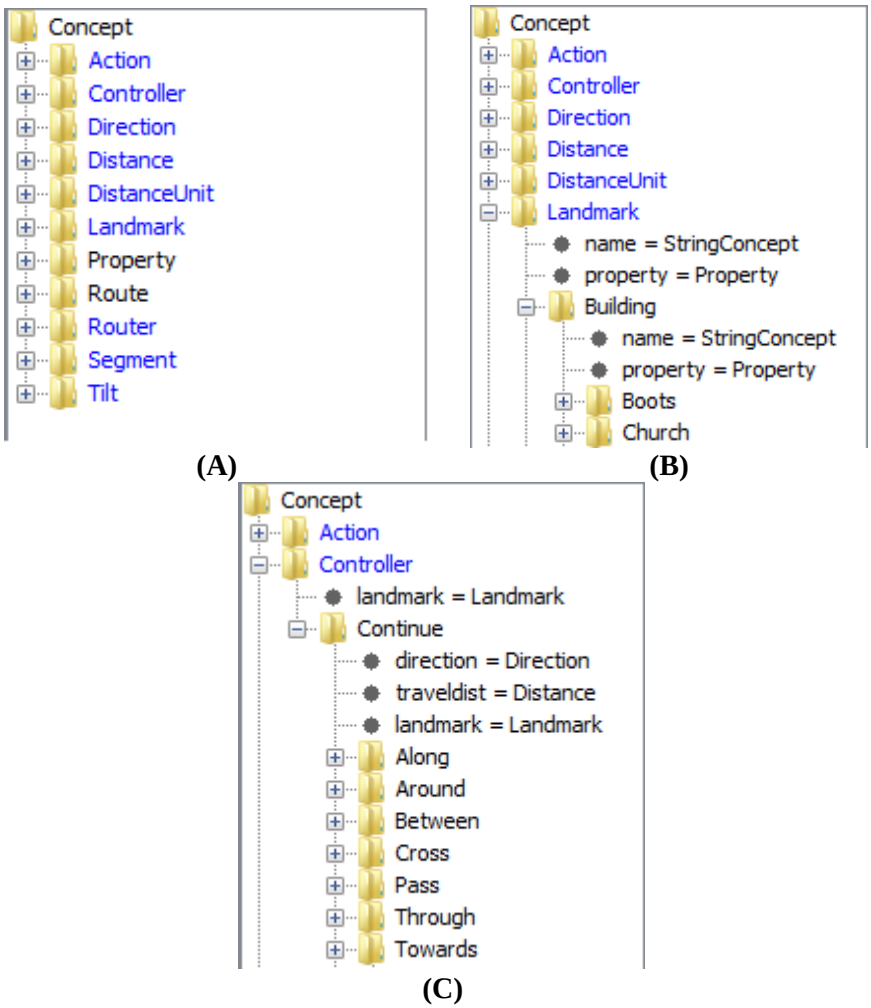


Figure 3.2: An example ontology of the concepts in route instruction task domain. (A) The generic domain concepts, (B) and (C) show some of the specific domain concepts and their inheritance relation with the basic concepts.

frameworks exist for developing ontologies (e.g., OWL—web ontology language), and it is possible to apply logical inferences to derive further knowledge about the concepts. Such a framework also allow for querying the knowledge base for relational information.

In this work, we define the ontology in the Jindigo framework (Skantze & Hjalmarsson, 2010) which is a Java-based framework for implementing

incremental dialogue systems. The framework provides methods for defining domain model for limited domains in an XML format. The domain models are then compiled into Java classes. A type hierarchy is specified, where each concept type may have a set of typed attributes. An example hierarchical ontology is illustrated in Figure 3.2. There is generic concept LANDMARK (cf. Figure 3.2-(B)), which is a subtyped by BUILDING, which is in turn subtyped by the concept CHURCH. Similarly, Concepts about traversal along the paths are subtyped from the generic concept CONTROLLER

A route graph instance can be illustrated with a conceptual tree, where nodes represent concepts and edges represent arguments. An example representation of a route comprising of three segments is shown in Figure 3.3. The graph exemplifies the deep structural relations among the various concepts in the semantic representation that as argued earlier in Section 3.2 are essential for capturing the semantics of route directions.

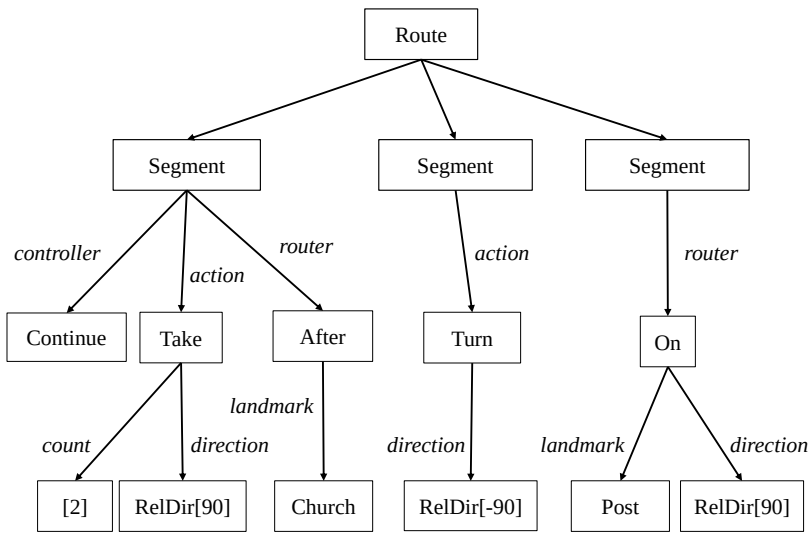


Figure 3.3: A conceptual route graph representing "*Yes sure, go straight and take the second right after the church then ehm take eh a left the post is on the eh right hand*"

3.5 A Chunking parser for SLU

For a given route instruction, we are interested in interpreting the basic domain concepts contained in it, the relations among them, and also how

concepts combine together to form a route segment. The chunking parser introduced by Abney (1991) is primarily aimed at syntactic analysis. Johansson et al. (2011) presented a novel application of the parser for semi-automatic semantic interpretation of manually transcribed route instructions. While their framework used data-driven models for learning basic concepts and relations among them, heuristic rules were used to learn the *route segments*. Here, we extend their framework for a fully automatic semantic interpretation and train the chunking parser to also learn route segments.

Abney's chunking parser is composed of two stages: the chunker and the attacher. The chunker is given the task of chunking a stream of words and the attacher attaches the chunks to form a parse tree. We propose a *Chunking parser* with a three stage process: the *Chunker*, the *Segmenter* and the *Attacher*. The Chunker is given the task of finding the base (generic) concepts in the sequence of words. The Segmenter chunks a sequence of base concepts into route segments. The Attacher is given the task of assigning the basic concepts more specific concepts (given by the type hierarchy of the domain) and to attach concepts within a segment as arguments. The possible arguments for concepts are defined as concept attributes in the domain ontology. For example, from Figure 3.2-(C), we can see that the concept CONTINUE has an attribute *landmark* (of type LANDMARK), which is inherited from the generic concept CONTROLLER, in addition to other attributes such as *direction* and *traveldist*.

Using the Chunking parser, the route instruction in Figure 3.3 could be chunked in the following way.

Route instruction: "*Yes sure go straight and take the second right after the church then ehm take eh a left the post is on the eh right hand*"

Chunker:

(14) [$_{DM}$ yes sure] [$_{CONTROLLER}$ go straight] [$_{DM}$ and] [$_{ACTION}$ take] [$_{COUNT}$ the second] [$_{DIRECTION}$ right] [$_{ROUTER}$ after] [$_{LANDMARK}$ the church] [$_{DM}$ then] [$_{FP}$ ehm] [$_{ACTION}$ take] [$_{FP}$ eh] [$_{DIRECTION}$ a left] [$_{LANDMARK}$ the post] [$_{ROUTER}$ is on] [$_{DIRECTION}$ the eh right hand]

The chunks in (14) are similar to the chunks in shallow parsing, e.g., a LANDMARK roughly corresponds to a NP (noun phrase). However, unlike shallow parsing where the chunks are syntactic categories here they

represent semantic classes. Discourse phenomena, such as fillers and discourse markers can be represented as a separate category, e.g. FP and DM. To turn chunking into a classification problem, a common practice is to define two labels for each type of chunk: one with the prefix B- for the first word in the chunk, and one with prefix I- for the following words. This labelling scheme is illustrated in the column 2 of Table 3.1.

Table 3.1: The correct output of the Chunker, the Segmenter and the Attacher for the example in Figure 3.3.

Word	***Chunker***	***Segmenter***	***Attacher***
yes	B-DM	B-SEGMENT	
sure	I-DM	I-SEGMENT	
go	B-CONTROLLER	I-SEGMENT	*class*: CONTINUE
straight	I-CONTROLLER	I-SEGMENT	
and	B-DM	I-SEGMENT	
take	B-ACTION	I-SEGMENT	*class*: TAKE *count:* → *direction:* →
the	B-COUNT	I-SEGMENT	*value*: 2
second	I-COUNT	I-SEGMENT	
right	B-DIRECTION	I-SEGMENT	*class*: RIGHT
after	B-ROUTER	I-SEGMENT	
the	B-LANDMARK	I-SEGMENT	*class*: CHURCH
church	I-LANDMARK	I-SEGMENT	
then	B-DM	B-SEGMENT	
ehm	B-FP	I-SEGMENT	
take	B-ACTION	I-SEGMENT	*class*: TAKE *direction:* →
eh	B-FP	I-SEGMENT	
a	B-DIRECTION	I-SEGMENT	*class*: LEFT
left	I-DIRECTION	I-SEGMENT	
the	B-LANDMARK	B-SEGMENT	*class*: POSTOFFICE
post	I-LANDMARK	I-SEGMENT	
is	B-ROUTER	I-SEGMENT	*class*: ON *direction:* → *landmark:* ←
on	I-ROUTER	I-SEGMENT	

the	B-DIRECTION	I-SEGMENT	*class*: RIGHT
eh	I-DIRECTION	I-SEGMENT	
right	I-DIRECTION	I-SEGMENT	
hand	I-DIRECTION	I-SEGMENT	

Segmenter:
As with the Chunker, the tasks of Segmenter can be also modeled as a classification problem. For a segment we define two labels: B-SEGMENT for the first chunk in a segment and I-SEGMENT for the following chunks. This scheme is illustrated in column 3 of Table 3.1. The Chunker output in (14) can be segmented into three route segments (domain concept SEGMENT) as shown in (15).

(15) **[**[SEGMENT] [DM yes sure] [CONTROLLER go straight] [DM and]
[ACTION take] [COUNT the second] [DIRECTION right] [ROUTER after]
[LANDMARK the church]**]** **[**[SEGMENT] [DM then] [FP ehm]
[ACTION take] [FP eh] [DIRECTION a left]**]** **[**[SEGMENT] [LANDMARK the post]
[ROUTER is on] [DIRECTION the eh right hand]**]**

Attacher:
The Attacher takes as input the Segmenter output and is responsible for performing two tasks. First, it may assign a more specific concept class (like *class*: CHURCH) to a generic chunk concepts. To allow it to generalize, it also assigns all the ancestor classes, based on the domain mode (i.e., BUILDING for CHURCH; this, however, is not shown in (16)).

(16) [SEGMENT [DM yes sure] [CONTINUE go straight] [DM and]
[TAKE(*count*:→, *direction*: →) take] [2 the second] [RIGHT right]
[AFTER(*landmark*: →) after] [CHURCH the church]] [SEGMENT [DM then]
[FP ehm] [TAKE(*direction*: →) take] [FP eh] [LEFT a left]]
[SEGMENT [POSTOFFICE the post] [ON(*landmark*: ←, *direction*: →) is on]
[RIGHT the eh right hand]]

The second task for the Attacher is to assign attributes. Some attributes are filled in with new concepts (like *property*: RED), while others are attached to the nearest concept that fits the argument type according to the domain model (like *direction*: →, which means that the interpreter should look for a matching argument in the right context). Thus, while the tasks of chunking and segmenting can be described as *single-label*

classification, the task of attachment is a *multi-label classification* where none, one or several labels may be assigned to the chunk (see column 3 in Table 3.1.).

The conceptual route graph in Figure 3.3 is the graphical representation of (16). The graph excludes details of fillers and discourse markers as they are not the domain concepts. Furthermore, route segments are simply combined in their order to produce a route node. In the next section, we discuss the algorithms and features used to train the data-driven models for the Chunker, the Segmenter and the Attacher.

3.5.1 Machine learning

For implementing the Chunking parser—the three classifiers—we used the Learning Based Java (LBJ) framework (Rizzolo & Roth, 2010). LBJ is a special-purpose programming language based in Java and is intended for machine learning and natural language processing. The framework has shown competitive performance on the CoNLL-2000 shared task (Sang & Buchholz, 2000). To train the classifiers we have tested two basic types of machine learning algorithms: Naïve Bayes (NB) and Linear Threshold Units (LTUs). Only the latter can be used for multi-label learning in the Attacher. LTU's represent their input with a weight vector whose dimensions correspond to features of the input, and outputs a real number score for each possible label as output. For single-label learning, the label with the highest score is selected. For multi-label learning, all labels with a score above a certain threshold (which is learned) are selected. Two types of LTU's were tested: Sparse Perceptron (SP), Sparse Average Perceptron (SAP) (Collins, 2002). The complete process of interpretation is automatic. However, to handle numbers (as in "*walk two hundred meters*") a simple rule-driven parser is applied in the attachment stage, which would otherwise require a large amount of data.

3.5.2 Features

We wanted to build an interpreter that is able to generalize to some extent when encountered with unseen data, and be able to back off to more general concepts, without breaking the conceptual structure. Towards this, we use not only the specific words as features, but also their Part-of-speech details. For the Chunker the following features were used:

- ***Word***: The *word instance*, as well as a *word window* comprising of the word itself, one previous and two next words.
- ***Previous tag***: The two previous chunking tags.

- ***Part-of-speech* (POS)**: The Part-of-speech tags of the words in the word window. For obtaining POS tags we used the Illinois Part-of-speech Tagger (Roth & Zelenko, 1998), as an off the shelf tagger.

For the Segmenter and the Attacher we used the following features:

- ***Bag-of-words* (BOW)**: An ordered set of words in the chunk (both the Segmenter and the Attacher process the chunks output by the Chunker)
- ***Bag-of-POS***: an ordered set of POS tags of words in the chunk
- ***Chunk label***: The label produced by the Chunker, as well as a window of two previous labels and one next label (***chunk label window***).

3.6 Quantitative Evaluation

A common measure for evaluating the performance of automatic speech recognition is the word error rate (WER). A reference string of words is compared to the string of recognized words using minimum edit distance, and the number of insertions, deletions and substitutions are counted (Jurafsky & Martin, 2000). Similarly, concept error rate (CER) can be used to evaluate a semantic interpreter. However, this is not as straightforward. First, the concepts are tree structured, which means they have to be flattened in a consistent manner. For this, a depth-first traversal of the tree is applied, where the order of the concept is preserved if the order is important (e.g., the SEGMENT children of a ROUTE), and otherwise alphabetically ordered. Second, not all concept substitutions should be treated equal. For example, substituting CHURCH with BUILDING should not be penalized as much as substituting STREET with BUILDING. To handle this, the domain model is used in the evaluation, where an insertion or deletion gives the error score of 1, a wrong base concept gives 1.5, a too specific concept gives 1, a too generic concept gives 0.5, and a wrong branch in the concept hierarchy gives 1. The total error score is then divided with the number of concepts in the reference and multiplied by 100 to obtain the CER. Although the error scores are quite arbitrary, they might give a better indication of the performance than one crude overall error score.

We will use the n-fold cross-validation scheme (discussed in Section 2.8.7) for evaluating model performances. A 5-fold cross-validation

Figure 3.4: A subject instructing the robot during the IBL corpus collection. Inset: remote-brained miniature robot. Figure copied from Bugmann et al. (2004) with permission from the first author.

scheme is followed. Thus, the reported CERs are averaged over the performances obtained over 5 cycles of training and testing. In each cycle, the training set has 25 examples of route instructions and the test set has 5 examples.

3.6.1 Data

The main focus of our work is extracting conceptual rout graphs from speech recognition results of spoken route instructions. For this we used the Instruction-Based Learning (IBL) corpus (Bugmann et al., 2001), which was collected to investigate the real-world instruction based-learning in a generic route instruction task. In an experimental setup (shown in Figure 3.4), 24 human subjects were recorded as they gave route instructions to the miniature robot in the model-town. The town environment comprised of various landmarks from real urban setup (e.g., supermarkets, roundabouts, crossings) and provides a close approximation of the urban settings. The subjects were told that the robot was remote-controlled, and that, at a later date, a human operator would use their instruction to drive the robot to its destination. The operator was located in another room and used only the images from the wireless on-board the video camera to navigate the robot (at a later date). Detailed analysis of the instruction types (Bugmann et al., 2001) and performance

of the miniature-robot in route following using the manual mapping of route instructions into primitive procedures has been reported in the literature (Bugmann et al., 2004; Kyriacou et al., 2005). The corpus has been widely referred to in the spatial language processing community (Mandel et al., 2006; Kollar et al., 2010; Pappu & Rudnicky, 2012; MacMahon et al., 2006).

The corpus contains manual transcriptions and audio recordings of 144 spoken route instructions given in English. The average length of instructions in IBL is 56 words. The corpora contain transcriptions of phenomena such as filled pauses, self-corrections and repetitions. However, no annotation indicating these categories has been made. The audio recordings were made in a normal indoor setup, and contain speaker pauses and silences during route instruction giving. This makes the corpus suitable for our task.

3.6.2 Procedure

As a first step, the Chunking parser's performance was evaluated on the manual transcriptions available in the IBL corpus. This gave us the baseline for comparing the parser's relative performance on the speech recognition results. The IBL experimental setup had three conditions. Route instructions in two of these conditions were monologue and dialogue in the third. For our work we have used 30 route instructions from the monologues as the experimental dataset. Two criteria for selecting the samples were used. First, natural instruction styles, e.g., "go *straight and turn left at the junction*," were preferred over the procedural-style, e.g., "*move 10 cm in front and then turn 90 degrees*". Since there were only few samples of the latter style, including them would make the task of machine learning difficult. Second, we excluded instructions that referred to a route segment from previously given route instructions, e.g. "*go to the post-office as shown earlier and then continue to...*".

The 30 route instructions were manually annotated for the conceptual route graphs. Since we first wanted to do a proof-of-concept study, all the annotations were made by this author under the supervision of an expert annotator. For the same reason, no inter-annotator agreement scores were obtained. The final dataset, on an average, contained 31.1 words, 13.13 concepts and 2 route segments per route instruction and was used as the cross-validation set.

In the next step, we trained the language model of an off-the-shelf ASR system with 113 route instructions from the IBL corpora (excluding the 30 used in the cross-validation set). The trained ASR had a vocabulary

size of 213 words. The audio recordings of the route instructions in the cross-validation set were then recognized by the trained ASR. The speech recognition hypothesis for each instruction was chosen for validating the Chunking parser's performance corresponding to the ASR's word error rate.

3.6.3 Results

The results presented here are obtained using the cross-validation scheme with 25 instructions in the training set and 5 in the testing set.

Baseline: For drawing comparisons we used the baseline performances of the Chunking parser corresponding to a *keyword spotting* based method, i.e., a model that uses the feature word instance alone for classification. Table 3.2 shows the performances for all the three stages of the Chunking parser using this feature. The baseline CER for the Chunker is 50.83 and obtained by the Naïve Bayes (NB) algorithm. The input to the Segmenter is the sequence of chunks identified by the Chunker. Using the bag-of-words representation of words in a chunk the Segmenter obtains a baseline CER of 77.22 using the NB. The Attacher takes the Segment chunk as input and processes the chunks contained in it. Using the bag-of-words representation of words in a chunk the Sparse Averaged Perceptron (SAP) algorithm achieves a baseline CER of 77.70. In general, the Linear Threshold Units algorithms performed better than Naive Bayes for the chunking task using only the word instance and BOW features.

Table 3.2: Baseline performances of the Chunking parser.

Component	Features	CER_{NB}	CER_{SP}	CER_{SAP}
Chunker	**Word instance**	**50.83**	46.15	45.17
Segmenter	**Bag-of-words**	**77.22**	60.83	53.06
Attacher	**Bag-of-words**	--	77.62	**77.70**

Chunker performances: We use the additional features discussed earlier in Section 3.5.2. The performances for the Chunker with the additive feature sets for the three algorithms are shown in Table 3.3. Using the word window in addition to word instance gives a large boost to the performance of all the algorithms. This supports the intuition behind the Chunker—that humans process information in chunks, and not word by

word (Abney, 1991). When complemented with the tag information about the previous two tags the Chunker achieves the best performance, a CER of 10.64, using the Sparse Averaged Perceptron learner. This is a large gain over the baseline mode, a reduction in CER of about 40 points.

Table 3.3: Chunker performances with additive features

Features	CER_{NB}	CER_{SP}	CER_{SAP}
Word instance	50.83	46.15	45.17
+ Word window	17.31	21.33	20.82
+ Previous tags	18.16	10.86	**10.64**
+ POS window	19.61	11.01	11.81

Using the part-of-speech (POS window) information in addition does not lead to further improvements for the Chunker. On the contrary it results in a slightly higher CER (11.81). This could be attributed to the POS tagger's performance, but also to the fact that a POS tags can be observed in two different chunk types. For example, in the phrase "*cross the street,"* the word *cross*, has POS tag VP (verb phrase) and the chunk type is CONTROLLER. However, in the phrase "*turn right*," the word *turn* also has the POS tag VP, but is conceptually a chunk of type ACTION. Here, the POS tag feature fails to provide any discriminatory information to the learner to differentiate between the CONTROLLER and the ACTION concepts. On the contrary it may cause confusion for the learner and lead to a higher CER.

Table 3.4 provides the F-measure (the harmonic mean of the accuracy and precision) for the SAP learner using word instance, word window and the previous tag features (the best feature combination in Table 3.3). We can see that the learner is successful at predicting the beginning (the class labels with prefix B-) of most chunk types, but it still has problem with detecting B-Controller tags, i.e., detecting the beginning of a controller chunk. Investigating the cause for this has been left as a future work.

Segmenter performances: Table 3.5 illustrates the performances for the Segmenter using additive features. As can be seen, the Bag-of-words and Part-of-speech features of the words in a chunk are not very informative for the Segmenter to discriminate whether the chunk is a B-SEGMENT or I-SEGMENT. However, the label of the chunk along with that of two

Table 3.4: The F-measure for some class-labels (B- prefix) against the best performing model in Table 3.3.

Concept-Label	*F-Measure*
B-ACTION	0.97
B-CONTROLLER	0.74
B-DIRECTION	0.89
B-LANDMARK	0.92
B-COUNT	0.94
B-ROUTER	0.92
B-DM	0.94

previous chunks and one next chunk (i.e., the chunk label window) offers the best performances for all the three learners. The best among them is achieved with the Sparse Perceptron learner, which is a CER of 25.83. Also, the gain in performance for the Segmenter using this feature set over the baseline model is large: a reduction of about 51 points in CER.

Table 3.5: Segmenter performances with additive features.

Features	CER_{NB}	CER_{SP}	CER_{SAP}
Bag-of-words	83.89	64.17	59.44
+ Bag-of-POS	98.33	67.50	64.17
Chunk label window	31.67	**25.83**	28.89

Attacher performances: The Attacher performs two tasks on a chunk: assign a more specific concept type and assign it attributes. Some attributes are filled in with new concepts (like property: 2), while others are attached to the nearest concept that fits the argument type according to the domain model. The performance figures in the first two columns in Table 3.6 suggest that the Attacher achieves the best performance, a CER of 29.11, with the Sparsed Averaged Perceptron learner and using the bag-of-word representation of the words in the chunk alone. This is also a major gain over the baseline: a reduction in the CER of about 48 points.

Table 3.6: Attacher performances with additive features.

Features	CER_{SP}	CER_{SAP}	$sgCER_{SAP}$
Bag-of-words (BoW)	29.42	**29.11**	**19.99**
Chunk label window	49.32	49.74	50.93
+ BoW	29.62	29.44	21.75
BoW + Bag-of-POS	30.85	31.48	21.69

A closer look at the generated route segments indicated that while the Attacher had correctly identified the attributes and their context (left or right) for a concept the poor placement of route segment boundaries (by the Segmenter) resulted in a restricted search space for actually connecting the concepts in the route graph. This resulted in many unconnected concepts in the conceptual route graph. In order to get an estimate of the Attacher's performance independently from the Segmenter, we compared only the sub-graphs in all the route segments of a CRG with their counterparts in the reference CRG. The last column in Table 3.6 presents the Attacher's performance following this scheme. The SAP learner obtains the best CER of 19.99. In the rest of this work we will use this performance of the Attacher as the Chunking parser's best performance.

The fact that additionally using the chunk label window feature did not add much to the performance of the Attacher suggests that words (BOW) alone are sufficient to identify the attributes of a chunk. Analysis of phrases such as "*turn left*" and "*take the first right*" indicate that the ACTION concepts "turn" and "take" will need an attribute of type DIRECTION in their right context. Fundamentally, it is due to the selectional property of verbs (turn, take) a well-established fact in the linguistic and language processing community.

3.6.4 Impact of data on learning

A general observation in the machine learning community is that more training data typically improves the model performance. We investigated how the models performances for the Chunker, the Segmenter and the Attacher scale with availability of more training data. The dataset was divided in 6 sets, each containing 5 route instructions. One of this set was randomly chosen as the test set and the remaining 5 sets were incrementally used to train the models. In each round a new set was added to the existing training set. The models were trained using the best

performing feature combination and algorithms presented in the previous section. The learning curves in Figure 3.5 show that while the Chunker, the Segmenter and the Attacher were able to perform well with little training (just 15 samples) it seems like the performance could improve further with more training data.

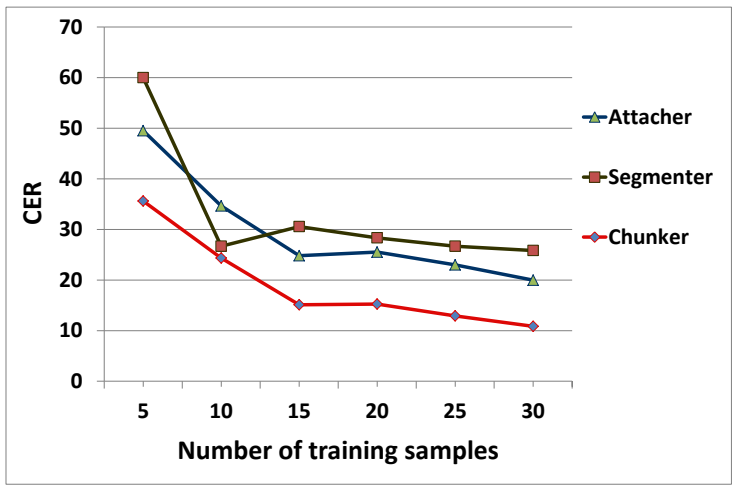


Figure 3.5: The learning curves for the Chunking parser

3.6.5 Performance on speech recognition

The evaluations above were all made on transcriptions of the spoken instructions. Next, we wanted to see the performance of the models on the output from an ASR. We also wanted to obtain an estimate of the increase in the CER of the Chunking parser with increased errors in the ASR. To simulate errors in speech recognition seven different configurations of the ASR systems were used. The configuration varied with regard to the beam search and weight on grammar parameters. The performances of the Chunking parser against these seven performances of the ASR are shown in in Figure 3.6. The curve *Baseline is the Chunking parser's performances on the transcribed route instructions: CER 19.99 when WER is 0 (i.e., perfect recognition). The general trends of the Chunker and the Attacher curves suggest that the model performances follow the WER. Against the best ASR performance, a WER of 25.02, the Attacher achieved a CER of 40.55. In comparison to the CER of 19.99 on transcriptions, the introduction of a WER of 25.02 resulted in a *relative* CER (i.e., CER minus WER) of 15.53. This is comparable with the

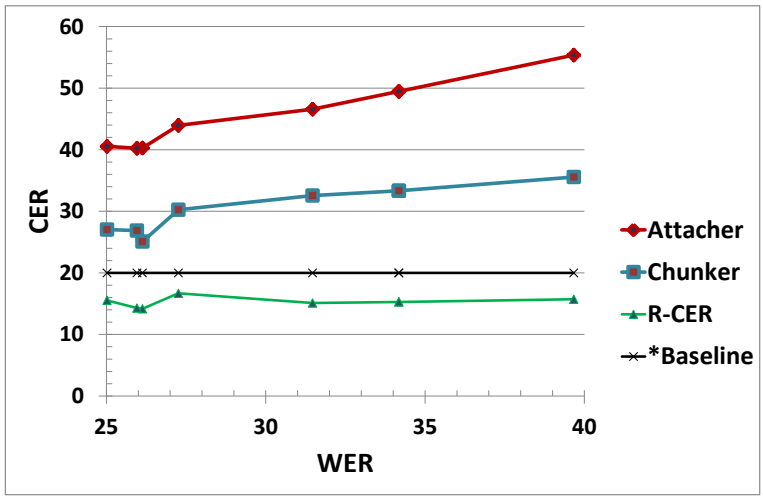


Figure 3.6: The performance of the Chunking parser with respect to ASR's WER (R-CER: relative CER).*Baseline is Chunking parser's CER on the manual transcriptions of route instructions

Attacher's original performance of 19.99 on the transcriptions. Moreover, the rather steady *relative CER* curve (R-CER) in Figure 3.6 highlights the robustness of our approach in dealing with errors in speech recognition.

3.6.6 Illustration of Chunking parsing

To illustrate the performance of the Chunking parser on speech recognition results, we provide two examples below. ASR errors are highlighted in gray in the speech recognition results.

Example 1

Human speech (manually transcription): "*okay if you erm if you get to the university and then take the third first exit just past the university and then turn right erm boots is on your left*"

Speech recognition hypothesis: "go front again the university and then take the third first exit just past the university and then turn right and trees is on your left"

Chunking parser output (independent of the Segmenter)**:**

(17) [CONTINUE(*landmark*:→) go front again] [UNIVERSITY the university] [DM and then] [TAKE(*count*: →, *landmark*:→) take] [COUNT the third] [EXIT first exit] [AFTER(*landmark*:→) just past] [UNIVERSITY the university] [DM and then] [TAKE(*direction*:→) turn] [RIGHT right] [DM and] [LANDMARK trees] [AT(*direction*: →,*landmark*: ←) is] [LEFT on your left]

The Chunking parser output shows that,

1. Despite errors in the recognized hypothesis the conceptual information about going to the University present in spoken utterance was identified and represented in the route graph (CONTINUE (*landmark*:→)).
2. On encountering the unseen concept "trees" that did not exist in our domain model, the Attacher has backed off and assigned to it the more general concept LANDMARK. In this way, the Attacher has not only preserved the fact that a landmark has been mentioned, but also maintained its structural relationship to the concept AT (the spatial relation) and LEFT in the route graph. This is where the system can initiate a spoken dialogue to obtain further details about the unknown concept.

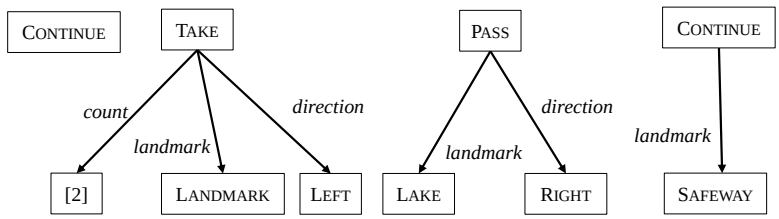
Example 2

Table 3.7 illustrates the performance of the Chunking parser with the visual aid of the conceptual route graphs for semantic representation. In the table, row 1 presents the manual transcription of the first segment of a route (we show only the first segment for the purpose of brevity). Row 2 presents the manually annotated CRG (actually the sub-graphs) for this route segment. Row 3 illustrates the sub-graphs extracted by the Chunking parser (independent of the Segmenter) on the transcription. Observe that the Attacher was unable to assign the concept LANDMARK as an argument (=attribute) to the action concept TAKE. Similarly, it failed to attach the concept RIGHT as the direction attribute of the controller concept PASS. Row 4 presents the speech recognition result of the route instruction in row 1. ASR errors are highlighted in gray. Row 5 presents the CRG extracted on this ASR result. The substitution of the word "*walking*" in the original instruction with the word "*going*" by the ASR did not affect the Chunker performance as it was identified to be part of the controller chunk (i.e., PASS). However, the presence of erroneous

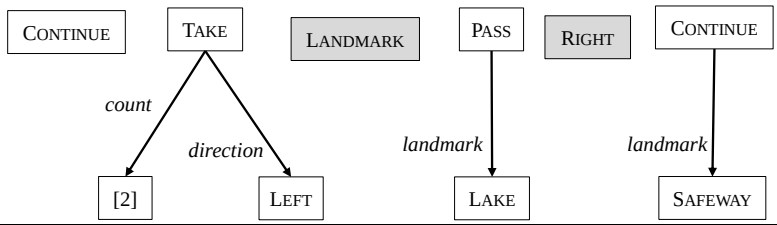
Table 3.7: An illustration of the Chunking parser. (1) Manual transcription from the IBL corpora, (2) Manually annotated sub-graphs in the CRG, (3) Sub-graph extracted by the Chunking parser from (1), (4) ASR result of the spoken route instruction (1), and (5) Sub-graph extracted by the parser from the ASR result.

(1) *"walk straight ahead and take the second turning on your left hand side keep walking past the lake on your right hand side keep walking until you come to Safeway…"*

(2)

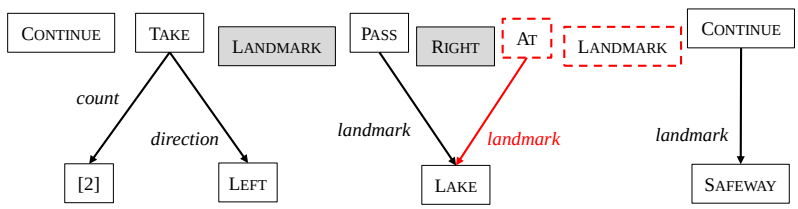


(3)



(4) *"walk straight ahead and take the second turning on your left hand side keep going past the lake on your right hand side is a parking until you come to safeway.."*

(5)



phrase "*is a parking*" in the ASR result led to addition of two erroneous concepts, AT and LANDMARK.

3.6.7 Summary of the quantitative evaluation

In contrast to the baseline performance of the Chunking parser using a *keyword spotting* based method for route instruction interpretation (CER

of 77.70) the final performance (CER of 19.99) using the features proposed here is very promising. Moreover, the rather steady *relative* CER on speech recognized route instructions shows that the presented framework is robust to speech recognition errors.

The performance of the automatic Segmenter was not in line with the performance of the Chunker, which means that we still might need to use heuristics for this task. In order to improve the performance of the *Segmenter* we would need both more data and better algorithms. We observed that the most useful feature for the task of segmenting is the chunk label window (cf. Table 3.5). However, the structure of a segment is not a rigid template. There may be two issues involved here. First, as discussed in Section 3.4, in a route segment the chunk elements: *controller*, *router* and *action* are not all mandatory. The requirement is that at least one of them is present. Second, within a route segment the three elements can come in different orders, e.g., in route instructions "*take a left after the church*" the segment begins with the action, whereas in "*after the church take a left*" the segment begins with a router. Due to lack of these two structural regularities, the classification task may not be easy and require more training data. Furthermore, in our dataset we had, on average, just two segments per route instructions, suggesting only few instances of the class B-SEGMENTER (a tag marking the beginning of a segment). As argued earlier and shown in Figure 3.5, more training data should improve the performance of the Segmenter. Since segment descriptions vary in length, using algorithms that are better suited for keeping long range dependencies may be topic for future work.

3.7 Qualitative Evaluation

While the quantitative evaluation above shows that the method is able to robustly capture much of the relevant concepts and structural information, it does not tell us to what extent the resulting route graphs are actually useful for route following. Since we do not have a full system that could make use of the route graphs (for route following), we have done a qualitative evaluation to see to what extent humans are able to make use of them. We realize this by asking human participants to sketch the described route on a map. Such an objective evaluation offers an alternative approach to evaluate our method: comparable human performances (in route drawing) using the manually transcribed CRGs and the CRGs produced from speech recognition results would confirm that our method is able to preserve vital conceptual information for route

following, despite speech recognition errors. In addition, a detailed analysis of human performances can help us (i) identify areas for further improvement in our method and the model, and also (ii) assess the usefulness of CRGs as a semantic representation for freely spoken route descriptions.

In the evaluation presented below, the task of sketching the path was formulated as a treasure-hunt task. The goal for the participants was to draw the path from the starting location (marked on the map) leading to the building containing the treasure, i.e., the destination of the IBL route description (not marked on the map). As part of the experimental-setup they would rely on different type of instructions to perform the task.

3.7.1 Method

Material: A new dataset of IBL instructions was created for this study. It contained 5 new route instructions, in addition to the set used in the quantitative evaluation presented in Section 3.6. On this dataset, a total of 35 route instructions, the Chunking parser obtained a CER of 18.04 on the manual transcriptions, a CER of 28.15 against a WER of 27.59, and a relative CER (i.e., CER minus WER) of 10.11. The relative increase in CER (R-CER) remains rather steady (SD = 2.80) with increase in WER.

Six IBL route descriptions from the set of 35 were used for human evaluation. Care was taken in selecting routes to ensure that subjects could not guess the destination. For each route we obtained four types of instructions, i.e., representations of the route instruction: (1) the manual transcriptions from the IBL corpus (Trans), (2) the manually annotated CRG (Annotation), (3) the CRG extracted by the Chunking parser from the manual transcription (Chunking), and (4) the CRG extracted by the parser from the speech recognized route description (ASR). A sample of these four instruction types has been earlier presented in Table 3.7 (namely rows 1, 2, 3 and 5 respectively).

The 24 items resulting from this combination (6 routes × 4 instruction types) were rearranged into four sets, each comprising six unique routes and contained the four instruction types. Due to the mismatch between the number of routes and instruction type, each set had two instruction types that were repeated. Each set was required to be evaluated by equally many numbers of participants to obtain a balanced observation set. Along with each item, a print of the IBL town map (see Figure 3.7, adapted from the original for this experiment) was provided for drawing the route. The starting locations for each route were marked on the map.

Figure 3.7: An edited picture of IBL town setup used in the evaluation

Subjects: A total of 16 humans (13 male and 3 female) participated in the evaluation. Participants ranged in the age from 16 to 46 (mean = 30.87, *SD* = 7.74). All, but one were researchers or graduate students in computer science at the School of Computer Science and Communication at KTH.

Procedure: Participants were asked to sketch the path from the starting location (marked) on the map to the target building (containing the treasure), following the specified instruction type. Each participant was individually introduced to the basic concept types in CRGs and shown how a route could be captured using the various nodes and sub-graphs in a CRG. They were also informed that loose nodes in the graphs may have relations with their neighboring concepts. A trial treasure-hunt task was used to ensure that the participants were more or less able to use the CRGs for drawing the routes. Participants were asked to also mark nodes and sub-graphs in the CRGs that they thought were absolutely necessary and strike-out what was redundant for route drawing. Each of the four sets was evaluated by four different participants.

3.7.2 Results and analysis

A total of 96 sketched routes (6 × 16) were obtained from the treasure-hunt task. We classified these human performances under three categories:

- FOUND: the participant arrived at the target building following the intended path,
- CLOSE: the participant has almost arrived at the target building following the intended path, but did not correctly identify it among the candidate buildings,
- NOT_FOUND: the participant lost her way and did not arrive at the target building.

Figure 3.8 provides an overview of the three performances across the four instruction types. A general observation can be drawn from this figure: the performances in route drawing using the conceptual route graphs (i.e., instruction types Annotation, Chunking, and ASR) differ from the ones using the transcribed route instructions (Trans). Among the performances using the conceptual route graphs it appears there is no major difference between Annotation and Chunking instruction types. The most number of failures in finding the target building was using CRGs extracted from speech recognition results (ASR).

A one-way between subjects ANOVA was conducted to compare the effect of instruction type on the performance in route drawing, in Trans, Annotation, Chunking, and ASR conditions. There was a significant effect of instruction type on performances in route drawing at $p < 0.05$ level for the four conditions [$F(3,92) = 3.07$, $p = 0.032$]. Post hoc comparison using the Benferroni test indicates that the mean score for the Trans condition was significantly different from the ASR condition ($p = 0.030$). However, there was no significant difference in other condition pairs. This affirms the general observation made above that there is apparently no difference between performances using the manually annotated CRG (Annotation) and the ones extracted by the Chunking parser (Chunking). This suggests that the conceptual information, required for human route following, present in Chunker parser produced CRGs is comparable with the information present in manually annotated CRGs. These results also suggest that improving the model (i.e. the CRG representation) to reduce the gap between human performances for Trans and Annotation instruction types (see Figure 3.8) will further enhance the human performances for Chunking parser extracted CRGs.

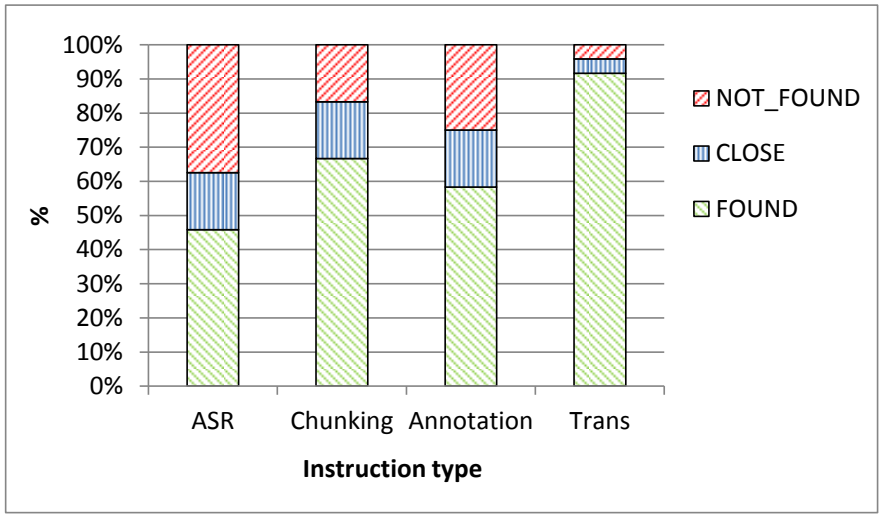


Figure 3.8: Human performances across the instruction types.

The performances in route drawing using the CRGs extracted from ASR results (WER of 27.39) were adversely affected by the increase in CER. This was expected. However, for most routes the subjects were able to find the target building or arrive close to it. These instances demonstrate the robustness of the Chunking parser in dealing with speech recognition errors and preserving the vital conceptual information required for route following.

Analysis of the failed performances: On 33 occasions the human participants failed to find the target building, either because they lost their way (NOT_FOUND on 20 occasions) or were unable to identify the target building among the candidate buildings (13 instances of CLOSE). We analyzed these 33 cases based on the route drawn on the map, the concepts that had been marked as useful or redundant for the task, and our knowledge about the target building.

At a first level of analysis, we observed that the problems in route drawing can be generally classified in five categories, four of which can be associated with the generic concepts in the route instruction task domain. These categories pertain to:

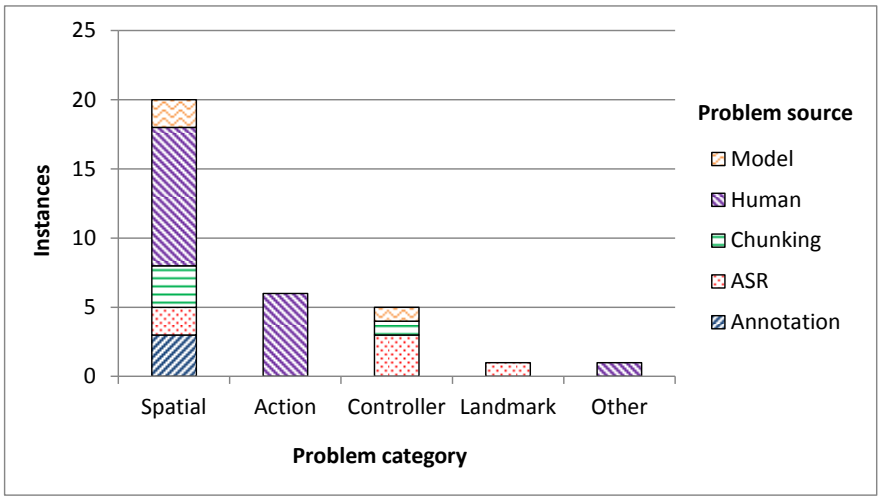


Figure 3.9: Distribution of error sources across the problem categories

1. **Spatial** relations in the route instructions, such as "***at*** *the post-office*", "***after*** *the post-office*", or "***on the right of*** *the post-office*". Correct interpretation of these instructions requires the knowledge about the spatial relation of the landmarks. Ignorance of these relations on part of the human follower, or the failure of the Chunking parser in detecting and preserving them, would hinder the correct interpretation of the instructions.
2. **Controller** concepts used for ensuring traversal along a path. For example, in the route instruction "*continue till the end of the street and turn left*" the knowledge about how far to go before changing the orientation is paramount to correct interpretation of the instruction.
3. **Action** concepts indicating the direction of reorientation. It is critical to take the exact direction at the exact location to be able to correctly follow the path.
4. **Landmark** concepts in the route instructions. They are used for ensuring traversal on path, resolving spatial relations, executing reorientation actions, and identifying the target building.
5. **Other**: this category contains instances of errors made by human participants, e.g., in not starting at the place marked in the map.

Figure 3.9 illustrates the distribution of these five problem categories as observed in the 33 instances of failed performances. The heights of the bars indicate the number of problems associated with the respective category. We can see from the figure that most problems concern **Spatial** relations. Other major problem categories are **Action** and **Controller**. Going a step further, we analyzed the cause of these problems. Some of the usual suspects are the Chunking parser and the ASR system. However, in our analysis of the failed performances we observed that the problem sources can be put in five categories. These include:

1. **Annotation**: an incorrect or underspecified manual annotation in the conceptual route graph.
2. **ASR:** insertion of erroneous concepts and deletion of relevant concepts in the speech recognition output.
3. **Chunking**: the Chunking parser introduced errors, e.g., failed to preserve the relations among concepts in the route graph.
4. **Model**: a limitation of the current domain model (the ontology) for representing the semantic of route instructions.
5. **Human**: incorrectly judging concept in the route graphs as redundant could lead the participant off the path. For example, ignoring spatial relations to landmarks or places for reorientation can lead to a wrong path. Also, we observed that some participants made mistakes in correctly orientating themselves, e.g., they marked the concepts TURN and RIGHT, useful, but drew a left turn instead.

Figure 3.9 also details the distribution of the five problem sources within the five problem categories. We can observe that a majority of issues related to spatial relation category are due to human judgement. This indicates that it was not always easy to make the right decision about discarding or using the concepts for spatial relations in the CRGs for route drawing. It is important to note here that these human errors indicate a likely area of problem in route following by the computer: deciding which of the concepts in the CRG are relevant and which are redundant.

The Chunking parser also contributed to problem when it assigned a wrong spatial relation concept to a chunk. As discussed earlier, while AT and AFTER are concepts of the same type (spatial relation), their semantic interpretation results in two different paths.

Incorrect manual annotations for the type of spatial relation and lack of appropriate representation in the domain model also led to a number of

errors in the identification of spatial relations. This analysis helps us identify areas where we can improve the system, e.g., fixing the annotations, extending the model, and learning better models for detection the specific type of spatial relations. As to why subjects had difficulty in judging their utility for route drawing may require further investigation.

In Figure 3.9, we can also observe that all the problems in the Action category are due to human errors. In fact, we observed that all these cases pertain to problems in changing orientation. For example, the instructions suggested taking a right turn but the subject took a left turn. It is important to note here that a computer is not likely to make these errors in route following.

We also observed that most errors from the ASR are due to deletion of relevant concepts. This led to problems in the Spatial and the Controller categories.

3.7.3 Summary of qualitative evaluation

The purpose of this study was to evaluate the extent to which the Chunking parser generated route graphs would actually be useful for route following. While the study is based on a qualitative evaluation of only 6 routes, the analysis has helped us verify that the Chunking parser is able to preserve vital conceptual details required for route following. The evaluation has also allowed us to identify areas of further improvement for the Chunking parser. We note that:

- Controllers with travel distance argument are vital for representing the extent of movement in a particular direction in route descriptions, such as "*follow the road to its end on the right is the treasure*" or "*a few buildings down from Pizza-Hut*". Our current model lacks representation for such instructions.
- A pre-requisite for proper grounding of the spatial relations in conceptual route graph is resolving their *direction* or *landmark* arguments (or even both depending on the situated context). For example, in the CRG "[$_{\text{BUILDING}}$ Tescos] [$_{\text{AT}}$ (*landmark*: ←) is on] [$_{\text{RIGHT}}$ right]", the Attacher's role in identifying the *direction* attribute for the spatial relation AT and attaching it to the following chunk (RIGHT), is essential for resolving the location of Tescos.
- The CRG representations for spoken route description contain redundant concepts that arise from speech phenomena, such as pronominal references, anaphoric descriptions, self-repair and repetitions, about *landmarks* and *actions*. The CRG

representation for "*you will take the third exit off…the third exit will be for Plymouth university…take this third exit*", contains two *actions* and four *landmarks*. Interpreting such utterances in an appropriate manner would require more sophisticated parsing techniques.

- ASR errors pose another challenge for an agent in route planning using the CRGs. Without access to the topological view of the environment a robot could not possibly infer erroneous concept insertions. To deal with this, we believe clarification or reprise of route segments would be a possible strategy, provided that the clarification sub-dialogue itself doesn't lead to further errors. Furthermore, concepts in the CRG representation can be assigned weights following the human participants' feedback on the nodes in a CRG they found useful and redundant. This should enable the dialogue system to strategically use the option to engage in clarification dialogue with the users.

3.8 Conclusion and Discussion

In this chapter, we set out with the task of semantic interpretation of spoken route instructions. The task presents two requirements: robust parsing of "ungrammatical" and erroneous user utterances (as recognized by the ASR) and extraction of deeper structural relations among the concepts in the semantic representations (such as the conceptual route graph (CRG)).

To meet these two requirements, we have proposed a novel application of the Chunking parser. In our methods, the Chunking parser comprises of three stages. In the first stage, the Chunker takes a sequence of words as input and produces a stream of chunks (basic domain concepts). Since the Chunker aims at identifying only the domain relevant concepts in the input and does not have to account for every single word, it is able to perform robust parsing of "ungrammatical" and erroneous inputs. In the next stage, the Segmenter, which works on the same principle as the Chunker, takes the Chunker output and chunks it into a stream of route-segments (i.e., a large chunk made up of other basic chunks). Finally, the Attacher assigns the basic chunks a more specific concept and attributes. The attributes are attached to neighboring concepts as arguments (i.e., relations). In this way, the proposed Chunking parser performs robust parsing of spoken route instructions and extracts the deeper structural relations in the conceptual route graphs.

We took a data-driven approach and modelled the three stages as classification tasks. We used the Naïve Bayes and Linear Threshold Units algorithms for training the models. The models were trained on various features including word instance, word window, previous chunker tags, part-of-speech (POS), bag-of-words, bag-of-POS, and chunk label window. We conducted two types of assessments of the proposed approach: a quantitative and a qualitative evaluation.

We trained and tested the models on a corpus of route instructions that were collected in a Wizard-of-Oz setup (the IBL corpus). The performance of the Chunking parser was evaluated on both manual transcriptions and speech recognized route instructions. On the manual transcriptions, using the best feature combination presented here, the Chunking parser obtained substantial gains over using only word instance as a feature (i.e., a *keyword spotting* based scheme for interpretation). While the keyword spotting scheme obtained a concept error rate (CER) of 77.70, the best feature combinations achieved a CER of only 29.11, i.e., a reduction in CER of about 50 points. When the Chunking parser was evaluated independent of the Segmenter, the CER was 19.99. Going further, we have also observed that the models presented here are likely to benefit from more training data.

In another experiment, we have shown that the performance of the Chunking parser follows ASR WER. Against a WER of 25.02, the Chunking parser achieved a CER of 40.55. This implies that the *relative* increase in CER (i.e., CER minus WER) is 15.53. This is comparable to the best CER (of 19.99) obtained on the manual transcriptions. Moreover, we have shown that the relative increase in CER is rather stable over increasingly noisy ASR.

Going further, we have presented a qualitative evaluation in order to assess the utility of the Chunking parser extracted conceptual route graphs. Since we did not have a full system that could make use of the route graphs, we asked human participants to sketch the described route on a map. In a study, 16 human participants were asked to draw the routes using four different types of route instructions: the transcription of the route instructions, the manually transcribed CRGs, the Chunking parser extracted CRGs and the CRG's extracted from speech recognized results. Analysis of participants' performance in route drawing indicated that the performance using the three types of conceptual route graphs were comparable. That is, there is no significant difference in route drawing using the manually annotated CRGs and the ones extracted from the speech recognized route instructions. This affirms the robustness of the

Chunking parser in preserving the vital concepts required for route drawing under speech recognition errors.

The presented approach makes new contributions to the task of spoken language understanding. Earlier, Johansson et al. (2011) applied the Chunking parser on a Swedish corpus and were able to obtain a CER of 25.60 viz.-a-viz. a baseline CER of 83.34 (obtained using the keyword spotting based Chunking parser). In conjunction with our results for the Chunking parser on the English corpus (the IBL corpus), a CER of 19.99 vs. baseline 77.70, we have strong evidence suggesting that the proposed data-driven approach could be easily extended for understanding route instructions in other languages, using simple features. Features such as affixes and part-of-speech which were found informative for interpreting route instructions in Swedish were not useful for English. This might be explained by the fact that Swedish has a richer morphology in comparison to English.

In the literature, grammar based approaches for extracting deeper structures (Charniak, 1997; Uszkoreit, 2000) and data-driven approaches for robust parsing (He & Young, 2006; Meza-Ruiz et al., 2008; Wong & Mooney, 2007) have been presented. However, they only cater to only one of the two requirements for the semantic interpretation of spoken route instructions. In contrast, the approach presented here offers robust parsing and extraction of deeper structural relations.

In the area of human–robot interaction, various schemes for interpretation of route directions have been presented (Bugmann et al., 2004; Mandel et al., 2006; Kollar et al., 2010; Pappu & Rudnicky, 2012; MacMahon et al., 2006). However, most of these works have focused on interpreting manually transcribed or human written route descriptions. In contrast, our approach has been tested for semantic interpretation of spoken route instructions and has shown to be robust to ASR errors. These are encouraging results for using the proposed Chunking parser in real life human–robot interaction settings.

3.9 Directions for future work

In the current model of Chunking parser, the performance of the Segmenter is not very satisfactory. From the analysis of the impact of training data size on Segmenter performance, we have observed that more data is likely to improve the model performance. In the current dataset there are on average just two route segments per instruction. In other words, there are only few instances for training a learner for recognizing

the beginning of a route segment. Furthermore, as the constituent concepts in route-segments do not exhibit structural rigidness, the learning task is much harder. For example, route instructions "*at the church turn left*" and "*turn left at the church*" have the same semantics; however, the order of concepts and hence the beginning of the route-segment is marked with different concepts.

In addition to using more data, the proposed approach may also benefit from using better algorithms. Perhaps the task of chunking can be modelled as a sequence labelling task, using Hidden Markov Models and discriminative methods such as Conditional Random Field (Collins, 2002). These methods have been widely used for the chunking task (Sang & Buchholz, 2000).

The proposed Chunking parser has been developed and tested on monologues of route instructions. However, in order for the parser to be useful for real human–computer interactions, it will need to be adapted for processing incremental inputs. As the Chunking parser presented in this work was able to achieve reasonably good performance using very little details from the right context, we believe the proposed method can be easily adapted for evaluation in an incremental framework for dialogue processing. Towards training and evaluating the Chunking parser on incremental inputs, we can use the set of IBL instructions that contain a dialogue between the human route giver and the wizard. In the setup for data collection, the wizard only gave acknowledgements and backchannels to the human subject as she gave the route instruction. The utterance segments of the subjects (interleaved by wizard's feedback) can be used to train a first incremental model of the Chunking parser.

From the analysis of human performances in route following, we observed that speech phenomena such as repetitions, revisions and anaphoric references could lead to the presence of redundant concepts and sub-graphs in the conceptual route graphs. This had adverse effect on participants' performance on route drawing. An interesting line of work would be to develop methods for dealing with these speech phenomena.

Chapter 4

Turn-taking

4.1 Introduction

A very important property of spoken dialogue systems is *responsiveness*. This involves giving feedback to the user (using short utterances such as "*uh-hun"* and "*mm-hmm*") to show continued attention and producing responses when appropriate or expected by the user. In the absence of timely feedback and responses from the system, the user may think that the system has not understood her. This may cause the user to either restart or respond (repeat/rephrase the initial request or seek clarification) or silently wait until the system responds. Such situations in interaction lead to inefficiency in human–computer dialogues and are likely to adversely affect user experience (Ward et al., 2005). As discussed earlier in Section 2.5, dialogue systems have generally relied on very simple model for turn-taking: the system uses a fixed silence threshold to detect the end of the user's utterance, after which the system responds. Such a simplistic model can result in systems that frequently produce responses at inappropriate occasions, or produce delayed responses or no response at all when expected, thereby causing the system to be perceived as interruptive or unresponsive.

More advanced models of turn-taking and backchanneling have been investigated in the literature. This includes the two theoretical models on human turn-taking mechanism developed by Duncan (1972) and Sacks et al. (1974), and corpora-based studies inspired by these two models (Ward, 1996; Koiso et al., 1998; Cathcart et al., 2003; Gravano & Hirschberg, 2011). In these models, conversational cues such as content, syntax, intonation, paralanguage, and body motion, are used for modelling the timing of turn-taking and back-channels. These models have been suggested to be useful for modelling human-like turn-taking behavior in spoken dialogue systems.

However, very little attention has been paid to actually using these models online in dialogue systems and assessing their impact on the interaction. We believe there are two problems in doing so. First, the data used in the studies mentioned above are from human–human dialogue,

and it is not obvious to what extent the models derived from such data transfers to human–computer dialogue. Second, many of the features used in the proposed models were manually extracted. This is especially true for the transcription of utterances, but several studies also rely on manually annotated prosodic features (e.g., the patterns of pitch contours, such as *rising* and *falling*, were manually labelled).

In this chapter, we present a data-driven approach to automatic detection of relevant feedback response locations in the user's speech. The setting is that of a Map Task[6], in which the user describes a route and the system may respond with, for example, acknowledgements or clarification requests. Gravano & Hirschberg (2011) have observed that turn-yielding and backchannel inviting cues vary in their characteristics. This suggests that the underlying models for managing turn-taking and backchannel in human–human interaction differ and a computational model should account for these differences. However, in the kind of task we are studying here, short feedback utterances such as "*okay*" are perhaps not best described as backchannels, but rather as "acknowledgements" (Allen & Core, 1997). These do not simply signal "continued attention", but rather "acceptance" (i.e., that the next route segment has been identified) according to Clark's "levels of understanding" (Clark, 1996) (cf. Section 2.5.3). Thus, while backchannels might be optional to a larger extent, acknowledgements are often required for the task to be continued. Also, the feedback generated in a Map Task (such as acknowledgements or clarification requests) are in general not intended to take the initiative or claim the floor. We therefore train a model that can detect general feedback response locations, and we call this *Response Location Detection* (RLD). The type of feedback that should actually be generated is not in the scope of this study

In comparison to previous approaches we propose models for online use and hence rely on only automatically extractable features—comprising syntax, prosody and context. Thus, when using patterns of pitch contour as a feature, we use automatic estimate of the slope instead of using any manual labelling of patters. Furthermore, the models have been trained on human–computer dialogue data and one of them has been implemented in a dialogue system that is in turn evaluated by users.

[6] The Map Task was first designed by Anderson et al. (1991) to elicit behaviors that answer specific research questions in linguistics, e.g., turn-taking in a dialogue. In their setup, one human subject described a route to another human subject whose task was to draw the route on a map.

We start in Section 4.2 with a discussion on previous studies on cues that human interlocutors use to manage turn-taking and backchannels. We will also elaborate on some of the proposed computational models and discuss the features used. In Section 4.3, we describe the test-bed that we used for boot-strapping a Map Task dialogue system to collect human–computer interaction data and develop an improved incremental version of the system. The various data-driven models that we have trained in this work will be discussed in Section 4.4. We describe the various features explored in this work, and discuss their performance using various learning algorithms, for online use. In Section 4.5, we discuss the subjective and objective evaluation schemes used for verifying the contributions of one of the trained model in user interactions. Finally, in Section 4.6, we discuss the key contributions and limitations of the models presented in this paper, and conclude with some ideas for future extensions of this work.

4.2 Background

Much of the work in developing computational models for turn-taking and backchanneling is influenced by the theories of Duncan (1972) and Sacks et al. (1974). While these theories have offered a function-based account of turn-taking (cf. Section 2.5.4), another line of research has looked into corpora-based techniques to build models for detecting turn-transition and feedback relevant places in speaker utterances. In this section, we will first review these models and then make a case for our model.

Ward (1996) suggested that a 110 millisecond (ms) region of low pitch is a fairly good predictor for backchannel feedback in casual conversational interactions. He also argued that more obvious factors, such as utterance end, rising intonation, and specific lexical items, account for less than they seem to. He contended that prosody alone is sometimes enough to tell you what to say and when to speak. Truong et al. (2010) extended this model by including pause information. They observed that the length of a pause preceding a backchannel is one of the important features in their model, next to the duration of the pitch slope at the end of an utterance.

Koiso et al. (1998) analyzed prosodic and syntactic cues to turn-taking and backchannels in Japanese Map Task dialogs. They observed that some part-of-speech (POS) features are strong syntactic cues for turn-change, and some others are strongly associated with turn-holds. Using

manually extracted prosodic features for their analysis, they observed that falling and rising F0 patterns are related to changes of turn, and flat, flat-fall, and rise-fall patterns are indications of the speaker continuing to speak. Extending their analysis to backchannels, they suggested that syntactic features, such as filled pauses, alone might be sufficient to discriminate when backchanneling is inappropriate, whereas the presence of backchannels is always preceded by certain prosodic patterns.

Cathcart et al. (2003) presented a shallow model for predicting the location of backchannel continuers in the HCRC Map Task Corpus (Anderson et al., 1991). In their model they explored features such as word count in the preceding speaker utterance, POS tag, and silent pause duration. A model based on silent pauses only inserted a backchannel in every speaker pause longer than 900 ms, and yet performed better than a baseline word model that predicted a backchannel every seventh word. A tri-gram POS model predicted that nouns and pronouns before a pause are the two most important cues for predicting backchannel continuers. The combination of the tri-gram POS model and pause duration model offered a five-fold improvement over the baseline model.

Bell et al. (2001) presented an approach to incrementally classify fragmentary user utterances as *closing* (more input is unlikely) and *non-closing* (more input is likely to come) in a multimodal dialogue system to decide whether the system should respond or wait for additional spoken or graphical input. They observed that dialogue context, lexical content, and semantic features were useful cue for the classification task.

Gravano & Hirschberg (2011) examined seven turn-yielding cues that occur with a significantly higher frequency in speaker utterances prior to a turn transition than those preceding turn holds. These events are: (i) a falling or high-rising intonation at the end of speaker turn; (ii) an increased speaker rate; (iii) a lower intensity level; (iv) a lower pitch level; (v) a longer duration; (vi) a higher value of three voice quality features: jitter, shimmer, and noise-to-harmonic ratios; and (vii) a point of textual completion. Gravano & Hirschberg (2011) also investigated whether backchannel-inviting cues differ from turn-yielding cues. They examined a number of acoustic and lexical cues in the speaker utterances preceding smooth turn-changes, backchannels, and holds. As in the case of turn transition, they identified six measureable events, but with different patterns then in the case of transitions, that are strong predictors of a backchannel at the end of an inter-pausal unit (IPU).

Another example is the line of work on building virtual rapport with animated agents, in which the agent should, for example, be able to

produce head nods at relevant places. Using data on human–human interaction, Morency et al. (2010) investigated the use of multi-modal features (including gaze, prosody and transcribed speech) for estimating the probability of giving non-verbal feedback and obtained statistically significant improvement over an approach based on hand-crafted rules.

When it comes to using these features for making online turn-taking decisions in dialogue systems, however, there is very little related work. One notable exception is Raux & Eskenazi (2008) who presented an algorithm for dynamically setting *endpointing* silence thresholds based on features from discourse, semantics, prosody, timing, and speaker characteristics. The model was also applied and evaluated in the Let's Go dialogue system for bus timetable information. However, that model only predicted the endpointing threshold based on the previous interaction up to the last system utterance; it did not base the decision on the current user utterance to which the system response is to be made. Another case is Huang et al. (2011) who presented a virtual interviewer which used a data-driven backchannel model (for producing head nods) in combination with a handcrafted turn-taking model (for asking the next question). An evaluation showed that subjects did indeed experience the turn-taking capabilities of the system as better compared to a baseline system. However, their system did not explore if word-level features play any role in predicting backchannel.

In order to improve current systems, we need a better understanding of the phenomena of human interaction, better computational models, and better data to build these models. As the review above indicates, a common procedure is to collect data on human–human dialogue and then train models that predict the behavior of the interlocutors. However, we think that it might be problematic to use a corpus of human–human dialogue as a basis for implementing dialogue system components. One problem is the interactive nature of the task. If the system produces a slightly different behavior than what was found in the human–human training data, this would likely result in a different behavior in the interlocutor. Another problem is that it is hard to know how well such a model would work in a dialogue system, since humans are likely to behave differently towards a system as compared to another human (even if more human-like behavior is being modelled). Yet another problem is that much dialogue behavior is optional and therefore makes the actual behavior hard to use as a gold standard. Indeed, although many of the classifiers in the studies reported above show a better performance than baseline, they typically have a fairly low accuracy or F-score. It is also

possible that much of human behavior that is "natural" is not necessarily preferable for a dialogue system to reproduce, depending on the purpose of the dialogue system.

A common practice for collecting realistic human–computer interaction data in the absence of a working prototype is to use a Wizard-of-Oz setup. A human wizard operates "behind the curtain" while the users are made to believe that they are interacting with a real dialogue system. While this methodology has proven to be useful, it has its limitations. For example, the wizard's performance may not be consistent across users or even for the same users (Dahlbäck et al., 1993). The responsiveness of the wizard in responding to user behavior is another issue, which makes the method hard to use when the issue under investigation is time-critical behaviors such as turn-taking and backchannels.

An alternative to Wizard-of-Oz studies is using a "boot-strapping" procedure, where more and more advanced (or human-like) versions of the system are built iteratively. After every iteration, users interact with the system and data is collected. This data is then used to improve the data-driven models in the system. A problem here, however, is how to build the first iteration of the system, since many components, such as Automatic Speech Recognition (ASR), need some data to be useful at all. In a previous study, Skantze (2012) presented a test-bed for collecting realistic human–computer interaction—a fully automated spoken dialogue system that can perform the Map Task with a user. By implementing a trick, the system could convincingly act as an attentive listener, without any speech recognition. The data from user interaction with the system was used to train an online model of *Response Location Detection* (RLD). Based on automatically-extractable prosodic and contextual features, 200 ms after the end of the user's speech, the trained model was able to identify response locations with a significantly higher accuracy as compared to the majority class baseline. The trained model was, however, not evaluated in user interactions.

In this work, we extend the approach presented in Skantze (2012) in the following ways: First, we use an ASR component in order to model lexico-syntactic features. Second, we explore a range of automatically-extractable features for online use—covering prosody, lexico-syntax, and context—and different classes of learning algorithms. We explore the contribution of each of these modalities to the task of RLD, individually as well as in combination. Third, we evaluate one of the proposed models by integrating it in the same system that was used for data collection, and testing the model in live interaction with users.

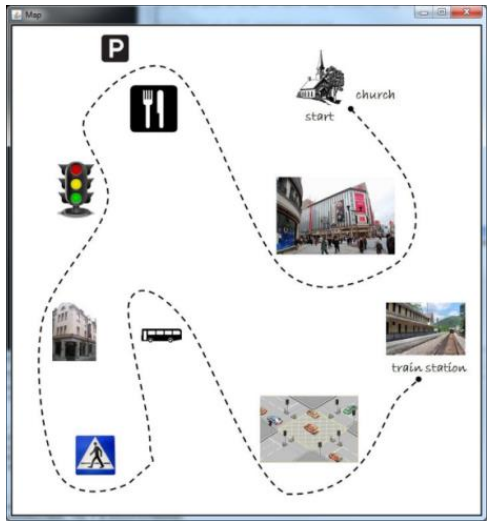


Figure 4.1: The user interface of the Map Task dialogue system, showing the map.

4.3 Bootstrapping a Map Task dialogue system

The Map Task is a common experimental paradigm for studying human–human dialogue, where one subject (the information *giver*) is given the task of describing a route on a map to another subject (the information *follower*). In our case, the user acts as the giver and the system as the follower. The choice of the Map Task is partly motivated because the system can allow the user to keep the initiative during the whole dialogue, and thus only produce responses that are not intended to take initiative, most often some kind of feedback.

Implementing a Map Task dialogue system with full speech understanding would indeed be a challenging task, given the state-of-the-art in automatic recognition of conversational speech. In order to make the task feasible, we have implemented a trick: the user is presented with a map on a screen (see Figure 4.1) and instructed to move the mouse cursor along the route as it is being described. The user is told that this is for logging purposes, but the real reason for this is that the system tracks the mouse position and thus knows what the user is currently talking about. It is thereby possible to produce coherent system behavior without any speech recognition at all, but only using voice activity detection (VAD). This often results in a very realistic interaction as compared to what users are typically used to when interacting with dialogue systems—

in our experiments, several users thought that there was a hidden operator behind it[7].

The system is implemented using the IrisTK dialogue system framework (Skantze & Al Moubayed, 2012). The basic components of the system can be seen in Figure 4.2. The system uses a simple energy-based speech detector to chunk the user's speech into inter-pausal units (IPUs), that is, periods of speech that contain no sequence of silence longer than 200 ms. Such a short threshold allows the system to give backchannels (seemingly) while the user is speaking, or take the turn with barely any gap. Similarly to Gravano & Hirschberg (2011) and Koiso et al. (1998), we define the end of an IPU as a candidate for the Response Location Detection (RLD) model to identify as a response location. We use the term turn to refer to a sequence of IPUs which do not have any interlocutor responses between them.

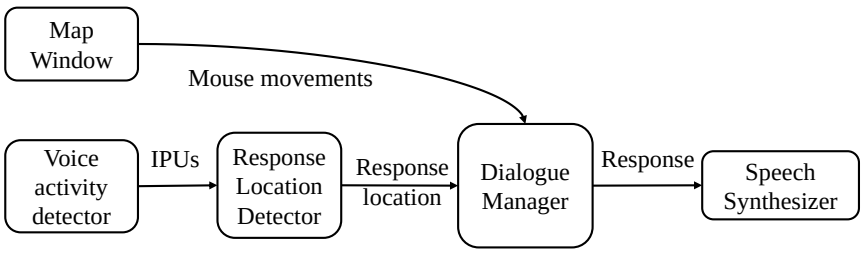


Figure 4.2: The basic components of the Map Task dialogue system (Iteration 1) used for data collection

Since we initially did not have any sophisticated model of response location detection, the system was simply set to wait for a random period between 0 and 800 ms after an IPU ended. If no new IPUs were initiated during this period, a RL was detected, resulting in random response delays between 200 and 1000 ms. Each time the RLD model detected a response location, the dialogue manager produced a response, depending on the current state of the dialogue and the position of the mouse cursor. Table 4.1 shows the different types of responses the system could produce. The dialogue manager always started with an Introduction and ended with an Ending, once the mouse cursor had reached the destination. Between these, it selected from the other responses, partly randomly, but also depending on the length of the last user turn and the current mouse

[7] An example video can be seen at http://www.youtube.com/watch?v=MzL-B9pVbOE.

location. Longer turns often led to Restart or Repetition Requests, thus discouraging longer sequences of speech that did not invite the system to respond. If the system detected that the mouse had been in the same place over a longer time, it pushed the task forward by making a Guess response. We also wanted to explore other kinds of feedback beyond simple acknowledgements, and therefore added short Reprise Fragments and Clarification Requests (see for example Skantze (2007) for a discussion on these).

Table 4.1: Different responses from the system

Response type	Example expressions
Introduction	"*Could you help me to find my way to the train station?*"
Acknowledgement	"*Yeah*", "*mm-hmm*", "*Okay*", "*Uh-hun*"
Reprise Fragment	"*A station, yeah*"
Clarification Request	"*A station?*"
Restart	"*Ah, I think I lost you at the hotel, how should I continue from there?*"
Repetition Request	"*Sorry, could you take that again?*"
Guess	"*Should I continue above the church?*"
Ending	"*Okay, thanks a lot.*"

4.3.1 Data collection and annotation

Ten subjects participated in the data collection. Although the task was conducted in English, no participants were native speaker (though proficient). They were seated in front of the display showing the map, wearing a headset. The instructor told them that they were supposed to describe a route to the computer. They were told that they should imagine the system having a similar picture as seen on the screen, but without the route. Each subject did five consecutive tasks with five different maps, resulting in a total of 50 dialogues. Table 4.2 shows an example interaction with the system.

The users' speech was recorded and all events in the system were logged. Each IPU in the corpus was manually annotated into three categories: HOLD (a response would be inappropriate), RESPOND (a response is expected) and OPTIONAL (a response would not be inappropriate, but it is perfectly fine not to respond). To validate the

coding scheme two human annotators labelled 20% of the corpus separately. For all three categories the kappa score was 0.68, which is substantial agreement (Landis & Koch, 1977). Since only 2.1% of all the IPUs in the corpus were identified for category OPTIONAL they would not have been sufficient for statistical learning of this class. Therefore we excluded them from the corpus and used the data instances for the RESPOND and HOLD categories only. The dataset contains 2272 IPUs in total, the majority of which belong to the class RESPOND (50.7%), which we take as our majority class baseline. Since the two annotators agreed between RESPOND and HOLD in 87.2% of the cases, this can be regarded as an approximate upper limit for the performance expected from a model trained on this data.

A simplifying assumption behind the current model is that the system will only consider response locations at the end of IPUs. While other models have applied continuous decisions for producing backchannels (e.g., Ward, 1996), we follow the approach taken in many of the related studies mentioned above (e.g., Koiso et al., 1998; Gravano & Hirschberg, 2011). This is again partly motivated by the fact that the acknowledgements produced by the system should perhaps not be considered as backchannels. Indeed, none of the annotators felt the need to mark relevant response locations within IPUs.

Table 4.2: An example Map Task dialogue between a user (U) and system (S)

S:	Introduction	*Could you help me to find my way to the bus stop?*
U:		start from the department store
S:	Acknowledgement	*yeah*
U:		and eh
S:	Guess	*Should I start by going west?*
U:		yeah do that
U:		then you will get to a meadow and when you get to the meadow
S:	Repeat	*Eh, could you repeat that?*
U:		you go straight and you see a meadow on your right side
S:	Clarification Request	*A green field?*

U:		ehm yeah a field
S:	Acknowledgement	*mm-hmm*
U:		pass the meadow and turn right so you are going north
S:	Acknowledgement	*okay*
U:		you get to a restaurant
S:	Reprise Fragment	*A restaurant*
U:		then you have to turn right and that will take you to
S:	Restart	*Ah, I think I lost you at the restaurant, could you take that again?*
…		
U:		at the junction go south and then you will get to the bus stop
S:	Ending	*Okay, thanks a lot.*

4.4 Data-driven models for response location detection

We used the human–computer Map Task corpus described in the previous section for training various new models for response location detection. Here, we discuss the various prosodic, contextual and lexico-syntactic features that we explored while training these models. We describe how we extracted the feature values from the IPUs, and also present the contributions of the features—individually as well as in combination—in classifying user IPUs as either RESPOND or HOLD type. As for the choice of models, we explored four data-driven models: the Naïve Bayes classifier (NB) as a generative model, and three discriminative models, namely a J48 decision tree classifier, a Support Vector Machine (SVM) with radial basis kernel function, and a Voted Perceptron (VP). We compare the performances of these models against the majority class baseline of 50.7%. For all three classifiers we used the implementations available in the WEKA toolkit (Hall et al., 2009). All results presented here are based on 10-fold cross-validation.

4.4.1 Prosodic features

We extracted pitch and intensity (sampled at 10 ms) for each IPU using ESPS in Wavesurfer/Snack (Sjölander & Beskow, 2000). The values were transformed to log scale and z-normalized for each user. We then identified the final 200 ms voiced region for each IPU. For this region, the **mean pitch** and **pitch slope** (using linear regression) were calculated. We used mean pitch in conjunction with its **absolute value** as a feature. Pitch slope was also used as a feature in combination with its correlates such as the **correlation coefficient *r*** for the regression line and the **absolute value** of the slope. In addition to these, we also used the **duration** of the voiced region as a feature. The last 500 ms of each IPU were used to obtain the **mean intensity** and **intensity slope** measures. As with the pitch features, we used these two measures in combination with the absolute value and the two correlates of slope, respectively.

The individual and collective performances of these prosodic features for the four classifiers in identifying user IPUs as either RESPOND or HOLD type is presented in Table 4.3. The combined pitch features offer the best accuracy of 66.2% using the SVM classifier. Using the intensity features in combination, the best accuracy of 60.7% is obtained by the J48 classifier. All the prosodic features used together offer the highest accuracy of 66.9% using the SVM classifier.

Table 4.3: Percentage accuracy of prosodic features in detecting response locations

		Algorithm			
#	***Feature(s)***	***J48***	***NB***	***SVM***	***VP***
1.	**All pitch features**	65.3	63.4	**66.2**	64.3
2.	**All intensity features**	**60.7**	58.4	57.1	55.2
3.	***All prosodic features combined***	65.7	66.3	**66.9**	62.8

4.4.2 Contextual features

We explored discourse context features such as **turn** and **IPU length** (in terms of duration in seconds), and **last system dialogue act**. We also used the **pause duration** between the onset of a user IPU and the end of previous user/system IPU as a feature. The values for the duration

Table 4.4: Percentage accuracy of contextual features in detecting response locations

		Algorithm			
#	*Feature(s)*	*J48*	*NB*	*SVM*	*VP*
1.	**IPU length (in seconds)**	60.8	57.3	61.2	60.9
2.	**Pause duration before user IPU onset**	57.3	53.7	56.3	54.6
3.	**Turn length (in seconds)**	58.6	58.4	58.8	59.1
4.	**Last system dialogue act**	54.1	54.1	54.1	53.4
5.	***All contextual features combined***	62.5	59.7	**63.7**	62.8

features were automatically extracted from the output of the voice activity detector (VAD). These values are therefore subject to the performance of the VAD component. Table 4.4 shows the performance of these contextual features, individually as well as in combination, in discriminating user IPUs as either RESPOND or HOLD type. In general, all features offer an improvement over the baseline accuracy of 50.7%. We observed that the IPU length feature generally appears to offer slightly better performance compared to the turn length feature. Using all the contextual features together the best accuracy, 63.7%, is achieved by the Support Vector Machine classifier. This is significantly better than the baseline.

Various studies have observed that dialogue act history information is a significant cue for predicting a listener response when the speaker has just responded to the listener's request for clarification (Koiso et al., 1998; Cathcart et al., 2003; Gravano & Hirschberg, 2011; Skantze, 2012). Some of the rules learned by the J48 decision tree classifier suggest that this pattern is also observable in our Map Task corpus. One of the rules states: *if the last system dialogue act is Clarification or Guess (cf.* Table 4.1*) and the turn word count is less than or equal to 1, then* RESPOND. In other words, if the system had previously sought a clarification, and the user has responded with a yes/no utterance, then the system must respond to acknowledge. A more general rule in the decision tree suggests that: *if the last system dialogue act was a Restart or Repetition Request (cf.* Table 4.1*) and the turn word count is more than 4 then* RESPOND *otherwise* HOLD. In other words, having requested information from the user, the system should wait until it receives a certain *amount* of information from the user.

4.4.3 Lexico-syntactic features

As lexico-syntactic features, we used **word form** of the last two words in an IPU, and explored its performance in combination with other related features such as **part-of-speech (POS) tag, semantic tag,** and the **word-level confidence score** obtained for the speech-recognized user utterances. To obtain the word form feature values, all the IPUs in our Map Task corpus were manually transcribed. Filled pauses (e.g. *ehm*, *uh-hun*) were also orthographically transcribed and assigned the class tag FP. To obtain the POS tag information we used the Illinois Part-of-speech Tagger (Roth & Zelenko, 1998). In our training data we observed 23 unique POS tags. We added the FP tags to this set of POS tags. The five most frequent POS tag patterns for the last two words in IPUs corresponding to the RESPOND and HOLD categories are shown in Table 4.5. The differences in the phrase final POS tag patterns and their respective frequencies suggest that some POS tag patterns are strong cues for discriminating the RESPOND and HOLD type user IPUs.

Table 4.5: The five most frequent IPU phrase final POS tag patterns for the RESPOND and HOLD type classes

RESPOND			
POS tag pattern	***Count***	***Percent***	***Example***
DT NN	261	22.6%	*the church*
NN NN	160	13.9%	*grass field*
<s> UH	81	7.0%	*yes*
VB RB	79	6.9%	*walk south*
<s> NN	74	6.4%	*field*

HOLD			
POS tag pattern	***Count***	***Percent***	***Example***
PRP VBP	119	10.6%	*you go*
<s> FP	97	8.7%	*ehm*
DT NN	73	6.5%	*the garage*
<s> NN	58	5.3%	*hotel*
<s> RB	57	5.1%	*south*

The discriminatory power of the word form and part-of-speech tag features, corresponding to the four classifiers, is presented in Table 4.6. The figures under column sub-heading "Text" are accuracy scores achieved on feature values extracted from the manual transcriptions of the IPUs. Using only the last word a best accuracy of 83.9% was obtained by the SVM classifier. The addition of the penultimate word generally does not result in any further improvement. Using the last two words, the Voted Perceptron classifier achieved a best accuracy of 83.1%. The POS tag feature for the last two words in IPUs offer a best accuracy of 81.4% with the J48 classifier. While POS tag is a generic feature that would enable a model to generalize, using word form as a feature has the advantage that some words, such as *yeah*, are strong cues for predicting the RESPOND class, whereas fillers, such as *ehm*, are strong predictors of the Hold class.

Table 4.6: Percentage accuracy of lexico-syntactic features in detecting response locations.

		Algorithm							
#	***Feature(s)***	***J48***		***NB***		***SVM***		***VP***	
		Text	ASR	Text	ASR	Text	ASR	Text	ASR
1.	**Last word**	82.5	80.1	82.5	80.5	**83.9**	81.1	83.8	81.4
2.	**Last two words**	82.5	80.4	82.3	80.3	81.5	78.8	**83.1**	80.8
3.	**Last two word's POS tags**	**81.4**	76.5	80.3	75.8	80.5	75.3	81.1	76.3
4.	**Last two word's Semantic tags**	**83.4**	79.4	81.2	77.1	82.6	79.2	83.1	79.0
5.	**Last two words + ASR word confidence scores**	--	80.1	--	80.5	--	78.7	--	**80.8**
6.	**Last two words + Semantic tags + ASR word confidence scores**	--	81.8	--	81.0	--	79.0	--	**82.0**

4.4.4 Online use of lexico-syntactic features

A model for online prediction of response locations requires that the values of the lexico-syntactic features word form and POS tag are extracted from the output of a speech recognizer. Since speech recognition is prone to errors, a model trained on manual transcriptions alone—suggesting perfect recognition—would not be robust when making predictions on noisy data. For this reason, we trained and tested our models on actual speech recognized results. To this end, we did an 80-20 split of the Map Task corpus into training and test sets, respectively. The manual transcriptions of IPUs in the training set were used to train the language model of an off-the-shelf ASR system[8]. The trained ASR system was then used to recognize the audio recordings of the IPUs in the test set. After performing five iterations of splitting, training and testing, we had obtained speech-recognized results for all of the IPUs in the Map Task corpus. The mean Word Error Rate (WER) for the five iterations was 17.3% (*SD* = 4.45%).

The performances of the lexico-syntactic features word form and POS tag, extracted from the best speech recognized hypotheses for the IPUs, are presented under the columns with sub-heading "ASR" in Table 4.6. With the introduction of a word error rate of 17.3%, the performances of all the models using the feature word form slightly decline, as expected (cf. rows 1 and 2). However, in contrast to this, the corresponding decline in accuracy of the model using the POS tag feature alone is much larger (cf. rows 3). This is because the POS tagger itself uses the left context to make POS tag predictions. With the introduction of errors in the left context, the tagger's accuracy is affected, which in turn affects the accuracy of these models for detecting response locations. This performance is bound to decline even further with an increase in ASR errors. In order to identify the correlation between the ASR WER and the performance of our models, we first obtained five different ASR performances by using increasingly smaller training sets in the iterative process described above.

The performance of the J48 classifier using word form and POS tag features compared to these five ASR performances is illustrated in Figure 4.3. The results corresponding to 0 on the WER axis reflect the classifier's performance on feature values extracted from manual transcriptions. We can observe that while the performance of the model declines with an increase in WER, word form as a feature offers

[8] Due to licencing terms we cannot disclose the name of the ASR system.

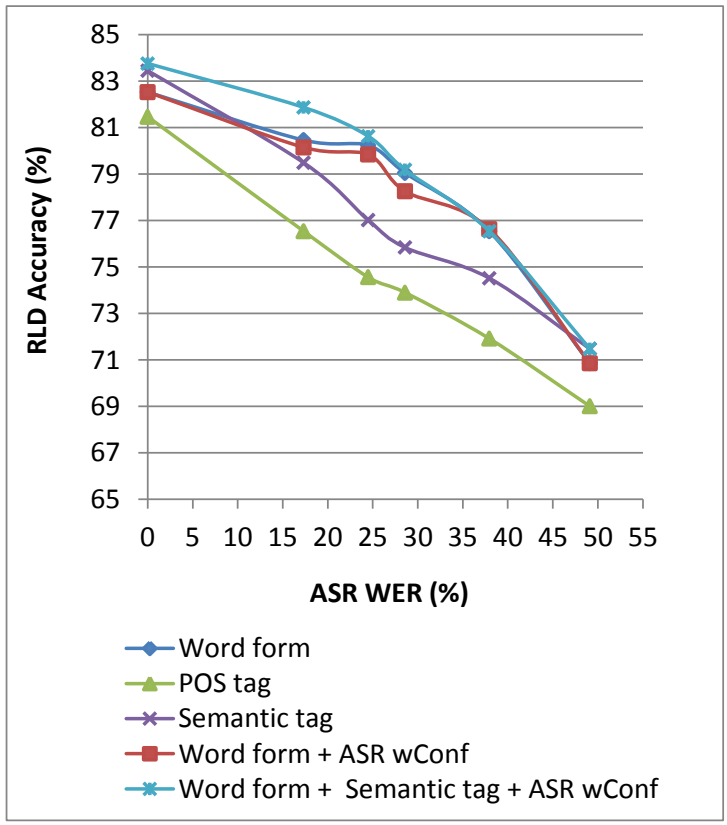


Figure 4.3: Performance of lexico-syntactic features using the J48 classifier, as a function of ASR WER.

constantly better performance in contrast to using the POS tag feature alone. This suggests that using context-independent lexico-syntactic features would offer better performance for an online model for response location detection. We therefore created a word class dictionary, which generalizes words into domain-specific **semantic tags** or classes in a simple way (much like a class-based n-gram model)**.** The semantic tags used in our dictionary were based on the domain ontology used in Chapter 3 for automatic semantic interpretation of verbally given route descriptions. For words that were not specific to this domain, their most frequent POS tag was used as a semantic tag. As a result, in our dictionary we had 12 classes, 4 of which were domain-specific (illustrated in Table 4.7), and the remaining 8 were POS tags.

Table 4.7: Domain specific semantic tags

Semantic tag	*Example words*
Landmark	building, train, station, garage, hotel, crossing
Direction	left, right, north, south, northeast, downwards
Action	take, turn
Spatial-Relation	after, from, at, before, front, around

The performance of the semantic tag feature for the last two words corresponding to the four classifiers is presented in row 4 in Table 4.6. The best accuracy using the semantic tag feature was achieved by the J48 classifier, 83.4% on manual transcriptions and 79.4% on ASR results. These figures are better than the corresponding performances of J48 classifier using the POS tag feature only, 81.4% and 76.5% respectively. The classifier performances on ASR results, suggest that using the semantic tag feature instead of POS tag (cf. row 3) generally improves the performance of the model for online predictions of response locations. This is also evident in Figure 4.3, where we observe that the semantic tag feature consistently performs better than the POS tag feature despite the increase in ASR errors. An additional advantage of using a semantic tag feature over word form is that new examples could be easily added to the model without having to retrain it. However, a model trained in this manner would be domain-specific.

We explored the **word-level confidence scores** (ASR wConf) from the ASR module as another lexico-syntactic feature that could possibly reinforce a classifier's confidence in trusting the recognized words. Using the word-level confidence score in combination with other lexico-syntactic features does not lead to any improvements over using word form alone (cf. Table 4.6 column "ASR", rows 2 vs. 4). The performance graph for this feature combination in Figure 4.3 also illustrates this observation.

The best accuracy for a model of response location detection, using lexico-syntactic features extracted from manual transcriptions, is achieved by the Voted Perceptron classifier (84.4%, using the features word form and semantic tag). For an *online* model of RLD, the best performance, 82.0%, is again achieved by the Voted Perceptron classifier using the features word form, semantic tag, and ASR word-level confidence score.

4.4.5 Combined models

We explored the performance of various model combinations among the three feature categories. Table 4.8 shows the contribution of these categories—individually as well as in combination—in discriminating RESPOND and HOLD type user IPUs. The top three rows show the best individual performances of the three categories. Since the prosodic and contextual features that we have used are independent of ASR WER, the performances of these features categories remain unaffected despite the introduction of an ASR WER of 17.32% (sub-column "ASR" in Table 4.8). For the models using lexico-syntactic features (Lex-Syntax), the figures in the "Text" column gave the results excluding the word-level confidence score feature, as it is not available for manual transcriptions.

Table 4.8: Percentage accuracy of combined models [* figures under column "Text" exclude the word-level confidence score feature]

		J48		***NB***		***SVM***		***VP***	
#	***Feature(s)***	***Text***	***ASR***	***Text***	***ASR***	***Text***	***ASR***	***Text***	***ASR***
1.	Prosody	65.7	65.7	66.3	66.3	**66.9**	**66.9**	62.8	62.8
2.	Context	62.5	62.5	59.7	59.7	63.7	63.7	**62.8**	**62.8**
3.	Lex-Syntax*	83.7	81.8	84.1	81.0	82.5	79.0	**84.4**	**82.0**
4.	Prosody + Context	68.1	68.1	69.2	69.2	**69.8**	**69.8**	66.9	66.9
5.	Prosody + Lex-Syntax*	84.1	81.6	**84.6**	**81.7**	80.5	77.5	80.8	78.3
6.	Prosody + Context + Lex-Syntax*	**84.2**	**82.4**	84.2	82.0	80.7	78.2	78.9	78.2

All the three feature categories offer significantly improved performance over the baseline of 50.7%. Model performances achieved through combining prosodic and contextual features exhibit improvement over using feature categories individually (cf. row 4, Table 4.8). The best accuracy for a model using context and prosody in combination, 69.8%, is achieved by the SVM learner. Lexico-syntactic features alone provide a large improvement over prosody and context categories. Using prosody in combination with lexico-syntax, the Naïve Bayes model achieves the best

accuracies—84.6% on feature values extracted from manual transcriptions and 81.7% on values extracted from ASR results. Using prosody, context and lexico-syntax in combination, the J48 classifier achieves the best accuracies, 84.2% on manual transcriptions and 82.4% on ASR output. These figures are significantly better than the majority class baseline of 50.7% and approach the expected upper bound—the inter-annotator agreement of 87.2% on HOLD and RESPOND classes in our Map Task corpus.

We performed significance tests on the performances of J48 classifier for some of the feature category combinations shown in Table 4.8. For these tests we used performance scores obtained from ten repetitions of 10-fold cross-validation using the Weka toolkit. Table 4.9 shows the results of these tests. Since the majority class baseline accuracies were not normally distributed, we applied the Mann-Whitney U test to compare the model performances with the baseline. For the remaining tests, a two-tailed t-test for independent samples was applied. The Mann-Whitney U test suggests that using prosodic features alone results in significant improvement over the majority class baseline of 50.7%. The t-test suggests that prosody achieves significantly better performance compared to using context alone. Using prosodic and contextual features in combination offers significant improvement in performance over using prosodic features alone. Lexico-syntactic features alone offer significant performance gain over using prosodic and context features together. Using prosody in addition to lexico-syntax does not offer any significant gains over using lexico-syntactic features alone. However, when context is used as an additional feature the resulting gain in performance is significant.

We also tested the performance of the J48, NB, SVM and VP classifiers against the majority class baseline, for significance. A Kruskal-Wallis H test indicates that one of the classifiers perform significantly better than one other classifier ($H(4) = 364.3$, $p < 0.001$). A pairwise (k-sample nonparametric) test indicates that all the four classifiers perform significantly better than the baseline ($p < 0.001$). Among the four learners, all learner performances except for those of pairs SVM and Voter Perceptron ($p = 0.393$), and Naïve Bayes and J48 ($p = 0.262$), are significantly different (with $p < 0.001$).

Table 4.9: Test of significance of various model combinations, with features extracted from ASR results and using the J48 classifier (< significant difference and ≅ no significant difference)

Significance tests for various feature category comparisons
Baseline < Prosody, Mann-Whitney U test (U = 0.000, $p < 0.001$)
Context < Prosody (t = -7.787, df = 198, p < 0.001)
Prosody < Prosody + Context (t = -5.338, df = 198, p < 0.001)
Prosody + Context < Lex-Syntax (t = -33.141, df = 198, p < 0.001)
Lex-Syntax ≅ Prosody + Lex-Syntax (t = -0.218, df = 198, p = 0.827)
Lex-Syntax < Prosody + Context + Lex-Syntax (t = -2.628, df = 198, p = 0.009)

An ideal model would have a high precision and recall for both RESPOND and HOLD prediction classes Table 4.10 shows the precision (proportion of correct decisions in all model decisions), recall (proportion of all relevant decisions correctly made) and F-measures for the baseline model and the J48 classifiers, using all the three feature categories together, with feature values extracted from ASR results. The J48 classifier appears to balance both recall and precision, and has the highest F-values: 0.84 for RESPOND and 0.81 for HOLD.

Table 4.10: Precision (P), Recall (R) and F-measures (F) of the baseline and J48 models using prosodic, contextual and lexico-syntactic features, with values extracted from ASR results.

	ZeroR			***J48***		
Prediction class	***P***	***R***	***F***	***P***	***R***	***F***
RESPOND	0.51	1.0	0.67	0.80	0.88	**0.84**
HOLD	0.0	0.0	0.0	0.86	0.77	**0.81**

4.4.6 RLD model performances vs. ASR performances

While the t-test results in Table 4.9 suggest that the gain in model performance achieved by using prosodic and contextual features in addition to lexico-syntactic features is significant, it is worthwhile to investigate when and which contributions were made by these two feature categories in the model decisions. The role of prosodic features is

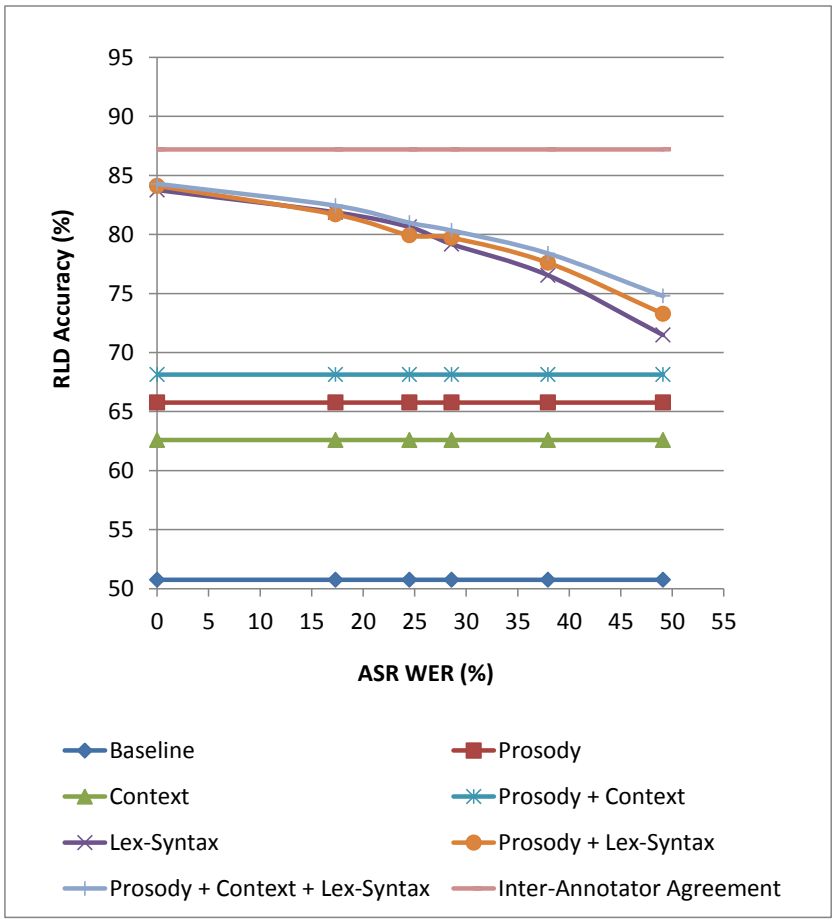


Figure 4.4: Performance of the J48 model of RLD as a function of ASR WER

emphasized when syntax alone cannot disambiguate (Koiso et al., 1998) or when errors in speech recognition impair the strength of lexico-syntactic features. Figure 4.4 shows the performances of the J48 classifier corresponding to various ASR WERs, and using various feature category combinations. As expected, the performance of the model using only lexico-syntactic features declines (linearly, R^2 = 0.91) with an increase in WER. In contrast, the models using only prosody or only context, or prosody and context in combination, are unaffected by an increase in ASR WER. Thus, prosody and context may provide features that are robust to noise, in contrast to lexico-syntactic features. This is demonstrated by the performance curve of the model combining prosodic, context, and lexico-

syntactic features. With a WER of 49.0%, the J48 classifier achieves an accuracy of 71.4%, using only lexico-syntactic features; however, together with the model combining prosodic and contextual features (an accuracy of 68.1%), the model achieves an accuracy of 74.7%—an absolute performance gain of 3.3%, which is substantial.

An example illustrates the role of prosodic features when lexico-syntactic features could be misleading. The user phrase "*pass through"* (also an IPU) was recognized as "*bus tunnel"* (against the ASR WER of 49.0%). As the user phrase is syntactically incomplete, the expected model decision is a HOLD. However, using the erroneous ASR output, which is syntactically complete, the lexico-syntactic model falsely identified the IPU as RESPOND type. The prosodic model, on the other hand, correctly classified the user phrase as HOLD type. The model using both lexico-syntactic and prosodic features correctly classified the user IPU as a HOLD type.

At times, the prosodic model led to incorrect model decisions as well. This is largely due to our simplistic method for extraction of prosodic feature values. For example, the user utterance "*then you will get to a corner where you have a church on your right*" was recognized as "*then the way yes the before way you eh the church all and garage.*" The expected model decision is to classify this user IPU as RESPOND type. The lexico-syntactic model correctly identified the user IPU as RESPOND type. However, the prosodic model incorrectly identified it as HOLD type. The model using both lexico-syntactic and prosodic features also falsely classified the IPU as HOLD type, suggesting that prosody may overrule syntax.

As regards the contribution of context to the model combinations, context plays a role when neither syntax nor prosody can disambiguate. As an illustration, the user utterance "*go south until you reach the pedestrian crossing*" was expected to be classified by the model as RESPOND type. The utterance was recognized as "*go south continue the church the the the station go see.*" The lexico-syntactic and the prosodic models incorrectly identified the ASR recognized user utterance as HOLD type. The model combining lexico-syntax and prosody also incorrectly identified it as HOLD type. However, the addition of contextual features to this model resulted in classifying the utterance as a RESPOND type. The IPU and turn length features contributed to this decision, suggesting that longer user turns should be followed by a system response. Two other instances where context made clear contributions are (a) the model decision to respond to acknowledge the user's response to a previously

asked clarification question, and (b) the model decision to wait (a HOLD) for some information from the user, after having requested the user to repeat the instructions (see the discussion on performance of contextual features in Section 0).

4.5 User evaluation

The best accuracy of 82.4% in discriminating user utterances as RESPOND and HOLD type, achieved by the J48 classifier using prosodic, contextual and lexico-syntactic features extracted from ASR results (with 17.3% WER), is significantly better than the majority class baseline performance of 50.7% (cf. Table 4.8). Would such a trained model also be perceived as significantly better in managing smooth interactions with real users? In order to evaluate the usefulness of a trained model in real user interactions we conducted a user evaluation at an early stage of this work. Two versions of the Map Task dialogue system that was used to collect the corpus (cf. Section 4.3) were created. One version used a Random model, which made a random choice between RESPOND and HOLD type when classifying the end of a user IPU as a response location. The Random model thus approximated our majority class baseline. Another version of the system used a Trained data-driven model to make the classification decision.

The Trained model used seven features in total: four prosodic features (mean pitch, pitch slope, pitch duration, and mean intensity), two contextual features (turn word count, and last system dialogue act), and two lexico-syntactic features (word form, and ASR word-level confidence score). A Naïve Bayes classifier trained on these seven features achieved an accuracy of 84.6% on manual transcriptions (excluding the feature word-level confidence score) and 82.0% on speech-recognized results. For both models, a decision was made whenever the system detected a silence of 200 ms. If the model decision was a Hold, the system waited 2 seconds and then responded anyway if no more speech was detected from the user. Figure 4.5 illustrates the components of the dialogue system using the Trained RLD model. In addition to the components used in the first iteration of the system during data collection (cf. Figure 4.2), this system had three new components: an ASR module for online extraction of lexico-syntactic features, a prosodic analysis component for online extraction of prosodic features, and a module to retrieve dialogue act history.

We hypothesize that since the Random model makes random choices, it is likely to produce false-positive responses as well as false-negative responses in equal proportion. While false-positive responses would result in occasional overlaps and interruptions in interactions, false-negative responses would result in gaps, delayed responses or simultaneous starts during the interactions. The Trained model, in contrast, should produce fewer overlaps and gaps, which would provide for smoother interactions and enhanced user experience. Users interacting with both models should be able to tell the systems apart based on the appropriate timing of system responses.

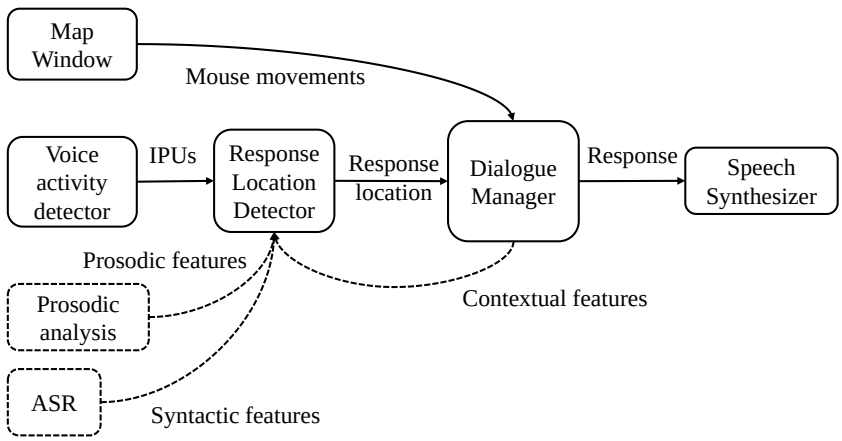


Figure 4.5: System architecture of the Map Task dialogue system (Iteration 2) used in user evaluation

In order to evaluate the two models, 8 subjects (2 female, 6 male) were asked to perform the Map Task with the two systems. Subjects were recruited from the school of Computer Science and Communication at KTH. Each subject performed five dialogues in total. This included one trial session with the system using the Trained model and two test sessions in which the subjects interacted with both versions of the system, one after the other. At the end of the first test session, subjects responded to a questionnaire asking them about which of the two systems responded at appropriate places (the timing), and which systems was more responsive. At the end of the second test session, subjects again provided feedback on the same questionnaire, but were asked to respond with the overall experience from both test sessions in mind. The purpose of the feedback questionnaire after the first test session was to prime subjects'

attention to the issues under evaluation. The trial session was used to allow the users to familiarize themselves with the dialogue system. Also, the audio recording of the users' speech from this session was used to normalize the user pitch and intensity for the online prosodic extraction. The order in which the systems and maps were presented to the subjects was varied over the subjects to avoid any ordering effect in the analysis.

We obtained 16 test dialogues each for the two systems. The 32 dialogues were, on average, 1.7 min long (*SD* = 0.5 min). The duration of the interactions with the Random and the Trained model were not significantly different. Both the models made about the same number of decisions: a total of 557 IPUs by the Random model and 544 IPUs by the Trained model. While the Trained model classified 57.7% of the IPUs as RESPOND type the Random model classified only 48.2% of the total IPUs as RESPOND type, suggesting that the Random model produced less responses.

It turned out that it was very hard for the subjects to perform the Map Task and at the same time make a valid subjective comparison between the two versions of the system. Some subjects reported that when responding to the questionnaire they were unable to recall their impressions of the first system they had interacted with, or their experience from the first test session at the end of second test session. From their oral feedback, we had the impression that at least some subjects based their judgements on the *type* of the response and not on *response timing*. For these reasons, we did not analyze the user feedback any further, and instead conducted another subjective evaluation to compare the two systems. We asked subjects to listen to the Map Task user interactions from the user evaluation and press a key whenever a system response was either lacking or inappropriate. The subjects were asked not to consider *how* the system actually responded, but only to evaluate the timing of the response.

4.5.1 Perception test

Eight users participated in the subjective judgment task. Although five of these were from the same set of users who had performed the Map Task in the user evaluation, none of them judged their own interactions. Our decision to not ask the participants to judge their own interactions was in order to avoid subjects' bias towards their own performances. The judges listened to the Map Task interactions in the same order as the original interaction, including the trial session. Whereas it had been hard for the subjects who participated in the dialogues to characterize the two versions

of the system, almost all of the judges could clearly tell the two versions apart, without being told about the properties of the two versions. They indicated that the system with the Trained model provided for a smoother flow of dialogue compared to the system with the Random model.

A total of 149 key presses for the Random model and 62 key presses for the Trained model were obtained. Since the judges were asked to simply press a key, we did not have access to the information whether a key was pressed due to perceived inappropriate response location (false-positive) or absolute lack of a system response (false-negative). To obtain this information we analyzed all the turn-transition instances where judges had pressed the key. The timing of the IPUs was aligned with the timing of the judges' key presses in order to measure the number of IPUs that had been given inappropriate response decisions.

We found that 11 instances of key press could be attributed to the failure of the voice activity detector in detecting user speech immediately after a system response or during the 2 seconds (timeout) following a Hold decision. Although judges were instructed not to judge the system utterance on the basis of the type of the response, 4 key presses were identified against responses that we believe were at appropriate response locations, but with inappropriate response type. There were 4 instances of key press where the system had correctly detected a Hold, but the current position of mouse cursor on the destination landmark triggered a system response of type End (cf. Table 4.1), which was perceived as inappropriate by the judges. Two key press instances could not be associated with any IPU and no plausible reason could be identified as to why the key presses occurred.

We excluded these 21 instances from our analysis as it would be inappropriate to hold the response location detection models responsible for these decisions. A Chi-Squared one-variable test suggests that the proportion of key presses received by the two systems (Random: 141 and Trained: 49) are significantly different ($\chi^2 = 44.54, \mathrm{df} = 1, p < .001$). We can therefore conclude that the Trained model was perceived to have significantly fewer inappropriate turn-transition instances than the Random model.

Responding at an inappropriate time, could be ascribed to a false-positive (FP) decision by the model. Similarly, the lack of response from system when it is expected could be ascribed to a false-negative (FN) decision by the model. Figure 4.6 illustrates the distribution of the perceived FP and FN decisions made by the Random and the Trained models. A Chi-squared test of independence of categorical variables

suggests that there is insufficient evidence to conclude that the number of key presses for one of the two models is influenced by the judges' perception of FN or FP ($\chi^2 = 1.0, \mathrm{df} = 1, p = .317$). In fact, both of the models seem to have received key presses for FN and FP decisions in equal proportion.

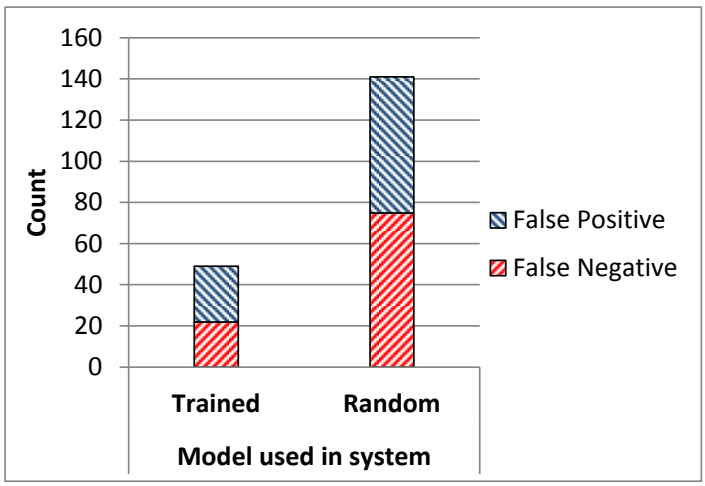


Figure 4.6: Distribution of perceived false-positive and false-negative model decisions

4.5.2 Response time

Responsiveness of a dialogue system has been identified as an important issue in turn-management that affects user experience (Ward et al., 2005). Delayed system responses, or lack of them, could be confusing for users. They have no idea whether the system is still processing their input and they should wait for system response, or whether the system has heard them at all, which may prompt them to repeat their utterances. Repeated user utterances may increase processing overhead for the system, which could result into even longer processing and response time. As mentioned earlier, it was difficult for the subjects who interacted with the Map Task system in the user evaluation to give proper feedback regarding the responsiveness of the systems. However, the judges in the perception test were able to perceive delayed system responses. While Figure 4.6 suggests that both systems received key presses for the FN and FP model decisions in equal proportion, we observed that the tendencies to press a key when the response did not appear in expected places (or was delayed) varies across judges. Our intuition is that some judges adapted to the

delay in system responses (on timeout after an actual FN model decision), and therefore didn't press the key, thereby causing the actual false-negative model decisions to be accounted as perceived true-positives. However, it would be preferable if the system could reply as quickly as possible. Therefore, we additionally compared the two system versions by measuring the response time for responses that were perceived as true positives.

Among the perceived true-positive system responses, a total of 267 for the Random and 312 for the Trained model, we identified two categories: responses which were produced timely (actual true-positive model decision) and those which were delayed (on timeout after an actual false-negative model decision). The mean response time for the timely responses was 301 ms ($SD = 2$ ms) whereas for the delayed responses it was 2324 ms ($SD = 4$ ms). These figures include the 200 ms silence threshold for triggering the response location detection task, and the additional 2 seconds of wait in case of FN decisions. Our system requires on average 100 ms for processing and decision making. Table 4.11 shows the distribution of timely and delayed responses for the Trained and the Random model. A Chi-Squared test suggests that the mean number of timely and delayed responses for the two models differ significantly ($\chi^2 = 27.844, \mathrm{df} = 1, p < .001$). We can therefore conclude that the Trained system has statistically significant faster response time than the Random system.

Table 4.11: Distribution of timely and delayed response for the Random and Trained model

Model (instances)	***Timely***	***Delayed***
Random (267)	78.7%	21.3%
Trained (312)	93.6%	6.4%

4.5.3 Precision, Recall and F-measure of the *live* models

An ideal data-driven model would have both high precision and high recall (i.e. a high F-measure). Table 4.12 shows that on the training dataset the precision, recall and F-measure of the Naïve Bayes model for online decision-making were significantly better than to the majority class baseline model. To measure whether similar performance was achieved by the model during the user evaluation (live usage) a performance

standard is needed. One way to obtain this is for the annotators who labelled the training data to annotate the interaction data from the user evaluation as well. However, such an evaluation would, at best, help confirm whether the models' live performance agreed with the annotators. An alternative—which is perhaps stronger and more viable—is to use the judges' key press feedback from the subjective evaluation as standards.

Table 4.12: Precision, recall and F-measure of the baseline and the NB models for online decision making

Prediction class	***Precision (%)***		***Recall (%)***		***F-measure (%)***	
	Baseline	***NB***	***Baseline***	***NB***	***Baseline***	***NB***
RESPOND	50.0	81.0	100.0	87.0	66.6	83.8
HOLD	0.0	85.0	0.0	78.0	0.0	81.3

In Section 4.5.1, we used judges' key presses to identify perceived false-positive and false-negative system decisions. We considered the remaining instances of system responses as True-Positive (TP) and system holds as True-Negative (TN). Using the counts of FP, FN, TP and TN we obtained the *perceived* precision, recall and F-measure scores for the two models, as shown in Table 4.13. When compared with Table 4.12 these figures confirm that during the real user interactions as well the Trained model achieved better recall, precision and F-measure for both RESPOND and HOLD type decisions, compared to the Random model.

Table 4.13: The *perceived* precision, recall and F-measure of the Random and NB models in live user interactions

Prediction class	***Precision (%)***		***Recall (%)***		***F-measure (%)***	
	Random	***NB***	***Random***	***NB***	***Random***	***NB***
RESPOND	75.4	91.4	73.2	92.8	74.3	92.1
HOLD	74.2	90.4	76.3	88.5	75.2	89.4

4.6 Conclusion and Discussion

We have presented a data-driven approach for detecting suitable response locations in user speech. We used human–computer interaction data that we collected using a fully automated dialogue system that can perform the Map Task with the user (albeit using a trick). Two annotators labelled all the user inter-pausal units (IPUs) with regard to whether a system response is appropriate at the end of each unit. We used automatically extractable features—covering prosody, context, and lexico-syntax—and trained various models (generative as well as discriminative) for online detection of feedback response locations. We tested the performances of these three feature categories, individually as well as in combination, for the task of response location detection.

To evaluate the contributions of such a trained model in real interactions with users, we integrated a trained model in the same dialogue system that was used to collect the training data, and tested it in user interactions. The results from a perception test show that the trained model exhibited significantly fewer instances of inappropriate turn-transitions compared to a baseline model. We also found that the trained model is significantly better at being responsive in comparison to the baseline system. To our knowledge, this is the first work on actual verification of the contributions of the type of models proposed in the literature for modelling human-like turn-taking and backchanneling behavior in dialogue systems. Our findings confirm that a model for turn-taking trained on prosodic, contextual and lexico-syntactic features offers both smooth turn-transitions and responsive system behavior.

Our work differs from earlier work in many regards. First, in contrast to the traditional procedure of using human–human interaction data to model human-like turn-taking behavior in dialogue systems, we used human–computer interaction data that we collected using a fully automated dialogue system. Second, most earlier work presented model performances on manual transcriptions only (Ward, 1996; Koiso et al., 1998; Cathcart et al., 2003; Gravano & Hirschberg, 2011; Morency et al., 2010). In contrast, we trained and demonstrated model performances based on automatically extractable features for online detection of response locations. Third, we explored the contributions of features pertaining to prosody, context, and lexico-syntax, whereas earlier works used features comprising only prosody (Ward, 1996), or combinations of lexico-syntax and prosody (Koiso et al., 1998), lexico-syntax and context (Cathcart et al., 2003), lexico-syntax, semantic and context (Bell et al.,

2001), prosody and context (Skantze, 2012), or prosody, context, and semantics (Raux & Eskenazi, 2008), or gaze, prosody and lexico-syntax (Morency et al., 2010), and pause and gaze (Huang et al., 2011). Fourth, very few of the models proposed earlier have been tested in user interactions. An exception is Raux & Eskenazi (2008), who trained models for online endpointing of user turns on human–computer interaction data. However, they excluded prosodic features from the online model used for live user interactions. Also, to evaluate model performance, Raux & Eskenazi (2008) used latency as the objective measure, but no subjective evaluation was carried out. We have evaluated the usefulness of our data-driven approach by integrating a trained model in a dialogue system and testing it in real user interactions. We have presented both subjective as well as objectives metrics for evaluating the improvements achieved by the trained model in contrast to a baseline model. In another work, Huang et al. (2011) trained a data-driven model (using pause and gaze information) for production of backchannel head-nods, and evaluated it in user interactions. Their model, however, does not use word-level information for predictions.

We tested the contribution of prosodic, contextual and lexico-syntactic feature categories, individually as well in combination, for online detection of response locations. Using the contextual features alone, the Voted Perceptron (VP) classifier achieved an accuracy of 65.2 %. This is a significant improvement over a majority class baseline of 50.7% in our data. Using the prosodic feature values extracted by our methods, a SVM classifier achieved an accuracy of 66.9%. While this performance is comparable to that achieved from using context alone, using prosody and context in combination offered the best accuracy of 69.5% using the Voted Perceptron classifier. This is again a significant improvement over the individual performances of the two feature categories. Using the lexico-syntactic features alone the best accuracy of 80.0% was achieved by the VP classifier taking into account an ASR WER of 17.3%. A model using prosodic, contextual, and lexico-syntactic features in combination achieved the best accuracy of 81.9% using the VP classifier. This is again significantly better than the majority class baseline of 50.7% and approached the expected upper bound—the inter-annotator agreement of 87.2% on Hold and Respond types in our Map Task corpus.

We found that lexico-syntactic features make a strong contribution to the combined model. This is consistent with earlier observations about their significant contribution in predicting turn-transition and backchannel relevant places (Koiso et al., 1998; Cathcart et al., 2003; Gravano &

Hirschberg, 2011). It has also been observed in these studies that POS tag alone is a strong generic lexico-syntactic feature for making predictions on manual transcriptions. However, our results show that the contribution of the POS tag is reduced when using it online in dialogue systems, due to errors in speech recognition. This is because the POS tagger itself uses the left context to make predictions, and is not typically trained to handle noisy input. We have shown that using context-independent lexico-syntactic features, such as word form or semantic tag (which generalizes the words into domain-specific semantic classes in a simple way, much like a class-based n-gram model) offers better and more robust performance despite speech recognition errors. However, this of course results in a more domain-dependent model.

Koiso et al. (1998) reported that prosodic features contribute almost as strongly to response location detection as lexico-syntactic features. We do not find such results in our data. This difference could be partly attributed to inter-speaker variation in our training data. All the users who participated in the collection of human–computer Map Task interaction data were non-native speakers of English. Also, our algorithms for extracting prosodic features are not as powerful as the manual extraction scheme used in Koiso et al. (1998). Although prosodic and contextual features do not seem to improve performance very much when lexico-syntactic features are available, they are clearly useful when no ASR is available (best accuracy of 69.8% as compared to the baseline of 50.7%, cf. Table 4.8) or when ASR performance is poor. As ASR WER approached 50% the performance of lexico-syntactic features approached the performance of the model combining both prosodic and contextual features (cf. Figure 4.4).

Koiso et al. (1998), Cathcart et al. (2003), and Skantze (2012) have observed that the last speaker dialogue act is useful for predicting a listener feedback response to acknowledge the speaker's response to a previously asked listener clarification request. One of the rules learned by the J48 decision tree classifier on our human–computer interaction data corroborates this observation. We also found that inclusion of additional contextual features, such as turn and IPU duration, enhances the model's discriminatory power when syntax and prosody cannot disambiguate.

During the user evaluation users found it difficult to recall their experiences and make subjective comparison based on only the *timing* of system responses and not the response *type*. However, when these users listened to other users' interactions, they were able to easily tell the two systems apart without being told about the properties of the two versions.

Thus the perception test of user interactions helped us identify instances of perceivable incorrect model decisions. The results show that the Trained model produced significantly fewer instances of inappropriate turn-transitions in contrast to the Random model.

Obtaining subjective feedback from real users is nonetheless vital for complete evaluation of dialogue system behavior. One possibility for evaluation in future studies could be to use the method of collecting users' expectations and experiences, first presented in Jokinen & Hurtig (2006), and used in Meena et al. (2012). In this scheme, subjects first fill out a questionnaire that has dual purpose: (a) to obtain a measure of users' expectations (on a Likert scale) regarding the system behavior(s) under investigation; and (b) to prime users' attention to these behavioral aspects so that the users are conscious about what to evaluate. Next the users interact with one version of the system, following which they fill the same questionnaire again. However, this time they are asked to provide feedback on their experience. The user repeats this step for the remaining system versions. The users' feedback from the questionnaires—on what they expect and what they actually experienced—could be then used to draw conclusions as to which system was perceived as being closer to their expectations.

Besides the issues with coordination of speaking turns, the responsiveness of a system is crucial for keeping the users engaged. We observed that in contrast to the Random model, the Trained model has a significantly larger number of responses that were produced as timely as possible, and significantly lower number of responses that were delayed. These results bring us to the conclusion that a model for feedback response location detection trained on features covering prosody, context, and lexico-syntax offers a dialogue system that is both more responsive and interrupts the user less often.

4.7 Directions for future work

While one of our trained models has been shown to enhance the turn-management performance of the Map Task dialogue-system, there is still room for further improvement. In the current model, lexico-syntactic features appear to dominate the decision making. This is evident from the respective performances of these feature categories, as well as from user comments that the system appeared to respond well when they mentioned a landmark on the map. Our observation is that if the model could better exploit prosodic cues, it could enhance the system's responsiveness to

other subtle, yet general behavioral cues present in users' speech. In this work we have mainly used pitch- and intensity-related prosodic features that we could extract automatically for online use. Also, the algorithms used for extracting these feature values are very simple, and perhaps using better extraction methods would improve the model's performance.

Other prosodic features such as intonation patterns, speaking rate, and acoustic cues such as jitter, shimmer and noise to harmonic ratio have also been identified as useful behavioral cues for prediction of turn-taking and backchannel relevant places (Koiso et al., 1998; Gravano & Hirschberg, 2011). As tools for online extraction of these cues become available, they could easily be incorporated in the current models. Thus, the dataset from this experiment could serve as a test for evaluating the applied usefulness of new models for prosodic analysis. To this end, we have released the complete dataset, both the training set and the data from user evaluation, for public use[9].

A general limitation of data-driven approaches is sparseness in training data. Due to the limited number of tasks in the Map Task user interactions, it is plausible that users share a common lexical space. However, due to inter-speaker variations in prosodic realization and usage, the prosodic space is sparse, which makes it difficult for the models to generalize across users. It would be interesting to explore algorithms that are better at generalizing across speakers.

We have so far explored prosodic, contextual, and lexico-syntactic features for predicting response location. An immediate extension to our model would be to bring more general syntactic features in the model. The motivation for this is syntactically incomplete user utterances such as "*and after the hotel…*" Our current model using lexico-syntactic features only, on observing the "DT NN" phrase final pattern, would predict a RESPOND. However, syntactic knowledge suggests that the predicate is missing in the user utterance, and therefore the system should predict a HOLD.

In a future version of the system, we do not only want to determine *when* to give responses but also *what* to respond. This would require processing at a higher level, more specifically understanding the semantics of spoken route descriptions, to play an active role in decision making. In Chapter 3, we presented a data-driven method for automatic

[9] The KTH Human–Computer Map Task corpus, available at http://www.speech.kth.se/maptask/index.html.

semantic interpretation of verbal route descriptions into *conceptual route graphs* (CRG)—a semantic representation that captures the way humans structure information in route descriptions. A CRG with missing concepts or relations should suggest incompleteness, and therefore a HOLD. However, the incompleteness could also be due to ASR misrecognitions. Perhaps the confidence scores from the ASR and the spoken language understanding component could be used to identify *what* to respond, and also to select between different forms of clarification requests and acknowledgements.

We would also like to test whether our models for response location detection will generalize to other domains. While both context and prosody offer features that are domain independent, POS tag would still be a more suitable lexico-syntactic feature (in contrast to word form and semantic tag features) for a domain independent model of RLD.

Another possible extension is to situate the Map Task interaction in a face-to-face Map Task between a human and a robot (similar to Skantze et al., 2013a) and add features from speaker's gaze, which has been identified as a visual cue that humans use to coordinate turn-taking in face-to-face interactions (Mutlu et al., 2009; Morency et al., 2010; Skantze et al., 2013b; Johansson & Skantze, 2015).

Chapter 5

Error Detection

5.1 Introduction

Problems in communication due to misunderstanding or non-understanding are frequent phenomena in both human–human and human–computer interactions. However, while human conversational partners are skilled at detecting and resolving problems, state-of-the-art dialogue systems often have problems with this. Various works have been reported on the detection of errors in human–computer dialogues. While the common theme among these works is to use error detection for making online adaption of dialogue strategies (e.g., implicit vs. explicit confirmations), they differ in what they model as error. For example, Litman et al. (1999) and Hirschberg et al. (2004) model problems in speech recognition as error. Walker et al. (2000) and Bohus & Rudnicky (2002) refer to incorrect understanding of user intentions as error. Walker et al. (2002) model failure in task completion as error. Schmitt et al. (2011) use the notion of interaction quality in a dialogue as an estimate of errors at arbitrary points in a dialogue. Krahmer et al. (2001) , Swerts et al. (2000), Bohus (2007) and Skantze (2007) model misunderstandings on the system's part as errors.

Awareness about errors in dialogues, however, has relevance not only for making online decisions, but also for off-line analysis, for example by dialogue system designers. Access to information about in which states the dialogue fails or runs into trouble could enable system designers to identify potential flaws in the dialogue design. Unfortunately, this type of error analysis is typically done manually, which is laborious and time consuming. Automation of this task is highly relevant for dialogue system developers in both industry and academia.

In this chapter, we present a data-driven approach for the detection of miscommunication in dialogue system interactions through automatic analysis of system logs. This analysis is based on the assumption that the onus of miscommunication is on the system. Thus, instances of non-understandings, implicit and explicit confirmations based on false assumptions, and confusing prompts are treated as problematic system

actions that we want to detect in order to avoid them. Since the proposed approach is targeted at offline analysis of interaction logs, we focus here largely on models for offline error detection. For this analysis, we have the full dialogue context (backward and forward at any stage in the dialogue) at our disposal, and use features that are both automatically extractable from the system logs and manually annotated. However, we also report the performance of these models using only online features and limited dialogue context, and demonstrate our models' suitability for online use in detection of potential problems in system actions.

We evaluate our approach on datasets from three different dialogue systems that vary in their dialogue modeling, dialogue strategy, language, user types, etc. Such an evaluation enables us to assess the applicability and generalizability of the features investigated here. We also report findings from an experimental work on cross-corpus analysis: using a model trained on logs from one system for analysis of interaction logs from another system. Such an analysis will show us if the proposed models can be directly applied for error detection on system logs of another system, without incurring additional costs, such as manual annotations. Finally, we present preliminary results from a study on root cause analysis, i.e., identifying the causes of problems in system behavior. Thus, the novelty of the work reported here lies in our models' relevance for offline as well as online detection of miscommunications, the applicability and generalizability of features across dialogue systems and domains, and the approach to root cause analysis.

This chapter is structured as follows: we start with a review of relevant works on error detection in Section 5.2. Following this, in Section 5.3, we explain our approach. In Section 5.4, we describe the three dialogue system corpora used for evaluation of our approach and the schemes for annotating the data for error detection. The complete set of features, algorithms, and the various models for error detection explored in this work are presented in Section 5.5. The performance of the trained models on the three datasets and cross-corpus is presented in Section 5.6. In Section 5.7, we present an approach to use the knowledge about errors to identify the root causes. We conclude and discuss directions for furthering this work in Section 5.8.

5.2 Background

As discussed earlier (in Section 2.5.3), one way to analyze miscommunication is to make a distinction between *non-understanding*

and *misunderstanding* (Hirst et al., 1994). In human–computer dialogue, non-understanding errors arise due to failure in obtaining any hypothesis during language understanding. On the other hand, misunderstanding errors happen when an incorrect interpretation is accepted by the system. This can be illustrated with the human–computer interaction in Table 5.1. The system, in turn 3, expresses that a non-understanding of user intentions (in turn 2) has occurred. In contrast, in turn 5—following the best assessment of user turn 4—the system makes a restaurant recommendation, but misunderstands the user's choice of cuisine. However, this problem does not become evident until turn 6. Thus, while non-understandings are noticed immediately by the listeners, the information about misunderstandings may surface only at a later stage in the dialogue.

Table 5.1: An illustration of miscommunication in human–computer interaction. S and U denote system and user turns, respectively. User turns have been manually transcribed.

1.	S:	*How may I help you?*
2.	U:	Can you recommend a Turkish restaurant in downtown area?
3.	S:	*Could you please rephrase that?*
4.	U:	A Turkish restaurant in downtown.
5.	S:	*Clowns, which serves Italian food, is a great restaurant in downtown area.*
6.	U:	I am looking for a Turkish restaurant

The various approaches to detection of errors presented in the literature can be broadly classified in two categories—*early error detection* and *late error detection*—based on at what turns in the dialogue the assessments about errors are made (Skantze, 2007). In early error detection approaches, the system makes an assessment of its current hypothesis of what the user has just said. Approaches for detection of non-understanding, such as confidence annotation (Bohus & Rudnicky, 2002), fall in this category. In the example above, the system determines that it does not have a good enough hypothesis for turn 2, but that it is confident enough to assume that the user is looking for an Italian restaurant after turn 4. In contrast, late error detection aims at finding out whether the system has made false assumptions about the user's intentions in previous

turns. An example of this would be if the system uses turn 6 in the example above to detect the previous misunderstanding. These distinctions are vital from our viewpoint as they point out the turns in dialogue that are to be assessed and the scope of dialogue context that could be exploited to make such an assessment.

We now present some of the related works and highlight what has been modeled as error, at what stage in the dialogue the assessment about errors are made, what types of features are used and what span of dialogue context is taken into account. Following this, in Section 5.3, we discuss the motivations and distinct contributions of our work.

Walker et al. (2002) presented a corpus based approach that used information from initial system-user turn exchanges alone to forecast whether the ongoing dialogue would fail. If the dialogue was likely to fail the call could be transferred to a human operator right away. A rule learner, RIPPER (Cohen, 1995), was trained to make a forecast about dialogue failure after every user turn. The model was trained on automatically extracted features from automatic speech recognizer (ASR), natural language understanding (NLU) and dialogue management (DM) modules.

Bohus & Rudnicky (2002) presented an approach to utterance level *confidence annotation* which aims at making an estimate of the system's understanding of the user's utterance. The model returns a confidence score which is then used by the system to select appropriate dialogue strategy, e.g. express non-understanding of user intention. The approach combines features from ASR, NLU and DM for determining the confidence score using logistic regression.

Schmitt et al. (2011) proposed a scheme to model and predict the quality of interaction at arbitrary points during an interaction. The task for the trained model was to predict a score, from 5 to 1 indicating very high to very poor quality of interaction, on having seen a system-user turn exchange. A Support Vector Machine model was trained on automatically extractable features from ASR, NLU and DM modules. They observed that additional information such as user's affect state (manually annotated) did not help the learning task.

In their investigations of a Dutch Train timetable corpus, Krahmer et al. (2001) observed that dialogue system users provided positive and negative cues about misunderstandings on the system's part. These cues include user feedback, such as corrections, confirmations, and marked disconfirmations, and can be exploited for late error detection.

Swerts et al. (2000) trained models for automatic prediction of user corrections. They observed that user repetition (or re-phrasing) is a cue to a prior error made by the system. They used prosodic features and details from the ASR and the DM modules to train the RIPPER learner to predict whether a user response is a correction. Their work highlights that user repetitions are useful cues for late error detection.

5.3 Approach

For our task, we have defined the problem as detecting miscommunication on the system's part. This could be misunderstandings, implicit and explicit confirmations based on false assumptions, or confusing system prompts. Since instances of non-understanding are self-evident cases of miscommunication for the system, we exclude them from the learning task. Detecting the other cases of miscommunication is non-trivial as it requires assessment of user responses in the context of the dialogue. In the rest of this text the term "error detection" is used to refer to these sets of problems. The proposed scheme can be illustrated in the example interaction in Table 5.2.

Table 5.2: An implicit confirmation based on a false assumption is an instance of a problematic system action. User turns are manual transcriptions.

1.	S:	*How may I help you?*
2.	U:	Sixty One D
3.	S:	*The 61C.What's the departure station?*
4.	U:	No

In the context of these four turns our task is to detect whether system turn 3 is problematic. If we want to use the model online for early error detection, the system should be able to detect the problem using only automatically extractable features from turns 1–3. Unlike *confidence annotation* (Bohus & Rudnicky, 2002), we also include what the system is about to say in turn 3 and make an anticipation (or forecast) of whether this turn will lead to a problem. Thus, it is possible for a system that has access to such a model to assess different alternative responses before choosing one of them. Besides using details from ASR and SLU components (exploited in the reported literature) the proposed *early model* is able to use details from Dialogue Manager and Natural Language Generation modules.

Next, we train another model that extends the anticipation model by also considering the user feedback in turn 4, similar to Krahmer et al. (2001) and Swerts et al. (2000). Such a model can also be used online in a dialogue system in order to detect errors after-the-fact, and engage in late error recovery (Skantze, 2007). The end result is a model that combines both anticipation and user feedback to make an assessment of whether system turns were problematic. We refer to this model as the *late model*.

Since both the early and late models are to be used online, they only have access to automatically extractable features. However, we also train an *offline model* that can be used by a dialogue designer to find potential flaws in the system. This model extends the late model in that it also has access to features that are derived from manual annotations in the logs.

In this work, we also investigated whether models trained on logs of one system can be used for finding problems in interaction logs from a different dialogue system. To achieve this, we will train the offline model using only features that are generic across application domains.

5.4 Datasets

Dialogue system logs from two publicly available corpora and one from a commercially deployed system were used for building and evaluating the three models. The first dataset is from the CamInfo Evaluation Dialogues corpus. The corpus comprises of spoken interactions between the Cambridge Spoken Dialogue System and users, where the system provides restaurant recommendations for Cambridge. The dialogue system is a research system that uses dialogue-state tracking for dialogue management (Jurcicek et al., 2012). As the system is a research prototype, users of these systems are not real users in real need of information but workers recruited via the Amazon Mechanical Turk[10] (AMT). Nevertheless, the dialogue system is state-of-the-art in statistical models for dialogue management. From this corpus 179 dialogues were used as the dataset, which we will refer to as the **CamInfo** set.

The second corpus comes from the **Let's Go** dialogue system. Let's Go (Raux et al., 2005) is developed and maintained by the Dialogue Research Center (DialRC) at Carnegie Mellon University. It provides bus schedule information for Pittsburgh's Port Authority buses during off-peak hours. While the system is a research prototype, the users of Let's Go system are real users that are in real need of the information. This makes the dataset

[10] Amazon Mechanical Turk is a crowdsourcing internet marketplace for work https://www.mturk.com/mturk/welcome

interesting for us. The dataset used here consists of 41 dialogues selected from the data released for the 2010 Spoken Dialogue Challenge (Black et al., 2010).

The third dataset, **SweCC**—Swedish Call Center Corpus—is taken from a corpus of call logs from a commercial customer service provider in Sweden providing services in various domains. The system tries to extract some details from customers before routing the call to a human operator in the concerned department. Compared to CamInfo and Let's Go datasets, the SweCC corpus is from a commercially deployed system with real users, and the interactions are in Swedish. From this corpus 219 dialogues were selected. Table 5.3 provides a comparative summary of the three datasets.

Table 5.3: A comparative summary of the three datasets

CamInfo	***Let's Go***	***SweCC***
Research	Research	Commercial
Hired users	Real users	Real users
User initiative	System initiative	System initiative
Mostly implicit confirmation	Mostly explicit confirmation	Only explicit confirmation
Stochastic DM	Rule based DM	Rule based DM
English	English	Swedish
179 dialogues	41 dialogues	219 dialogues
5.2 exchanges on average per dialogue	19 exchanges on average per dialogue	6.6 exchanges on average per dialogue

5.4.1 Annotations

We take a supervised approach to detect problematic system turns in the interaction logs. This requires each system turn in the training datasets to be labeled as PROBLEMATIC (if the system turn reveals a miscommunication in the dialogue) or NOT-PROBLEMATIC. There are different schemes for labeling data. One approach is to ask one or two experts (having knowledge of the task) to label data and use inter-annotator agreement to set an acceptable goal for the trained model. Another approach is to use a few non-experts but use a set of guidelines so that the annotators are consistent (and to achieve a higher Kappa score,

(Schmitt et al., 2011)). We took the crowdsourcing[11] approach for annotating the CamInfo data and used the AMT platform. Thus, we avoided using both experts and guidelines. The key, however, was to make the task simple for the AMT-workers. Based on our earlier discussion (in Section 5.3) on the role of dialogue context and type of errors assessed in early and late error detection approaches, we set up the annotation tasks such that AMT-workers saw two dialogue exchanges (4 turns in total), as shown in Table 5.2. The workers were asked to label system turn 3 as PROBLEMATIC or NOT-PROBLEMATIC, depending on whether it was appropriate or not, in the context of these four turns. They were asked to use the label PARTIALLY-PROBLEMATIC when it was not straightforward to choose between the former two labels.

In the Let's Go dataset, we observed that whenever the system engaged in consecutive confirmation requests, the automatically extracted sub-dialogue (any four consecutive turns) did not always result in a sub-dialogue with a meaningful context. Therefore the Let's Go data was annotated by a human expert. The SweCC data could not be used on the AMT platform due to the agreement with the data provider, and was annotated by the same human expert. Table 5.8 and Table 5.9 illustrate sample interactions and annotations from the CamInfo and the Let's Go corpus, respectively.

Since the interaction logs in the CamInfo Evaluation Dialogues corpus contained the user feedback to the questionnaire asked during the evaluation, we investigated whether the problematic turns identified by the AMT-workers reflected the overall interaction quality, as experienced by the users. We observed a visibly strong correlation between the user feedback and the fractions of system turns per dialogue labelled as PROBLEMATIC by the AMT-workers. Figure 5.1 illustrates the correlation for one of the four questions in the questionnaire. This shows that the detection and avoidance of problematic turns (as defined here), will have bearing on the users' experience of the interaction.

Each system turn in the CamInfo dataset was initially labeled by two AMT-workers. In case of a tie, one more worker was asked to label that instance. In total 753 instances were labeled in the first step. We observed

[11] Crowdsourcing is the process of obtaining information by soliciting contributions from a large group of non-experts, recruited via online community, rather than traditional employees.

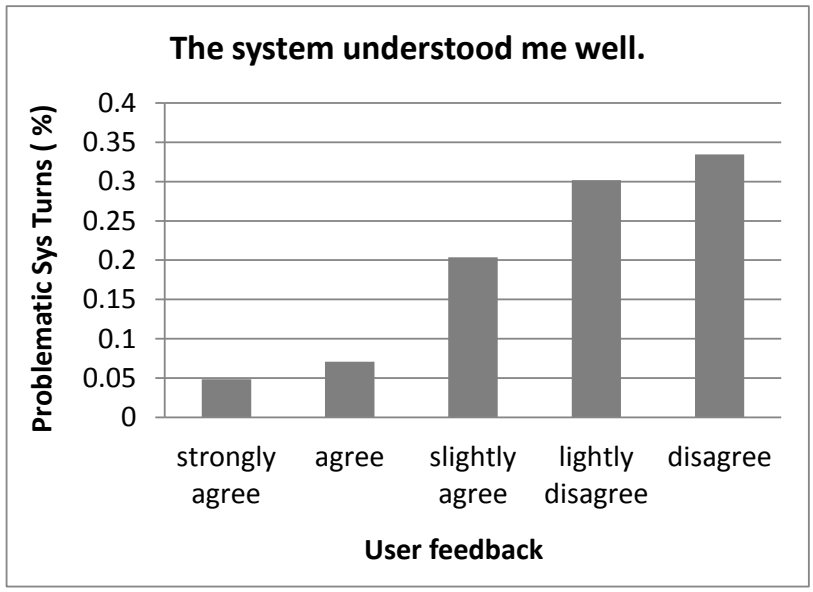


Figure 5.1: Correlation of system turns labelled as problematic with user feedback to the statement "The system understood me well."

an inter-annotators agreement of 0.80 (Fleiss's Kappa[12]) among the annotators. Only 113 instances had a tie and were annotated by a third worker. For these instances the label with the majority vote was chosen as the final class label. Table 5.4 shows the distributions for the three annotation categories seen in the three datasets. In general, for all the three datasets, the system actions have been mostly NOT-PROBLEMATIC. The rather low percentage of PARTIALLY-PROBLEMATIC class for the Let's Go dataset in comparison to the CamInfo dataset can be attributed to the fact that the Let's Go system mostly uses explicit confirmations that seek *yes*/*no* type responses. Assessing such system turns (turn 3 in the sub-dialogue) in conjunction with the user response (in turn 4) is a rather simple task for the AMT-annotators; hence most system actions were labelled as either PROBLEMATIC or NOT-PROBLEMATIC.

[12] When assessing the agreement between more than two raters, Fleiss's Kappa is used instead of Cohen's Kappa. In our crowdsourcing set-up we obtained two annotations for every instance, but the set of annotators was not restricted to any two specific AMT-workers; hence we have used the Fleiss's Kappa.

Table 5.4: Distribution of annotation categories for the three datasets

Dataset **(#instances)**	***CamInfo*** **(753)**	***Let's Go*** **(760)**	***SweCC*** **(968)**
PROBLEMATIC	16 %	42%	31%
NOT-PROBLEMATIC	73 %	57%	61%
PARTIALLY-PROBLEMATIC	11 %	1%	8%

Due to the imbalance of the PARTIALLY-PROBLEMATIC class in the three datasets we excluded this class from the learning task and focused only on classifying system turns as either PROBLEMATIC or NOT-PROBLEMATIC. System turns expressing non-understanding were also excluded from the learning task. The final datasets had the following representation for the NOT-PROBLEMATIC class: CamInfo (615) 86.0%, Let's Go (744) 57.5, and SweCC (871) 65.7%. To mitigate the high class imbalance in CamInfo, another 51 problematic dialogues (selected following the correlations of user feedback from Figure 5.1) were annotated by the author of this work. The resulting CamInfo dataset had 859 instances of which 75.3% were from the NOT-PROBLEMATIC class.

5.5 Features and Models

5.5.1 Features

We wanted to train models that are generic and can be used to analyze system logs from different dialogue systems. Therefore we trained our models on only those features that were available in all three datasets. Table 5.5 presents the various feature categories explored in this work. This includes features and manual annotations that were readily available in system logs. A range of higher-level features were also derived from the available features. Since all three dialogue systems are based on slot-filling, we use the term *concept* to refer to the slot-value pairs as well as the unmatched slot-types and slot-values, observed in the parse results.

Table 5.5: A comprehensive list of features used for error detection

A.	***Automatic Speech Recognition***
	1. Recognized hypothesis with the highest confidence score. A word-vector representation was used.

2. ASR confidence score
3. Number of words in the recognized hypothesis

B. Natural Language Understanding

4. Parse result obtained for the ASR recognized hypothesis (*nlu_asr*). A word-vector representation was used.
5. Dialogue act(s) for the user utterance. When n-best NLU hypotheses were available, parse result and dialogue act from the one with the highest NLU confidence score were selected.
6. Parse result obtained on the manual transcription of the user utterance (*nlu_trn*)
7. User dialogue act in *nlu_trn*
8. Number of concepts in *nlu_asr*
9. Number of concepts in *nlu_trn*
10. Concept error rate (CER): the Levenshtein distance between *nlu_trn* and *nlu_asr*
11. Correctly transferred concepts (CTC): the fraction of concepts in *nlu_trn* observed in *nlu_asr*

C. Natural Language Generation

12. System dialogue act(s)
13. Number of concepts in system act
14. System prompt (in a word-vector representation)
15. Number of words in the system prompt

D. Manual annotations

16. Manual transcriptions of the best ASR hypothesis (in a word-vector representation)
17. Number of words in the transcription
18. Word error rate (WER): the Levenshtein distance between the recognized hypothesis and the manual transcription
19. Correctly transferred words (CTW): fraction of words in the transcription observed in the ASR hypothesis

E. Discourse features

20. ***Position in dialogue***: fraction of turns completed up to the decision point
21. ***New information***: fraction of new information in the successive utterances of a speaker. This is measured using (a) words at utterance level and (b) concepts in the respective dialogue acts.
22. ***Repetition***: two measures for estimating repetition in successive turns of a speaker were used: (a) *cosine similarity*: the cosine angle

between vector representation of the two turns (words as well as concepts) and (b) simply the number of common concepts.

23. ***Marked disconfirmation***: whether the user response to a system clarification request (explicit or implicit) contains a marked disconfirmation, e.g., "no", "not" (or "nej" in Swedish).
24. ***Corrections***: the number of slots-values mentioned in previous system turn that were given a new value in the following user turn. When extracted from manual transcription this would indicate the extent of actual user corrections.
25. ***Misunderstanding***: the number of slots-values mentioned in previous user turn that were given a new value in the following system turn. When extracted from manual transcription this would indicate the extent of actual misunderstandings by the system.
26. ***Rectifications***: the number of slots-values mentioned in previous system turn that were given a new value in the following system turn.
27. ***Change-of-intention***: the number of slots-values mentioned in previous user turn that were given a new value in the following user turn. When extracted from manual transcription this would indicate the extent of actual change in user intentions.

5.5.2 Models and Algorithms

As mentioned earlier in Section5.3, we trained three models for error detection: the *early* and *late* models are aimed at online use in dialogue systems, whereas the *offline model* is to be used for offline analysis of interaction logs. As with the annotation setup, a window of 4 turns (cf. Table 5.2) is used to limit the dialogue context for extraction of features. Accordingly, the *early model* uses features from turns 1–3; the *late model* uses features from the complete window, i.e., turns 1–4. The *offline model*, like the *late model*, uses the complete window, but additionally uses the manual transcription features or features derived from them, e.g. word error rate.

In order to model the discourse context in these three models, we introduced variables for each turn. Thus, we used two variables for representing the ASR confidence scores of user turn 2 and 3. Similarly, features of string type (e.g., ASR hypothesis, NLU parse, system prompt) were represented as a word-vector representation. In this representation each token in the string (the feature-value) is treated as an independent variable. The dialogue context of the token is represented by prefixing the

Table 5.6: The four feature sets used for performance analysis of the three models.

#	*Feature set*	*Description*
1.	**Bag-of-word (BoW)**	A word-vector representation of system and user turns.
2.	**Derived-from-turns (DrT)**	A set containing all the features derived from the user utterances and system prompts. This includes, e.g., turn length (measured in number of words), cosine similarity in speaker turns as an estimate of speaker repetition, correctly transferred words, and word error rate.
3.	**Bag-of-concept (BoC)**	A word-vector representation of the system and user communicative intentions. For the system, this includes the dialogue acts and the concepts in the communicative goal. For the user, it includes the dialogue act and the concepts observed in the NLU parse hypothesis.
4.	**Derived-from-concepts (DrC)**	A set with all the features derived from dialogue acts and NLU parse results. This includes, e.g., turn length (measured in number of concepts), cosine similarity in NLU parses as an estimate of speaker repetition, correctly transferred concepts, and concept error rate.

token with the turn number in the window. For example, variables *2_busNum* and *3_busNum* represent two independent variables and the presence of the binary flag true for both these features in the word-vector captures the discourse context where the user turn (turn 2) contained the concept *busNum* and the following system turn (turn 3) also contained the concept *busNum*. The intuition behind such a representation is that it is able to capture the discourse event that the system was able to partly understand what the user said (cf. Table 5.2).

Table 5.5 provides a comprehensive list of the various feature categories and features used for training the models for error detection. However, the values for these features were extracted from multiple sources. For example, user repetition can be estimated on the user utterances, as recognized by ASR, as well as on the parse results obtained

by the NLU. Furthermore, as the offline model for error detection uses manual features, user repetition can also be estimated on the manual transcriptions of the user utterances as well as the corresponding parse results from the NLU. Thus, the actual list of features used for training the models is much larger and have not been presented here for the purpose of brevity. Instead, we group these features in four sets, namely *bag-of-words* (BoW), *derived-from-turns* (DrT), *bag-of-concepts* (BoC), and *derived-from-concepts* (DrC), and present the model performances using different feature set combinations. The description of the four feature sets is presented in Table 5.6. We will illustrate the performances of the three models of error detection using the feature sets BoW and BoC, and feature set combinations BoW+DrT and BoC+DrC+DrT. A complete list of actual features (87 of them) used in this work has been made available in Appendix A.1.

Given the skew in distribution of the two classes in the three datasets (cf. Section 5.4.1) accuracy alone is not a good evaluation metric. A model can achieve high classification accuracy simply by predicting the value of the majority class (i.e. NOT-PROBLEMATIC) for all predictions. However, since we are equally interested in the recall for both PROBLEMATIC and NOT-PROBLEMATIC classes, we use the ***un-weighted average recall*** (UAR) to assess the model performance, similar to Higashinaka et al. (2010) and Schmitt et al. (2011).

We explored various machine learning algorithms available in the Weka toolkit (Hall et al., 2009), but report here models trained using two different algorithms: JRip, a Weka implementation of the RIPPER rule learning algorithm, and Support Vector Machine (SVM) with linear kernel. The rules learned by JRip offer a simple insight into what features contribute in decision making. The SVM algorithm is capable of transforming the feature space into higher dimensions and learning sophisticated decision boundaries. The figures reported here are from a 10-fold cross-validation scheme for evaluation.

5.6 Results

5.6.1 Baseline

To assess the improvements made by the trained models we need a baseline model to draw comparisons. We can use the simple majority class baseline model that will predict the value of majority class for all predictions. The UAR for such a model is shown in (row 1) of Table 5.7. The UAR for all the three datasets using this model is 0.50.

Table 5.7: Un-weighted average recall (UAR) of models trained on the three datasets

#			*CamInfo*		*Let's Go*		*SweCC*	
1.	Majority class baseline		0.50		0.50		0.50	
	Feature set	***Model***	***JRip***	***SVM***	***JRip***	***SVM***	***JRip***	***SVM***
2.	**MDisCnf**	*late*	0.50	0.50	0.68	0.68	0.87	0.83
		offline	0.50	0.50	0.74	0.73	0.89	0.84
3.	**BoW**	*early*	0.72	0.75	0.72	0.74	0.78	0.80
		late	0.73	0.79	0.80	0.81	0.88	0.88
		offline	0.78	0.80	0.84	0.82	0.90	0.89
4.	**BoW**	*early*	0.75	0.77	0.71	0.75	0.84	0.82
	+DrT	*late*	0.71	0.82	0.82	0.80	0.92	0.91
		offline	0.77	0.79	0.85	0.84	0.92	0.90
5.	**BoC**	*early*	0.80	0.81	0.76	0.76	0.81	0.81
		late	0.81	0.82	0.86	0.84	0.89	0.88
		offline	0.81	0.82	**0.88**	0.85	-	-
6.	**BoC**	*early*	0.80	0.83	0.70	0.80	0.84	0.82
	+DrC	*late*	0.78	0.82	0.84	0.85	**0.93**	0.89
	+DrT	*offline*	0.82	**0.84**	0.87	0.86	0.92	0.89

All three dialogue systems employ confirmation strategies, which are simple built-in mechanisms for detecting miscommunication online. Therefore, a model trained using the *marked disconfirmation* feature alone (cf. Table 5.5, row E) could be a more reasonable baseline model for comparison. Since the awareness about the error using this feature comes after-the-fact, the model will fall in the category of late error detection. The performances of the *late* and *offline* models using this feature are shown in row 2, in Table 5.7 (feature MDisCnf). The performance scores suggest that while the marked disconfirmation feature is not at all useful for the CamInfo dataset (UAR = 0.50 for both JRip and SVM) it makes substantial contributions to the models for the Let's Go and SweCC datasets. The *late model*, using the online values for the feature marked disconfirmation and the JRip algorithm, obtained a UAR of 0.68 for the Let's Go dataset and 0.87 for the SweCC dataset. The

offline models corresponding to these two datasets additionally use the manual feature for marked disconfirmation and achieved even better results: UAR of 0.74 and 0.89, respectively. These figures clearly illustrate two things: Firstly, while Let's Go and SweCC systems often use explicit confirmations, CamInfo hardly uses them. Secondly, the majority of problems in the Let's Go and SweCC are due to misunderstandings on the system's part.

5.6.2 Word-related features

When using the bag-of-word (**BoW**) feature set alone (cf. row 3 in Table 5.7), we observed that for the CamInfo dataset, the SVM algorithm achieved a UAR of 0.75 for the *early* model, a UAR 0.79 for the *late model*, and a UAR of 0.80 for the *offline model*. These are major gains over the baseline of 0.50. The figures for the *early model* suggest that by looking only at (i) the most recent user prompt, (ii) the system prompt preceding it, and (iii) the current system prompt which is to be executed, the model can anticipate, well over chance (the baseline of 0.50) whether the chosen system prompt will lead to a problem.

For the Let's Go and SweCC datasets, using the BoW feature set, the *late model* achieved modest gains in performance over the corresponding MDisCnf baseline model (row 2 in Table 5.7). For example, using the SVM algorithm, the late model for Let's Go achieved a UAR of 0.81. This is an absolute gain of 0.13 points over the UAR of 0.68 achieved using the marked disconfirmation feature set alone. This gain can be attributed partly to the features used in the *early model* (which led to a UAR of 0.74) and the late error detection features which add another 0.07 absolute points and raise the UAR to 0.81. For the SweCC dataset, although the gains made by the JRip learner models over the MDisCnf baseline are marginal, the fact that the *late model* gains in UAR scores over *early model* points to the contributions of words that indicate user disconfirmations, e.g. *no* or *not* (or 'nej' in Swedish).

Next, using the BoW feature set in combination with the **DrT** feature set (containing features derived from turns, such as prompt length (number of words), speaker repetitions, and ASR confidence score) we achieved both minor gains and losses for the CamInfo and Let's Go datasets (cf. row 4 in Table 5.7). The *offline* models for Let's Go made a gain of approx. 0.04 over the *late* models. A closer look at the rules learned by the JRip model indicates that features, such as word error rate, cosine similarity measure of user repetition and number of words in user turns contributed to rule learning.

In the SweCC dataset we observed that for the *early* and *late* models the combination of BoW and DrT feature sets offered improved performances over using BoW alone. The rules learned by the JRip learner indicate that in addition to the marked disconfirmation feature, the model is able to make use of features that indicate whether the system takes the dialogue forward (contributes new information), the ASR confidence score for user turns, the position in dialogue and the user turn length.

5.6.3 Concept-related features

Next, we analyzed the model performances using the bag-of-concept (**BoC**) feature set alone. A cursory look at the performances of the models for the three datasets suggests that for both CamInfo and Let's Go the BoC feature set offers modest and robust improvement over using BoW feature set alone (cf. row 5 in Table 5.7). In comparison, for the SweCC dataset the gains made by the models over using BoW alone are marginal. This is not surprising given the high UARs achieved for SweCC corresponding to the MDisCnf feature (cf. row 2, Table 5.7), suggesting that most problems in the SweCC dataset are inappropriate confirmation requests and simply detecting user disconfirmations is a good enough cue for error detection.

We also observed that the contribution of the *late model* is very clearly seen in Let's Go and SweCC datasets while this is not true for CamInfo. In view of the earlier observation that explicit confirmations are seldom seen in CamInfo, we can say that users use strategies such as repetitions to correct false assumptions by the system. These cues of corrections are much harder to assess than the marked disconfirmations.

The best performances were in general obtained by the *offline* models: UAR of 0.82 on the CamInfo dataset using the SVM algorithm and 0.88 for Let's Go using the JRip learner. Some of the features used by the JRip rule learner include: number of concepts in parse hypothesis being zero (i.e., parse failure), the system dialogue act indicating open prompts "*How may I help you?*" during the dialogue (suggesting that the system restarted the dialogue), and slot-types which the system often had difficulty understanding. These slot-types concerned the user requests for price range and postal codes in the CamInfo dataset, and the time of travel and place of arrival in the Let's Go dataset. As the NLU hypothesis for manual transcription is not available for the SweCC dataset, the corresponding row (row 5) for the *offline model* in Table 5.7 is empty.

Next, we trained the models using the feature sets BoC, DrC and DrT in combination (cf. row 6, Table 5.7). We observed that while the majority of models achieved marginal gains over using the BoC set alone, the ones that did lose did not exhibit a major drop in performance. The best performance for the CamInfo dataset is obtained using the *offline model* and the SVM algorithm: a UAR of 0.84. For Let's Go the JRip model achieved the best UAR, 0.87 for the *offline model*. For the SweCC the *late model* performs better than the *offline model* and achieved a UAR of 0.93, using the JRip rule learner. These are comprehensive gains over the two baseline models.

To recap, the three models have exploited the features presented in this work differently. The *early* models have relied on using features that contribute to the assessment of user turn (turn 2), e.g., the ASR confidence score, the number of word in user turn, the position in dialogue (dialogue length), and whether the system takes the dialogue forward (by adding new information in turn 3). When using features derived from concepts (NLU parse results), the models have used features such as number of concepts in parse hypothesis being zero (i.e., parse failure), the system dialogue act indicating dialogue restart during a conversation, and the slot-types which the system often had difficulty understanding.

The *late* models benefit from addition inputs from user response in turn 4. These include measures of repetition (cosine similarity, number of repeated words) for the CamInfo dataset, and the marked disconfirmation feature for the Let's Go and SweCC datasets.

The *offline* model is similar to the late model, but additionally uses manual features. We have seen that the offline models for the CamInfo and Let's Go datasets extensively relied on manual features such as correctly transferred concepts, word error rate, cosine similarity measure of user repetition and number of words in user turns. However, a comparison of the best performances of the late and the offline models indicates no large differences. This suggests that on these three datasets a reasonably good performance in error detection can be obtained using only the online features and the *late* model presented here.

5.6.4 JRip rules for error detection

In order to get some insights into what features contributed to the detection of models we present here few of the (top) rules learned by the JRip rule learner using the feature set BoC+DrC+DrT (cf. row 6, Table 5.7) for the *offline error detection* models. A comprehensive list of the

rules learned for all the three models (early, late and offline) and the three datasets (CamInfo, Let's Go and SweCC) has been made available in Appendix A.2.

***CamInfo* corpus**: One of the first rules learned by the JRip algorithm on the CamInfo dataset for offline error detection is:

(18) (CTC-usr-2 <= 0.5) and
(NewInfo-NLU-ASR-usr-2 <= 0.5)
=>class=PROBLEMATIC (83.0/9.0)

The rule in (18) uses features derived from manual transcriptions (i.e., CTC cf. Table 5.5, row B) as well as online components (e.g., NLU and ASR). It suggests that, if the fraction of correctly transferred concept in user turn 2 (CTC-usr-2) is <= 0.5, and if the fraction of new information (cf. Table 5.5, row E) contained in user turn 2 (in comparison to the previous user turn) is <= 0.5, then label the system turn 3 as PROBLEMATIC. The rule captures a situation where in user turn 2 the NLU has hardly preserved half of the concepts actually mentioned by the user. At the same time, the second feature (NewInfo-NLU-ASR-usr-2) suggests that the user utterance does not contain much new information when compared with the previous turn, suggesting that the user is rephrasing or repeating herself in turn 2. This is indicative of a typical problem in dialogue system interaction: the system prompt (turn 1) reveals that some miscommunication has occurred, the user in turn 2 repeats (or rephrases) to initiate recovery, but as the ASR and NLU have difficulty understanding the user and the system lacks sophisticated measures to detect corrections, the actions (turn 3) is most likely to be based on a poor judgement of user request (in turn 2), which leads to problems in the dialogue. The rule in (18) suggests that there is scope for further improvements in the model for early error detection in the Cambridge Dialogue System used in the CamInfo Evaluation.

A total of 83 instances of system actions were labelled as problematic by the rule in (18). Among these instances 9 were cases of false predictions (thus, obtaining a precision of 89.1% in its prediction power). Table 5.8 illustrates the predictions made by the *offline model* on a sample CamInfo interaction.

Table 5.8: Offline error detection on the CamInfo interaction. Text within square brackets is the top ASR hypothesis. T and P show the training and predicted labels, respectively. ☒ is PROBLEMATIC, ☑ is NOT-PROBLEMATIC, and ⦸ is an instance of non-understanding therefore excluded from the learning task.

#			T	P
1.	S:	*Welcome to Cambridge system. How may I help you?*		
2.	U:	I want to find a [I want to find a pub]		
3.	S:	*The Avery is a great pub.*	☑	☑
4.	U:	What is the phone number [what is the phone number]		
5.	S:	*The phone number of The Avery is 01223 323405.*	☑	☑
6.	U:	Does it have internet [Uh uh have in there]		
7.	S:	*Can I help you with anything else?*	☒	⦸
8.	U:	Does that pub have internet [Where is that er have in]		
9.	S:	*The Avery is on 69-73 Regent Street.*	☒	☒
10.	U:	Thank you bye.		

Let's Go **corpus**: Two of the rules learned by the JRip algorithm on the Let's Go dataset for offline error detection are:

(19) (WER-usr-2 >= 20) and (4-MDisCnf-tr >= 1)
=>class= PROBLEMATIC (121.0/3.0)
(20) (CTC-usr-2 <= 0.5) and (4-DAact-tr_yes <= 0)
=>class= PROBLEMATIC (115.0/23.0)

The rule in (19) states that if the word error rate (WER) for user turn 2 is more than 20 and the user turn 4 is a marked disconfirmation (cf. discourse features in Table 5.5, row E) then label the system response in turn 3 as PROBLEMATIC. The rule in (20) is similar to the rule in (19) but uses different features (e.g., CTC instead of WER). It states: if the fraction of correctly transferred concept in user turn 2 is <= 0.5 and if in turn 4 the user act was not affirmative (e.g., "yes") then the system action in turn 3 was PROBLEMATIC. Thus, the two rules use late error detection cues such as marked disconfirmations to make an assessment of the system clarification requests. These rules are indicative of a typical problem in dialogue system interaction: the system makes an explicit or implicit clarification request that is based on false assumptions. This suggests that the confidence annotation scheme used in the Let's Go

system can be further improved. Table 5.9 illustrates the predictions made by the *offline model* on a sample Let's Go interaction.

Table 5.9: Offline error detection on the Let's Go interaction. Text within square brackets is the top ASR hypothesis. T and P show the training and predicted labels, respectively. ☒ is PROBLEMATIC, ☑ is NOT-PROBLEMATIC, and ⦸ is an instance of non-understanding therefore excluded from the learning task.

#			T	P
7.	S:	*What can I do for you?*		
8.	U:	Next bus [65e]		
9.	S:	*The 65E. Did I get that right?*	☒	☒
10.	U	No [no]		
11.	S:	*What can I do for you?*	☑	☑
12.	U:	61A [61A]		
13.	S:	*The 61A. Did I get that right?*	☑	☑
14.	U:	Yes [yes]		
15.	S:	*Alright Where would you like to leave from?*	☑	☑
16.	U:	Atwood [Duquesne]		
17.	S:	*For example, you can say, NORTH BRADDOCK, FORBES AND MURRAY, or DOWNTOWN.*	☒	☒
18.	U:	Forbes and Atwood [Forbes big at Atwood in wall]		
19.	S:	*ATWOOD IN WALL. Did I get that right?*	☒	☒
20.	U:	No [no]		

5.6.5 Impact of data size on model performances

We also analyzed the impact of the amount of training data used on model performances. A hold-out validation scheme was followed. A dataset was first randomized and then split into 5 sets, each containing equal number of dialogues. Each of the sets was used as a hold-out test set for models trained on the remaining 4 sets. Starting with only one of the 4 sets as the training set, four rounds of training and testing were conducted. At each stage one whole set of dialogue was added to the existing training set. The whole exercise was conducted 5 times, resulting in a total of 5×5=25 observations per evaluation. Each point in Figure 5.2 illustrates the UAR averaged over these 25 observations by the *offline model* which uses the JRip learner and feature combination BoC+DrC+DrT (cf. row 6 in Table

5.5). The performance curves and their gradients suggest that all the models for the three datasets are likely to benefit from more training data, particularly the CamInfo dataset.

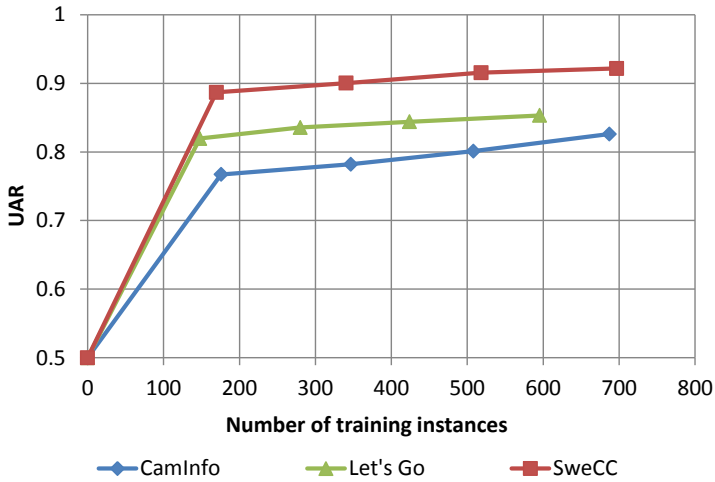


Figure 5.2: Gains in UAR made by the *offline model* (using the JRip learner and feature sets BoC+DrC+DrT, cf. row 6 in Table 5.5) with increasing training set size.

5.6.6 A model for cross-corpus analysis

We also investigated whether a model trained on annotated data from one dialogue system can be used for automatic detection of problematic system turns in interaction logs from another dialogue system. For a given corpus pair only those features were used that were common to both. For example, word error rate (WER) was used in all corpus pairs. However, as the concept error rate (CER) feature is not available for SweCC, it was not used in pairs involving SweCC corpus. Furthermore, this experiment mostly used numeric features such as turn length and word error rate, and dialogue acts that are generic across domains, e.g., request for information, confirmations, and marked disconfirmations. Table 5.10 illustrates the cross-corpus UARs obtained by the *offline model* using the JRip learner and the feature set BoC+DrC+DrT, cf. row 6 in Table 5.5).

All corpus-pair, except one, obtained better UARs than the majority class baseline of 0.50. We observed that using the Let's Go dataset as the training set, we can achieve a UAR of 0.89 for SweCC and 0.72 for CamInfo. These are large improvements over the baseline scores. For

example, the JRip rule in (19) correctly labels 111 system turns as PROBLEMATIC in the CamInfo corpus (a precision of 97.4%). Since both SweCC and Let's Go use explicit clarifications, using them vice-versa for training should offer comparable results. However, as the SweCC dataset exhibits fewer error patterns (mostly explicit clarification on false assumptions) a UAR of only 0.73 is obtained for the test set Let's Go using a model trained on SweCC. Models trained on CamInfo seem more appropriate for Let's Go than for SweCC. Details of all the rules learned by the JRip algorithm for the cross-corpus error detection are presented in Appendix A.3.

Table 5.10: Cross-corpus performances of the *offline model* (using the JRip learner and feature sets BoC+DrC+DrT, cf. row 6 in Table 5.5)

Training set	**Test set**		
	CamInfo	***Let's Go***	***SweCC***
CamInfo	-	0.62	0.50
Let's Go	0.72	-	0.89
SweCC	0.54	0.73	-

5.7 Root cause analysis

The three models for error detection proposed in this work can be used in many different ways. A simple application of the online models could be to build an "error awareness" module in a dialogue system. For offline analysis, the *late error detection* model could be trained on a subset of data collected from a system, and then applied to the whole corpus in order to find problematic turns. By doing this, only these turns (or dialogues) would need to be transcribed and analyzed further, eliminating a lot of manual work. The transcribed dataset could also be used to train the *offline model* and again perform error detection on this set to ascertain the predictions made by the *late model*. As seen in Section 5.6, the offline model offers the best results for error detection among the three models (early, late and offline).

The transcribed dataset can also be used to identify main root causes of the problems, in order to help the dialogue designer to mitigate them. For example:

- Instances with high word error rate (WER) could be used for identification of (i) out-of-vocabulary (OOV) words or phrases that should be added to the language model, and (ii) instances on which the language model could be retrained.
- Instances with a low WER (or high recognition accuracy) but high concept-error rate (CER) could be used for identifying problems with language understanding.
- Instances with low WER and CER could point to possible flaws in dialogue policy.

Such analyses will require identification of the optimal thresholds which is not a trivial task; however, approaches to identifying thresholds in a principled manner have been proposed in the literature (Bohus & Rudnicky, 2005b; Lison, 2013). This type of analyses can be used by dialogue designers to identify errors that are caused by the actual shortcomings of dialogue system components, e.g., missing words in the vocabulary for the ASR and missing concepts in the grammar for parsing. It is a common observation that most errors in spoken dialogue systems are introduced by the speech recognition component (Hirschberg et al., 2004; Bohus & Rudnicky, 2005b). Even if OOV words are added to the grammar there will still be speech recognition errors, due to the mismatch in training and usage conditions in the underlying models. In order to deal with these errors the system needs error awareness and robust processing. An important aspect of the robust processing is having a good dialogue policy for handling the flow of interaction. It is often the case that spirals of error in speech recognition are introduced by poor dialogue policy for error recovery (Bulyko et al., 2005). One often cited example of such a situation (or poor policy) is the strategy to ask the user to "repeat" after a non-understanding (Skantze, 2005). This may lead an already dissatisfied user to repeat the request with hyper-articulation (in an effort to be heard) which is likely to introduce errors in speech recognition, thereby resulting in a spiral of errors. In the following, we present a preliminary work on automatic identification of situations (or patterns) in dialogues that lead to problematic system behavior.

5.7.1 Proposed approach

To aid dialogue designers, the proposed scheme aims at automatically extracting descriptions of situations in a dialogue where the system policy is likely to lead to a problem. With "description" we mean a model of a typical situation that leads up to the problem, e.g., a rule of the form "*if(A)*

and if(B) then PROBLEMATIC." This is similar to the actual error detection described above. However, whereas the error detection can utilize both *symptoms* (e.g. user repetitions) and *causes* (e.g. ASR hypothesis with low confidence score) to identify problematic turns, the root cause analysis should only describe the problem using *causes*, so that the dialogue designer can take actions to remedy these causes. Thus, in our approach, we could first use the automatic error detector to identify problematic turns, and then use the automatic root cause analysis to come up with descriptions of possible causes.

For illustration of the proposed approach, consider the hypothetical interaction in Table 5.11, where we use only the dialogue act and a single slot (DepartureStop) for representing the dialogue discourse. Furthermore, the system turns have been annotated with labels indicating whether they are PROBLEMATIC or NOT-PROBLEMATIC.

Table 5.11: A hypothetical dialogue with discourse represented by dialogue act and slot types only. ☒ is PROBLEMATIC, ☑ is NOT-PROBLEMATIC

#		***DialogueAct*(*Slot-type*)**	***Label***
1.	S:	*Request*(*DepartureStop*)	☑
2.	U:	Inform(DepartureStop)	
3.	S:	*Confirm*(*DepartureStop*)	☒
4.	U:	Inform(DepartureStop)	
5.	S:	*Confirm*(*DepartureStop*)	☒
6.	U:	Inform(DepartureStop)	
7.	S:	*Request*(*DepartureStop*)	☒
8.	U:	Inform(DepartureStop)	

In turn 3, the system confirmation was found PROBLEMATIC, suggesting that it was based on a false assumption. At this stage we can only speculate that the error was caused by limitations of the ASR system (in recognizing user turn 2), or the dialogue policy, or any other source. However, as system turns 5 and 7 were also found to be problematic we need to consider if the dialogue policy in this situation is appropriate. For example, maybe the system could "move on" to fill other slots and return to this slot (DepartureStop) later on, a strategy that has been found useful in recovering from errors in information seeking dialogue (Bohus, 2007).

If such dialogue patterns are often seen in the interactions, it would certainly require a dialogue system designer's attention to learn a better dialogue policy or optimize the one that is currently in place. In the example in Table 5.11, we have only used the dialogue act and slot type as feature for the ease of explanation. However, other features can also be used, e.g., system repetitions. Bulyko et al. (2005) investigated the impact of the form of system responses on speech recognition of user turns, following an error. It was observed that if the system prompt were rephrased the recognition rate of user turns improved in contrast to using repeated prompts. Although the differences were not statistically significant, Bulyko et al. (2005) argued that rephrasing of system prompts caused users to also rephrase their responses, which helped in improved speech recognition of the user requests. It would be interesting to use the system repetition features (cf. Table 5.5, row E) used for error detection for the kind of root cause analysis we are proposing here.

5.7.2 A proof-of-concept analysis

In the proposed framework, we want to identify what happened in turns 3–5 that led to problems in turns 6–7, in the example interaction in Table 5.11. To achieve this, we started by analyzing two attributes of the interactions, namely the dialogue acts and the slot-types used in turns 3–5, and looked for patterns that were most often observed with a PROBLEMATIC turn 7.

Data: For the analysis, the Let's Go and CamInfo datasets introduced earlier were used. As we want to identify the ground truth of the causes of errors, the analysis has to be performed on the values for the two attributes (dialogue-act and slot-type) obtained from the NLU results on the manual transcriptions of the user utterances. Since the Swedish corpus does not contain NLU results on manual transcriptions it was excluded from this analysis. One key difference between the data used for root cause analysis and for training the models for error-detection (used earlier in Section 5.5.2) is that here we also include system turns that expressed non-understandings (cf. Section 5.4.1). While non-understandings are self-evident cases of error in a dialogue we are interested in identifying what caused them.

An example of analysis in the Let's Go dataset

An example interaction from the Let's Go dataset is shown in Table 5.12. The predictions about problematic system turns, in column P, are made by

Table 5.12: Problematic dialogue patterns in a Let's Go dialogue. T and P contain the training and predicted labels, respectively.☒ is PROBLEMATIC, ☑ is NOT-PROBLEMATIC.

#			T	P
7.	S:	*What can I do for you?*	☒	☑
8.	U:	leaving from Dynamo Way to Swissvale [leaving from Brownsville Road to Swissvale]		
9.	S:	*Leaving from Brownsville Road. Did I get that right?*	☒	☒
10.	U:	No Dynamo Way [Dynamo]		
11.	S:	*Leaving from Dynamo. Is this correct?*	☒	☒
12.	U:	Dynamo Way [Dynamo Way]		
13.	S:	*Leaving from Dynamo. Did I get that right?*	☒	☑
14.	U:	Leaving from Dynamo [Lincoln]		
15.	S:	*Leaving from Dynamo. Is this correct?*	☒	☒
16.	U:	East Pittsburgh [East Pittsburgh]		

the *offline model* using the JRip rule learner and the combined features: BoC+DrC+DrT (cf. row 6, Table 5.7). In this interaction, we find a problematic dialogue pattern in turns 9, 10 and 11 where the system is engaged in consecutive clarifications for the same slot-type. This leads to problem in turn 13 (as annotated in column T). The pattern is again observed in turns 11, 12 and 13 which lead to problem in turn 15. Next, we present the statistics on the different problematic dialogue patterns observed in the two datasets.

5.7.3 Results

Let's Go dataset: The dataset contains 754 instances and 322 (42%) of these belong to the PROBLEMATIC class. Table 5.13 illustrates the six discourse events (represented by the dialogue act and slot-type features) in turns 3–5 that figured on the top when observing the label PROBLEMATIC in system turn 7. These events in turns 3–5 make up the rules or the "description" that we talked about in the introduction of this section—a model of a typical situation that leads up to the problem.

Table 5.13: The discourse context in which the Let's Go system action is likely to be problematic (TP is truly-problematic, FP is falsely-problematic, and Pr. is precision or the likelihood of error).

#	*Turn 3 (system)*	*Turn 4 (user)*	*Turn 5 (system)*	*TP*	*TP + FP*	*Pr.*
1.	confirm-depatureplace	inform-depatureplace	request-depatureplace	16	26	0.62
2.	confirm-arrivalplace	inform-timetravel	request-timetravel	13	19	0.68
3.	request-depatureplace	inform-depatureplace	confirm-depatureplace	10	11	**0.91**
4.	confirm-depatureplace	inform-depatureplace	confirm-depatureplace	10	12	**0.83**
5.	request-depatureplace	NA	confirm-depatureplace	6	7	**0.86**
6.	request-timetravel	NULL	request-timetravel	6	6	**1.00**

Of particular interest to us are the patterns where the precision is above 80%, i.e., row 3, 4, 5, and 6.

- Rule 3 indicates a dialogue flow pattern that is problematic 91% of the time (high precision, but low coverage—just 11). The rule suggests that the dialogue pattern of the system requesting a departure station, the user giving the departure station, and the system clarifying the departure station will on 91% occasions lead to the next system action that is problematic.
- Rule 4 captures 12 instances of another dialogue pattern where the next system action is problematic (with a likelihood of 83%).
- Rule 5 captures 7 instances where it was not possible to manually transcribe the audio for the departure station (state NA indicates that no transcription for user turn is available). We speculate this is due to poor audio quality or noisy speech signal.
- Rule 6 captures all the 6 instances where filling the 'time' slot is problematic. The parse result of the manual transcription returned NULL, suggesting issues with the grammar for interpreting time relation information in user utterances.

CamInfo dataset: The dataset has 915 instances of which 245 (26%) belong to the PROBLEMATIC class. Table 5.14 illustrates the 3 discourse events in turn 3-5 (using dialogue act and slot type as feature) that figured on the top when observing the label PROBLEMATIC in system turn 7.

Table 5.14: The discourse context in which the Cambridge system action is likely to be problematic (TP is truly-problematic, FP is falsely-problematic, and Pr. is precision or the likelihood of error).

	Turn 3 (system)	***Turn 4 (user)***	***Turn 5 (system)***	***TP***	***TP + FP***	***Pr.***
1.	inform-name	request-postcode	inform-name	35	61	0.57
2.	inform-name	BYE	inform-name	10	10	**1.00**
3.	inform-name	confirm-hasTV	inform-name	8	10	**0.80**

Of particular interest to us are the rules where the precision is above 80%, i.e., rule 2 and 3.

- Rule 2 captures all the 10 instances where the user indicated end of dialogue (by saying "BYE") but the system continued to recommend a restaurant name.
- Rule 3 captures situation where the system response to the user clarification regarding "*whether the recommended restaurant has a television*" leads to the follow up system actions that is problematic (with a likelihood of 80%).

5.7.4 Automatic learning of descriptions

From our usage of the RIPPER algorithm (Cohen, 1995) for error detection, we believe it should be possible to automatically obtain the descriptions of problematic patterns (such as the ones shown in Table 5.13 and Table 5.14) as rules. The RIPPER algorithm learns rules for classification (from annotated data) as follows: during the training phase the dataset is recursively split into smaller subsets until a subset containing sufficiently high examples of only one class is obtained. The successive splitting is done using features that are optimal for making a good split on the current subset. The end result is a set of rules each corresponding to a unique subset of the original dataset and describing a

specific property of the instances in that subset. For example, the two rules "*if(A) and if(B) and if(C) then* class Z" and "*if(A) and if(B) and if(D) then* class X" create two subsets, classes Z and X, from the main dataset. The rules also describe how elements of class Z differ from those in class X.

In leveraging the RIPPER learner for splitting the dataset into subsets, each representing a unique error situation, the important problem is to identify the relevant features (A, B, C and D), which describe the cause of the problem. Examples of such attributes are the dialogue-act and slot-type attributes used in the proof-of-concept analysis in Section 5.7.2.

The RIPPER algorithm also gives for each rule the accuracy and precision in assigning a particular class label for a subset. Thus, the automatically learned descriptions of problematic dialogue patters can be assessed with regard to their precision in describing problems. The RIPPER algorithm uses the parameter "tolerance error" to specify the acceptable rate of misclassifications in a subset. The parameter values can be experimented with for identifying rules (descriptions of problematic dialogue pattern) that have very high **precision** (but necessarily a high recall).

5.7.5 Summary of root cause analysis

In this section, we have presented a proof-of-concept analysis on identifying dialogue patterns that are highly likely to lead to problems in the dialogue. We have observed that using carefully designed features we can explain a small fraction of problematic patterns in a dialogue albeit with high precision. For the Let's Go and CamInfo dataset we were able to recover 32 and 18 instances, respectively, where the likelihood of the observed dialogue pattern leading to a problematic system action is as high as 80%. These are encouraging results even though the coverage of the rule is not as high (it could only explain approx. 10% of the errors in each corpus). The key here is that we are aiming for rules that have high precision. Thus we may only be able to explain a tiny fraction of errors in the complete dataset with these kinds of general rules. However, it is possible that a larger number of problems could be identified in systems that have not yet been tuned to the same extent. We have pitched an idea that using the RIPPER learner (Cohen, 1995) it should be possible to automatically derive descriptions of dialogue situations that lead to problematic system behavior.

5.8 Conclusion and Discussion

In this chapter, we have presented a data-driven approach to the detection of problematic system turns by automatic analysis of dialogue system interaction logs. Features that are generic across dialogue systems were automatically extracted from the system logs (of ASR, NLU and NLG modules) and the manual transcriptions. We also created abstract features to estimate discourse phenomena such as user repetitions and corrections, and discourse progression. The proposed scheme has been evaluated on interaction logs of three dialogue systems that differ in their domain of application, dialogue modeling, dialogue strategy and language. The trained models achieved substantially better recall on the three datasets, compared to a baseline. We have also shown that it is possible to achieve reasonable performance using models trained on one system to detect errors in another system.

The models for error detection presented here can be used in different ways. For example, the early and later error detection models can be used online for enhancing a system's error awareness. The offline model can be used for root cause analysis. We have proposed a scheme for automatically obtaining descriptions of dialogue patters that result in problematic system behavior. In a proof-of-concept analysis, we were able to identify approximately 10% of the problematic system actions in the CamInfo and Let's Go corpus with specific dialogue patterns. The likelihood of these patterns resulting in problematic system behavior was as high as 80%.

The work presented here differs from the earlier approaches on many accounts. In comparison to Walker et al. (2002), where task failure is treated as error, we have modelled problems in system behavior as error and presented an approach for turn-level detection of errors. We believe knowledge about errors at turn-level is necessary for narrowing down potential flaws in system design. Our approach to turn-level error detection shares the aspect of turn-level analysis in Bohus & Rudnicky (2002), Schmitt et al. (2011), and Krahmer et al. (2001). However, in contrast to Bohus & Rudnicky (2002) and Krahmer et al. (2001), in our model for early error detection we also include the system turn to make an anticipation of the consequences of the system's own actions. Some of the abstract features used in our work to estimate discourse phenomena such as user repetitions and corrections have earlier been found to be useful, e.g., in Krahmer et al. (2001). However, while in Krahmer et al. (2001) features for repetition and correction have been manually extracted, we

have relied on automatic extraction, i.e., using simple measures such as cosine similarity and number of common words.

Yet another contribution of this work is that we have evaluated our approach on datasets from three different dialogue systems that vary in their dialogue modeling, dialogue strategy, language, and user types. This has allowed us to assess the generalizability of the features investigated here. Furthermore, it is due to these generic features that we were able to demonstrate that a model for error detection trained on logs from one system can be readily used for analysis of interaction logs from another system. The results obtained were much better than chance for systems that share similar clarification strategies.

A final novel contribution of our work is the approach for root cause analysis. While for the task of error detection we have exploited the symptoms and causes of error, in the analysis of root causes for these errors we focused on events leading up to problematic system behavior. In contrast to the traditional error cause analysis, where the task is to see if the problem is in ASR (Litman et al., 1999; Hirschberg et al., 2004) and NLU (Walker et al., 2000; Bohus & Rudnicky, 2002), the approach presented here has the potential to identify problems in dialogue policy.

5.9 Directions for future work

An obvious next step is to validate the proposed idea of using the RIPPER learner (Cohen, 1995) for automatic identification of problematic dialogue patterns. For this task we would use additional discourse features such as system repetition, which has been observed to contribute to poor ASR (Bulyko et al., 2005)

It would be very interesting to verify the usefulness of the proposed methodology for error detection and root cause analysis in a bootstrapping approach to dialog system development. Following the analysis of a first version of a system that is tested with users, a new version of the system can be built, and re-analyzed using the same scheme. Improvements in system interactions observed through objective as well as subjective measures would be useful in a qualitative assessment of the proposed framework.

Chapter 6

Knowledge Acquisition

6.1 Introduction

Dialogue managers have to generally rely on various knowledge sources to perform its functions. While information from dialogue and discourse context are used for contextual processing of user utterances, external knowledge bases such as domain models are required for processing domain specific concepts and relations. However, only a finite amount of information can be pre-specified in the knowledge bases. As discussed in Section 2.5.5, even if the dialogue manager has access to some online knowledge-bases, it cannot be guaranteed that all the knowledge is represented there. Missing or incomplete information may cause gaps in dialogue system's knowledge. These gaps in turn may lead to non-understanding or misunderstanding in the dialogues. In an analysis of dialogues with a system for conference room reservation, Bohus & Rudnicky (2005), found that as many as 38% of user utterances get rejected by the system as they contained some information that the system lacked (e.g., our-of-application tasks, out-of-grammar concepts). Pappu (2014) argue that these rejected utterances were opportunity for the system to learn the missing information from the user and presents various approaches to discover missing knowledge and its acquisition in an interactive setting. Leveraging the interactive setting to acquire knowledge from the human counterpart has increasingly gained interest in system development in the field of human–robot interaction. The interactive setting of situated dialogue has been proposed to be a useful source for acquiring knowledge (Kruijff et al., 2007; Johnson-Roberson et al., 2011).

In the speech technology community, harnessing knowledge of various types, such as speech data acquisition, transcription/labelling, and assessment of speech technology applications has been fundamental to development of the field. Traditionally, researchers have used the services of expert annotators to harness the information. Recently, *crowdsourcing*—seeking knowledge from a crowd of non-experts—has

become very popular. Parent & Eskenazi (2011) present an overview of works that have used crowdsourcing for various aspects of spoken dialogue systems.

However, we are not aware of any attempts where a dialogue system is the *vehicle* for crowdsourcing rather than the object of study, that is, where a spoken dialogue system is used to harness knowledge from a large body of non-experts. A task where such crowdsourcing dialogue systems would be useful is to populate geographic databases. While there are now open databases with geographic information, such as OpenStreetMap (Haklay & Weber, 2008), these are typically intended for map drawing, and therefore lack detailed street-level information about city landmarks, such as colors and height of buildings, ornamentations, facade materials, balconies, conspicuous signs, etc. Such information could for example be very useful for pedestrian navigation (Tom & Denis, 2003; Ross et al., 2004). With the current growing usage of smartphones, we might envisage a community of users using their phones to contribute information to geographic databases, annotating cities to a great level of detail, using multi-modal method including speech. The key reason for using speech for map annotation is convenience; it is easy to talk into a mobile phone while walking down the street, so a user with a little experience will not be slowed down by the activity of interacting with a database. This way, useful information could be obtained that is really hard to add offline, sitting in front of one's PC using a map interface, things like: Can you see X from this point? Is there a big sign over the entrance of the restaurant? What color is the building on your right?

Another advantage of using a spoken dialogue system is that the users could be asked to freely describe objects they consider important in their current view. In this way, the system could learn new objects not anticipated by the system designers, and their associated properties.

In this chapter we present a proof-of-concept study of how a spoken dialogue system could be used to enrich geographic databases by crowdsourcing. To our knowledge, this is the first attempt at using spoken dialogue systems for crowdsourcing in this way. In Section 6.2, we elaborate on the need of spoken dialogue systems for crowdsourcing geographic information. In Section 6.3 we describe the dialogue system implementation. Section 6.4 presents our in-lab crowdsourcing experiment. We present an analysis of crowd-sourced data in Section 6.5, and discuss directions for future work in Section 6.6.

6.2 The pedestrian routing domain

Routing systems have been around quite some time for car navigation, but systems for pedestrian routing are relatively new and are still in their nascent stage (Bartie & Mackaness, 2006; Krug et al., 2003; Janarthanam et al., 2012; Boye et al., 2014). In the case of pedestrian navigation, it is preferable for way-finding systems to base their instructions on *landmarks*, by which we understand distinctive objects in the city environment. Studies have shown that the inclusion of landmarks into system-generated instructions for a pedestrian raises the user's confidence in the system, compared to only left-right instructions (Tom & Denis, 2003; Ross et al., 2004).

Basing routing instructions on landmarks means that the routing system would, for example, generate an instruction "Go towards the red brick building" (where, in this case, "the red brick building" is the landmark), rather than "Turn slightly left here" or "Go north 200 meters". This strategy for providing instructions places certain requirements on the geographic database: It has to include many landmarks and many details about them as well, so that the system can generate clear and unambiguous instructions. However, the information contained in current databases is still both sparse and coarse-grained in many cases.

Our starting point is a pedestrian routing system we designed and implemented, using the landmark-based approach to instruction-giving (Boye et al., 2014). The system performs visibility calculations whenever the pedestrian approaches a waypoint, in order to compute the set of landmarks that are visible for the user from his current position. OpenStreetMap (Haklay & Weber, 2008) is used as the data source. Figure 6.1 shows a typical situation in pedestrian routing session. The blue dot indicates the user's position and the blue arrow her direction. Figure 6.2 shows the same situation in a first-person perspective. The system can now compute the set of visible landmarks, such as buildings and traffic lights, along with distances and angles to those landmarks. The angle to a building is given as an interval in degrees relative to the direction of the user (e.g. 90° left to 30° left). This is exemplified in Figure 6.1, where four different buildings are in view (with field of view marked with numbers 1–4). Landmarks that are not buildings are considered to be a single point, and hence the relative angle can be given as a single number.

Figure 6.1: A pedestrian routing scenario

Figure 6.2: The visual scene corresponding to the pedestrian routing scenario in Figure 6.1

When comparing the map with the street view picture, it becomes obvious that the "SEB" bank office is very hard to see and probably not very suitable to use as a landmark in route descriptions. On the other hand, the database does not contain the fact that the building has six stories and a façade made of yellow bricks, something that would be easily recognizable for the pedestrian. This is not due to any shortcoming of the OpenStreetMap database; it just goes to show that the database has

been constructed with map drawing in mind, rather than pedestrian routing. There are also some other notable omissions in the database; e.g. the shop on the corner, visible right in front of the user, is not present in the database. Since OpenStreetMap is crowd-sourced, there is no guarantee as to which information will be present in the database, and which will not. This also highlights the limitation of existing approaches to crowdsourcing geographic information: Some useful information is difficult to add off-line, using a map interface on a PC. On the other hand, it would be a straightforward matter given the kind of crowdsourcing spoken dialogue system we present next.

6.3 A dialogue system for crowdsourcing

To verify the potential of the ideas discussed above, we implemented a spoken dialogue system that can engage in spoken conversation with users and learn details about landmarks in visual scenes (such as Figure 6.2). To identify the kind of details in a visual scene that the system could potentially ask the users, we first conducted a preliminary informal crowdsourcing dialogue: one person (the receiver), was instructed to seek information that could be useful for pedestrian navigation from the other person (the giver). The receiver only had access to information available in the maps from OpenStreetMap, as in Figure 6.1, but without any marking of field of views, whereas the giver only had access to the corresponding visual scene (as in Figure 6.2). Interaction data from eight such dialogues (from four participants, and four different visual scenes) suggested that in a city environment, buildings are prominent landmarks and much of the interaction involves their properties such as color, number of stories, color of roof, signs or ornamentations on buildings, whether it has shops, etc. Seeking further details on mentioned signs, shops, and entities (whether mapped or unmapped) proved to be a useful strategy to obtain information. We also noted that asking for open-ended questions, such as "*Is there anything else in this scene that I should be aware of*?" towards the end has the potential of revealing unknown landmarks and details in the map.

Obtaining specific details about known objects from the user corresponds to slot-filling in a dialogue system, where the dialogue system seeks a value for a certain slot (= attribute). By engaging in an open-ended interaction the system could also obtain general details to identify new slot-value pairs. Although slots could be in some cases be multi-valued (e.g., a building could have both color red and yellow), we

have here made the simplifying assumption that they are single valued. Since users may not always be able to specify values for slots we treat *no-value* as a valid slot-value for all type of slots.

We also wanted the system to automatically learn the most reliable values for the slots, over several interactions. As the system interacts with new users, it is likely that the system will obtain a range of values for certain slots. The variability of the answers could appear for various reasons: users may have differences in perception about slot-values such as colors, some users might misunderstand what building is being talked about, and errors in speech recognition might result in the wrong slot values. Some of these values may therefore be in agreement with those given by other users, while some may differ slightly or be in complete contradiction. Thus the system should be able to keep a record of all the various slot-values obtained (including the disputed ones), identify slot-values that need to be clarified, and engage in a dialogue with users for clarification.

In view of these requirements, we have designed our crowdsourcing dialogue system to be able to (1) take and retain initiative during the interactions for slot-filling, (2) behave as a responsive listener when engaging in open-ended dialogue, and (3) ask *wh–* and *yes–no questions* for seeking and clarifying slot-values, respectively. Thus when performing the slot-filling task, the system mainly asks questions, acknowledges, or clarifies the concepts learned for the slot-values. Apart from requesting repetitions, the user cannot ask any questions or by other means take the initiative. A summary of all the attributes and corresponding system prompts is presented in Appendix B.

The top half of Figure 6.3 illustrates the key components of the dialogue system. The Dialogue Manager queries the Scene Manager (SM) for slots to be filled or slot-values to be clarified, engages in dialogue with users to learn/clarify slot-values, and informs the SM about the values obtained for these slots. The SM manages a list of scenes and the predefined slots—for each type of landmark in visual scenes—that need to be filled, maintains a record of slot-values obtained from all the users, and identifies slot-values with majority vote as the current reliable slot-value. To achieve these objectives, the scene manager uses an XML representation of visual scenes. In this representation, landmarks (e.g., buildings, junctions) are stored as *scene-objects* (cf. Figure 6.4). The details about the landmarks are acquired automatically from the OpenStreetMap database using the system for visibility computation mentioned in Section 6.2.

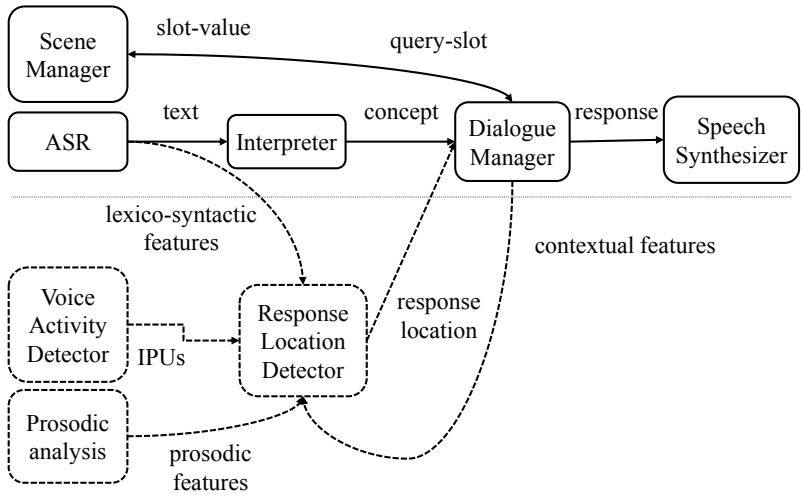


Figure 6.3: Dialogue system architecture

The Dialogue Manager (DM) uses scene-object attributes, such as type, angle or interval of a building, to generate referential expressions, such as "Do you see a *building* on the *far left*?" or "Do you see a *shop* on the *left*?" to draw the users' attention to the intended landmark in the scene. During the course of interaction, the Scene Manager (SM) extends scene-objects with a set of predefined attributes (= slots) that we identified in the preliminary study, along with their various slot-values (cf. Figure 6.5). For each slot, the SM keeps a record of slot-values obtained through wh–questions as well as the ones disputed by the users in yes–no questions (cf. `obtained` and `disputed` tags in the XML in Figure 6.5), and uses their tally to identify the slot-value in majority. The system assumes this slot-value (or one of them in case of a tie) as its best estimate of a slot-value pair, which it could clarify with another user using a yes–no query.

During the slot-filling mode the DM switches to open-ended interaction mode to seek general details if the user suggests/agrees that there are signs on/at a scene-object, or a building has shops or restaurants. The systems uses prompts such as "*Could you describe it/them?*" or "*How does it look like?"* to achieve this objective. Once all the slots for all the scene-objects in a visual scene have been queried, the DM once again switches to the open-ended interaction mode and queries the users whether there are any other relevant signs or landmarks that the system may have missed and should be aware of. On completion of the open-

ended queries the SM selects the next visual scene, and the DM engages in a new dialogue.

```
<scene xmlns="cityCS.scene" name=" view7.jpg"
lat="59.34501" lon="18.0614" fovl="-60" fovr="60"
bearing="320" dist="100">

      <scene-object>
          <id>35274588</id>
          <type>building</type>
          <from>-60</from>
          <end>-39</end>
      </scene-object>

      <scene-object>
          <id>538907080</id>
          <type>shop</type>
          <distance>34.82</distance>
          <angle>-39</angle>
          <bearing>281</bearing>
      </scene-object>

      <scene-object>
           <id>280604</id>
           <type>building</type>
           <from>-38</from>
           <end>6</end>
      </scene-object>

      <scene-object>
          <id>193906</id>
          <type>traffic_signals</type>
          <distance>40.77</distance>
          <angle>-14</angle>
          <bearing>306</bearing>
      </scene-object>
  ...
</scene>
```

Figure 6.4: XML representation of visual scenes

For speech recognition and semantic interpretation the system uses a context-free grammar with semantic tags (SRGS[13]), tailored for the domain. The output of semantic interpretation is a concept. If the concept type matches the type of the slot, the dialogue manager informs the scene manager about the obtained slot-value. If the concept type is inappropriate the DM queries the user once more (albeit using different utterance forms). If still no appropriate concept is learned the DM requests the SM

[13] http://www.w3.org/TR/speech-grammar/

for the next slot and proceeds with the dialogue. For speech synthesis, we use the CereVoice system developed by CereProc[14]. The dialogue system has been implemented using the IrisTK framework (Skantze & Al Moubayed, 2012).

```
<scene-object>
    <id>35274588</id> <type>building</type>
    <from>-60</from> <end>-39</end>
    <slot slotName="VISIBLE">…    </slot>
    <slot slotName="COLOR">
     <obtained>
       <value slotValue="Green">
         <userlist>
           <usrDtls uid="u01" asrCnf="0.06" qType="WH"/>
         </userlist>
       </value>
       <value slotValue="no-value">
         <userlist>
           <usrDtls uid="u02" asrCnf="0.46" qType ="WH"/>
         </userlist>
       </value>
       <value slotValue="Gray">
         <userlist>
           <usrDtls uid="u03" asrCnf="0.19" qType ="WH"/>
         </userlist>
       </value>
     </obtained>
     <disputed>
       <value slotValue="Green">
         <userlist>
           <usrDtls uid="u02" asrCnf="0.92" qType ="YN"/>
         </userlist>
       </value>
     </disputed>
    </slot>
    <slot slotName="STORIES">…    </slot>
    <slot slotName="ROOF_COLOR">…    </slot>
    …
</scene-object>
```

Figure 6.5: Every slot-value is recorded

In contrast to the slot-filling mode, when engaging in an open-ended interaction, the system leaves the initiative to the user and behaves as a responsive listener. That is, the system only produces feedback responses, such as backchannels (e.g., *okay*, *mh-hmm*, *uh-huh*), repetition requests for longer speaker turns (e.g., *Could you repeat that?*), or continuation prompts such as "*anything else?*" until the user is finished speaking. Unless the system recognized an explicit closing statement from the user

[14] https://www.cereproc.com/

(e.g., "*I can't*"), the system encourages the user to continue the descriptions for 2 to 4 turns (chosen randomly).

To detect appropriate locations in users' speech where the system should give feedback response, the system uses the model for Response Location Detection (RLD) presented in Section 4.5 in Chapter 4. When the voice activity detector detects a silence of 200 ms in users' speech, the model uses prosodic, contextual and lexico-syntactic features from the preceding speech segment to decide whether the system should produce a feedback response. The lower half of Figure 6.3 shows the additional components of the dialogue system used in open-ended interaction mode (cf. Figure 4.5 which illustrates the components for the system for RLD, in iteration 2). In this mode, the ASR system uses a language model that is trained on verbally given route instruction from the IBL corpus (used in Chapter 3), and the study conducted in Boye et al. (2014), in parallel to the SRGS grammar for understanding.

6.4 In-lab crowdsourcing experiment

Nine visual scenes (wide-angle pictures in first-person perspective and taken in Stockholm city, cf. Figure 6.2) were used for the task of crowdsourcing. Fifteen human participants (4 females and 11 males) participated in the crowdsourcing exercise. All participants either studied or worked at the School of Computer Science and Communication, KTH, Stockholm.

Participants were placed in front of a computer display and were told that the system will engage them in a spoken conversation to seek or clarify details about landmarks and other objects in visual scenes. They were told that the details would be used for pedestrian routing and therefore they are free to choose and specify details (in open-ended questions) that they thought would be useful when giving route instructions to another person.

Each participant did the nine visual scenes in the same order, with a 1 minute pause between each of them. The first visual scene was used as a trial in order to familiarize participants with the interaction scenario. For this reason, the trial interaction was specifically designed to engage the participants in both wh– and yes–no type questions. We also wanted to investigate whether the use of wh– or yes–no questions alone has any impact on the interactions. For the remaining eight interactions, the dialogue system therefore alternatively used wh– and yes–no questions throughout the whole dialogue. The order of query types was switched for

each successive participant. Thus visual scenes presented with wh–queries to a participant p_i were presented to participant p_{i+1} with only yes–no queries, and vice-versa. After each interaction participants filled in a questionnaire that asked them (1) whether the system appeared to be familiar with the visual scene; and (2) whether the interactions went smoothly.

An example[15] of a conversation with wh–type queries is presented in Table 6.1. The second column indicates the various types of utterances (dialogue acts) used by the system during this interaction. Note that during the slot-filling mode, the system has the initiative, whereas in the open-ended queries (cf. rows 15–25 in Table 6.1) the system gives the initiative to the participant and only produces feedback responses. Table 6.2 presents a conversation with yes–no queries following the conversation in Table 6.1. In a conversation with only yes–no queries, a wh– query may be presented if a slot has been observed with *no-value* (cf. rows 9–10 in Table 6.1) or when the participant disputes the system's current best estimate (cf. rows 6–9 in Table 6.2).

Table 6.1: An example interaction with wh– queries for the visual scene in Figure 6.2. S: system, P: participant (here, uid=u01 from Figure 6.5). ASR results are shown within square brackets.

1	S:	Introduction	*Could you help me with some details about this view?*
2	P:		Yes
3	S:	wh– query	*Do you see a building on the far left?*
4	P:		Yes
5	S:	Backchannel	*mh-hmm*
6	S:	wh– query	*What color does the building have?*
7	P:		Cream color [Green color]
8	S:	Reprise	*Green*
9	S:	wh– query	*How many floors is this building?*

[15] An example video of the interaction can be seen at https://www.youtube.com/watch?v=T4vtKD8NKVs

10	P:		I don't know [I do no]
11		…	
12	S:	wh– query	*Are there any important signs in this scene that I may have missed?*
13	P:		No
14	S:	Backchannel	*uh-huh*
15	S:	wh– query	*Is there anything peculiar in this scene that I should be familiar with*
16	P:		There is a bike painted on street just in front of me [is the right pink university in front of may]
17	S:	Backchannel	*Anything else?*
18	P:		Yeah its white paint [Yes light pink]
19	S:	Backchannel	*mh-hmm*
22	P:		That's it [that that]
23	S:	Backchannel	*ehm*
24	P:		I am done [okay]
25	S:	Ending	*Okay, thanks*

Table 6.2: An example interaction with yes–no queries corresponding to the visual scene in Figure 6.2. S: system, P: participant (here uid=u02 from Figure 6.5). ASR results are shown within square brackets.

1	S:	Introduction	*Could you help me with some details about this view?*
2	P:		Yes
3	S:	yes–no query	*Is the building on the far left visible to you?*
4	P:		Yes
5	S:	Backchannel	*mh-hmm*
6	S:	yes–no	*Does this building have green color?*

		query	
7	P:		No
8	S:	Backchannel	*uh-huh*
9	S:	wh– query	*But than what color is the building?*
10	P:		white [light]
11	S:	Repeat	*Sorry, what color did you say?*
12	P:		white [light]
13	S:	wh– query	*How many stories is this building?*
14	P:		I don't know [I do rental]
15	S:	Backchannel	*ehm*
16		…	
17	S:	wh– query	*Are there any important signs in this scene that I may have missed?*
18	P:		ehm no [nine no]
19	S:	Backchannel	*ehm*
20	S:	yes–no query	*I am informed that there are other important landmarks or structures in this scene that I should be aware of, is it true?*
21	P:		No
22	S:	Backchannel	*uh-huh*
23	S:	Ending	*Okay, thanks*

6.5 Data analysis

We analyzed the data (15×8 = 120 interactions) collected from the experiment along the following tracks: first, we compare the majority value of the slots to the ground truth as given by a human annotator; second, we explore how the ground truth of slot-values could be estimated automatically; third, we also analyzed the instances where the participants disputed the system's current estimate of slot-values; and fourth, we examined the post-experimental questionnaires.

6.5.1 Rate of learning slot-values

A total of 197 slots were learned in the experiment. We analyzed how many slot-values had been correctly retrieved after 1, 2… 15 users. In

Figure 6.6, the curve "Majority" illustrates the fraction of slot-values correctly learned with each new user, under the assumption that the slot-values with majority votes—from all the 15 users—constitute the ground truth. Thus, after interacting with the first user, the system had obtained 67.0% of slot-values correctly (according to the majority), and 96.4% of slot-values after interacting with the first six users. Another eight users, or fourteen in total, were required to learn all the slot-values correctly. The progression curve thus provides an estimate of how many users are required to achieve a specific percentage of slot-values correctly if majority is to be considered the ground truth.

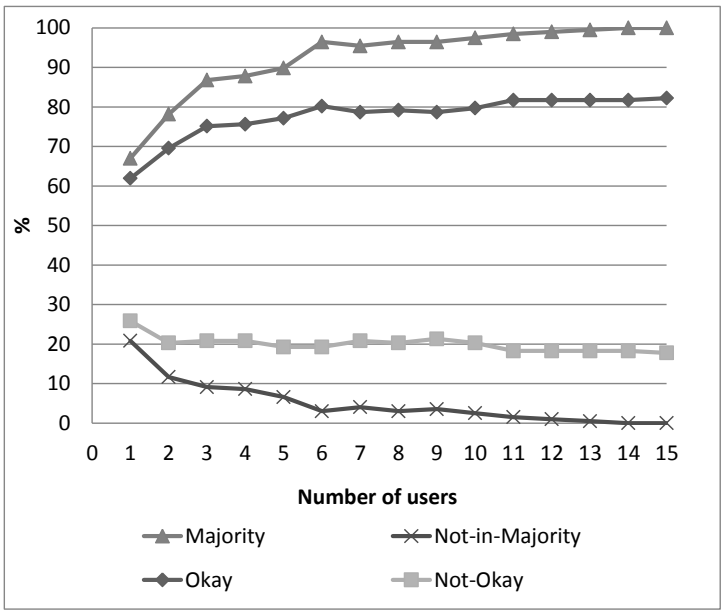


Figure 6.6: Rate of learning slot-values with two different estimates of ground truth

The curve "Not-in-Majority" indicates the number of slot with values that were not in the majority. Thus after interacting with the first user 20.8% of the slot-values the system had obtained were not in majority and could be treated as incorrect. Note that the curves Majority and Not-in-Majority do not sum up to 100%, this is because we consider *no-value* as a valid slot-value, and treat the slot as unfilled. For example, 12.2% of the slots remained unfilled after interacting with the first user.

We also investigated how close the majority is to the actual truth. The author of this thesis labeled all the obtained slot-values with a judgement: Okay or Not-Okay. A slot value was considered Okay if it made sense in view of the combined knowledge from the corresponding map from OpenStreetMap, the visual scenes, and the set of obtained values. If the value did not make sense, the label Not-Okay was assigned. A slot could have many acceptable values. For example, various parts of a building could be painted in different colors. The progression curves "Okay" and "Not-Okay", in Figure 6.6 illustrate the fraction of total slots for which the learned values actually made sense and no sense, respectively. While the curve Okay follows the same pattern as the progression curve Majority, the percent of slot-values that were labelled okay is always lower than the majority as ground truth, and it never reached 100%. The constant gap between the two curves suggests that some slot-values learned by the majority were not actually the ground truth (as per the annotator).

As mentioned earlier, much of the slot-filling interaction involved buildings and their properties. Figure 6.7 illustrates the progression of learning the properties of the landmark "building" with every new user. The curves suggest that meaningful values for most slots (pertaining to whether a building is visible, whether it is residential, whether it has shops, and the color of roof) were obtained by interacting with only few participants. In contrast, properties such as color of the building and number of stories required many more participants. This could be attributed to the fact that participants may have differences in perception about slot-values. As regards to whether there are signs on the buildings, we observed that the recall is relatively low. This is largely due to lack of common ground among participants about what could be considered a sign. However, some element in this low recall is also due to the annotator's definition of what defines a sign. Our intentions with designing this prompt was to retrieve any peculiar detail on the building that is easy to locate: for us a sign suggesting a name of restaurant is as useful as the knowledge that the building has blue sunshade on the windows. Some participants understood this while other didn't. The constant gap in the Majority and Okay curve in Figure 6.6 can also be partly ascribed to this lack of common ground between the annotator and the participants, as well as across participants.

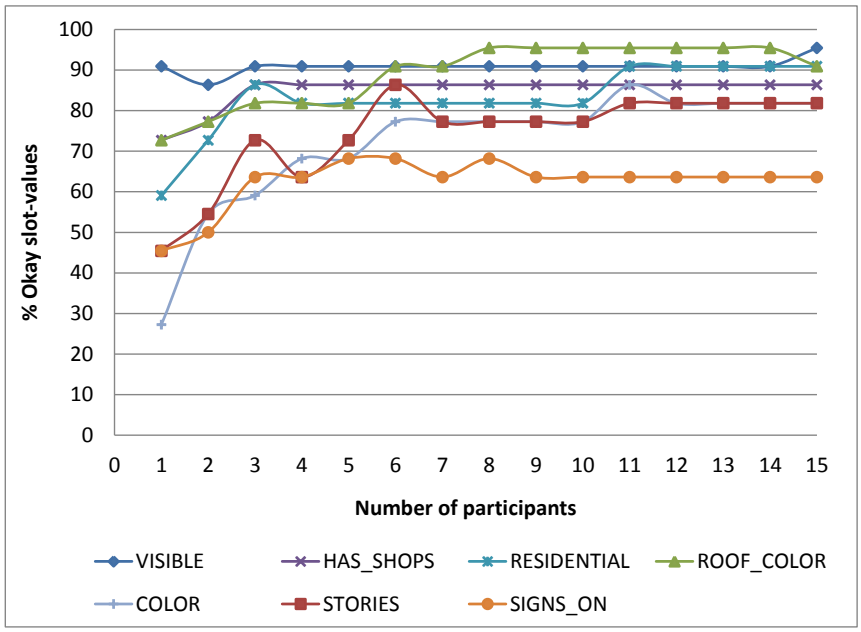


Figure 6.7: Learning rate of various slots for landmark type *building*

6.5.2 Estimated ground truth of slot-values

The 15 subjects in the in-lab experiment were all asked for the same information. In a real application, however, we want the system to only ask for slots for which it has insufficient or conflicting information. If the ground truth of a certain slot-value pair can be estimated with a certainty exceeding some threshold (given the quality requirements of the database, say 0.8), the system can consider the matter settled, and need not ask about that slot again. We therefore want to estimate the ground truth of slot-values along with a certainty measure. To this end, we use the CityCrowdSource Trust software package (Dickens & Lupu, 2014), which is based on the probabilistic approach for supervised learning when we have multiple annotators providing labels (possibly noisy) but no absolute gold standard, presented in Raykar et al. (2009).

Using this approach, a question concerning the color of a building, say with ID 24, (e.g. "What color is the building?") would be translated into several binary predicates COLOR_Red(24), COLOR_Brown(24), COLOR_Orange(24), etc. The justification for this binary encoding is that the different color values are not mutually exclusive: A building might of

course have more than one color, and in many cases more than one color name might be appropriate even though the building has only one dominating color (e.g. to describe the color either as “brown” and “red” might be acceptable to most people). Figure 6.8 shows the incremental estimates for different colors for a certain building (OpenStreetMap ID 163966736) after 1, 2… 15 subjects had been asked. The answer from the first subject was erroneously recognized as “pink”. The next 9 subjects all referred to the building as “brown”. Among the final subjects, 3 subjects referred to building as “red”, and 2 subjects as “brown”. The final truth estimates are 0.98 for “brown”, 0.002 for “red”, and 0.00005 for “pink”. The diagram shows that if the certainty threshold is set to 0.8, the value “brown” would have been established already after 4 subjects.

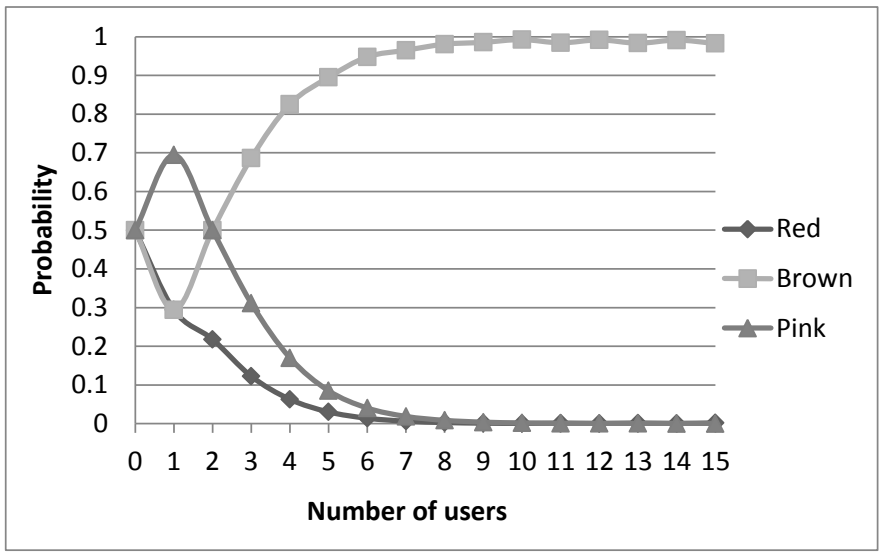


Figure 6.8: Probabilities of different estimated ground truth values for the color of a certain building

6.5.3 Disputed slot-values

We also examined all system questions of yes–no type that received negative answers, i.e. instances where the participants disputed the system’s current best estimate (based on majority vote) of a slot-value. Among the 95 such instances, the system’s current best estimate was actually Okay (made sense) only on 43 occasions. In 30 of these instances the participants provided a rectified slot-value that was meaningful (can could be labelled Okay). For the remaining 13 instances the new slot-

values proposed by the participant did not made sense, i.e., were labelled Not-Okay.

The remaining 52 instances were false disputations, i.e. the system's current estimate of a slot-value made sense (Okay), but the participants disputed it. 6 of these occurrences were due to errors in speech recognition, but for the remaining 46 occasions, error in grounding the intended landmark (15 cases), users' perception of slot-values (3), and ambiguity in what the annotator terms as Okay slot-values (28), (e.g. whether there are signs on a building (as discussed in Section 6.5.1)) were identified as the main reasons. This suggests that slots (i.e. attributes) that are often disputed may not be easily understood by users.

6.5.4 Post-experimental questionnaire

As described in Section 6.4, the participants filled in a questionnaire after each interaction with the system. They were asked to rate the system's familiarity with the visual scene based on the questions asked. A Mann–Whitney *U* test suggests that participants' perception of the system's familiarity with the visual scene was significantly higher for interactions with yes–no queries than interactions with wh– queries (U=1769.5, p= 0.007). This result has implications for the design choice for systems that provide as well as ask for information from users. For example, a pedestrian routing system can already be used to offer routing instructions as well as crowdsourcing information. The system is more likely to give an impression of familiarity with the surrounding, to the user, by asking yes–no type questions than wh–questions. This may influence a user's confidence or trust in using the routing system.

Since yes–no questions expect a "yes" or "no" in response, we therefore hypothesized: interactions with yes–no questions would be perceived smoother in comparison to interactions with wh– questions. However, a Mann–Whitney *U* test suggests that the participants perceived no significant difference between the two interaction types (U=1529.0, p= 0.248). Feedback comments from participants suggest that abrupt ending of open-ended interactions by the system (due to the simplistic model of detecting whether the user has anything more to say) gave users an impression that the system is not allowing them to speak.

6.6 Discussion and future work

We have presented a novel approach for knowledge acquisition. The presented approach combines the interactive setting of spoken dialogue with crowdsourcing. We have presented a proof-of-concept study on

using a spoken dialogue system for crowdsourcing street-level geographic information. To our knowledge, this is the first attempt at using spoken dialogue systems for crowdsourcing in this way. The system is fully automatic, in the sense that it (i) starts with minimal details—obtained from OpenStreetMap—about a visual scene, (ii) prompts users with wh–questions to obtain values for a predefined set of attributes; and (iii) assumes attribute-values with majority vote as its beliefs, and engages in yes–no questions with new participants to confirm them. In a data collection experiment, we have observed that after interacting with only 6 human participants the system acquires more than 80% of the slots with actually Acceptable values.

We have also shown that the majority vote (as perceived by the system) could also be incorrect. To mitigate this, we have explored the use of the CityCrowdSource Trust software package (Dickens & Lupu, 2014) for obtaining the probabilistic estimate of the ground truth of slot-values in a real crowdsourcing system. However, it is important not only to consider the ground truth probabilities per se, but also on how many contributing users the estimate is based and the quality of information obtained. We will explore these two issues in future work.

We have observed that through open-ended prompts, the system could potentially collect a large amount of details about the visual scenes. Since we did not use any automatic interpretation of these answers, we transcribed key concepts in participants' speech in order to obtain an estimate of this. However, it is not obvious how to quantify the number of concepts. For example, we have learned that in Figure 6.2, at the junction ahead, there is: a *traffic-sign*, a *speed-limit* sign, a sign with *yellow* color, a sign with *red* color, a sign with *red boarder*, a sign that is *round*, a sign with some *text*, the *text* says *50*. These are details obtained in pieces from various participants. Looking at Figure 6.2 one can see that these pieces when put together refer to the speed-limit sign mounted on the traffic-signal at the junction. How to assimilate these pieces together into a unified concept is a task that we have left for future work.

Chapter 7

Conclusion

7.1 Thesis summary

A central challenge for spoken dialogue systems is to deal with the ubiquity of *variability*—in user behavior, in the environment of system usage, in the performances of the various speech and language processing sub-components, and in the nature of the tasks. However, as the predominant methodology for dialogue system development is to handcraft the sub-components, these systems typically lack robustness in dealing with the challenges arising from variability. The lack of robustness is also a major hindrance towards realizing the ambition of building human-like dialogue systems.

Due to their robustness to variability, data-driven approaches have recently gained interest in development of spoken dialogue technologies, and have been shown to offer promising results (especially in speech recognition and speech synthesis). However, there are various other tasks that can potentially benefit from this methodology. In this thesis, we have investigated the application of data-driven methodologies, and contributed four novel methods towards modelling robust solutions to tasks that are of key importance to dialogue systems:

- A data-driven method for semantic interpretation of spoken route instructions. The presented approach is robust in parsing "ungrammatical" spoken utterances and also extracts the structural relations among the concepts in the semantic representation.
- A data-driven method for timing system responses, i.e., identifying places in a user's speech where it is appropriate for the system to give feedback responses.
- A data-driven method for detection of problematic system actions through automatic analysis of interaction logs. Three different models have been presented, two of which can be used online for enhancing the system's error awareness and one for offline error

analysis towards helping the dialogue system designer to identify potential flaws in dialogue design.

- An implicitly-supervised method for knowledge acquisition. We have presented a system that leverages the interactive setting of spoken dialogue and the crowdsourcing scheme to acquire new information through interactions with users, without requiring any manual supervision.

In the rest of this section, we present a brief summary of these four works. In the next section, we discuss some ideas on how the methods presented here could generalize to other tasks.

7.1.1 Spoken Language Understanding

Semantic interpretation of spoken route instructions must address two issues: (a) robust parsing of "ungrammatical" and erroneous user utterances, as recognized by the speech recognizer, and (b) extraction of deeper structures—the structural relations among the concepts—in the semantic representation. The issue of "ungrammatical" utterances arises from speech phenomena such as restarts, hesitations, repetitions and revisions. Errors in utterance arise from problems in speech recognition—deletion, insertion, and substitution of words. The need to extract relational structures arises from the semantic representations needed for representing structured information contained in for example route instructions.

The existing grammar based approaches for interpretation can extract deeper structures (Charniak, 1997; Uszkoreit, 2000), but are not adequate for robust parsing of spoken utterances. On the other hand, data-driven methods for semantic interpretation of spoken utterances (He & Young, 2006; Meza-Ruiz et al., 2008) exhibit robustness to errors, but are only suitable for flat, slot-value pairs semantic representations and do not account for structured representations.

In Chapter 3 of this thesis, we have presented a novel application of the *Chunking parser* (Abney, 1991) for addressing these two issues. In our approach, the parser takes a route instruction as input, processes it through three stages: the *Chunker*, the *Segmenter*, and the *Attacher*, and as an end result produces a *Conceptual route graph* (*CRG*). The Chunker and Segmenter follow the intuition of parsing by chunks, i.e., they do not have to account for every single word (or concept) in the input, but simply focus on identifying the chunks—the basic concepts relevant to the task domain. This enables the Chunking parser to robustly deal with

"ungrammatical" and erroneous utterances. The Attacher takes a sequence of chunks as input and may assign to each basic concept therein a more *specific* domain concept and attributes (as structural relations) to neighboring concepts. In this way, the three components of the Chunking parser are able to perform robust parsing and preserve the structural relations among the concepts in the semantic representation, i.e., the *conceptual route graphs*.

In the reported data-driven approach, we modelled each of the three components of the Chunking parser as a *classification* task and used various linear classifiers. The models were trained using a range of lexico-syntactic features such as the *word instance*, *word-window*, *bag-of-words*, *part-of-speech* (POS), *bag-of-POS*, and domain level features such as the previous *chunking tags*, *chunk-label* of previous chunks, and a *chunk-label window*.

In experimental evaluations, we were also able to show that the proposed model was robust to speech recognition errors, and that more training data should help to improve the performance. The comparable performances of human subjects in route drawing using the automatically extracted conceptual route graphs from spoken and transcribed instructions, further strengthens our claim of robustness of the approach.

7.1.2 Turn-taking

In order to maintain a smooth flow in the interactions, dialogue systems should give feedback (such as "*mh-hmm*", "*uh-hun*") to show continued attention to the users and produce responses when appropriate or expected by the user. This necessitates dialogue systems to be able to detect when it is appropriate to take the turn and when to give feedback. A conventional method in dialogue systems is to use silence (of some fixed threshold) in user speech to detect the end of the user's utterance and produce responses. However, such a model is not very accurate, since speakers sometimes hesitate while no turn-change is intended, and sometimes the turn changes after barely any silence (Sacks et al., 1974). Using such a simplistic model can result in systems that frequently produce responses at inappropriate occasions, or produce delayed responses, or no response at all when expected. This affects the efficiency of the dialogue and user experience. Following the observations on turn-taking and backchanneling mechanism in human–human interactions, more advanced models for timing system responses have been suggested (Ward, 1996; Koiso et al., 1998; Cathcart et al., 2003; Gravano & Hirschberg, 2011). However, hardly any work has been done in actually

assessing their contribution in actual human–computer interactions (with the exception of Raux & Eskenazi (2008) and Huang et al. (2011)).

In Chapter 4 of this thesis, we have presented a data-driven model for timing system responses and demonstrated its application in real user interactions. We were able to accomplish this because unlike the previous works, we have trained the model on relevant data, i.e., human–computer interaction data and have used only automatically extractable features (i.e., requiring no manual labelling) from the interactions. The presented approach follows the bootstrapping scheme for system development. First, a fully automated dialogue system using a naïve model for response location detection (RLD) was built and used to collect human–computer interaction data. Next, the data was manually annotated and used to train a more sophisticated model for RLD, integrated in the same system, and evaluated in real user interactions.

We formulated the task of RLD as a classification task: given a user speech segment ending in 200 ms of silence, identify whether it is an appropriate response location. A range of linear classifiers and decision tree learner algorithms were trained using features spanning over *prosody*, *context*, and *lexico-syntactic* categories.

In experimental evaluations we observed that while syntactic features contribute the most to the classification task, prosodic and contextual features alone were able to obtain a reasonably good performance. These models could be useful for RLD when speech recognition is poor or not available. An early version of the proposed models was evaluated in a user study and was found to reduce the number of errors (in turn-taking) by two thirds, as compared to a naïve model.

7.1.3 Error detection

Awareness about errors in human–computer dialogue is useful not only for online error handling, but also for the dialogue system designer to identify potential flaws in the dialogue system design. The knowledge about errors and their causes can help system designers to identify limitations of system components such as speech recognition and language understanding, but also to identify dialogue situations (patterns) that lead to problems in dialogue. Unfortunately, this kind of analysis is largely done manually which is laborious and expensive. In this thesis, we have presented a first-of-its-kind approach towards automating this task. We have proposed data-driven models for error detection and a scheme for identifying problematic dialogue patters.

The approach, presented in Chapter 5, is based on the assumption that the onus of error in on the dialogue system and therefore error detection should involve detection of not just non-understandings and misunderstandings but also clarification requests based on false assumption and confusing system prompts. Thus, the task of error detection has been formulated as a classification task where the objective is to learn whether a system action is problematic in a given dialogue context.

Three different models for error detection, namely *early*, *late*, and *offline*, have been presented. The early model can be used online to anticipate if the selected system action is likely to be perceived as problematic. The late error detection model is useful for detecting errors online, but after-the-fact, i.e., any misunderstandings made by the system during the interaction. The offline error detection model is meant for error analysis and therefore uses additional manual features (e.g., transcriptions of user utterances).

The models have been trained using automatically extractable features that are accessible to the dialogue systems, covering ASR, SLU, DM, and NLG. In addition, we have used methods to capture discourse events such as corrections and repetitions, which are cues indicating problems in conversations. The features used are generic to dialogue systems and has enabled us to evaluate the proposed models on interaction logs from three different dialogue systems that vary in their dialogue design, tasks and user types. All the three models obtained much better performance in recovering instances of problematic system behavior in the three datasets.

The generic nature of the models has enabled us to show that a model trained on the interaction logs of one system can be used for error detection on logs of another system that share common dialogue strategies. Finally, we have presented a scheme to identify problematic dialogue patterns, which makes use of the models for error detection presented in this work.

7.1.4 Knowledge acquisition

Dialogue systems rely on various knowledge sources for performing various tasks. However, only a limited amount of knowledge can be pre-specified in the knowledge bases available to the agent. Even if knowledge bases are connected to a large body of information, such as the Internet, it cannot be guaranteed that all the knowledge is there and it is correct. This is because we live in an ever changing world and it is not feasible to always keep the knowledge bases up-to-date. Missing or

incomplete knowledge can create gaps in the system's knowledge and could lead to errors in interaction or task failure.

In Chapter 6 of this thesis, we investigated whether dialogue systems can make use of another knowledge source—the resourcefulness of human users. We developed a dialogue system that interacts with the users to acquire information about landmarks in pictures. An important aspect of the automatically acquired information is its reliability. Here again, we have used the interactive setting of spoken dialogue and designed the system to verify (i.e., cross-check) the information with other users. Thanks to the interactions with different users, the system is able to build a reliable hypothesis about the concept values it has acquired, by simply using the majority vote. In this manner, the presented system exhibits the crowdsourcing scheme for soliciting information from non-experts.

The analysis of the data acquired using this approach in an in-lab experiment shows that after interacting with only 6 unique users, the system is able to acquire a correct hypothesis for 96% of the slots. The correctness criterion is here based on the majority vote on the slot-values. In another analysis, which is based on a probabilistic measure of the correctness of slot-values, we observed that interactions with only 4 users are sufficient to learn the color of a building, with a certainty of 80%. Thus, the presented scheme offers encouraging results for conducting the exercise in the real world.

The system also engaged its users in open-ended interactions such as "*Could you describe it?*" whenever the user responded in affirmation to a clarification request such as "*I think there is a shop in this building, is it true?*". The interaction data from the open-ended dialogues has the potential of discovering new knowledge. Endowing dialogue systems with such skills in obtaining information from humans will add increased autonomy to systems.

7.2 Discussion and future work

7.2.1 Integration and further improvements

Although the four methods reported in this thesis have been presented as independent pieces of work, they can be integrated in a dialogue system for enabling more human-like interactions with the users. Doing this would allow us to not only assess their contribution to the system as a whole, but also outline new requirements for further developments. In a first step towards this, we have integrated the model of response location

detection (RLD), trained and evaluated in the Map Task scenario in Chapter 4, for timing the backchannel responses of the system for crowdsourcing street-level geographic information. The RLD model was used during the open-ended dialogues with the users (presented in Chapter 6). Since the model was not retrained for the crowdsourcing task, the interaction data from the open-ended dialogues offers an excellent opportunity to assess the utility of the proposed model of RLD for a different task (domain). We have not analyzed the data yet, but a perception based judgement of the appropriateness of system responses (as used in Section 4.5.1) could be a first simple scheme to assess the model's contributions in the interactions.

In the system for crowdsourcing, we know that during the open-ended dialogues the system does not have a sophisticated model to detect if the user has finished describing the landmarks. We have used a naïve model that relies on a threshold (a random number in the range 2–4) to set the number of turn-exchanges after which the model assumes that the user has finished speaking. This model is not optimal as it predicts turn completion even when the user may want to continue speaking or waits for further inputs even though the user is finished speaking. The instances of abrupt ending of user turns by the system were reported by the users as annoying and gave them the impression that the system was not interested in listening to them. This suggests that developing a sophisticated model for predicting not only when to give feedback, but also whether a user is finished speaking during open-ended interaction, is important for indicating earnestness to the user and ensuring a satisfactory interaction experience.

The Chunking parser based method for spoken language understanding (presented in Chapter 3) can be integrated in the system for crowdsourcing street-level details in order to enhance the level of interactivity during the open-ended interactions. Currently, the crowdsourcing system only gives backchannel responses (e.g., "*okay*", "*uh-hun?*") to the user. However, the Chunking parser can be used to detect presence of landmarks and spatial relations among them. These pieces of concepts can then be used by the system to take initiative in asking questions and give evidences about its knowledge. This can enable a mixed initiative dialogue for information gathering, as imagined in Table 7.1.

The response location detection model can also be integrated in the dialogue system for human–robot interaction scenario, introduced in Chapter 3, where an urban robot seeks route directions from human

Table 7.1: A mixed initiative dialogue for information gathering.

1.	S:	*Is there anything peculiar in this scene?*
2.	U:	well…
3.	S:	*mhm*
4.	U:	There is a tea shop on the corner
5.	S:	*a shop?*
6.	U:	ya, a tea shop
7.	S:	*mhm…Is it to the right of the ATM?*
8.	U:	well no…it's diagonally across the street
9.	S:	*across?*

passers-by. As the human subject describes a route, the robot should be able to provide acknowledgements when expected by them. Enabling this would require adapting the Chunking parser for processing incremental user inputs during the dialogue. This would inevitably require the parser to be trained on features from only the left context. In this system, the response location detection model can be equipped with additional knowledge from the interpretation module for timing the system response, e.g., the information about semantic completeness. This can be extracted from the detection of chunks, route-segments, and the route.

In the qualitative evaluation of the automatically extracted conceptual route graphs, presented in Section 3.7, we have observed that the current method does not deal with phenomena such as repetitions and revisions in user speech. As a consequence, the extracted route graphs contained duplicate chunks or sub-graphs, which in turn affected the human performances in route drawing. There are different potential approaches to resolve the errors due to these speech phenomena. Heeman & Allen (1999) have presented an approach for detection of repairs (revisions) in user utterances. In their approach, the language model for speech recognition has been trained on data containing tags indicating parts that were revised and the corresponding revisions. During speech recognition, the recognized utterance contains tags indicating the parts that are repaired and the revisions. We believe that in the task domain of interpreting route directions, the statistics on the various structures in the conceptual route graph can be used to train a model to detect redundancy in the structure arising due to repairs or revisions. For this, a data-driven approach similar to the *early error detection* model, presented in Chapter

5, would be interesting to explore. The objective of the model would be to assess the correctness of the extracted route graphs and predict whether an error in understanding has occurred. When an error is detected, a dialogue for recovery would need to be initiated. However, this would require the system to have access to various dialogue strategies and a policy for determining when to use them, e.g., whether the system should clarify the specific chunk only, or the complete route segment.

Yet another line of future work is to exploit the prosody in user utterances for further improving the Chunking parser for semantic interpretation of route instructions. As introduced in Section 3.3, the intuition behind the Chunking parser is that human process information in chunks and the information about chunk units is revealed by the prosodic patterns—of stresses interleaved by pauses. In conventional approaches to language understanding (where only the speech recognition result is used for processing) this valuable information about how speakers segment information is lost. We believe prosodic features such as pauses and patterns of pitch contour can be useful cues for detection of route-segments.

7.2.2 Generalization

The majority of the methods presented here follow the supervised learning approach for machine learning. One of the underlying assumptions of this approach to learning is that the training and the validation datasets are from the same distribution as the test data. Thus, if during online use the interactions deviate from what was observed in the training, the model performances are likely to be affected. This assumption also limits the applicability of the models to dialogue tasks from another domain. In the simplest case, the models would need to be retrained on the interaction data from another domain. The generalizability of the presented methods needs to be evaluated in each application domain.

The intuition behind using the model of RLD in the system for crowdsourcing street-level information was that in both tasks, the system is the information receiver while the user is the giver. In both cases the system has to mostly provide feedback while the user retains the initiative. As mentioned earlier, the effectiveness of the RLD model in the crowdsourcing task domains remains to be evaluated. Furthermore, the model of RLD is trained on interaction data from the Map Task where the dialogue initiative is mostly with the user. It would be worthwhile to see if the features and the approach presented in this thesis can be used to

train models for RLD in mixed initiative dialogues. Here, as both the participants make contributions to continue the dialogue, the dynamics of floor management are likely to be different than the one where only one speaker has the initiative. Thus, interaction phenomena such as barge-in and interruptions, which are listener's attempts at taking the conversational floor, might be common. Since system barge-ins and interruptions are intended to disrupt the user's speech (with could be perceived negatively by the user), it will be necessary for the model to account for the effectivity and appropriateness of the strategy to use them.

The approach to Chunking parsing for semantic interpretation has been developed for extracting complex semantic structures from route instructions. However, the models can be used for the detection of domain specific concepts in any other task domain where using a domain ontology is useful. Since the models rely on syntactic and domain level concept features, the models will need to be trained on task specific data. We have investigated the contribution of more generic features such as Part-of-speech (POS). However, as off-the-shelf POS tagging systems are trained on well-formed written text, their performance on speech recognition results is not very encouraging for online use. For this reason, in this thesis, we have trained the models for online use on speech recognition results. The trained models thus had the ability to robustly deal with speech recognition errors.

The models for error detection, presented in Chapter 5, have been tested on interaction logs of three different dialogue systems and were found to be reasonably generic. We have trained the models on many generic measures of sub-component performances e.g., number of words in speech recognition and parse results, and also more abstract features such as corrections and discourse progression. This has enabled us to test models trained on one dataset for error detection on another dataset. Although the models have relied on more direct cues of errors, such as speech recognition failure or issues in parsing, we believe more abstract cues, such as repetition, correction, new contributions, would be useful for error detection in other task domains. In this work, we have used lexical features to measure user repetitions (e.g., computing cosine similarity between user turns); however, since ASR can be problematic, the estimates may not be robust for online models. In this case, using alternative method for repetition detection, such as phonetic distance (Lopes et al., 2015), can possibly offer robustness.

One downside of supervised learning is that it requires labelled training examples. Annotating data can be laborious and expensive. While internet

based crowdsourcing offers a scheme for cheaper and quicker annotations, it may not be suitable for all the tasks, e.g., annotating geographic databases with visually salient details of landmarks. The approach to implicitly-supervised learning for acquiring street-level information, presented in Chapter 6, suggests an interesting direction for dialogue system development where the system learns from direct supervision of users. Bohus & Rudnicky (2007) have used the implicitly-supervised learning for training models for error detection. It would be interesting to explore other dialogue modelling tasks where this methodology can be useful.

Appendix A Details on error detection

A.1 List of features for error detection

In this section, we present an exhaustive list of features used in Chapter 5, for training models for error detection. The list is made available in Table A.2. The table contains the feature name and its description. The name for feature also encodes some details, e.g. which turn in the context window it was extracted from (1–4), whether it is extracted from ASR (asr) or manual transcriptions (tr), whether it is extracted from speaker turns (utter) or the parse result of language understanding. An example context window of four turns is illustrated in Table A.1 for ease of decoding the feature names along with the descriptions.

Table A.1: A context window of four turns

1	S:	*How may I help you?*
2	U:	Sixty One D
3	S:	*The 61C.What's the departure station?*
4	U:	No

Table A.2 An exhaustive list of features used to train the models for error detection

#	*Feature name and descriptions. Source of extraction*	
	ASR confidence score of user turn	
1	asr-conf-usr-2	ASR
2	asr-conf-usr-4	ASR
	Word Error Rate (WER): The Levenshtein distance between output of speech recognition results and the transcriptions	
3	wer-tr-usr-2	Transcription
4	wer-tr-usr-4	Transcription
5	dlen-pcplt-usr-4	Percentage of dialogue completed until user turn-4

A.1 List of features for error detection

	Concept error rate (CER): the Levenshtein distance between the NLU parses obtained on speech recognition results and transcription of the user turn, respectively	
6	cer-psltvals-usr-2	Transcription
7	cer-psltvals-usr-4	Transcription
	Number of concepts: in the NLU parses for the user turn, system dialogue acts, system prompts and user turns	
8	cnum-asr_dact-usr-2	NLU on ASR output
9	cnum-asr_dact-usr-4	NLU on ASR output
10	cnum-tr-dact-usr-2	NLU on Transcription
11	cnum-tr-dact-usr-4	NLU on Transcription
12	cnum-dact-sys-1	System dialogue act (Sys DAct)
13	cnum-dact-sys-3	System dialogue act (Sys DAct)
14	cnum-utter-sys-1	System prompt
15	cnum-utter-sys-3	System prompt
16	cnum-utter-asr-usr-2	ASR
17	cnum-utter-asr-usr-4	ASR
18	cnum-utter-tr-usr-2	Transcription
19	cnum-utter-tr-usr-4	Transcription
	Correctly transferred concepts (CTC): the fraction of concepts in NLU parse output of transcriptions that were observed in the NLU parse of speech recognition results	
20	ctc-psltvalprs-usr-2	Transcription
21	ctc-psltvalprs-usr-4	Transcription
	Correctly Transferred Words (CTW): The fraction of words in transcription of user turn that were observed in the ASR output	
22	ctw-tr-usr-2	Transcription
23	ctw-tr-usr-4	Transcription
	Speaker repetition: The **cosine similarity** score of the current (speaker) turn with the last (same speaker) turn. Measured from dialogue acts (NLU)	
24	dact-cosine-sim_p1-sys-1	System dialogue act
25	dact-cosine-sim_p1-sys-3	System dialogue act
26	dact-cosine-sim_p1-usr-2	NLU on ASR output

27	dact-cosine-sim_p1-usr-4	NLU on ASR output
28	dact-tr-cosine-sim_p1-usr-2	NLU on Transcription
29	dact-tr-cosine-sim_p1-usr-4	NLU on Transcription
30	utter-asr-cosine-sim_p1-usr-2	ASR
31	utter-asr-cosine-sim_p1-usr-4	ASR
32	utter-cosine-sim_p1-sys-1	System prompt
33	utter-cosine-sim_p1-sys-3	System prompt
34	utter-tr-cosine-sim_p1-usr-2	Transcription
35	utter-tr-cosine-sim_p1-usr-4	Transcription
	Speaker repetition: Number of slot types in current speaker turn repeated from the last turn (same speaker)	
36	num_repeated_slots-asr-pslts-usr-2	NLU on ASR output
37	num_repeated_slots-asr-pslts-usr-4	NLU on ASR output
38	num_repeated_slots-pslts-sys-1	System dialogue act
39	num_repeated_slots-pslts-sys-3	System dialogue act
40	num_repeated_slots-tr-pslts-usr-2	NLU on Transcription
41	num_repeated_slots-tr-pslts-usr-4	NLU on Transcription
	Speaker repetition: Number of words in the current speaker turn repeated from the last turn (of the same speaker)	
42	num_repeated_words-asr-usr-2	ASR
43	num_repeated_words-asr-usr-4	ASR
44	num_repeated_words-sys-1	System prompt
45	num_repeated_words-sys-3	System prompt
46	num_repeated_words-tr-usr-2	Transcription
47	num_repeated_words-tr-usr-4	Transcription
	New Information: Fraction of slot-types in the current speaker turn that were not observed in the last (not the same) speaker's turn	
48	frc_newinfo-asr_slts-usr-2	NLU on ASR output
49	frc_newinfo-asr_slts-usr-4	NLU on ASR output
50	frc_newinfo-psltvals-sys-1	System dialogue act
51	frc_newinfo-psltvals-sys-3	System dialogue act
52	frc_newinfo-tr_slts-usr-2	NLU on Transcription
53	frc_newinfo-tr_slts-usr-4	NLU on Transcription

54	frc_newinfo-utter-asr-usr-2	ASR
55	frc_newinfo-utter-asr-usr-4	ASR
56	frc_newinfo-utter-sys-1	System prompt
57	frc_newinfo-utter-sys-3	System prompt
58	frc_newinfo-utter-tr-usr-2	Transcription
59	frc_newinfo-utter-tr-usr-4	Transcription
	Marked disconfirmation: whether the user response is a marked disconfirmation to last system clarification request.	
60	mark-disconfirm-asr-usr-2	NLU on ASR output
61	mark-disconfirm-asr-usr-4	NLU on ASR output
62	mark-disconfirm-tr-usr-2	NLU on Transcription
63	mark-disconfirm-tr-usr-4	NLU on Transcription
64	mark-disconfirm-utter-asr-usr-2	ASR
65	mark-disconfirm-utter-asr-usr-4	ASR
66	mark-disconfirm-utter-tr-usr-2	Transcription
67	mark-disconfirm-utter-tr-usr-4	Transcription
	Rectifications: Number of slot-types in the last system turn that have changed slot-value in the current turn	
68	num-sys-corrects-self-sys-1	System dialogue act
69	num-sys-corrects-self-sys-3	System dialogue act
	Misunderstandings: Number of slot-types in the last user turn that have a different slot-value in the current system turn	
70	num-sys-misundrstd-usr-asr-1	Sys DAct, NLU on ASR output
71	num-sys-misundrstd-usr-asr-3	Sys DAct, NLU on ASR output
72	num-sys-misundrstd-usr-tr-1	Sys DAct, NLU on Transcription
73	num-sys-misundrstd-usr-tr-3	Sys DAct, NLU on Transcription
	Change-of-intention: Number of slot-types in last user turn that have changed slot-value in turn	
74	num-usr-change-goal-asr-usr-2	NLU on ASR output
75	num-usr-change-goal-asr-usr-4	NLU on ASR output
76	num-usr-change-goal-tr-usr-2	NLU on Transcription
77	num-usr-change-goal-tr-usr-4	NLU on Transcription
	User corrections: Number of slot-types in last system turn that have	

	changed slot-value in current user turn	
78	num-usr-corrects-sys-asr-2	Sys DAct, NLU on ASR output
79	num-usr-corrects-sys-asr-4	Sys DAct, NLU on ASR output
80	num-usr-corrects-sys-tr-2	Sys DAct, NLU on Transcription
81	num-usr-corrects-sys-tr-4	Sys DAct, NLU on Transcription
	Word-vector representation of speaker dialogue act	
82	dact-sys-1	System dialogue act
83	dact-sys-3	System dialogue act
84	dact-tr-usr-2	NLU on Transcription
85	dact-tr-usr-4	NLU on Transcription
86	dact-usr-2	NLU on ASR output
87	dact-usr-4	NLU on ASR output

A.2 JRip rules for error detection

A.2.1. Early error detection model

CamInfo dataset

1. (cnum-asr_dact-usr-2 <= 1) and (cnum-utter-asr-usr-2 >= 4) => class=problematic (80.0/8.0)
2. (2-dact_request <= 0) and (num_repeated_slots-pslts-sys-3 >= 1) and (cnum-asr_dact-usr-2 <= 1) and (frc_newinfo-psltvals-sys-3 >= 0.571429) => class=problematic (11.0/0.0)
3. (2-dact_request <= 0) and (utter-asr-cosine-sim_p1-usr-2 <= 0.365148) and (frc_newinfo-asr_slts-usr-2 <= 0.5) => class=problematic (45.0/13.0)
4. (cnum-utter-sys-3 >= 24) and (3-dact_food >= 1) => class=problematic (31.0/11.0)
5. (asr-conf-usr-2 <= 0.970373) and (frc_newinfo-utter-sys-3 <= 0.333333) => class=problematic (8.0/1.0)
6. (3-dact_name <= 0) => class=problematic (21.0/10.0)
7. (cnum-utter-sys-3 >= 17) and (frc_newinfo-psltvals-sys-1 <= 0.571429) and (cnum-dact-sys-3 >= 9) and (2-dact_area <= 0) => class=problematic (23.0/8.0)
8. class=notproblematic (640.0/44.0)

The first of the two figures in brackets indicate total number of instances covered by the rule and the second figure is the number of false cases among the covered instances.

Using these eight rules, the JRip model for early error detection obtains a UAR of 0.80 for the CamInfo dataset.

Let's Go dataset

1. (3-dact_request <= 0) and (num_repeated_slots-pslts-sys-3 >= 1) => class=problematic (127.0/22.0)
2. (3-dact_request <= 0) and (dlen-pcplt-sys-3 <= 74) and (frc_newinfo-psltvals-sys-1 <= 0.666667) => class=problematic (109.0/34.0)
3. (num_repeated_slots-pslts-sys-3 >= 1) and (cnum-utter-sys-3 <= 16) => class=problematic (43.0/12.0)
4. (cnum-dact-sys-1 <= 2) and (frc_newinfo-utter-sys-3 >= 0.926829) => class=problematic (37.0/12.0)
5. class=notproblematic (428.0/80.0)

Using these five rules, the JRip model for early error detection obtains a UAR of 0.70 for the Let's Go dataset.

SweCC dataset

1. (frc_newinfo-utter-sys-3 <= 0.5) and (dlen-pcplt-sys-3 <= 87.5) and (dlen-pcplt-sys-3 >= 43.75) and (3-dact_request <= 0) and (cnum-asr_dact-usr-2 <= 1) => class=problematic (50.0/0.0)
2. (frc_newinfo-utter-sys-3 <= 0.5) and (cnum-utter-asr-usr-2 >= 7) and (dlen-pcplt-sys-3 <= 87.5) => class=problematic (153.0/20.0)
3. (3-dact_confirm >= 1) and (frc_newinfo-asr_slts-usr-2 <= 0) => class=problematic (56.0/17.0)
4. (cnum-dact-sys-1 <= 2) and (asr-conf-usr-2 <= 0.62) and (dlen-pcplt-sys-3 >= 50) and (cnum-utter-asr-usr-2 >= 8) => class=problematic (17.0/4.0)
5. (cnum-dact-sys-1 <= 2) and (num_repeated_words-asr-usr-2 >= 5) and (cnum-utter-sys-3 >= 6) and (asr-conf-usr-2 >= 0.55) and (asr-conf-usr-2 <= 0.61) => class=problematic (9.0/0.0)

6. (asr-conf-usr-2 <= 0.47) and (dlen-pcplt-sys-3 >= 38.888889) and (cnum-utter-sys-3 <= 9) => class=problematic (8.0/0.0)
7. (cnum-utter-asr-usr-2 >= 3) and (asr-conf-usr-2 <= 0.35) => class=problematic (6.0/1.0)
8. class=notproblematic (572.0/42.0)

Using these eight rules, the JRip model for early error detection obtains a UAR of 0.84 for the SweCC dataset.

A.2.2. Late error detection model

CamInfo dataset

1. (cnum-asr_dact-usr-2 <= 1) and (num-usr-corrects-sys-asr-2 >= 0) => class=problematic (56.0/2.0)
2. (cnum-asr_dact-usr-2 <= 1) and (utter-cosine-sim_p1-sys-1 >= 0.308607) => class=problematic (43.0/3.0)
3. (frc_newinfo-asr_slts-usr-4 <= 0.666667) and (cnum-utter-sys-3 >= 17) => class=problematic (48.0/17.0)
4. (dact-cosine-sim_p1-sys-3 >= 1) and (asr-conf-usr-2 <= 0.993137) and (cnum-utter-asr-usr-4 >= 4) => class=problematic (14.0/3.0)
5. (2-dact_request <= 0) and (cnum-asr_dact-usr-2 <= 3) and (cnum-utter-sys-1 <= 8) and (frc_newinfo-psltvals-sys-3 >= 0.545455) => class=problematic (26.0/5.0)
6. (4-dact_missing_s_type >= 1) and (asr-conf-usr-2 <= 0.990746) and (num_repeated_words-asr-usr-4 >= 1) => class=problematic (7.0/0.0)
7. class=notproblematic (665.0/48.0)

Using these seven rules, the JRip model for late error detection obtains a UAR of 0.78 for the CamInfo dataset.

Let's Go dataset

1. (mark-disconfirm-utter-asr-usr-4 >= 0) and (4-dact_yes <= 0) => class=problematic (253.0/35.0)
2. (frc_newinfo-psltvals-sys-3 <= 0) and (cnum-utter-sys-3 <= 9) and (dact-cosim-sys-1 >= 1) => class=problematic (24.0/0.0)
3. (cnum-asr_dact-usr-2 <= 0) => class=problematic (41.0/15.0)

4. class=notproblematic (426.0/48.0)

Using these four rules, the JRip model for late error detection obtains a UAR of 0.84 for the Let's Go dataset.

SweCC dataset

1. (4-dact_no >= 1) => class=problematic (258.0/31.0)
2. (frc_newinfo-utter-asr-usr-4 <= 0.9) and (frc_newinfo-utter-sys-3 <= 0.5) => class=problematic (52.0/5.0)
3. (cnum-utter-asr-usr-2 >= 8) and (cnum-utter-asr-usr-4 >= 2) => class=problematic (8.0/2.0)
4. class=notproblematic (553.0/19.0)

Using these four rules, the JRip model for late error detection obtains a UAR of 0.93 for the SweCC dataset.

A.2.3. Offline error detection model

CamInfo dataset

1. (ctc-psltvalprs-2 <= 0.5) and (frc_newinfo-asr_slts-usr-2 <= 0.5) => class=problematic (83.0/9.0)
2. (ctc-psltvalprs-2 <= 0.666667) and (utter-tr-cosine-sim_p1-usr-4 >= 0.258199) and (frc_newinfo-utter-asr-usr-4 >= 0.7) => class=problematic (33.0/5.0)
3. (cnum-asr_dact-usr-2 <= 1) and (utter-cosine-sim_p1-sys-1 >= 0.323381) => class=problematic (29.0/3.0)
4. (ctc-psltvalprs-2 <= 0.75) and (frc_newinfo-utter-asr-usr-4 <= 0.6) and (2-dact-tr_missing_s_type >= 1) => class=problematic (12.0/1.0)
5. (dact-cosine-sim_p1-usr-4 >= 0.408248) and (cnum-utter-sys-3 >= 15) => class=problematic (33.0/14.0)
6. (ctw-tr-2 <= 0.4) and (wer-tr-2 >= 100) and (wer-tr-2 <= 125) and (asr-conf-4 <= 0.998421) => class=problematic (18.0/4.0)
7. class=notproblematic (651.0/40.0)

Using these seven rules, the JRip model for offline error detection obtains a UAR of 0.82 for the CamInfo dataset.

Let's Go dataset

1. (wer-tr-usr-2 >= 20) and (4-dact-tr_no >= 1) => class=problematic (121.0/3.0)
2. (ctc-psltvalprs-usr-2 <= 0.5) and (4-dact-tr_yes <= 0) => class=problematic (115.0/23.0)
3. (mark-disconfirm-utter-tr-usr-4 >= 0) and (4-dact-tr_yes <= 0) and (cnum-utter-sys-3 <= 8) => class=problematic (74.0/13.0)
4. (cnum-tr-dact-usr-2 <= 0) and (frc_newinfo-psltvals-sys-1 <= 0.666667) and (num_repeated_slots-pslts-sys-3 >= 1) => class=problematic (23.0/3.0)
5. class=notproblematic (411.0/25.0)

Using these five rules, the JRip model for offline error detection obtains a UAR of 0.87 for the Let's Go dataset.

SweCC dataset

1. (4-dact_no >= 1) => class=problematic (258.0/31.0)
2. (frc_newinfo-utter-sys-3 <= 0.5) and (frc_newinfo-utter-asr-usr-4 <= 0.9) => class=problematic (52.0/5.0)
3. (wer-tr-usr-2 >= 10) and (asr-conf-usr-4 <= 0.48) and (frc_newinfo-utter-sys-3 <= 0.909091) => class=problematic (14.0/4.0)
4. class=notproblematic (547.0/15.0)

Using these four rules, the JRip model for offline error detection obtains a UAR of 0.93 for the SweCC dataset.

A.3 JRip rules for cross-corpus error detection

A.3.1. Train on CamInfo dataset and test on Let's Go dataset

There are 95 features that are common between the two datasets.

1. (cnum-asr_dact-usr-2 <= 1) and (utter-cosine-sim_p1-sys-1 >= 0.267261) => class=problematic (88.0/7.0)

2. (ctc-psltvalprs-2 <= 0.666667) and (ctc-psltvalprs-2 <= 0.333333) => class=problematic (83.0/24.0)
3. (frc_newinfo-tr_slts-usr-4 <= 0.5) and (cer-psltvals-2 >= 36.363636) and (frc_newinfo-utter-asr-usr-4 <= 0.714286) => class=problematic (18.0/2.0)
4. (cnum-utter-asr-sys-3 >= 29) and (cnum-utter-asr-sys-1 >= 31) => class=problematic (11.0/3.0)
5. (cnum-tr-dact-usr-2 <= 0) and (ctw-tr-2 <= 0.333333) => class=problematic (10.0/2.0)
6. class=notproblematic (649.0/40.0)

Using these 6 rules, the JRip model for offline error detection trained on CamInfo dataset obtains a UAR of 0.62 on the Let's Go dataset.

A.3.2. Train on CamInfo dataset and test on SweCC dataset

There are 69 features that are common between the two datasets.

1. (cnum-asr_dact-usr-2 <= 1) and (utter-cosine-sim_p1-sys-1 >= 0.267261) => class=problematic (88.0/7.0)
2. (ctw-tr-2 <= 0.571429) and (ctw-tr-2 <= 0.111111) and (frc_newinfo-utter-tr-usr-4 <= 0.833333) => class=problematic (26.0/1.0)
3. (num-usr-corrects-sys-asr-2 >= 0) and (cnum-asr_dact-usr-2 <= 0) => class=problematic (12.0/0.0)
4. (dact-cosine-sim_p1-usr-4 >= 0.57735) and (wer-tr-4 >= 50) and (cnum-utter-asr-sys-3 >= 16) => class=problematic (14.0/1.0)
5. (frc_newinfo-utter-sys-3 <= 0.333333) and (frc_newinfo-utter-asr-usr-2 >= 0.833333) and (asr-conf-2 <= 0.980979) => class=problematic (12.0/2.0)
6. (dact-cosine-sim_p1-usr-4 >= 0.707107) and (num_repeated_words-asr-usr-2 >= 9) => class=problematic (5.0/0.0)
7. (cnum-utter-asr-sys-3 >= 29) and (cnum-utter-asr-sys-1 >= 30) => class=problematic (11.0/3.0)

8. class=notproblematic (691.0/58.0)

Using these 8 rules, the JRip model for offline error detection trained on CamInfo dataset obtains a UAR of 0.50 on the Let's Go dataset.

A.3.3. Train on Let's Go dataset and test on CamInfo dataset

There are 95 features that are common between the two datasets.

1. (wer-tr-2 >= 20) and (mark-disconfirm-tr-4 >= 1) => class=problematic (114.0/3.0)
2. (num_repeated_slots-pslts-sys-3 >= 1) and (cnum-utter-asr-sys-1 <= 18) and (num-sys-corrects-self-sys-3 <= 0) and (dact-cosine-sim_p1-sys-1 >= 1) => class=problematic (56.0/4.0)
3. (ctc-psltvalprs-2 <= 0.5) and (ctw-tr-2 <= 0.333333) => class=problematic (60.0/12.0)
4. (mark-disconfirm-tr-4 >= 1) => class=problematic (51.0/12.0)
5. (wer-tr-2 >= 50) and (num-usr-corrects-sys-asr-4 >= 0) => class=problematic (22.0/3.0)
6. (dact-tr-cosine-sim_p1-usr-4 >= 0.816497) and (wer-tr-2 >= 25) and (frc_newinfo-psltvals-sys-1 <= 0.5) => class=problematic (10.0/1.0)
7. (frc_newinfo-utter-tr-usr-4 <= 0.333333) and (cnum-utter-asr-sys-3 <= 8) => class=problematic (24.0/9.0)
8. class=notproblematic (407.0/23.0)

Using these 8 rules, the JRip model for offline error detection trained on Let's Go dataset obtains a UAR of 0.72 on the CamInfo dataset.

A.3.4. Train on Let's Go dataset and test on SweCC dataset

There are 75 features that are common between the two datasets

1. (wer-tr-2 >= 20) and (mark-disconfirm-utter-tr-usr-4 >= 1) => class=problematic (112.0/3.0)

2. (wer-tr-2 >= 25) and (utter-tr-cosine-sim_p1-usr-4 >= 0.447214) and (frc_newinfo-psltvals-sys-1 <= 0.666667) => class=problematic (35.0/2.0)
3. (wer-tr-2 >= 60) and (num-usr-corrects-sys-asr-4 >= 0) => class=problematic (35.0/5.0)
4. (num_repeated_slots-pslts-sys-3 >= 1) and (cnum-utter-asr-sys-3 <= 6) => class=problematic (35.0/4.0)
5. (4-dact_no >= 1) => class=problematic (52.0/13.0)
6. (wer-tr-2 >= 20) and (frc_newinfo-asr_slts-usr-4 >= 1) and (dlen-pcplt-usr-4 <= 45) => class=problematic (24.0/7.0)
7. (num-usr-corrects-sys-asr-4 >= 0) and (utter-tr-cosine-sim_p1-usr-4 >= 0.408248) => class=problematic (14.0/3.0)
8. (mark-disconfirm-utter-tr-usr-4 >= 1) => class=problematic (7.0/1.0)
9. (ctw-tr-2 <= 0.8) and (utter-cosine-sim_p1-sys-3 <= 0.070711) => class=problematic (10.0/2.0)
10. (num_repeated_slots-pslts-sys-3 >= 1) and (frc_newinfo-utter-sys-1 >= 0.625) and (dact-cosine-sim_p1-sys-1 >= 1) and (dlen-pcplt-usr-4 >= 56.25) => class=problematic (9.0/0.0)
11. class=notproblematic (411.0/23.0)

Using these 11 rules, the JRip model for offline error detection trained on Let's Go dataset obtains a UAR of 0.89 on the SweCC dataset.

A.3.5. Train on SweCC dataset and test on CamInfo dataset

These two datasets have 69 features in common.

1. (mark-disconfirm-asr-4 >= 1) => class=problematic (254.0/30.0)
2. (frc_newinfo-utter-sys-3 <= 0.5) and (frc_newinfo-utter-asr-usr-4 <= 0.9) => class=problematic (52.0/5.0)
3. (wer-tr-2 >= 10) and (asr-conf-4 <= 0.52) and (wer-tr-2 <= 30) => class=problematic (7.0/0.0)
4. class=notproblematic (558.0/21.0)

Using these 4 rules, the JRip model for offline error detection trained on SweCC dataset obtains a UAR of 0.54 on the CamInfo dataset.

A.3.6. Train on SweCC dataset test on Let's Go dataset

There are 75 features common between these two datasets.

1. (4-dact_no >= 1) => class=problematic (258.0/31.0)
2. (frc_newinfo-utter-sys-3 <= 0.5) and (frc_newinfo-utter-asr-usr-4 <= 0.9) => class=problematic (52.0/5.0)
3. (wer-tr-2 >= 10) and (wer-tr-4 >= 75) and (frc_newinfo-utter-asr-usr-4 >= 1) and (cnum-utter-asr-usr-2 >= 3) => class=problematic (12.0/1.0)
4. class=notproblematic (549.0/14.0)

Using these 4 rules, the JRip model for offline error detection trained on SweCC dataset obtains a UAR of 0.73 on the Let's Go dataset.

Appendix B Crowdsourcing system

A list of all the system response types used by the dialogue system for crowdsourcing street-level geographic details, presented in Chapter 6, are reported here, in Table B.1.

Table B.1: The table below lists slots (= landmark attributes) and the corresponding wh– and yes–no system questions. For attributes marked with * the dialogue manager switches to open-ended interaction mode.

Slot (=attribute)	***System wh– questions***	***System yes–no questions***
Visible: whether a particular landmark is visible from this view.	• Do you see a building on the far left? • Do you see another building in front of you? • Is there a junction on the right? • Do you see a traffic-signal ahead?	• Is the building on the far right visible to you? • I think there is another building in front of you, do you see it? • Can you see the junction on the right? • Are you able to see the traffic-signal ahead?
Color of the building	• What color does the building have? • What color is the building?	• I think this building is *red* in color, what do you think? • Does this building have *red* color?
Size of the building (in number of stories)	• How many floors do you think are there in this building • How many stories is this building	• I think there are *six* floors in this building, what do you think? • Is this building *six* storied?
Color of the building's *roof*	• What color does the roof of this building have? • What color is the roof of this building?	• I think the roof of this building is *orange* in color, what do you think? • Do you think that the roof of this building is *orange*?

Signs or ornamentation on the building	• Do you see any signs or decorations on this building?	• I think there is a sign or some decoration on this building, do you see it? • There may be a sign or a name on this building, do you see it?
Shops or restaurants in the building	• Are there any shops or restaurants in this building?	• I am informed that there are some shops or restaurants in this building, is it true? • I think there are some shops or restaurants in this building, what do you think?
Signs at landmarks	• Are there any important signs at the junction/crossing?	• I believe there is a sign at this junction/crossing, do you see it? • Do you see the sign at this junction/crossing?
**Description* of sign	• Could you describe this sign? • What does this sign look like? • Does the sign say something?	• Could you describe this sign? • What does this sign look like? • Does the sign say something?
**Signs* in the visual scene	• Are there any important signs in this scene that I may have missed? • Have I missed any relevant signs in this scene?	• There are some important signs in this scene that could be useful for my knowledge, am I right? • I am informed that there are some signs in this scene that are relevant for me, is it true?
**Landmarks* in the visual scene	• Are there any other important buildings or relevant structures in this scene that I should be aware of? • Is there anything particular in this scene that I should be familiar with? • Have I missed any	• I am informed that there are some important landmarks or structures in this scene that I should be aware of, is it true? • I have been told that there are some other things in this scene that I are relevant for me, is

	relevant buildings or landmarks in this scene?	it true? • I believe I have missed some relevant landmarks in this scene, am I right?
Description of unknown *landmarks* e.g. shop, restaurant, building, etc.	• Could you describe it? • Could you describe them? • How do they look like?	• Could you describe it? • Could you describe them? • How do they look like?

Bibliography

Abney, S. (1991). Parsing by chunks. In Berwick, R. C., Abney, S. P., & Tenny, C. (Eds.), *Principle-Based Parsing: Computation and Psycholinguistics* (pp. 257–278). Dordrecht: Kluwer.

Abney, S. (1997). Part-of-Speech Tagging and Partial Parsing. In Young, S., & Bloothooft, G. (Eds.), *Corpus-Based Methods in Language and Speech Processing* (pp. 118-136). Springer Netherlands.

Allen, J. F., & Core, M. (1997). *Draft of DAMSL: Dialog act markup in several layers.* Unpublished manuscript.

Allen, J. F., & Perrault, C. R. (1980). Analyzing intention in utterances. *Artificial Intelligence, 15*(3), 143-178.

Allen, J. F., Ferguson, G., & Stent, A. (2001). An architecture for more realistic conversational systems. In *Proceedings of the 6th international conference on Intelligent user interfaces (IUI-01)* (pp. 1-8). Santa Fe, NM, US.

Allwood, J., Nivre, J., & Ahlsen, E. (1992). On the semantics and pragmatics of linguistic feedback. *Journal of Semantics, 9*(1), 1-26.

Anderson, A., Bader, M., Bard, E., Boyle, E., Doherty, G., Garrod, S., Isard, S., Kowtko, J., McAllister, J., Miller, J., Sotillo, C., Thompson, H., & 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.

Bartie, P. J., & Mackaness, W. A. (2006). Development of a Speech-Based Augmented Reality System to Support Exploration of Cityscape. *Transactions in GIS, 10*(1), 63-86.

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.

Black, A. W., Burger, S., Langner, B., Parent, G., & Eskenazi, M. (2010). Spoken Dialog Challenge 2010.. In Hakkani-Tür, D., & Ostendorf, M. (Eds.), *SLT* (pp. 448-453). IEEE.

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). 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. (2005b). 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.

Bohus, D. (2007). *Error awareness and recovery in conversational spoken language interfaces*. Doctoral dissertation, Carnegie Mellon University, Pittsburgh, PA.

Boye, J., Fredriksson, M., Götze, J., Gustafson, J., & Königsmann, J. (2014). Walk This Way: Spatial Grounding for City Exploration. In Mariani, J., Rosset, S., Garnier-Rizet, M., & Devillers, L. (Eds.), *Natural Interaction with Robots, Knowbots and Smartphones* (pp. 59-67). Springer New York.

Bugmann, G., Lauria, S., Kyriacou, T., Klein, E., Bos, J., & Coventry, K. (2001). Using Verbal Instructions for Route Learning: Instruction Analysis. In *Proceedings of TIMR 01 - Towards Intelligent Mobile Robots*. Manchester.

Bugmann, G., Klein, E., Lauria, S., & Kyriacou, T. (2004). Corpus-Based Robotics: A Route Instruction Example. In Groen, F. (Ed.), *Intelligent autonomous systems 8* (pp. 96-103).

Bulyko, I., Kirchhoff, K., Ostendorf, M., & Goldberg, J. (2005). Error-correction detection and response generation in a spoken dialogue system. *Speech Communication, 45*(3), 271-288.

Cathcart, N., Carletta, J., & Klein, E. (2003). A shallow model of backchannel continuers in spoken dialogue. In *10th Conference of the European Chapter of the Association for Computational Linguistics*. Budapest.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. *AAAI/IAAI*, *2005*(598-603), 18.

Clark, H. H., & Schaefer, E. F. (1989). Contributing to discourse. *Cognitive Science, 13*(2), 259-294.

Clark, H. H. (1996). *Using language.* Cambridge, UK: Cambridge University Press.

Cohen, W. (1995). Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of ACL* (pp. 1-8). Philadelphia, PA.

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).

Daniel, M-P., & Denis, M. (1998). Spatial Descriptions as Navigational Aids: A Cognitive Analysis of Route Directions Wegauskünfte als Navigationshilfen - eine kognitive Analyse. *Kognitionswissenschaft, 7*(1), 45-52.

De Mori, R., Bechet, F., Hakkani-Tur, D., McTear, M., Riccardi, G., & Tur, G. (2008). Spoken language understanding. *Signal Processing Magazine, IEEE, 25*(3), 50-58.

Dickens, L., & Lupu, E. (2014). *Trust service final deliverable report*. Technical Report, Imperial College, UK.

Domingos, P. (2012). A Few Useful Things to Know About Machine Learning. *Communications of the ACM, 55*(10), 78-87.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern Classification.* Wiley.

Duncan, S. (1972). Some Signals and Rules for Taking Speaking Turns in Conversations. *Journal of Personality and Social Psychology, 23*(2), 283-292.

Edlund, J. (2011). *In search of the conversational homunculus - serving to understand spoken human face-to-face interaction*. Doctoral dissertation, KTH.

Edlund, J., Al Moubayed, S., Tånnander, C., & Gustafson, J. (2013). Temporal precision and reliability of audience response system based annotation. In *Proc. of Multimodal Corpora 2013*. Edinburgh, UK.

Filisko, E., & Seneff, S. (2005). Developing city name acquisition strategies in spoken dialogue systems via user simulation. In *Proceedings of 6th SIGDial Workshop on Discourse and Dialogue*.

Flycht-Eriksson, A. (2001). *Domain knowledge management in information-providing dialogue systems*. Licentiate dissertation, Linköping University.

Frampton, M., & Lemon, O. (2009). Recent research advances in reinforcement learning in spoken dialogue systems. *Knowledge Engineering Review, 24*(4), 375-408.

Gravano, A., & Hirschberg, J. (2011). Turn-taking cues in task-oriented dialogue. *Computer Speech & Language, 25*(3), 601-634.

Grosz, B. J., & Sidner, C. L. (1986). Attention, intentions, and the structure of discourse. *Computational Linguistics, 12*(3), 175-204.

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.

Haklay, M., & Weber, P. (2008). OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Computing, 7*(4), 12-18.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations, 11*(1).

He, Y., & Young, S. (2006). Spoken language understanding using the hidden vector state model. *Speech Communication, 48*(3-4), 262-275.

Heeman, P., & Allen, J. (1999). Speech repairs, intonational phrases, and discourse markers. *Computational Linguistics, 25*(4), 527-571.

Higashinaka, R., Minami, Y., Dohsaka, K., & Meguro, T. (2010). Modeling User Satisfaction Transitions in Dialogues from Overall Ratings. In *Proceedings of the SIGDIAL 2010 Conference* (pp. 18-27). Tokyo, Japan: Association for Computational Linguistics.

Hirschberg, J., Litman, D., & Swerts, M. (2004). Prosodic and Other Cues to Speech Recognition Failures. *Speech Communication, 43*, 155-175.

Hirst, G., McRoy, S., Heeman, P., Edmonds, P., & Horton, D. (1994). Repairing conversational misunderstandings and non-understandings. *Speech Communication, 15*, 213-230.

Hone, K. S., & Baber, C. (1995). Using a simulation method to predict the transaction time effects of applying alternative levels of constraint to user utterances within speech interactive dialogues. In *ESCA Workshop on Spoken Dialogue Systems* (pp. 209-212).

Huang, X., Acero, A., & Hon, H-W. (2001). *Spoken language processing: a guide to theory, algorithm and system development.* Prentice Hall.

Huang, L., Morency, L-P., & Gratch, J. (2011). Virtual Rapport 2.0. In *Intelligent Virtual Agents* (pp. 68-79). Reykjavik, Iceland.

Janarthanam, S., Lemon, O., Liu, X., Bartie, P., Mackaness, W., Dalmas, T., & Goetze, J. (2012). Integrating Location, Visibility, and Question-Answering in a Spoken Dialogue System for Pedestrian City Exploration. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and*

Dialogue (pp. 134-136). Seoul, South Korea: Association for Computational Linguistics.

Johansson, M., & Skantze, G. (2015). Opportunities and Obligations to Take Turns in Collaborative Multi-Party Human-Robot Interaction. In *Proceedings of SIGDIAL*. Prague, Czech Republic.

Johansson, M., Skantze, G., & Gustafson, J. (2011). Understanding route directions in human-robot dialogue. In *Proceedings of SemDial* (pp. 19-27). Los Angeles, CA.

Johnson-Roberson, M., Bohg, J., Skantze, G., Gustafson, J., Carlson, R., Rasolzadeh, B., & Kragic, D. (2011). Enhanced Visual Scene Understanding through Human-Robot Dialog. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Jokinen, K., & Hurtig, T. (2006). User expectations and real experience on a multimodal interactive system.. In *INTERSPEECH*. ISCA.

Jurafsky, D., & Martin, J. (2000). *Speech and language processing.* Englewood, NJ, US: Prentice Hall, Inc.

Jurcicek, F., Thomson, B., & Young, S. (2012). Reinforcement learning for parameter estimation in statistical spoken dialogue systems. *Computer Speech & Language, 26*(3), 168-192.

Keizer, S., Gasic, M., Jurcicek, F., Mairesse, F., Thomson, B., Yu, K., & Young, S. (2010). Parameter estimation for agenda-based user simulation. In *Proceedings of the SIGDIAL 2010 Conference* (pp. 116-123). Tokyo, Japan: Association for Computational Linguistics.

Koiso, H., Horiuchi, Y., Tutiya, S., Ichikawa, A., & Den, Y. (1998). An analysis of turn-taking and backchannels based on prosodic and syntactic features in Japanese Map Task dialogs. *Language and Speech, 41*, 295-321.

Kollar, T., Tellex, S., Roy, D., & Roy, N. (2010). Toward understanding natural language directions. In *Proceedings of the 5th ACM/IEEE international conference on Human-robot interaction* (pp. 259-266). Piscataway, NJ, USA: IEEE Press.

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.

Krieg-Brückner, B., Frese, U., Lüttich, K., Mandel, C., Mossakowski, T., & Ross, R. J. (2005). Specification of an ontology for route graphs. In *IN SPATIAL COGNITION IV, LECTURE NOTES IN ARTIFICIAL INTELLIGENCE* (pp. 390-412). Springer.

Krug, K., Mountain, D., & Phan, D. (2003). Webpark: Location-based services for mobile users in protected areas.. *GeoInformatics*, 26-29.

Kruijff, G-J. M., Zender, H., Jensfelt, P., & Christensen, H. I. (2007). Situated Dialogue and Spatial Organization: What, Where\ldots and Why?. *International Journal of Advanced Robotic Systems, 4*(2).

Kyriacou, T., Bugmann, G., & Lauria, S. (2005). Vision-based urban navigation procedures for verbally instructed robots. *Robotics and Autonomous Systems, 51*(1), 69-80.

Landis, J., & Koch, G. (1977). The measurement of observer agreement for categorical data. *Biometrics, 33*(1), 159-174.

Larsson, S. (2015). The State of the Art in Dealing with User Answers. In *Proceedings of the 19th Workshop on the Semantics and Pragmatics of Dialogue.* (pp. 190-191).

Lemon, O., & Pietquin, O. (2007). Machine learning for spoken dialogue systems. In *In Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)*. Antwerp, Belgium.

Levelt, W. J. M. (1989). *Speaking: From Intention to Articulation.* Cambridge, Mass., USA: MIT Press.

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.

Lison, P. (2010). *Robust processing of spoken situated dialogue: a study in human-robot interaction.* Hamburg, Germany: Diplomica Verlag.

Lison, P. (2013). *Structured Probabilistic Modelling for Dialogue Management*. Doctoral dissertation, Department of Informatics Faculty of Mathematics and Natural Sciences.

Litman, D., Walker, M., & Kearns, M. (1999). Automatic Detection of Poor Speech Recognition at the Dialogue Level. In *Proceedings of the 37th Annual Meeting of the Association of Computational Linguistics*.

Lopes, J., Salvi, G., Skantze, G., Abad, A., Gustafson, J., Batista, F., Meena, R., & Trancoso, I. (2015). Detecting Repetitions in Spoken Dialogue Systems Using Phonetic Distances. In *INTERSPEECH-2015* (pp. 1805-1809). Dresden, Germany.

MacMahon, M., Stankiewicz, B., & Kuipers, B. (2006). Walk the talk: connecting language, knowledge, and action in route instructions. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2* (pp. 1475-1482). AAAI Press.

Mandel, C., Frese, U., & Rofer, T. (2006). Robot navigation based on the mapping of coarse qualitative route descriptions to route graphs. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 205-210). Beijing, China.

McTear, M. (2002). Spoken dialogue technology: enabling the conversational user interface. *ACM Computing Surveys, 34*(1), 90-169.

Meena, R., Jokinen, K., & Wilcock, G. (2012). Integration of Gestures and Speech in Human-Robot Interaction. In *Proceedings of 3rd International Conference on Cognitive Infocommunications (CogInfoCom 2012)* (pp. 673-678). Kosice, Slovakia: IEEE.

Meza-Ruiz, I. V., Riedel, S., & Lemon, O. (2008). Accurate statistical spoken language understanding from limited development resources. In *Proceedings of ICASSP 2008* (pp. 5021-5024). Las Vegas, Nevada.

Mitchell, T. M. (1997). *Machine learning.* Singapore: McGraw-Hill Book Co.

Morency, L. P., de Kok, I., & Gratch, J. (2010). A probabilistic multimodal approach for predicting listener backchannels. *Autonomous Agents and Multi-Agent Systems, 20*(1), 70-84.

Mutlu, B., Shiwa, T., Ishiguro, T., & Hagita, N. (2009). Footing in human-robot conversations: how robots might shape participant roles using gaze cues. In *Proceedings of the 2009 ACM/IEEE Conference on Human-Robot Interaction (HRI)*.

Müller, R., Röfer, T., Lankenau, A., Musto, A., Stein, K., & Eisenkolb, A. (2000). Coarse qualitative descriptions in robot navigation. In Freksa, C., Brauer, W., Habel, C., & Wender, K-F. (Eds.), *Spatial Cognition II* (pp. 265-276). Springer.

Nenkova, A., Gravano, A., & Hirschberg, J. (2008). High Frequency Word Entrainment in Spoken Dialogue. In *ACL-08* (pp. 169-172). Columbus, Ohio, USA.

Paek, T. (2003). Toward a Taxonomy of Communication Errors. *ISCA Workshop on Error Handling in Spoken Dialogue Systems*, 53-58.

Pappu, A., & Rudnicky, A. I. (2012). The Structure and Generality of Spoken Route Instructions. In *SIGDIAL Conference* (pp. 99-107). ACL.

Pappu, A. (2014). *Knowledge Discovery Through Spoken Dialog*. Doctoral dissertation, School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213.

Parent, G., & Eskenazi, M. (2011). Speaking to the Crowd: Looking at Past Achievements in Using Crowdsourcing for Speech and Predicting Future Challenges. In *INTERSPEECH* (pp. 3037-3040). ISCA.

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.

Potjer, J., Russel, A., Boves, L., & den Os, E. (1996). Subjective and objective evaluation of two types of dialogues in a call assistance service. In *Interactive Voice Technology for Telecommunications Applications, 1996. Proceedings., Third IEEE Workshop on* (pp. 89-92).

Purver, M., Ginzburg, J., & Healey, P. (2001). On the means for clarification in dialogue. In *Proceedings of SIGdial 2001* (pp. 235-255).

Quinlan, R. J. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, San Mateo, CA.

Raux, A., & Eskenazi, M. (2008). Optimizing endpointing thresholds using dialogue features in a spoken dialogue system. In *Proceedings of SIGdial 2008*. Columbus, OH, USA.

Raux, A., Langner, B., Bohus, D., Black, A. W., & Eskenazi, M. (2005). Let's go public! Taking a spoken dialog system to the real world.. In *INTERSPEECH* (pp. 885-888). ISCA.

Raykar, V. C., Yu, S., Zhao, L. H., Jerebko, A., Florin, C., Valadez, G. H., Bogoni, L., & Moy, L. (2009). Supervised Learning from Multiple Experts: Whom to Trust when Everyone Lies a Bit. In *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 889-896). New York, NY, USA: ACM.

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*.

Reiter, E., & Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering, 3*(1), 57-87.

Rieser, V., & Lemon, O. (2012). *Reinforcement Learning for Adaptive Dialogue Systems.* Berlin: Springer-Verlag.

Rizzolo, N., & Roth, D. (2010). Learning Based Java for Rapid Development of NLP Systems. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC-2010)* (pp. 957-964). Valletta, Malta.

Ross, T., May, A., & Thompson, S. (2004). The Use of Landmarks in Pedestrian Navigation Instructions and the Effects of Context. In Brewster, S., & Dunlop, M. (Eds.), *Mobile Human-Computer Interaction - MobileHCI 2004* (pp. 300-304). Springer Berlin Heidelberg.

Roth, D., & Zelenko, D. (1998). Part of Speech Tagging Using a Network of Linear Separators. In *Coling-Acl, The 17th International Conference on Computational Linguistics* (pp. 1136-1142).

Sacks, H., Schegloff, E., & Jefferson, G. (1974). A simplest systematics for the organization of turn-taking for conversation. *Language, 50*, 696-735.

Sang, E. F. T. K., & Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL* (pp. 127-132). Lisbon, Portugal.

Schegloff, E., & Sacks, H. (1973). Opening up closings. *Semiotica, 8*, 289-327.

Schlangen, D., & Skantze, G. (2009). A general, abstract model of incremental dialogue processing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*. Athens, Greece.

Schmitt, A., Schatz, B., & Minker, W. (2011). Modeling and Predicting Quality in Spoken Human-computer Interaction. In *Proceedings of the SIGDIAL 2011 Conference* (pp. 173-184). Stroudsburg, PA, USA: Association for Computational Linguistics.

Searle, J. S. (1979). *Expression and meaning: studies in the theory of speech acts.* Cambridge University Press.

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., & Al Moubayed, S. (2012). IrisTK: a statechart-based toolkit for multi-party face-to-face interaction. In *Proceedings of ICMI*. Santa Monica, CA.

Skantze, G., & Hjalmarsson, A. (2010). Towards Incremental Speech Generation in Dialogue Systems. In *Proceedings of SIGdial* (pp. 1-8). Tokyo, Japan.

Skantze, G., & Schlangen, D. (2009). Incremental dialogue processing in a micro-domain. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*. Athens, Greece.

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

Skantze, G. (2007). *Error Handling in Spoken Dialogue Systems - Managing Uncertainty, Grounding and Miscommunication*. Doctoral dissertation, KTH, Department of Speech, Music and Hearing.

Skantze, G. (2012). A Testbed for Examining the Timing of Feedback using a Map Task. In *Proceedings of the Interdisciplinary Workshop on Feedback Behaviors in Dialog*. Portland, OR.

Skantze, G., Hjalmarsson, A., & Oertel, C. (2013a). Exploring the effects of gaze and pauses in situated human-robot interaction. In *14th Annual Meeting of the Special Interest Group on Discourse and Dialogue - SIGDial*. Metz, France.

Skantze, G., Oertel, C., & Hjalmarsson, A. (2013b). User feedback in human-robot interaction: Prosody, gaze and timing. In *Proceedings of Interspeech*.

Stoyanchev, S., & Stent, A. (2009). Lexical and Syntactic Priming and Their Impact in Deployed Spoken Dialog Systems. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers* (pp. 189-192). Stroudsburg, PA, USA: Association for Computational Linguistics.

Swerts, M., Hirschberg, J., & Litman, D. (2000). Corrections in spoken dialogue systems. In *Proceedings of the International Conference on Spoken Language Processing*. Beijng, China.

Tanenhaus, M. K., & Brown-Schmidt, S. (2008). Language processing in the natural world. In Moore, B. C. M., Tyler, L. K., & Marslen-Wilson, W. D. (Eds.), *The perception of speech: from sound to meaning* (pp. 1105-1122).

Tom, A., & Denis, M. (2003). Referring to Landmark or Street Information in Route Directions: What Difference Does It Make?. In Kuhn, W., Worboys, M., & Timpf, S. (Eds.), *Spatial Information Theory. Foundations of Geographic Information Science* (pp. 362-374). Springer Berlin Heidelberg.

Truong, K., Poppe, R., & Heylen, D. (2010). A rule-based backchannel prediction model using pitch and pause information.. In Kobayashi, T., Hirose, K., & Nakamura, S. (Eds.), *INTERSPEECH* (pp. 3058-3061). ISCA.

Turunen, M. (2004). *Jaspis - a spoken dialogue architecture and its applications*. Doctoral dissertation, University of Tampere, Department of Computer Sciences.

Tversky, B., & Lee, P. U. (1998). How Space Structures Language. In Freksa, C., Habel, C., & Wender, K. (Eds.), *Spatial Cognition* (pp. 157-175). Springer Berlin Heidelberg.

Uszkoreit, H., Flickinger, D., Kasper, W., & Sag, I. A. (2000). Deep linguistic analysis with HPSG. *Wahlster 00*, 216-237.

Walker, M. (1989). Evaluating Discourse Processing Algorithms. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics* (pp. 251-261). Vancouver, British Columbia, Canada: Association for Computational Linguistics.

Walker, M., Wright, J., & Langkilde, I. (2000). 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-Geary, I., Hastie, H. W., Wright, J., & Gorin, A. (2002). Automatically Training a Problematic Dialogue Predictor for a Spoken Dialogue System. *Journal of Artificial Intelligence Research, 16*, 293-319.

Ward, N. (1996). Using prosodic clues to decide when to produce backchannel utterances. In *Proceedings of the fourth International Conference on Spoken Language Processing* (pp. 1728-1731). Philadelphia, USA.

Ward, N., Rivera, A., Ward, K., & Novick, D. (2005). Root causes of lost time and user stress in a simple dialog system. In *Proceedings of Interspeech 2005*. Lisbon, Portugal.

Werner, S., Krieg-Brückner, B., & Herrmann, T. (2000). Modelling navigational knowledge by route graphs. In Freksa, C., Brauer, W., Habel, C., & Wender, K-F. (Eds.), *Spatial Cognition II* (pp. 295-316). Springer.

Williams, J. D., & Young, S. (2007). Partially observable markov decision processes for spoken dialog systems. *Computer Speech and Language, 21*(2), 393-422.

Wong, Y. W., & Mooney, R. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of ACL-07* (pp. 960-967). Prague, Czech Republic.

Yngve, V. H. (1970). On getting a word in edgewise. In *Papers from the sixth regional meeting of the Chicago Linguistic Society* (pp. 567-578). Chicago.

Zen, H., Tokuda, K., & Black, A. W. (2009). Statistical parametric speech synthesis. *Speech Communication, 51*(11), 1039-1064.