

EXPLORING ADAPTATION IN DIALOG SYSTEMS

BY SVETLANA STOYANCHEV

SUBMITTED IN PARTIAL FULFILLMENT  
OF THE REQUIREMENTS  
FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY

SUPERVISED BY  
DR. AMANDA STENT

COMPUTER SCIENCE DEPARTMENT  
STONY BROOK UNIVERSITY  
STONY BROOK, NEW YORK

January 28, 2008

## Abstract

Numerous psycholinguistic studies have demonstrated that people adapt their language use in conversation to that of their conversational partners. Human conversational partners adapt to each others choice of words (particularly referring expressions), syntax, gesture and eye gaze. For example, if Susan uses the phrase my friend Mary to refer to her boss while talking with Sam, Sam will tend also to use your friend or Mary even if he knows that Mary is Susan's boss. There is also some evidence that people adapt their language use in conversation with computer partners - spoken dialog systems. Spoken dialog systems allow users to access computer databases and accomplish tasks using speech (and sometimes multimodal input). The interaction with a dialog system is impaired by frequent speech recognition and understanding errors. The goal of much dialog systems research is to reduce the frequency of system errors and find strategies for minimizing their impact. In my thesis I investigate how adaptation is exhibited by users of dialog systems and how it may be used to reduce the frequency of speech recognition and understanding errors and recover from them more quickly. I explore variation in the users' phrasing of concepts like dates and times and investigate whether users may be implicitly directed into using particular phrasings through interaction with the system. To do this, I use a combination of corpus analysis, dialog system engineering and experimental research.

In my experimental research, I am planning user studies in which a user interacts with a dialog system. In one of my planned studies I will measure the effect of prompt variation on the user. This experiment evaluates the potential for user adaptation to the system. In another I will measure the effect on speech recognizer performance of speech recognizer adaptation based on the type of the users utterance. Users utterances will be classified as corrections or non-corrections and the

system will use a specific language model based on this classification. This experiment evaluates the potential for system adaptation to the user.

# Contents

<b>1</b>	<b>Introduction and Problem Statement</b>	<b>6</b>
1.1	Evidence for Adaptation in Dialogs . . . . .	6
1.2	Definitions of Terms . . . . .	8
1.2.1	Terms Describing Components of a Dialog System . .	8
1.2.2	Terms Related to Adaptation . . . . .	9
1.3	Types of Adaptation in Dialog Systems . . . . .	11
1.4	Motivation and Outline of the Proposed Experiments . . . . .	12
<b>2</b>	<b>Related Work</b>	<b>16</b>
2.1	Adaptation in Dialogs . . . . .	17
2.1.1	Psycholinguistic Perspective . . . . .	17
2.1.2	Computer Science Perspective . . . . .	19
2.1.3	User Modeling in Generation Module of Dialog Systems	21
2.2	Improving Speech Recognition . . . . .	23
2.2.1	Improving Language Models . . . . .	24
2.2.2	Targeted Help . . . . .	25
2.2.3	Guiding Speakers to Use Simpler Grammar . . . . .	26
2.2.4	Topic-Based Language Model Adaptation . . . . .	26
<b>3</b>	<b>Systems</b>	<b>27</b>
3.1	RavenCalendar Dialog System . . . . .	28
3.1.1	System Architecture . . . . .	29
3.1.2	Flexible Dialog Management . . . . .	30
3.1.3	Multimodality . . . . .	31
3.1.4	Adaptation in RavenCalendar . . . . .	32
3.2	<i>Rate-a-course</i> Survey Dialog System . . . . .	33

3.2.1	System Description . . . . .	33
3.2.2	<b>Adaptation in Rate-A-Course</b> . . . . .	36
3.3	<i>Let's Go</i> System: Online Bus Information . . . . .	37
3.3.1	Adaptation in Let's Go . . . . .	38
<b>4</b>	<b>Performed and Proposed Corpus Studies and Corpus Ex-</b>	
	<b>periments</b>	<b>39</b>
4.1	Analysis of Variation in Communicator Corpus (Performed and Proposed studies) . . . . .	41
4.1.1	Performed Corpus Study: Variation in Function Verbs	43
4.1.2	Proposed Corpus Study: Analyzing Variation in Con- cepts . . . . .	44
4.2	Performed Experiment: Analysis of Rejections in the Com- municator Corpus . . . . .	48
4.2.1	Causes of <i>Slu-Reject</i> in the Communicator Corpus . .	50
4.2.2	System Strategies for Handling Non-Understandings .	53
4.2.3	Experiment: Predicting System's Action . . . . .	54
4.2.4	Users' Strategies on Rejections . . . . .	57
<b>5</b>	<b>Directive Adaptation: Exploring Prompt Strategies</b>	<b>59</b>
5.1	Performed Experiment: Adaptation in User's Lexical and Syntactic Choices in Rate-a-Course system . . . . .	61
5.1.1	Data Analysis of the Rate-a-Course Experiment . . .	63
5.1.2	Results . . . . .	65
5.2	Proposed Experiment: Matching NLU and NLG for the <i>time</i> Concept . . . . .	66
5.2.1	Experiment . . . . .	68

5.2.2	Hypotheses . . . . .	69
5.2.3	Measurement . . . . .	70
5.2.4	Implications of the Experiment . . . . .	70
5.2.5	Using <i>Let's Go</i> System for Matching of NLU and NLG Experiment . . . . .	71
5.3	Proposed Optional Experiment on Calendar System . . . . .	72
5.3.1	Wizard-of-Oz Type Experiment . . . . .	72
5.3.2	Changes to Calendar System . . . . .	73
5.3.3	Experimental Participants . . . . .	73
<b>6</b>	<b>Responsive Adaptation: Experiment on Speech Recognition Component</b>	<b>73</b>
6.1	Performed Experiment: Adapt Language Model to the Ques- tion Topic in a Question Answering System . . . . .	74
6.1.1	Problem Description . . . . .	74
6.1.2	Approach . . . . .	76
6.1.3	Experiments and Results . . . . .	79
6.1.4	Future Work . . . . .	82
6.2	Proposed Experiment: Adapt ASR Language Model to the Correction State . . . . .	83
6.2.1	Motivation . . . . .	83
6.2.2	Method . . . . .	86
<b>7</b>	<b>Schedule</b>	<b>88</b>
	Bibliography . . . . .	90

# 1 Introduction and Problem Statement

## 1.1 Evidence for Adaptation in Dialogs

The dictionary definition of adaptation is “the process of bringing one thing into correspondence with another . . . implies a modification according to changing circumstances”. [Merriam-Webster] Numerous psycholinguistic studies have demonstrated that people adapt their language use in conversation to that of their conversational partners. For example, conversational partners adapt to each other’s choice of words, particularly referring expressions [Brennan and Clark1996], converge on certain syntactic choices [Pickering et al.2000, Lockridge and Brennan2002], adapt their prosody to help their partners disambiguate syntactic ambiguities [Kraljic and Brennan2005], and also adapt using audiovisual information [Kraut et al.2003]. Some of these results have been duplicated using corpus studies: researchers have found evidence of within-speaker and between-speaker convergence to certain syntactic constructions in dialog corpora [Dubey et al.2006a, Reitter et al.2006a]. Finally, there is some evidence that people adapt their language use in conversation with computer partners. For example, researchers have shown that users of dialog systems adapt to the system’s choice of referring expressions [Brennan1996], the system’s choice of modality for referring [Bell et al.2000, Skantze2002], and the system’s choice of words [Gustafson et al.1997].

Spoken dialog systems allow users to access computer interfaces using speech (and text). Most of the US companies today including banks, telephone companies, airlines, etc. provide over-the-phone automatic help for their customers. It is now possible to check Amtrak train and bus availability using a dialog system over the phone (1-800-872-7245), find out the closest gas station with the cheapest gas or traffic information in the

Seattle area [Tashev et al.2007]. Several free directory information services appeared in the past year, Free 411 (1-800-FREE-411), Live Search 411 (1-800-CALL-411), Google 411 (1-800-GOOG-411), are summarized in [Stenchikova2007]. Dialog systems are a rich application field for a different research areas, like artificial intelligence and psycholinguistics. The richness of natural language allows for a great variability in the grammar and vocabulary of user utterances. A user is not guaranteed to follow a predefined grammar. For example, an answer to the first query of bus information system's question "How can I help you" can be a general specification of the task, e. g. "*scheduling*", a general request for information on a particular bus route "*i need /uh/ information on the 56e*", or a question about a specific route: "*when is the next 28x from downtown to the airport?*". Variation exists even in simple utterances specifying a concept like time: *four pm, four o'clock, around four*. Variability in natural language complicates dialog system designer's task and often causes misrecognitions in dialogs. Modern dialog systems are greatly impaired by poor speech recognition. Minimization of, and efficient recovery from misrecognitions is a necessary capability of a dialog system.

In my thesis I will explore the possibility of using adaptation to improve users' experience in dialogs with automatic dialog systems. I will focus on two types of adaptation in system behavior: responsive adaptation, in which the system modifies its own behavior to better fit user's behavior; and directive generation, in which the system uses behaviors designed to guide a user into less error-prone behaviors. In this chapter I first define the terms that will be used throughout the proposal, in section 1.3 I discuss the types of adaptation exhibited by dialog systems, and in section 1.4 I give motivation and an overview the proposed experiments.



## 1.2 Definitions of Terms

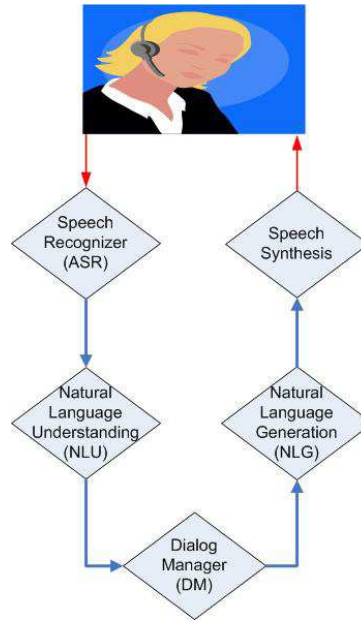


Figure 1: Dialog flow and main components of a dialog system

### 1.2.1 Terms Describing Components of a Dialog System

Main components of a dialog system and data flow are illustrated in figure 1.

- **Automatic Speech Recognition (ASR)** takes audio of a user's utterance and converts it to text.
- **Natural Language Understanding (NLU)** converts text recognized by an ASR into a system-internal semantic representation.
- **Natural Language Generation (NLG)** converts a system-internal semantic representation into a natural language utterance.
- **Dialog Manager (DM)**. Dialog system's "brain". Takes as input a semantic representation from the NLU, performs some action

(database access, table lookup, logical inference, etc. ) and produces a new semantic representation that is passed to the NLG.

Other terms related to a dialog system:

- **Language model.** A probabilistic model of words in context, used in a statistical ASR.
- **Grammar.** A set of rules in the form  $A \rightarrow BC$ . A grammar can be used in a grammar-based ASR component and in a NLG component.
- **Acoustic model.** A probabilistic model mapping syllables to pronunciation. This model is used in a statistical ASR.

### 1.2.2 Terms Related to Adaptation

- **Adaptation.** “The process of bringing one thing into correspondence with another . . . implies a modification according to changing circumstances”. [Merriam-Webster]
- **Responsive adaptation in a dialog system.** An adjustment that happens when a system changes its behavior to accommodate a user. This may involve a change in ASR, NLU, DM, or NLG.
- **Directive adaptation in a dialog system.** System guides a user into using particular grammar or vocabulary. DM and NLG modules can be responsible for directive adaptation.
- **Partner (user or system) model.** Structured information pertinent to the dialog and held by a dialog participant about his/her dialog partner. This model may be partially constructed from prior knowledge (what the other partner is expected to know) or general

information (e.g. the sky is blue). The model is dynamically updated in the course of a dialog. The model can be held by a human dialog participant or by a system.

- **User model in a dialog system.** System's *partner model*. The model may hold any information that pertains to the specific user or a group of users: general or domain preferences, language model, acoustic model, etc. The model may be used for predicting user's actions or generating responses.
- **System model.** User's internal representation (*partner model*) of his/her understanding for the system. By reading directions or by communicating with a system, users build their understanding of the system's capabilities: its vocabulary, grammar, and accessible functionality.
- **Priming.** A process that influences linguistic decision-making. An instance of priming occurs when a syntactic structure or lexical item giving evidence of a linguistic choice (prime) influences the recipient to make the same decision, i.e. re-use the structure, at a later choice-point (target) [Reitter et al.2006b]
- **Concept.** A piece of information that a system needs to obtain from a user. In an flight booking system concepts would include departure city, date, time, airline, etc. There is usually a number of ways to specify each concept, e.g. *four pm/four o'clock/four*.
- **Directive prompt.** System's prompt that mentions a concept in a particular form. We call it *directive* because the user gets *primed* by a system about the representation form of a *concept*.

### 1.3 Types of Adaptation in Dialog Systems

Adaptation of a dialog system to its conversation partner may be classified along several dimensions: time frame, type of a user model, part of the system adapted, and objective of adaptation.

Time frame for adaptation can be *short-term* or *long-term*. In the *short-term* adaptation model a dialog system changes its behavior by observing a speaker in each separate dialog. This type of adaptation is applicable to systems where a user's identity is not known and repeat users may not be identified. [Komatani et al.2003] presents an example of a system with *short-term* adaptation. The authors describe an adaptive dialog system presenting bus information to users over a phone. The system does not recognize repeating callers and a *user model* is identified by the system automatically when a call is received. In *long-term* adaptation the system collects information about a user over time and uses the information to update a *user model*, which is then used to adapt system behavior. [Stent et al.2004, Walker et al.2004, Thompson et al.2004] describe systems that use long-term adaptation in response generation.

A user model for adaptation can be defined by a user's skill level, urgency, or linguistic preferences. [Komatani et al.2003]'s bus system automatically identifies a user's skill level (beginner/advanced) and adapts the system by providing more detailed help for beginner users. It also identifies the user's knowledge level about the city: a first-time visitor does not know the areas and landmarks, while a person living in the city has more knowledge. Systems vary in how they obtain user models:

- by hand-coding (for classes of users)
- by offline learning (from collections of data)
- by surveying users

- by online learning (during a conversation)

Adaptation to the users' knowledge level is a type of domain-specific adaptation. A domain-specific model determines user's preferences within a domain, for example the user's favorite type of food or preferred price range in a restaurant recommendation system [Thompson et al.2004, Walker et al.2004]. In a calendar system, a user model for language understanding may contain the user's occupation: students are more likely to schedule classes, while business people are likely to schedule meetings and appointments. A preference for information presentation in a calendar may differ for users with different occupations.

Speech recognizers use an acoustic model and a language model. Work on speaker-dependent speech recognition [Shi and Chang2004, Huang1991] focuses on adapting an acoustic model of a speech recognizer to a particular user allowing systems to understand speakers with accents. Speaker-dependent speech recognizers have achieved remarkable quality and are good candidates for use in dialog systems with recognized and repeated users. While speaker-adaptive recognizers focus on adapting the acoustic model to a particular speaker, in my work I focus on adapting the language model to a particular dialog state.

#### **1.4 Motivation and Outline of the Proposed Experiments**

In my work I focus on task-oriented mixed-initiative human-computer dialogs. User's communication with a dialog system is affected by the user's perception of the system. In [Giesemann and Stenneken2006] the authors gave different instructions to two groups of users communicating with a robot. One group was told to talk to the robot in a child-like manner, while

the other group was not. The perception of the system given to the users by this instruction led to qualitative and quantitative differences in users' communicative behavior. I hypothesize that users build a *partner model*<sup>1</sup> for a dialog system which affects their choices made during interaction. This model may be partially affected by phenomena over which a dialog system designer has no control: user's prior experience with dialog systems, user's background and computer savvy. In the course of communication with a dialog system a user's partner model gets updated to reflect user's understanding of system's capabilities. A user's *partner model* for a system may include the user's perception of which grammatical structures the system can or can not understand, whether the system can understand free speech, the form of a *concept* understood by the system, or whether the system can understand multiple *concepts* in a single utterance. A user may not be consciously aware of his/her model of the system, and this model does not determine 100% the utterances that the user will use, but it influences the user in a certain direction making some utterances more likely than others. A human-computer dialog is more successful when a user's model matches actual capabilities of a system. A system should communicate its capabilities to the user through explicit or implicit channels.

One of common approach for improving a dialog system is to expand grammars and language models to recognize a wider variety of user utterances. This approach aims at allowing users to communicate with the

---

<sup>1</sup>Partner model is a structured information pertinent to the dialog and to the dialog partner, held by a dialog participant . This model may be partially constructed from prior knowledge (what the other partner is expected to know) or general information (e.g. "the sky is blue"). The model is dynamically updated in the course of a dialog (e.g. "my dialog partner likes to ski").

system as “naturally” as possible, using free speech. However this approach is not necessarily preferred by users [Tomko and Rosenfeld2004] because of the poor speech recognition when free user responses are allowed in a dialog system. Larger grammars and language models increase coverage but decrease recognition performance of in-vocabulary utterances. An alternative approach where the system used a limited grammar and vocabulary called *speech graffiti* and through help and confirmation messages implicitly “taught” users to utilize *speech graffiti* [Tomko and Rosenfeld2006]. Speech graffiti and free speech are two opposite approaches to the design of a dialog system. Evidence of lexical and syntactic convergence in human-human and human-computer dialogs provides motivation for using a middle-ground approach: allowing users to use free speech while maintaining a low error rate by adapting grammars and language models to the system’s dialog state and to particular users. For example, a travel dialog system may be flexible to understand different ways of specifying a departure location with user’s utterances like “I would like to leave from Dallas in the evening”, “I will depart from Dallas at four”, “leaving from Dallas”. Suppose, that the system frequently uses a confirmation prompt of the form: “You are leaving from LOCATION at TIME, is this correct?”. In case of a repeating customer who has heard the confirmation prompt (and assuming that *directive adaptation* in dialog systems is a valid theory), this user may be more likely to use words and syntax of the system’s prompt: “leaving from LOCATION at TIME”.<sup>2</sup>

In my previous research on adaptation in dialogs I have devised new measures for adaptation and compared adaptation due to recency and to

---

<sup>2</sup>However, in reality multiple aspects of dialog history as well as user’s preferences can affect user’s word and grammar choice in an utterance.

partner effect in human-computer dialogs [Stenchikova and Stent2007]. I have performed corpus studies analyzing causes of system's rejections in communicator corpus and modeled prediction of system's action (described in section 4). I have performed an experiment improving ASR of a question answering system by adapting a language model to the question's target [Stoyanchev(Stenchikova) et al.2008]. In the proposed experiments I will use a combination of corpus studies and system experiments. My experiments explore *responsive* adaptation in speech recognition and *directive* adaptation in natural language generation. In order to more easily examine the impact of these types of adaptation, I will focus on dialog states that immediately precede, contain or follow a system error. In terms of responsive generation, I will look at characterizing misrecognitions, and adapting the system's speech recognizer for different types of misrecognition. In terms of directive generation, I will look at variations on system prompts during error recovery, to see how these prompts affect users' utterances and find those prompts that lead to most efficient recovery from misrecognitions. Errors in a human-computer dialog are common and unavoidable. The errors impair system's performance, but they also help users shape their *partner model* of the system and discover the system's capabilities.

To sum up, in my thesis I will look at how *directive adaptation* (in system prompts) and *responsive adaptation* (in system prompts and in ASR) can be used to reduce the frequency of occurrence of errors, and to more efficiently recover from errors, in spoken dialog systems. I will use corpus studies and empirical system studies. My empirical studies will be conducted using the SBU RavenCalendar Dialog System, the SBU Rate-a-Course dialog system, and the CMU Let's Go dialog systems. My preliminary corpus studies include a study of variation in function verbs and analysis of rejections in



Communicator Corpus (done; see Chapter 4).

**My studies on directive adaptation include:**

- A corpus study on analysis of variations in concepts (proposed; see section 4.1.2)
- Empirical study: Matching NLU and NLG for the *time* concept (proposed; see Chapter 5.2)
- Empirical study: adaptation in user's lexical choices (performed: See section 5.1)

**My studies on responsive adaptation include:**

- Evaluation of ASR language models: Adapting language model to the question topic in a question answering system(performed: see section 6.1)
- Evaluation of ASR language models: adapting ASR language model to the correction state (proposed: see section 6.2)

**The main contributions that I expect from my experiments are:**

- An understanding of which factors affect user's form of *concept* choices.
- A practical strategy for improving system's ASR by adapting it to a *correction* state.
- Functional dialog systems that exploit the phenomena of directive and responsive adaptation.

## 2 Related Work

In this chapter I describe related work in psycholinguistics and in computer science on adaptation in human-to-human and human-to-computer dialogs (section 2.1). One of my proposed experiments uses adaptation to improve

speech recognition component in a dialog system. In section 2.2 I describe related work on other approaches to speech recognition improvement.

## 2.1 Adaptation in Dialogs

In this section I will summarize prior research that provides evidence of adaptation in dialog. Most prior research on adaptation in dialog has been done by psycholinguists, who study adaptation by conducting controlled experiments. Recently, some research in this area has been done by computational linguists, who adopt a data-driven approach, mining existing corpora of dialog.

### 2.1.1 Psycholinguistic Perspective

Psycholinguists find two types of adaptation in dialog, one based on *convergence* and the other on *complementarity* [Brennan et al.2004].

*Convergence* happens when dialog participants change their language use to be more similar over time. As an argument for convergence, researchers show that while there is great deal of lexical variability across conversations, there is far less lexical variability within a conversation [Brennan and Clark1996, Garrod and Doherty1994, Furnas et al.1987]. [Brennan1998] finds evidence for lexical convergence in human-human dialogs and shows that frequency of a term use by individual speakers affects the durability of the effect. [H. Branigan2004, Pickering et al.2000, Lockridge and Brennan2002] find that dialog participants converge on certain syntactic choices. [Garrod and Anderson1987] make an argument for tight coupling of production and comprehension in dialog. The authors describe an output/input coordination principle which predicts that a speaker formulates an utterance according to the same model

and semantic rules used to formulate or interpret the most recent utterance. [Pickering and Garrod2004] argue for alignment of pragmatic and semantic representation for generating output with representations for interpreting input. According to the authors this alignment occurs via priming (see definition in 1.2) mechanisms during interaction. Interactive alignment process between the dialog partners explains why the dialog partners may complete each other's phrases and reuse the same expressions. The authors that alignment is central to successful dialog.

*Complementarity* – when dialog participants' language use becomes or remains more different over time, usually for some particular reason e.g. deafness, first language. Native English speakers use a simplified wording when speaking with a non-native speakers or young children, using shorter and more direct expressions when speaking with computer systems [Brennan1990, Oviatt1995]. [Brennan1998] finds that in human-human conversations expression referring to a particular concept get shortened over-time. [Kraljic and Brennan2005] show that dialog participants may adapt their prosody to help their partners disambiguate syntactic ambiguities. Speakers also adapt using audiovisual information [Kraut et al.2003].

From experiments with real and simulated dialog systems researchers have shown that users of dialog systems adapt system's choice of referring expressions [Brennan1996], system's choice of modality for referring [Bell et al.2000, Skantze2002], and system's choice of words [Gustafson et al.1997]. [Brennan1998] concludes with proposition for dialog system designers:

Spoken language systems should be supported by an architecture that explicitly supports (and expects) feedback and negotiation. Because of the ubiquity of the vocabulary problem, feedback and negotiation are necessary in any spoken duologue system that aims to handle

	in prime	not in prime
in target	a	c
not in target	b	d

Table 1: Contingency table

spontaneous utterances from real users.

### 2.1.2 Computer Science Perspective

Computer scientists have also studied adaptation in dialogs by analyzing corpora and developing systems with adaptation capabilities. In general, these studies confirm the experimental results summarized above. However, each computational study also produces either (a) an algorithm for measuring adaptation; or (b) an algorithm for modeling/reproducing adaptation in dialog.

[Church2000] introduced a method for measuring lexical 'adaptation' in text. This method determines whether appearance of a lexical feature in the *priming (earlier) portion* of a document affects the likelihood of its appearance in the *target (later) portion*. This method requires the construction of a contingency table illustrated in table 1. For each feature in a corpus of texts, the table shows how many of the texts contained the feature: (a) in both *priming* and *target* portions, (b) in the *target* only, (c) in the *priming portion* only, and (d) in neither portion. Church applied this method to a study of a corpus of text documents, treating the first half of each document as the *priming portion* and the second half as the *target*. He showed that positive lexical adaptation does occur, more strongly for content words than for function words. [Dubey et al.2006b] used Church's method to evaluate adaptation for selected syntactic constructions in coordinating structures in the Brown news text and Switchboard dialog corpora. The *priming* and *tar-*

*get portions* in Dubey’s coordinating structure experiment were the left and the right sides of coordinating constructions (*and* and *or*) in the corpus. The authors reported positive adaptation for each of the syntactic constructions they considered. Although it was used for measuring adaptation, Church’s measure was developed to identify the most useful features for information retrieval, rather than for study of adaptation *per se*.

In [Stenchikova and Stent2007] my adviser and I propose two new measures, one that measures the presence of adaptation and another that measures its strength. Our adaptation measure is a modification of Church’s measure in two ways. First, it uses the frequency of occurrence of each feature in each document rather than merely its presence or absence. Second, instead of using Church’s prior we use an estimate of the probability of feature co-occurrence in prime and target by *chance*. In the adaptation *strength* measure, instead of using binary values for each feature indicating presence or absence of that feature in a document, we use the actual frequency of occurrence of the feature in the document. These measures allow us to identify adaptation within a single document or between documents; to identify the strength of adaptation as well as its presence; and to identify the source of the adaptation. Our research addresses a question that is currently debated in the psycholinguistics community about whether adaptation in dialogs is:

- *partner adaptation* - adaptation based on a model of the partner. This type of adaptation is sometimes called entrainment or audience design [Brennan and Clark1996, Horton and Gerrig2002]
- *recency adaptation* - adaptation due to the representations of words, concepts etc. being activated, or brought to the forefront during lan-

guage production, by previous perception or alignment [Brown and Dell1987, Pickering and Garrod2004, Chartrand and Bargh1999] or comprehension.

We study adaptation in the Maptask spoken dialog corpus and show that for syntactic features, recency adaptation is stronger than partner adaptation; however, we find no significant differences for lexical adaptation using these measures.

In [Reitter et al.2006b] the authors compare syntactic adaptation in task oriented and spontaneous conversations showing that between-speaker priming effect is stronger in task-oriented dialogs. [Reitter et al.2006a] analyzes human-to-human dialog corpora and detects strong priming effect for syntactic rules in task oriented dialogs. The authors used logistic regression to examine and compare short-term priming effects within a small window of time in separate dialogs. They show rapid degradation of priming effect in a dialog over time.

Similarly to psycholinguists' conclusions in [Pickering and Garrod2004] about the importance of alignment in dialog success, [Reitter and Moore2007] shows that lexical adaptation positively correlates with task success in human-human task-oriented Maptask corpus.

### **2.1.3 User Modeling in Generation Module of Dialog Systems**

This section describes several dialog systems designed to adapt their generation to particular aspects of a user. The results of the experiments on these systems indicate that adaptation to a user benefits the systems' performance.

Adaptive Place Adviser [Thompson et al.2004] is a personalized spoken

dialog system for recommendations of books, movies, and restaurants. It uses a personalized long-term user model <sup>3</sup> based on domain-specific user preferences. During interaction the system’s dialog manager selects a dialog move based on the user’s input query, user model, and match of the user’s query with a database. Possible system moves include suggesting user to constrain or to relax a query, recommending an item from a database, providing a list of choices, or asking the user for a clarification. The experiment compared users of adaptive and generic systems. The results show a decrease in conversation length (both in number of turns and in time) for the users of the adaptive system.

MATCH [Walker et al.2004] is a multimodal dialog system for giving suggestions about restaurants. The system tailors its generated utterances to a user model. The user model incorporates the level of importance of decor, price, and food quality to the user. The model is static and is generated from the user’s answers to a set of questions. The authors evaluated the system by asking users to rate the information quality and conciseness of user-tailored and generic system recommendations and comparisons of restaurants. They found that user-tailored recommendations and comparisons are preferred.

[Stent et al.2004] describes trainable sentence planning for complex information presentations in spoken dialog systems. The syntax of generated system utterances is adapted based on user preferences collected in a rating experiment. An evaluation showed that user-adapted presentations were preferred over generic presentations.

[Komatani2005] describes a bus information system that generates co-

---

<sup>3</sup>The term *user model* is defined in section 1.2

operative help messages based on a user model. The user model includes skill level, knowledge level, and urgency of the user. The user model is determined automatically at the time of the user's call based on the initial user utterances. Evaluation results show that novice users learn to communicate with the system more efficiently when the user model is utilized.

[Purver and Kempson2004] introduce a novel theoretical approach to generation in dialog: context-based incremental tactical generation. This approach reflects both *convergence* and *complementarity* adaptation in dialog. A tree structure represents a semantic interpretation for a string. Parsing and generation use the same tree representations. When a string is parsed, a tree with a corresponding semantic representation is generated. During parsing, trees are built in a word-by-word incremental fashion allowing analysis of anaphora and ellipses. In generation process, a semantic tree structure is converted into a string. During generation, lexicon search considers the most recently used words first, enabling lexical adaptation. Dynamic syntax models generation with an idea of minimizing effort on the part of the speakers. Incremental nature of generation process allows bare fragments taking structure from previous sentence (rather than postulate a new rule for the fragment), e.g. *What did you eat for breakfast? Porridge.*

## 2.2 Improving Speech Recognition

Speech recognition is one of the largest causes of errors in human-computer dialog. In the past twenty years, since the breakthrough in basic speech recognition [Rabiner and Juang1986], researchers have been working on incremental improvements to the Hidden Markov Model (HMM)-based algorithm by enhancing language models, adapting them to a dialog state,



or adapting acoustic models to a particular speaker for speaker-dependent recognition.

### 2.2.1 Improving Language Models

Although domain-dependent speech recognition in a dialog system is more tractable than open-domain recognition, user variation in grammar and vocabulary causes problems for robust speech recognition. In [Riccardi and Bangalore1998] the authors improve the system's language model by learning phrase grammars with unsupervised clustering techniques (iterative entropy reduction). The automatically learned phrase grammars allow for generalization. Using these grammars one may automatically generate phrases never seen in the training corpus, yet similar to the phrases in the training corpus. Evaluation shows improvement in a call classification task. [G. Riccardi2000] describe a machine learning approach that shifts the burden of learning jargon from users to the system. The language model is learned using the HMM Viterbi algorithm and conditioned on the current state of the dialog system. Speech recognition and language understanding modules of the system are working as a whole and the result shows improvement of word-error-rate.

In spoken dialog systems, on the other hand, it is a usual practice to use dialog state specific language models [Bechet et al.2004]. For example after the confirmation prompt it is more likely the user will say *yes* or *no*.

Binding speech recognizer and natural language understanding (semantic parsing) modules was shown to benefit both modules. [Young1994] use output from the NLU module along with acoustic probabilities to detect misrecognized words on a second pass through the recognizer. Expansion of speech recognition vocabulary using the web is proposed and tested on

the meetings data [Yu et al.2000] resulting in significant improvement of recognition for out-of-vocabulary words.

[Gorrell et al.2002, Hockey et al.2003] use a combination of grammar-based and statistical speech recognition in a two-pass approach. First, a user's utterance is passed through a grammar-based speech recognizer. Using a threshold on confidence level, a system either accepts an utterance or passes it to a statistical recognizer.

### **2.2.2 Targeted Help**

[Gorrell et al.2002, Hockey et al.2003, Fukubayashi et al.2006] provide targeted help to unrecognized users' utterances. Gorrell describes two approaches where automatically recognized users' utterances with low recognition confidence score are used to identify a help message. In the first approach a help message is one of the predefined messages. A machine learning approach using features from speech recognizer classifies an appropriate help message to be played to the user. In the second approach a help message is generated by matching user's utterance to the closest in-grammar utterance. The first approach is implemented in a command and control application and the second – in a question answering application. Both applications are tested with experimental users and show significant improvement in completion rate and user satisfaction.

[Hockey et al.2003] assists users in becoming experts by providing informative help messages. The researchers identified three major types of errors in their push-to-talk command and control application: 1) endpointing error (when a initial word is cut off) 2)out-of-vocabulary word, 3) subcategorization (in-vocabulary but out-of-grammar). The authors designed rules to

handle each type of the error. For example, out-of-vocabulary error gets a message “the system does not understand the word X”.

In [Komatani et al.2007] the authors identify a cause of error as out-of-grammar or out-of-domain utterance. A system keeps track of a *known degrees* of each node in a system domain concept tree. A domain concept tree contains four layers: system, function, element, and content word. *known degrees* are updated after each user’s utterance. For example, if a user utters a content word, a *known degrees* of a corresponding concept in the content layer is incremented. A user’s utterance is classified using thresholds as 1) in-grammar, 2) in-grammar but incorrect language understanding result 3) out-of grammar (reject). The system suppresses unnecessary help messages for utterances in the second category using the *known degrees* score of a user.

### 2.2.3 Guiding Speakers to Use Simpler Grammar

In another approach researchers try to cause users to speak in simpler phrases to improve speech recognition. [Tomko and Rosenfeld2006] “teaches” users to utilize “speech graffiti”, a limited language of simple commands when speaking with the system. These systems use limited grammar for language understanding and assist users in become experts by guiding them with explicit or implicit help messages into using grammar and vocabulary understood by the system.

### 2.2.4 Topic-Based Language Model Adaptation

4

---

<sup>4</sup>This section is a modification of related work from [Stoyanchev(Stenchikova) et al.2008]

Iyer and Ostendorf [Iyer and Ostendorf1999] used mixture models for broadcast news recognition. The documents in the training data are clustered using the expectation maximization (EM) algorithm to obtain topically coherent sentences. For each topic, separate language models are created. Then either offline (static) or online (dynamic) mixture language model adaptation is proposed. They obtain 21% reduction in perplexity and 4.5% lower word error rate on the Wall Street Journal corpus using static mixture modeling. They also applied this approach for conversational telephone speech recognition using the Switchboard corpus [Godfrey et al.1992]. The conversations in this corpus have already been marked with 71 topics. They still employed the clustering approach using this annotation as the seed and came up with five clusters. They obtained a humble 1.2% relative reduction in the word error rate. Gildea and Hofmann proposed combining the topic language model with a generic model during run-time in a dynamic fashion [Gildea and Hofmann1999]. Similar to the previous work the EM algorithm is used for clustering the documents. Model combination is done using a linear or log-linear interpolation. While they obtain 16% lower perplexity using the resulting language model, the word error rate increased by 2.5% relative on the TDT-pilot corpus. [Gruenstein et al.2006] improve recognition of proper nouns by limiting a language model for a current context.

### 3 Systems

In this chapter I describe three dialog systems that I will use in my experiments. Two of the systems, the SBU RavenCalendar dialog system and the SBU Rate-a-Course spoken dialog system, were developed at Stony Brook. I

took a leading role in developing both of these systems. RavenCalendar is an interactive speech-based front-end to Google Calendar and Google Maps. It was developed by me and two other graduate students using the RavenClaw dialog system framework. I chose the framework for the system, designed the architecture, developed the dialog management part of the system, developed grammars for parsing, and designed templates for generation. I supervised another graduate student (Barbara Mucha) in developing the back end of the system and Sarah Hoffman who worked on enhancing the speech recognition. Rate-a-Course is a spoken dialog survey system. It was developed by my adviser Amanda Stent, me, and an undergraduate student Matthew Marge. I restructured the system for easier maintainability, implemented and debugged Voice XML dialog forms for the system. I also took a leading role in deploying and testing the system before the experiments described in Section 5.1. I ran half (24) of the experimental subjects for that experiment. The third system described in this chapter, Lets Go, was developed at Carnegie Mellon University. I established a research connection with the Language Research group at Carnegie Mellon and submitted two proposals for experiments on the system which were accepted.

### 3.1 RavenCalendar Dialog System

5

[Stenchikova et al.2007] describes the RavenCalendar dialog application.

Dialog applications for managing calendars have been developed for every generation of dialog systems research, from Heidorn's 1978 text-based system [Heidorn1978] to speech-based calendar applications designed to study

---

<sup>5</sup>This section is partially taken from [Stenchikova et al.2007]

different approaches to dialog management [Yankelovich1994, Constantinides and others1998, Horvitz and Paek2000] and multimodal systems developed in the last decade [Vo and Wood1996, Huang and others2001]. Today, Web-based calendar applications are freely available and widely used. A spoken dialog interface to a Web-based calendar application would permit convenient use of the system on a hand-held device or over the telephone.

*RavenCalendar* is a multimodal dialog system built around the Google Calendar and Google Maps web applications. *RavenCalendar* allows the user to create, modify and remove calendar events, query for events, and hear descriptions of events.

### 3.1.1 System Architecture

Figure 2 illustrates the *RavenCalendar* system architecture. *RavenCalendar* is developed using the Ravenclaw/Olympus [Bohus and Rudnicky2003] architecture . Olympus is a dialog system framework; Ravenclaw is the Olympus dialog manager. Olympus architecture allows transparent swapping of dialog system components. I have taken several off-the-shelf speech recognition and synthesis components and plugged them into our system. As a speech recognition component we have an option of using the speaker-independent Sphinx 4 [W.Walker and et al.2004] speech recognizer with a hand-written grammar, or a speaker dependent language model-based Microsoft ASR. The result of the recognition is parsed by Phoenix, a robust parser using an extended Context Free Grammar. It allows the system to skip unknown words and perform partial parsing [Ward and Issar1994]. For speech synthesis we use off-the shelf Microsoft TextTS as well as the default Festival Speech Synthesis component. *RavenCalendar* is one of the

first dialog systems based on the Ravenclaw/Olympus framework to be developed outside of CMU. Other research Olympus-based dialog systems developed at CMU include the Let’s Go [Raux et al.2005], Room Line, and LARRI [Bohus and Rudnicky2002] systems.

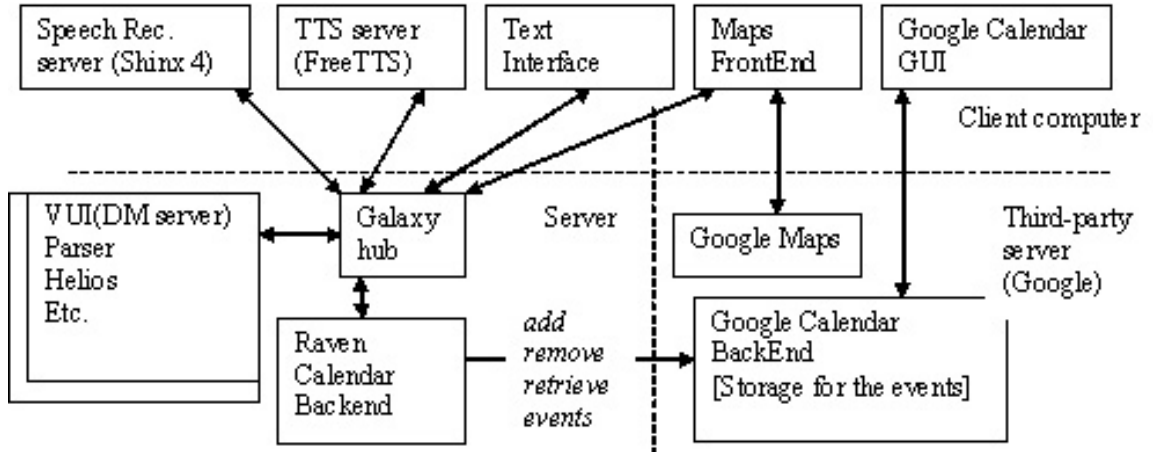


Figure 2: RavenCalendar Design

### 3.1.2 Flexible Dialog Management

Most of my work with RavenCalendar has focused on building the dialog manager. In *RavenCalendar*, during event management or event querying a user can provide event information either in a single utterance (e.g. “Add a meeting for tomorrow from two to three in the computer science lounge”) or by engaging in a subdialog with the system (Table 2). This functionality is made possible because we use the Ravenclaw dialog manager [Bohus and Rudnicky2003]. The dialog manager allows “object-oriented” specification of a dialog structure. In *RavenCalendar*, we define the dialog as a graph. Each node in the graph is a minimal dialog component that performs a specific action and has pre- and post-conditions. The dialog flow

is determined by edges between nodes. With this structure, we maximize the reuse of minimal dialog components inside the dialog manager.

S: Welcome to RavenCalendar, the automated dialog calendar system.
S: What can I do for you today?
U: add a meeting for Monday
S: For what time do you want to add the meeting?
U: from ten until eleven
S: The event was added type meeting for tomorrow starting at ten a.m.
U: List all events for the next week

Table 2: Sample dialog with *RavenCalendar*

### 3.1.3 Multimodality

A *RavenCalendar* user may interact with the calendar directly using the Google Calendar interface, or may interact through *RavenCalendar* using text, speech, map gestures or a combination of these media. A user may use the Google Maps interface to specify the location of an event; the system uses Google Maps to display the locations of events (Figure 3).

*RavenCalendar* integrates with Google Calendar for storing event data. The back-end server communicates with the Google Calendar database through the Google Calendar Java API. Google Maps communicates with *RavenCalendar* through the *maps front end* server. In addition to the locations recognized by Google Maps, an XML file with pre-selected location-name mappings (e.g. buildings at Stony Brook University) helps the user specify locations.





Figure 3: RavenCalendar Map

### 3.1.4 Adaptation in RavenCalendar

RavenCalendar runs on the user's own computer, providing an excellent platform for *long-term responsive* adaptation experiments. The system may observe the user's behavior over time and generate a user model based on these observations. This user model may, at some point in the future, include:

- System categorizations of the user – their type (student/business person, etc.), their level of expertise with the system
- Knowledge about the user's use of language and other modalities – e.g. grammars with vocabularies adapted to the contents of the user's calendar, user specific speech recognition
- Knowledge about user preferences – e.g. for particular types of event

(Seawolves games, or concerts), or for particular system actions (e.g. reminders)

According to psycholinguistic research, dialog partners in human-human conversations reflect each other's grammar and vocabulary. The calendar system can be used to test whether in a conversation with dialog systems a user would notice any difference or prefer a system which adapts to the user's use of language. For example, adaptation on the level of grammar and vocabulary can be implemented by matching grammars in the NLU and the NLG components of the system.

### **3.2 *Rate-a-course* Survey Dialog System**

<sup>6</sup> [Stent et al.2006] describe the Rate-a-course system and the experiment performed on it.

Surveys are a natural and commercially viable application for spoken dialog systems. Survey dialog systems also present interesting opportunities for research on spoken dialog and on survey design.

#### **3.2.1 System Description**

The Rate-A-Course system is implemented in VoiceXML, XML and Javascript. It runs on the BeVocal Cafe platform and uses its proprietary speech recognizer [BeVocal]. The survey questions, and potential answers, take the form of an XML document that can be automatically generated from a web-based survey design interface. This interface permits the selection of choice points (for subdialogs), question types and error-handling strategies. The XML document is used to automatically generate speech recognition grammars

---

<sup>6</sup>This section is partially taken from [Stent et al.2006]

Topic	Synonyms	Answers/Ratings
Instructor	teacher, professor	very good/100, good/75, okay/50, bad/25, very bad/0
Exams	tests, midterms	too hard/0, hard/50, about right/100, easy/50, too easy/0
Class size	course size, size of the class	too packed/0, packed/50, about right/100, small/50, too small/0
Assignments	homeworks, course work	too hard/0, hard/50, about right/100, easy/50, too easy/0
TA	t a, teachers assistant	very good/100, good/75, okay/50, bad/25, very bad/0

Figure 4: Topics used in Rate-A-Course system experiment

System	S: We will now ask your opinion on the following aspects of your course: the instructor, the assignments and the exams. Is the instructor: very good, good, okay, bad, or very bad?
Mixed	S: Which topic was the next most important to you? Your choices are ...
User	S: Would you like to discuss another topic? U: Yes S: Which topic is the next most important to you? Your choices are ...

Figure 5: System questions by initiative condition

and to populate VoiceXML forms that act as templates for different question types. Javascript embedded in the VoiceXML forms permits automatic logging of all system and respondent interactions.

Survey respondents are asked about five topics for a course that is being evaluated. For each topic, they are first asked to rate that aspect of the course (e.g. “Was the instructor very good, good, okay, bad or terrible?”). Then, they were asked to explain their rating (e.g. “Why did you think the instructor was okay?”). Figure 4 gives information about the course topics.

Possible answers to close-ended questions (e.g. “Was the instructor very good, good, okay, bad or terrible?”) and question-related keywords taken from the XML document are used to create recognition grammars; these permit respondents to answer close-ended survey questions using full or partial sentences, using the terms specified in the question or using synonyms of question terms. In this version of the system, no attempt is made to automatically process the answers to open-ended questions (e.g. “Why did you think the instructor was okay?”) during the survey.

The Rate-A-Course system permits respondents to ask for the last question to be repeated and to ask for help at any time. A request for help is interpreted as a request for clarification of the current question. The system also provides help on a recognition failure or no input; this help can be a simple repetition of the question, an explanation of the answers or an example answer, or a subdialog, depending on the XML specification for the survey. Respondents in the experiment described here were only allowed to go back or cancel for certain questions (e.g. course department and number).

The Rate-A-Course system generates structured logs in the form of question-answer pairs for all questions, as well as a complete dialog history with pointers to the audio files containing respondents’ speech.

### 3.2.2 Adaptation in Rate-A-Course

The Rate-A-Course system implements several different dialog behaviors leading to different amounts of system interactivity and adaptation:

- **Choice of question type:** In the XML document, survey designers can specify whether a question should be open-ended or close-ended and can specify valid answers to a close-ended question.
- **Question ordering:** The VoiceXML forms implement random ordering of questions when the survey designer does not specify question order.
- **Lexical adaptation:** The VoiceXML forms implement lexical adaptation, so that the system can adapt its choice of words and tense to the user's word choice.
- **Initiative:** There are VoiceXML forms for a survey with *system initiative* (the system chooses the question order), *mixed initiative* (the respondent chooses in which order to answer survey questions), or *user initiative* (the respondent chooses which survey questions to answer, as well as the order in which to answer them). Figure 5 shows how the system moves from topic to topic in each initiative condition.

The survey designer can give the respondent a code that pre-specifies system behaviors. This means that in addition to collecting survey data, the system can be used to perform research about spoken dialog and survey design.

**Lexical adaptation** is a type of a *directive adaptation* implemented in the system. In my experiments I will measure user adaptation for different

system initiative conditions.<sup>7</sup>

SYSTEM:	Welcome to the CMU Lets Go! Bus Information System. How may I help you?
USER:	Id like to go to Forbes and Murray.
SYSTEM:	To Forbes and Murray. Where are you leaving from?
USER:	Carnegie Mellon.
SYSTEM:	From Carnegie Mellon. There are five buses running between Carnegie Mellon and Forbes and Murray: the 501, the 59U, the 61A, the 61B, and the 61C. What else can I do for you?
USER:	When is the next one?
SYSTEM:	The next 61C going to Forbes and Murray will leave Carnegie Mellon at 5:13 PM.

Table 3: Sample dialog with *Let's Go*

### 3.3 *Let's Go* System: Online Bus Information

The *Let's Go* [Raux et al.2005] system is developed, maintained, and deployed at Carnegie Mellon University. This telephone-based system provides information about bus routes, departure times, and bus connections in Pittsburgh. The system is reachable through the local Port Authority number outside of the hours when human operators answer the phone lines. The system receives calls from users of all ages and language backgrounds.

*Let's Go* is developed using the Olympus framework. It has an architecture similar to the RavenCalendar application and uses many of the same components. The system uses the Sphinx 2 speech recognizer. The Phoenix parser performs extraction of concepts. The Rosetta template-based gen-

---

<sup>7</sup>Please see Chapter sec:RACexperiment for more details

erator is used for response generation. The Festival Speech Synthesizer converts system utterances to speech. The dialog manager is built on the RavenClaw system and implemented as a tree of agents. Similar flexibility (as described in 3.1) in the order of requesting concepts is achieved in Let's Go. To accommodate the diverse user population and noisy speaking conditions the dialog system is designed as a linear system-initiative dialog. The system asks the user for several concepts sequentially: an optional bus route number, a departure place, a destination and a desired travel time. Each concept is explicitly confirmed. The backend of the system is the database of all stops and bus routes in Pittsburgh provided by the Port Authority. Table 3 shows a sample dialog with the system.

The system receives on average 40 calls per day. Average call length is 12.9 turns with a large standard deviation. A 2005 call analysis showed a speech recognition error rate of 68%. The task success rate is estimated at 43%.

Since 2007, CMU researchers have provided the system as a resource for outside developers. It is a valuable resource for the dialog system research community as it provides a testbed on a real system for experimenters and a large and stable pool of users. I was granted permission to run two adaptation experiments on the Let's Go system. These experiments are described in Chapters 5 and 6.

### **3.3.1 Adaptation in Let's Go**

Let's Go system receives calls from anonymous users and it is not possible to identify repeating callers. This makes it impossible to use the system for *long-term* adaptation experiments, however the system can be used for

*directive* and *responsive short-term* adaptation experiments. The advantage of using the Let's Go system is that it has a large pool of real users. In Chapter 5 I describe proposed experiments on directive adaptation where I compare the effect of prompts and system's understanding in a form of named entity in users' utterances.

The speech recognition error rate in the Let's Go system is estimated to be 68%. This error rate is very high because of the diversity of the callers, their accents, age, and experience with dialog systems. Improving the speech recognition performance would benefit real users of the system. In my experiment I will try to improve the speech recognition performance by implementing responsive adaptation in the speech recognizer. With the high percentage of system errors we expect to see a high percent of user corrections in dialogs. In Chapter 6 I describe a proposed responsive adaptation experiment where the language model of the speech recognizer on correction utterances<sup>8</sup> is adapted to the dialog state.

## 4 Performed and Proposed Corpus Studies and Corpus Experiments

In this chapter I describe my preliminary analysis of the Communicator corpus of human-computer dialogs and propose further corpus studies on the Communicator and Let's Go corpora. In my preliminary studies I describe variations in users' utterances. I observe the forms of users' expressions, variation in function verbs, and concept forms used by experiment participants. Function verbs are an important for system's understanding of users' intention. Similarly to concepts, function verbs generally have multiple syn-

---

<sup>8</sup>Corrections are identified automatically. Please see Chapter 6 for more details



Speaker	utterance	system speech act/task-subtask
S	on what day would you like to depart?	request_info / depart_arrive_date
U	the twenty september twenty sixth	<b>system fails to recognize</b>
S	sorry, i misunderstood. please say the month and day of your departure.	apology / <b>slu_reject</b> request_info / depart_arrive_date
U	september twenty sixth	
S	what time do you want to leave?	request_info / depart_arrive_time
U	about two two o'clock	<b>system fails to recognize</b>
S	sorry, i misunderstood. please give the approximate time you would like to depart, or say anytime.	apology / <b>slu_reject</b> request_info / depart_arrive_time
U	anytime	

Table 4: Examples of dialogs snippets from Communicator corpus around system error

onymous realizations. I observe frequencies of several synonymous function verbs in travel domain and analyze users' choices for using a particular function verb. I also observe variation in the forms of concepts specified by users. System's understanding of concepts is essential to successful communication between a user and a system. Dates and times are ubiquitous concepts that are required in Communicator travel domain as well as in most other dialog system domains. I look at user utterances that mention dates and times concepts and observe users' and systems' strategies when concepts in user utterances are not recognized by the system. The goal of much dialog systems research is to reduce the frequency of system errors and find strategies for minimizing their impact and understanding users' strategies after system's misrecognitions is important for successful recovery from a system error. [Walker and Passonneau2001] describe annotation scheme of

Communicator corpus and table 5 lists the *speech act*, *task\_subtask*, and *conversational domain* labels of Communicator corpus. In this chapter I introduce a novel way for describing system’s policies: in terms of adding, deleting, or changing a speech act with respect to the previous system’s utterance. This analysis allows a more general description of a system policy. I analyze users’ strategies after a system error by comparing them with an utterance before a rejection and evaluate success rate of various user actions. Table 4 shows two example of Communicator dialog snippets where the system’s action and the user’s utterances before and after an error can be observed. In the first example user’s action is to rephrase its previous utterance and in the second one the user changes the request after a system’s error.

#### **4.1 Analysis of Variation in Communicator Corpus (Performed and Proposed studies)**

Communicator [Walker et al.2000a] is a human-computer dialog corpus in a travel assistance domain. The corpus contains over 3000 dialogs. Participants called one of nine dialog systems to arrange both real and imaginary travel plans. In each dialog, users had to fill in four slots: departure city, arrival city, departure date, departure time and optionally airline, hotel, and rental car. All of the participants called one system four times over six month period to arrange real travel plans. Half of the participants also called to arrange a fixed scenario plans that were given to them. The scenarios include a combination of simple round trips and multi-leg trips. In some cases user was also asked to reserve a car and/or a hotel, or request a particular airline. System messages of the corpus are annotated in a con-

Conversational Domain Dimension	Speech Act Dimension	Task-Subtask Dimension
about_communication, about_situation_frame, about_task	acknowledgment apology explicit_confirm implicit_confirm instruction opening_closing present_info status_report request_info	meta_error meta_inconsistent_info meta_slu_reject meta_correction meta_repetition meta_greeting_goodbye meta_hangup meta_instruct meta_situation_info meta_start_over meta_request_change meta_ambiguity_resolution unknown top_level_trip price_retrieval flight flight_booking hotel_booking rental_booking ticket_type trip_type trip continue_trip orig_city dest_city orig_dest_city orig_dest_date depart_arrive_date return_date depart_arrive_time airline_or_date_or_time depart_time_airline airline ground

Table 5: Communicator Annotations

versational domain dimension, a speech act dimension, and a task-subtask dimension. The labels are listed in the table 5. The following concepts in user utterances are annotated: dates, times, city, airport, airline, car. In the next section I present a descriptive study of variations in users' utterances.

#### 4.1.1 Performed Corpus Study: Variation in Function Verbs

One type of variation is a function verb variation. Let me define **trivial** utterances as those utterances that contain only a named entity, e.g. *four pm*, or *Dallas airport*. The **nontrivial** utterances are those that have other words than a named entity, for example *arriving at four pm*, or *from Dallas airport*. Only 30% of the users' utterances in Communicator corpus are nontrivial. I looked at variation of a function verb in a subset of nontrivial utterances. A common phrases specifying an itinerary in the corpus: "I'd like to ...". The paraphrases in the corpus are: "I'd like to fly", "I'd like to travel", "I'd like to go", and "I'd like to leave". I analyze whether each user tends to stick to a single function word or varies their choice of function words throughout his/her dialog(s). Table 6 lists frequencies of occurrence of these verbs in a phrase "I'd like to" across the corpus as well as the number of unique users who used each verb.<sup>9</sup> The table shows that *fly* is the predominant function verb in this phrase, followed by *go*, *leave*, and *travel*. Users tend to repeat the same function verb in the selected phrase approximately 3 times (average in Table 6) over all dialogs. Table 7 lists variation of the verb within a user over all dialogs: the numbers of users who used at least two of the function verbs throughout the experiment, not

---

<sup>9</sup>I chose to look at the phrase "I'd like to ..." because I wanted to narrow down on a context. Alternatively, I could have chosen a different phrase, like "I want to", or "<verb>ing to/on/from ..."

necessarily in the same dialog. For example, 13% of users who used word *fly*, also used the word *travel*. This data shows that there is a significant variation in a function verb choice within a user. Table 8 shows variation within a user in a single two of the verbs in the same dialog. Variation of a function verb within a user in a single dialog, the number of users who used at least two different function verbs in a single dialog, is lower than across dialogs, but it still exists. I hypothesize that this variation reflects the context sensitivity of the synonym function verbs. *Fly* and *travel* are not used in the same dialog because these verbs are mutually interchangeable in all contexts. They can be used with either date, time, departure, or destination. However *leave* can not be used with destination, so if a users may uses *leave* with departure or date concept “I’d like to leave on June fifth”, s/he will be likely to use *fly* or *go* (or nothing) with destination concept in the same dialog, e.g. “i’d like to go to Osaka Japan” .

Table 9 lists the function verbs in users’ utterances specifying departure or arrival time along with the occurrences of each of the terms in the dataset. In this experiment I narrow down to a concept (only looking at utterances specifying date or time) to avoid context sensitivity variation of the verbs. This analysis shows a rich overall variability<sup>10</sup> in use of function verbs by Communicator experiment participants.

#### 4.1.2 Proposed Corpus Study: Analyzing Variation in Concepts

Another type of variation is a variation in specifying concepts. Communicator corpus is an air travel dialog. In each dialog, users had to fill in four slots: departure city, arrival city, departure date, departure time and

---

<sup>10</sup>this does not reflect variability within a user

expression	number of times used	number of unique users	ratio (non-unique/unique)
I'd like to fly	148	38	3.8
I'd like to travel	45	13	3.46
I'd like to go	77	23	3.35
I'd like to leave	54	28	1.9
average			3.13

Table 6: Counts on paraphrases

I'd like to	fly	travel	go	leave
fly	X	13%	32%	44%
travel	30%	X	31%	8%
go	52%	17%	X	52%
leave	61%	4%	43%	X

Table 7: Number of users that used at least two verbs in a phrase “I'd like to ...” throughout the experiment, e.g. 13% of users who used word *fly*, also used the word *travel*.

I'd like to	fly	travel	go	leave
fly	X	0	21%	32%
travel	0	X	15%	0
go	35%	9%	X	39%
leave	43%	0	32%	X

Table 8: Number of users that used at least two verbs in a phrase “I'd like to ...” in a single dialog, e.g. fly and travel, fly and go, etc.

verb	count
fly	248
flying	10
travel	50
traveling	5
go	91
going	46
leave	253
leaving	150
depart	41
departing	44

Table 9: Counts on synonymous function verbs used with DATETIME concept

optionally airline, hotel, and rental car. There are typically not very many ways to refer to each departure or arrival city, but there are several ways to describe dates and times. *Let's Go* corpus is a human-computer dialog corpus where users request information about buses. *Let's Go* system, similarly to Communicator systems, requests a user to specify time. Table 23 shows variation of users specifying time. In my analysis I will look at forms of concepts before and after system misrecognition. This will allow me to study how users change the forms of the concepts in their utterances. In the Chapters 5 I describe experiments on directive adaptation in dialog systems. The corpus study proposed here will give an insight for the system experiments of the next chapter.

Next, I describe several hypotheses for this study and my approaches to proving these hypotheses.<sup>11</sup>

---

<sup>11</sup>I will perform this analysis on date and time concepts in Communicator corpus and

**Hypothesis 1. I hypothesize that users tend to use the concept form used by a system, but if the concept is not understood by the system, user will paraphrase the concept.** In both Communicator and in Let's go dialogs system generally use the same form of date and time concepts: "Monday, January fifth", "X pm" . Users's forms of concepts, however, contain variations. I will first estimate prior probability for each form of time concept realization using all communicator and Let's Go corpora. Then I will compute the proportion of the first user's mention of date/time concept for each realization. I will compare this proportion with the prior probability of the the system's form ("X pm"). If it is greater and the difference is significant, I will conclude that users tends to follow the system in the form of specifying concepts.

**Hypothesis 2. I hypothesize that some concept forms are understood better than others.** I will compute average error rate for each concept form realization and perform ANOVA to determine whether errors in recognizing different concepts are equal. If they are not equal, I will use Tuckey pairwise comparison test to determine the best recognized form of concept. I will do this analysis both on Communicator and Let's Go data.

**Hypothesis 3. Over trial and error users find the optimal (most frequently recognized) form for a named entity.** I will track average error rate for recognizing users' date and time concepts over time, e. g. average error rate in the first dialog, second, third, etc. and graph the result.<sup>12</sup> I expect the error rate in the later dialogs to be lower than in 

---

on time concept of *Let's Go* corpus (except when comparing between systems). I may decide to add other concepts to this analysis

<sup>12</sup>I may have a more fine-grain breakdown of dialogs



the earlier dialogs. I will compare proportion of the most frequently used time concept form in the last mention of a time concept by each user in the last dialog of this user with the Communicator system<sup>13</sup>. If the second hypothesis turns out to be true and I can determine using Tuckey test the best recognized form of time concept, I will then see whether the most frequent time concept form in the last users' mention will be the one with the highest recognition rate. I will use the large sample inference on proportions to compare the difference between proportions of the highest and the second highest time concept form to determine whether users actually converge on the form of time concept.

## 4.2 Performed Experiment: Analysis of Rejections in the Communicator Corpus

I extracted utterances in a window around system misunderstandings across the corpus. To identify locations of system misunderstandings, I focused on the *slu\_reject* system speech act taken by every system in cases of non-understandings. I present analysis of *slu\_reject* system action in this chapter because analyzing errors in a dialog system is important for understanding of how a system may recover from an error.<sup>14</sup> I extracted a five-turn window (two before, two after) around each system turn that contained an utterance labeled *slu\_reject*. Table 10 describes the features extracted for my analysis.

---

<sup>13</sup>There are 180 unique users in Communicator

<sup>14</sup>I may choose to do analysis of some of the other system's speech acts *meta\_request\_change*, *meta\_ambiguity\_resolution*, *meta\_correction* *meta\_repetition*, *meta\_instruct* as well in the future.

feature	description
Sys NE -1	List of NE types mentioned in the previous system turn
Sys NE -1	List of NE types mentioned in the previous user turn
Sys NE 0	List of NE types mentioned in the current system turn
Sys NE 0	List of NE types mentioned in the current user turn
Sp. act -1	Speech acts present in previous system turn
Task -1	task-subtask present in previous system turn
Sp. act 0	Speech acts present in current system turn
Task 0	task-subtask present in current system turn
NumNE-sys-1	Number of NEs mentioned by the system
NumNE-sys-0	Number of NEs mentioned by the system
NumNE-usr-1	Number of NEs mentioned by the user
NumNE-usr-0	Number of NEs mentioned by the user
txt-match-sys (0/1)	match between previous and current system utterance 1: system repeats itself
NE-match-sys (0/1)	match between previous and current system NE types 1: system mentions the same NEs
task-match-sys (0/1)	match between previous and current System task-subtask 1: system repeats the task
spact-match-sys (0/1)	match between previous and current System speech act 1: system repeats the speechact
txt-match-sys0-usr-1 (0/1)	system repeats what user said
NE-match-sys0-usr-1 (0/1)	system repeats a named entity that user said
txt-match-usr (0/1)	match in user utterance before and after a misrecognition 1: user repeats an utterance
ne-match-usr (0/1)	match between previous and current user NE types 1: user repeats the NE

Table 10: Attributes extracted

### 4.2.1 Causes of *Slu-Reject* in the Communicator Corpus

The Communicator corpus contains a total of 4226 system rejections of user utterances. Approximately 300 are in response to empty user utterances and 980 are in response to non-empty user utterances that are recognized correctly by the automatic *speech recognizer* (ASR) but rejected by the *natural language understanding* NLU component, or parser. 80 (from the 980 recognized but rejected utterances) are negative answers to yes/no questions. 2518 utterances have no ASR error in a concept. This analysis indicates that more than half of rejections in the Communicator systems are caused by the NLU component and are either parsing failures or interpretation failures.

Many of the parser errors are generated when 1) the user attempts to take the initiative by asking a question, 2) the system asks a question that the user does not answer, instead addressing a different topic, or 3) the user initiates a correction. Table 11 shows examples of rejected user's utterances where the cause of the rejection is other than speech recognition error. It is difficult to tell why the correctly recognized utterance is rejected because it may depend on a state in a dialog system. It is possible that this system would never accept the rejected utterance because it is not in the grammar. It is also possible that the grammar rule that matches the rejected utterance is not activated at that state of the dialog.

I distinguish between the two outcomes following a *slu-reject*: **success** (successful recovery) and **failure** (failure to recover). Failure occurs when a rejection is followed by another rejection (see example in the Table 12), and success occurs when a rejection is followed by a non-rejection. I chose this metric because I would like to minimize the length of misunderstandings in a dialog by maximizing the rejection recovery rate. I use this as a metric to

dialog	system	user
1026 SRI	what time do you want to leave?	late morning
1029 SRI	what would you like to change?	time
1141 CMU	is that OK?	that's great
1665 SRI	is that correct?	correct yes
1418 Lucent	was the departure time around six a. m. or around six p. m.	around noon
1653 BBN	holiday inn ... do you want it?	what's the location ?
1219 Lucent	leaving chicago on what date?	correct arrival city
340 MIT	okay from zurich to denver. what date will you be traveling?	zurich to kansas city

Table 11: Examples of slu rejections not caused by the ASR

S:	sorry, i misunderstood. please say your departure city or airport. for example, san francisco or jfk.
U:	ASR failure
S:	sorry, i misunderstood. please say the name of an airport or a city that contains an airport. for example, heathrow airport or london.

Table 12: An example misunderstanding sequence

System	num reject	num second reject	% second reject
AT&T	181	19	10.5%
BBN	334	72	21.56%
CMU	482	114	23.46%
IMB	380	113	29.74%
Lucent	737	241	31.34%
MIT	292	74	25.34%
SRI	1706	999	52.76%
total rejects	4118	1632	39%

Table 13: Counts of rejects and consecutive reject per system

analyze the strategies used by the Communicator systems to recover from dialog errors.<sup>15</sup>

Table 13 shows the percentage of times a rejection is followed by another rejection on a per-system basis. The percentage of consecutive rejection ranges from 10% to 50% with a mean of 39%.

In the following analysis I try to identify causes of consecutive rejections and successful dialog strategies for recovering from a system misunderstanding. Both user and system behavior may contribute to the outcome after a rejection. I look at system *speech\_act* and *task\_subtask* labels and try to identify a correlation between these tags and success. Similarly I classify user behavior according to the types of named entities mentioned and correlate these tags with success/failure.

---

<sup>15</sup>Although it is not completely fair to consider a non-rejection as a successful outcome because in the case of change of topic, a misunderstood slot is still not filled.

speech act (1)	count occurring (2)	count omitted (3)
request_info	1909	
instruction	1590	706
offer	835	316
present_info	422	
implicit_confirm	220	300
explicit_confirms	187	132
acknowledgment		192
apology	4118	

Table 14: Counts of speech acts co-occurring most frequently with *slu\_reject* tag (first column), and number of times omitted (second column)

#### 4.2.2 System Strategies for Handling Non-Understandings

A system’s utterance after a non-understanding of a user’s utterance is labeled as a *slu\_reject* in *task\_subtask* dimension in all of the Communicator systems (see examples in Table 4). First, I look at speech acts co-labeled with *slu\_reject* *task\_subtask*. Table 14 (*count occurring* column) lists most frequent speech acts co-labeled with *slu\_reject* and a number of times that each of them is co-labeled with *slu\_reject*. Every Communicator systems implemented some strategy for handling non-understandings. All of the systems apologize for a mis-recognition, and then either repeat or paraphrase a question, add an instruction, or omit a part of the question.

Next, I look at system behavior after a failure to recognize a user’s utterance in terms of a system’s **action**. I define system’s **action** as the difference between the speech acts of a given system utterance with the speech acts of a previous system utterance. The possible system actions

are the Cartesian product of: ADD[0|1] OMIT [0|1] CHANGE[0|1]. ADD indicates that one or more new speech act(s) were added, OMIT – that one or more of the speech act(s) were omitted (the second column of the Table 14 shows the counts for omissions of several speech acts), and CHANGE means that one or more of the dialog act(s) kept a speech act while changing a task.<sup>16</sup> Table 15 lists all system *actions*, numbers of their occurrences in corpus, and percentage of success.<sup>17</sup> The most frequent *actions* used by the systems are OMIT and REPEAT with the recovery success rate of 69.6% and 71.4% respectively. The least successful actions are ADD and ADD&CHANGE with recovery success rate of 39.6% and 47.3% respectively. These proportions are significantly lower than success ratios of all other *actions*<sup>18</sup>. Also, REPEAT is significantly less successful than ADD&OMIT. Other actions that are also successful but not highly utilized are CHANGE and OMIT&CHANGE. I hypothesize that successful recovery depends on other factors besides system’s action. In the next section I describe how I use this data to train a classifier for predicting probability of success for each of the system **actions** using dialog history information such as the sequence of preceding speech acts, word error rate, etc.

### 4.2.3 Experiment: Predicting System’s Action

In this experiment I evaluate a possibility of automatically learning the best system action using current dialog state (task, speech act), dialog history (previous dialog states), match between user and system current and previous utterances and dialog acts, and the speech recognizer’s word er-

---

<sup>16</sup>Dialog act is a tuple: (speech act, task\_subtask)

<sup>17</sup>These counts correspond to the first reject in a sequence

<sup>18</sup>Using inference on proportion of large sample (p-value<.05)

ACTION	num	number (%) success	Example Speech act before reject	Example Speech act after reject
REPEAT	624	435 (69.6%)	reqinfo dep-arr-date	req-info dep-arr-date
OMIT	822	587 (71.4%)	<i>imp-conf dep-arr-date</i> , req-info dep-arr-time	req-info dep-arr-time
ADD	180	76 (39.6%)	req-info dep-arr-time	present-info flight , request-info depart-arrive-time
ADD&OMIT	584	462 (79.2%)	ack flight-booking	present-info flight, req-info price
CHANGE	104	79 (76.0%)	exp-conf dep-arr-time	exp-conf orig-city
OMIT&CHANGE	90	71 (79.8%)		
ADD&CHANGE	224	72 (47.3%)	explicit-confirm depart-arrive-time	apology slu-reject, explicit-confirm orig-city
ADD&OMIT&CHANGE	43	33 (76.7%)		

Table 15: Dialog strategies and their statistics



features	f-measure fail	f-measure success
baseline ( always predict success)	0	.7
system features ( WITH System id and WER)	.677	.817
system features ( no System id )	.617	.776
system features ( no System id , no WER)	.523	.775
system features (no System id , no WER, no spact of task-1)	.518	.769
system features, task 0 (no System id , no WER, no spact of task-1 and task0)	.518	.769
only system id and WER	.618	.812
only WER	.624	.751

Table 16: F-measure for predicting the outcome of the rejection

ror rate. I performed AdaBoost [Schapire and Singer1999] rule learning using the WEKA machine learning tool [Garner1995] to predict the outcome of rejection using system-related features from the Table 10. Following a misunderstanding, a system could use the learned model to automatically choose the action with the highest probability of a successful outcome. [Bohus et al.2006] describe a similar experiment in learning policies: system’s speech act as a response to a non-understanding. The authors show that the learned policies perform significantly better than the hand-coded baseline. In my experiment the policies or **actions** are specified in terms of the omission or addition of a speech act to the speech act in the utterance preceding the misunderstanding. Table 16 reports classification accuracy using 10-fold cross validation. The result shows that failure/success in the outcome is predicted best using just the word error rate feature (f-measure .62/.75). Predicting the outcome using only system features available at runtime (no system id or word error rate) and based on the match of sys-

behavior % match   reject (1)	count (2)	percent of occurrences (3)	percent fail to recover (4)	description (5)
0	1973	47.9%	31.9%	heightcomplete rephrase
0 ... 25]	2266	<b>55%</b>	32.7%	significant rephrase
(25 ... 75]	1101	26.7%	48.0%	rephrase
(75 ... 1]	763	<b>18.5%</b>	47.4%	very close rephrase
total	4118	100%	39.6%	

Table 17: Analysis of users’ paraphrase strategies after a rejection

tem’s current and previous speech acts and speech acts is significantly lower for a failure outcome (f-measure .52/.78). Taking *speech act* and *task* out of the feature set further lowers the f-measure for predicting both failure and success. These results show that word error rate (WER) is the most valuable feature in determining the outcome. Adding system features to WER does not improve the prediction f-measure. Both the system’s *action* (defined in terms of the difference between current and past system utterances) and the history (current and past speech acts and tasks) can be used to predict the user’s action, however less accurately than WER alone.

#### 4.2.4 Users’ Strategies on Rejections

In this section I analyze user behavior after a system fails to recognize a user’s utterance. System errors are inevitable and successful recovery (correct understanding of the next user’s utterance) is essential for successful communication between a user and a system. I consider consecutive rejection an indication of recovery failure. Understanding user behavior after a system error gives an important insight for a design of speech recognition and language understanding components of a dialog system. For example, if we determine that a user is very likely to repeat an utterance in a particular

situation, we may combine speech recognition hypotheses from an utterance before and after an error.<sup>19</sup> In my analysis I use **% match**, a measure of string similarity between user utterances before and after a system error. Match between two utterances is computed using dynamic programming. I observe how the **% match** correlates with system's failure to recover from an error.

To analyze user behavior after a system's rejection I cluster users' utterances after a rejection according to user's *strategies* derived from the **% match**: 0 % match (complete rephrase), 0 .. 25% (significant rephrase), 25 .. 75% (rephrase) , 75% .. 100% (close rephrase). The second and the third columns of table 17 show the numbers and proportions of the users' *strategies* after a rejection. According to this data, majority of users' utterances after a rejection are complete or significant rephrases (comprising 55% of all after-rejection utterances). Only 18.5% are close rephrasees. The fourth column shows the percentages for non-recovery from a rejection for each of the user strategies. It is interesting to notice that complete and significant rephrases have significantly lower proportion of recovery failures<sup>20</sup>. This result is not unexpected: a non-understanding may occur because a)user's utterance is out of system's grammar, b) there is external noise, c) a user breathes or mumbles. Repetition (or close rephrasing) will fix non-understanding only in cases (b) and (c). If an utterance is not in-grammar, a significant rephrase is more likely to create in-grammar that can be understood. This analysis indicates that users of Communicator systems make reasonable choices when rephrasing their out-of-grammar utterances.

In [Litman et al.2006] the authors analyzed the TOOT (train informa-

---

<sup>19</sup>This is a theoretical idea, I do not develop it in my proposal

<sup>20</sup>Compared using inference on proportions of a large sample (p-value<.05)

Directive System utterance	Adapted User utterance	Non-adapted User utterance
Which date will you be <i>departing on</i> ?	<i>Departing on</i> November first	I <i>would like to leave</i> on November first
Please say <i>the departure date</i>	My <i>departure date</i> is November first	<i>Departing on</i> November first
On what day <i>would you like to depart</i> ?	I <i>would like to depart</i> on November first	<i>Departing on</i> November first

Table 18: Examples of directive prompts, adaptive, and non-adaptive user’s utterances

tion) corpus for user strategies after a system rejection or a misrecognition. User’s actions were analyzed in terms of adding, omitting, or changing a concept. Contrary to Communicator corpus, the post-rejection utterances are more likely to be repeats in TOOT corpus. This supports the author’s suggestion of the “surprising power of system directions”. In TOOT system rejection message for tasks was always a close paraphrase of *Sorry, I cant understand you. Can you please repeat your utterance?* while Communicator systems used a variety rejection prompts.

The high probability of a consecutive system rejection gives motivation for adjusting a speech recognizer’s language model for user’s utterances after a system’s rejection. The proposal for this experiment is described in the Chapter 6.

## 5 Directive Adaptation: Exploring Prompt Strategies

*Directive* adaptation in a dialog system is exhibited when a system guides a user into using a particular grammar or vocabulary. For example, when a system prompt uses a particular function verb, form of a referring expression,

or form of a concept. Table 18 gives several examples of system utterances asking user for a departure date. In my studies of adaptation I assume that each of these system's utterances *primes*, or directs a user into using a particular form of utterance by mimicking a verb, a referring expression, or a concept form used by a system<sup>21</sup>. Second column of the table 18 gives examples of *adapted* user utterances that reflect the structure of the system's expression<sup>22</sup> and the third column gives examples of *non-adapted* user utterances that do not reflect the structure of the system's utterance.

It is interesting to study directive adaptation because it gives an insight on how a system may manipulate or direct a user into using a particular verb, a form of a concept, or a syntactic structure. It is known from psycholinguistic research that in human-human communication partners follow each others syntactic and lexical choices, and it is interesting to investigate whether this is the case in human-computer dialogs and to what extent. In this chapter I propose the experiments which will form a descriptive study on *directive adaptation* in dialog systems. While it is an interesting theoretical question, it may also have practical implications of improving performance in a dialog system.<sup>23</sup>

In my work I will focus on the particular points of adaptation: 1) in a form of concept, 2) in a lexical choice for a synonym, and 3) in a syntactic choice (present/past tense for the verb "to be"). I choose to look at variation in the form of a concept because in practical dialog systems users often avoid using full sentences and answer system's questions using single concepts. Rate-

---

<sup>21</sup>If the user uses a sentence and not a simple *concept* to answer a question

<sup>22</sup>Of course there may be several gradations of adaptation, e.g. adapted the verb, but not syntax or adapted syntax but not verb, etc.

<sup>23</sup>Please see Section 1.4 for the more details on motivation for using directive adaptation in dialog systems.

a-course system forms a suitable platform for studying variation of lexical choice for a synonym and syntactic choice for present/past tense of the verb “to be” as it was specifically designed for this purpose.

Section 4.1.2 describes a corpus analysis for directive adaptation. In the corpus study I did not have control over system’s prompts and language understanding. This experiment will allow me to analyze user behavior while varying system prompts and language understanding grammar. In my experiments with the Let’s Go and with the Calendar systems I look at places of errors in a dialog, where a user’s utterance is not understood by a system and has to be requested again. This allows me to formulate a *directive prompt* for a system’s confirmation or for a rejection of a user’s utterance; and to compare user’s utterance before and after an error. A user’s utterance after a system error may be a repetition or a paraphrase of the previous utterance, or a complete topic switch. This utterance may or may not follow the system’s *directive* prompt.

Quick and efficient error recovery is essential for a practical dialog system. Looking at locations of system errors also allows me to evaluate effect of system prompts on system performance and to determine the type of prompt that will allow quicker error recovery.

## **5.1 Performed Experiment: Adaptation in User’s Lexical and Syntactic Choices in Rate-a-Course system**

The Rate-a-Course dialog system [Stent et al.2006] allows users to evaluate their courses using a spoken language interface. Chapter 3.2 describes the Rate-a-Course survey system. The system runs on the BeVocal Cafe platform and uses its proprietary speech recognizer (<http://cafe.bevocal.com>).

id	utt type	S/U	Utterance
1		S:	Please choose one of the following: <b>instructor</b> , class size, etc.
2	topic choice	U:	<b>instructor</b>
3		S:	<b>Was</b> the <b>instructor</b> very good, good, okay, bad, or very bad
4	topic rating	U:	very good
5		S:	Please describe why do you think <b>instructor</b> was very good
6	free response	U:	the <b>professor was</b> very funny at times also enter- taining as well as very informative and [um] it was obvious to all the students that it was a very in- tellectual person [uh] definitely knew what he was talking about and [um] yeah it was a good class

Table 19: An example of user adapting to the system in Rate-a-Course system

A sample dialog with Rate-a-course system is illustrated in the table 19. The Rate-a-course system runs in one of the adaptation conditions (adaptive/ non-adaptive) in combination with one of the initiative conditions (system/mixed/user). The *concepts* of the system are the topics that may be evaluated by a user: teacher, exams, homework, etc. Each concept has several forms of realization (e.g. teacher/lecturer/instructor). The system at first uses a randomly-selected form of each survey concept. In the adaptive condition the system switches after a user’s utterance to the user’s choice of form for survey concepts, if it was different from the system’s. In the non-adaptive condition the system never switches its form of a concept. The three types of user’s utterances in the system are: topic choice, topic rating, and free response. User’s topic choice and topic rating are recognized automatically with a grammar-based recognition. Either of synonymous

word word	system prompts (count unique)	user’s topic choices	user’s free responses
lecturer	9	7	1
teacher	6	10	17
professor	4	6	25
instructor	28	26	13
exams	26	13	27
examinations	6	5	2
tests	5	8	15
quizzes	9	8	8

Table 20: Number of concept occurrences in system prompts, user’s free responses, user topic choices

forms of a concept (instructor/lecturer/teacher, etc.) can be recognized. A topic rating can be either a sentence, e.g. “Instructor was good” or a simple concept, e.g. “good”.<sup>24</sup> Free responses are hand-transcribed. In the data analysis I look whether a user follows a system’s choice of synonym for a concept (instructor/lecturer/teacher). In the *topic rating* of the example in table 19 user follows the system’s synonym choice (instructor). In the *free speech* response user picks a different synonym choice (professor).

### 5.1.1 Data Analysis of the Rate-a-Course Experiment

We have already ran 48 experiment participants. 16 subjects were in system initiative, 16 were in mixed initiative, and 15 in user initiative conditions.<sup>25</sup> In each of the initiative conditions half of the subjects were in the non-adaptive condition (total 24) the other half in adaptive condition (total 23). Each subject evaluated two courses. I analyze the data by counting numbers of occurrences of each concept (instructor, exams, TA, difficulty,

---

<sup>24</sup>Users predominantly used a simple concept in a topic rating and hence I do not include this field in my analysis

<sup>25</sup>One of the 48 subjects was disqualified because s/he was not a native speaker



condition	number of sub- jects	same in free	same in topic	diff in free	diff in topic
System Initiative TOTAL	16	12 (.414)	0	17 (.586)	0
System Initiative BOOL	16	7 (.43)	0	9 (.57)	0
Mixed Initiative TOTAL	16	6 (.400)	29	9 (.600)	2
Mixed Initiative BOOL	16	4 (.307)	16	9 (.693)	2
User Initiative TOTAL	15	2 (.167)	14	10 (.833)	4
User Initiative BOOL	15	2 (.182)	9	9 (.818)	4
System Adaptive TOTAL	23	10 (.385)	21	16 (.615)	4
System Adaptive BOOL	23	8 (.363)	13	14 (.636)	4
System Nonadaptive TOTAL	24	10 (.333)	22	20 (.667)	2
System Nonadaptive BOOL	24	6 (.315)	12	13 (.684)	2

Table 21: Number of *instructor* concept occurrences in user’s free responses

class size).<sup>26</sup> Table 20 lists the counts for the number of times each concept form is mentioned in:

- system’s prompt
- user’s topic choice
- user’s free responses

In a non-adaptive condition the system does not switch its word choice in the conversation with a user. In adaptive condition system switches a concept’s word choice only if a user uses a different word for a concept in a *topic choice* or *topic rating* utterance. However, user’s topic choice generally follows the system’s word except for 6 cases out of 49. I would not attribute this to *adaptation*, because a user is given an explicit set of options from which s/he makes a choice. In my experiments I will look at synonym choice adaptation in user’s free responses to get an insight on presence of adaptation. I compare the proportions of *same* and *different* concept word

---

<sup>26</sup>Here I have analyzed *instructor* and *exam* concepts. I am still to analyze the other 3 concepts.

choices in free responses. *Same* is the count of a user repeating the system’s initial choice of concept and *different* is the count of a user picking a different concept. Table 21 shows the counts of *same* and *different* word choices for instructor concept used in each condition. Rows labeled as TOTAL show total counts and proportions of usages of the *same* and *different* concept words as the system initial prompt. Rows labeled as BOOL show counts and proportions of individual users that use the *same* and *different* concept words as the system initial prompt.<sup>27</sup> System initiative condition has 0’s for topic choices because in this condition a user never specifies a topic choice. User initiative condition has overall lower counts because users in this condition do not have to describe every aspect of the class and they generally skip some of the aspects. The numbers in “same in free” and “different in free” do not add up to the total number of subjects because users can say the word in a free response zero times or multiple times.

### 5.1.2 Results

First I compare adaptation in *adaptive* and *nonadaptive* system conditions. I use a *large sample inference on proportions* by comparing the proportions of *same* concept choice in free responses. According to the test the difference between proportions is not significant, hence we can not make any conclusions about difference of users’ adaptation in system-adaptive and system-non-adaptive conditions (either when comparing TOTAL or BOOLEAN). It is hardly unexpected that I did not get a significant difference because in this experiment the system had a chance to adapt only 4 out of 24 times in adaptive condition and the system did not adapt when it could have

---

<sup>27</sup>If a user uses both same and different concept s/he is counted twice

Utterance before error	after error
british airways baltimore washington international houston hobby airport	british baltimore houston airport
two oh seven two p. m. september twenty fourth	two oh seven p. m. two o'clock p. m. september two four

Table 22: Example of a non-recognition

in 2 out of 23 non-adaptive conditions. The lack of opportunity for the system to “adapt” makes this data insufficient for comparing adaptive and non-adaptive conditions.

Another interesting question is to consider all data and to determine whether there is adaptation in the use of concepts. However, my data is highly unbalanced: the system uses one of the options over 50% of the time (table 20)<sup>28</sup>. Also, I do not know prior probabilities of each of the concept word choice (lecturer vs. professor vs. instructor etc.). Without balanced design or knowing the prior probabilities of each synonymous term I have no way of estimating the proportion of co-occurrences by chance. In order to answer the question “Is there adaptation in the use of concepts?” I need to run more subjects to balance out the data.

## 5.2 Proposed Experiment: Matching NLU and NLG for the *time* Concept

I classify errors in a dialog as misrecognitions and non-recognitions as in [Bohus et al.2006]. Misrecognitions are the places where user’s utterance is recognized incorrectly. Non-recognitions are the places where user’s utterance is not recognized at all. To handle non-recognitions a system generally

---

<sup>28</sup>This was not intended by the design

format	number
Am/pm	5
Ass soon as possible	7
Early morning	1
Now	54
NUM	30
NUM am	57
NUM MIN	33
NUM MIN AM	36
NUM oclock	42
NUM oclock am/pm	7

Table 23: Variations of time concepts specified by users of *Let's go* system

uses a *rejection* utterance notifying the user of an error and re-prompting the user, or asking to repeat or rephrase the previous utterance. Handling misrecognitions is a little more tricky as the system thinks that it understood the user. A user's correction utterance may follow system's implicit confirmation. While performing corpus studies on the Communicator corpus I have observed that after rejections users often try to vary the forms of *concepts*. Some examples variations are shown in Table 22. In this experiment I will investigate how users change their tactic for interacting with a dialog system when encountering errors and test whether a form of a prompt can influence a user. This experiment focuses on variation of a form of a *concept* specifying time. Time concept is interesting because it is used throughout different systems (including my systems and data that I am looking at: Let's go, Calendar, Communicator). From an initial analysis of users' utterances in *Let's Go* dialogs we see a high variability in user's utterances specifying time. The most frequent forms for users' specifying time in the *Let's Go* system are relative (*now*), HOUR, HOUR am, and HOUR o'clock (see table 23). We define **directive prompts** to be system prompts that (a) realize a non-understanding or explicit confirmation dialog act, and (b) express a *time*

*concept* in a form that can be understood by the system’s grammar. For example, the system’s utterance “leaving at *twelve am* and arriving at *one fifty seven pm* with a stop in dallas” primes a user for the form of time concept *X pm/am*.

### 5.2.1 Experiment

Table 24 describes six dialog situations that I will compare. The dialog situations differ in terms of the *system prompt type*, *NLU grammar*, and *match between the prompt and NLU*. The prompt type is either *generic* or *specific*. In the case of a *generic* prompt type, system does not mention the *concepts* of interest (time). Examples of such prompts are: “I did not understand you”, “I did not understand you, please repeat”, “Please say it again”, “Ha?”.

The *NLU grammar* of the system can be **specific** or **flexible**. In the case of a specific NLU, the system is limited to understand only one of the forms of time, e.g. “N o’clock”. In the flexible condition all known forms can potentially be understood.

When both prompt and NLU grammar are *specific* and use the same form, there is a *match* between prompt and NLU component (condition 2). When both prompt and NLU component are *specific* but use different forms, there is *no match* between prompt and NLU (condition 3). The last system condition (condition 6) is *adaptive*: the system switches its prompt to the form of a *concept* previously used by the user.

Other concepts in *Let’s Go* system are dates and bus routes. I will perform a similar experiment on variation of these concepts as well.

	prompt type	NLU	Match	Prompt realization	NLU accept format
1	Generic	Specific	n/a	I did not understand	N o'clock
2	Specific	Specific	yes	Did you say <b>X o'clock?</b>	<b>N o'clock</b>
3	Specific	Specific	no	Did you say <b>X o'clock</b>	N pm/am
4	Generic	Flexible	n/a	I did not understand	N pm/am, N o'clock, N
5	Specific	Flexible	n/a	Did you say <b>X o'clock</b>	N pm/am, <b>N o'clock</b> , N
6	Specific	Flexible	n/a	<b>adaptive</b>	N pm/am, N o'clock, N

Table 24: Dialog situations for exploring adaptation in mentions of time

### 5.2.2 Hypotheses

In this experiment I am interested to detect what affects user’s utterances more: the form of system prompts or the flexibility of the NLU grammar. I hypothesize that:

- Users are affected (or primed) by system prompts. Specific prompts and matching NLU grammars (conditions 2,5) will lead to more user adaptation than generic prompts (condition 1, 4)
- Users follow prompts more when the NLU grammar forces them to. Specific prompts and matching specific NLU grammars (condition 2) will lead to more user adaptation than specific prompts and flexible NLU grammars (condition 5)
- **Which has more effect: phrasing of the prompts or grammar of the NLU?** Specific prompts and flexible NLU grammars (condition 5) will lead to more user adaptation than specific prompts and non-matching specific NLU (condition 3).

I additionally hypothesize the following about misrecognition rate and recovery from misrecognition:

- In the case of “misleading” prompts, users can still learn from NLU

errors; however, I expect more errors and longer rejection recovery in the condition with specific prompts and non-matching specific NLU grammars (condition 3). I expect that it would be more difficult for users to figure out a workable way of referring to times if NLU grammar and system prompts do not match.

- I predict more errors and longer rejection recovery in the case of generic prompts and specific NLU grammars (condition 1) than in the case of specific prompts and specific matching NLU grammars (condition 2).

### 5.2.3 Measurement

I will compute the frequency of use of each form of referring to times (*N pm/am*, *N o'clock*, *N*) in user utterances in each condition. I will use unpaired t-tests to determine the effect on user utterances of (1) the system's prompt form, and (2) the NLU grammar. Then I will compare the strength of the two effects using ANOVA.

### 5.2.4 Implications of the Experiment

Whether the system uses statistical or grammar-based speech recognition, VoiceXML or some other type of the dialog management, it is important for the system to expose its capabilities to the user. This experiment looks at one aspect of system behavior (form of *concepts* used in a prompt) and evaluates the potential of using system prompts to implicitly prime a user to use a particular form. This experiment follows the idea introduced by [Tomko and Rosenfeld2006] in their *shaping* experiments.

The result from this experiment will be a descriptive study on the effect of directive prompts. Some of the findings may have a direct practical effect

on dialog systems:

- if the condition with specific prompt and flexible NLU significantly increases the chance of a user using the same format as the prompt, then adjusting grammar probabilities (so that NLU is still flexible) may be beneficial.
- If I can reliably predict the form of a *concept* the user will use, it may improve the possibility of concept spotting on the level of speech recognition.

### 5.2.5 Using *Let's Go* System for Matching of NLU and NLG Experiment

The Let's Go system is a live system receiving calls from real users, described in Chapter 3.

The modifications to the system for this experiment are limited to the parser and language generation (NLG) components. For the *generic prompt condition*, I will 1) change the NLG on confirmations to not use the time concept when possible to avoid it; and 2) change the confirmation strategy to never use implicit confirmations. In some cases the system will have to present users with bus time, also in some dialogs users never specify time. These dialogs (or the part of the dialog after the system presented the user with a time) can not be used in the experiment.

For the *specific prompt condition*, I will change the NLG to specify the format in a particular form. I propose to have 3 versions with 3 types of matching prompt and NLU grammar corresponding to the most frequent forms from previous Let's Go dialog data: *HOUR [MIN]*, *HOUR o'clock*, and *HOUR [MIN] am/pm*.



For the *specific NLU condition*, I will limit grammar rules for time recognition of the parser.

*Let's Go* is estimated to collect 500 dialogs in 2 weeks of running. If we conservatively assume that 20% of the dialogs from each condition will be usable, we would need to run the system for 2 months to get 40 dialogs for each condition.

### **5.3 Proposed Optional Experiment on Calendar System**

The *Let's Go* system is the optimal candidate for the experiment described in Section 5.2 because of access to a large number of potential participants who are real users of the system. However, *Let's Go* is a live system and if some of the conditions cause significant degradation of the system's performance it may not be possible to run those conditions on the *Let's Go* system. In this case I will use the calendar system for the experiment described in this chapter. In addition to using time concepts, calendar system requests users for a date concept. I will perform a similar experiment with date concepts on the Calendar system.

#### **5.3.1 Wizard-of-Oz Type Experiment**

Wizard-of-Oz type experiment is widely used in psycholinguistic studies. In this experiment, a user communicates with a system, while there is a human "wizard" present between the user and the system who determines system's actions. This enables researchers to evaluate ideas and aspects of a system before complete implementation of the system.

### 5.3.2 Changes to Calendar System

In a lab experiment it is not feasible to use hundreds of subjects. In this case I would make an experiment controlled for the types of errors that happen in the system. A dialog participant will communicate with a calendar system described in the chapter 3. The WOZ module will be plugged into the system such that if a system misrecognizes time mentioned by a user a wizard can override incorrect speech recognition result with the correct one. Also, the wizard will have power to “fake” the misunderstanding of time specified by a user.

### 5.3.3 Experimental Participants

Participants would receive instructions to create/remove/change events. The instructions will be as general as possible not to bias the participant to use a particular type of named entity.

I estimate that for each condition we would need approximately 10 subjects: 60 subjects total.

## 6 Responsive Adaptation: Experiment on Speech Recognition Component

*Responsive adaptation* happens when a system changes its behavior to accommodate a particular user or to adjust to a dialog state. Responsive adaptation can be handled by any of the modules of a dialog system: a dialog manager, a language generation, or speech recognition/understanding modules. The following are some of the examples of responsive adaptation. [Komatani et al.2003] describes a system that adjusts the type and amount

of help provided to a user depending on the user’s skill level. This is an example of *responsive adaptation* in dialog manager. In [Walker et al.2004]’s system response generation module incorporates user’s linguistic preferences. Using state-specific language models [Raux et al.2005] or acoustic models in a speech recognizer is another form of *responsive adaptation*. In this chapter I describes two experiments where speech recognition component is involved in *responsive adaptation* of the system.

## 6.1 Performed Experiment: Adapt Language Model to the Question Topic in a Question Answering System

We have successfully improved speech recognition in an open-domain Question Answering application by adapting the language model to the target of the question, or the type of the expected answer. In this experiment we allow the user to provide a target named entity before asking the question. Then we build a named entity specific language model using the documents containing the named entity. The question-specific model is obtained by merging the named entity specific model with the model built on a set of questions. We present a set of experiments using the TREC question set on the AQUAINT corpus. This experiment was done in collaboration with Gokhan Tur and Dilek Tur at SRI [Stoyanchev(Stenchikova) et al.2008].<sup>29</sup>

### 6.1.1 Problem Description

Question answering (QA) is the task of automatic retrieval of an answer given a question. Typically the question is linguistically processed, and search phrases are automatically extracted. The search phrases are then

---

<sup>29</sup>This section is partially taken from [Stoyanchev(Stenchikova) et al.2008]

used to extract the candidate documents and sentences for the answer.

Question answering provides a natural language interface for information retrieval. This interface also opens the possibility of access to the system using voice. User of a spoken question answering system may be a reporter on a job who needs to check a fact, a driver on the go, a researcher in the field, or a visually disabled person. Spoken question answering can be seen as a more sophisticated version of spoken information access systems such as phone-based directory assistance [Kellner et al.1998, among others] or weather/restaurant/flight/hotel information systems [Zue et al.2000, among others].

We address speech recognition performance for the spoken question answering task. The word error rates of the state-of-the-art open-domain speech recognition technology are around 25%-30% [Riccardi and Hakkani-Tür2003]. Performance is known to be even lower for names and rare words. If a question is asked about a person, an organization, or another named entity, the recognition of this named entity is essential for finding a correct answer.

We propose and evaluate a method for improving speech recognition of the question by allowing interaction during the question specification phase. The interactivity allows the system to dynamically change the language models based on the dialog state.

We design a voice interface to an open-domain interactive question answering system. We show that the interactivity feature improves speech recognition performance of the open-domain system. In an interactive system a user may first specify the named entity of interest: a person's name, an organization, and so on. A grammar for the named entities is created from a database of named entities existing in the target corpus. If this named entity is recognized, a model specific to the name is used by the speech rec-

ognizer. In this study we create models using matching documents from the dataset.

The main idea is that named entities are strongly associated with the content words. For example for the target name *Gordon Gekko*, one question used in TREC 2004 evaluations is *In what film is Gordon Gekko the main character?*, including non-function words related to the movie industry, such as *film* or *character*. Our goal is capturing these content words using the documents where this name appears frequently.

Note that in most question answering evaluations, such as TREC or GALE Distillation, the named entity in consideration is provided in an explicit way. For example in TREC, first the target named entity is given and then several questions are asked about the target. Similarly in the GALE Distillation task, the questions are organized in templates such as *Describe attacks in [LOCATION]* where the variable portion is the named entity. This is in parallel to our design of first getting the name in question.

### 6.1.2 Approach

We simulate the interactive system where the user first specifies a target named entity. The named entity concept is grounded: the user confirms that the named entity is recognized correctly. In the case of continuous misrecognition, a named entity may be spelled. This task has been widely studied in the framework of directory assistance systems [Kellner et al.1998, among others]. The idea is limiting the language model using the names in consideration and such systems perform with very high accuracy. A keypad aided spelling correction may be used as a back-off mechanism [Parthasarathy2004] where the user the phone keypad while spelling the name.

Original TREC question	How many times has <b>Limbaugh</b> been married?
Target NE	Rush Limbaugh
Modified with NE (set 3)	How many times has <b>Rush Limbaugh</b> been married?
Modified without NE (set 4)	How many times has <b>he</b> been married?

Table 25: Example of the question in the test set

Figure 6 shows the control flow of the simulated system. First, a user is asked to specify the target named entity. The recognition uses grammar generated from the AQUAINT Named Entity database [Lloyd et al.2005]. The system then asks the user to specify the question about the given named entity. Meanwhile, a question-specific language model is built.

The described experiments focus on the improvement in speech recognition of the question by employing interactivity. Once the target named entity is recognized by the system, a target-specific model is built. To this end, we employ a search engine to extract the documents matching the named entity in the target corpus and use these documents to build the name-specific language model. We hypothesize that these documents will be likely to contain the lexicon of the question resulting in a more relevant model for speech recognition. For example, a question *On what date did Michael Brown resign as head of FEMA*, the words *resign* and *FEMA* may have relatively low probability in a generic model, but higher probability in the top matching documents. The documents are extracted from the AQUAINT corpus indexed by the Lucene information retrieval engine [Luc]. We match the string pattern of the target named entity using the Lucene API.

While the name-specific language model,  $LM_{AperQ}$ , provides the context words, the questions from the earlier TREC evaluations provide the typical characteristics of questions, such as the Wh- words at the sentence initial po-

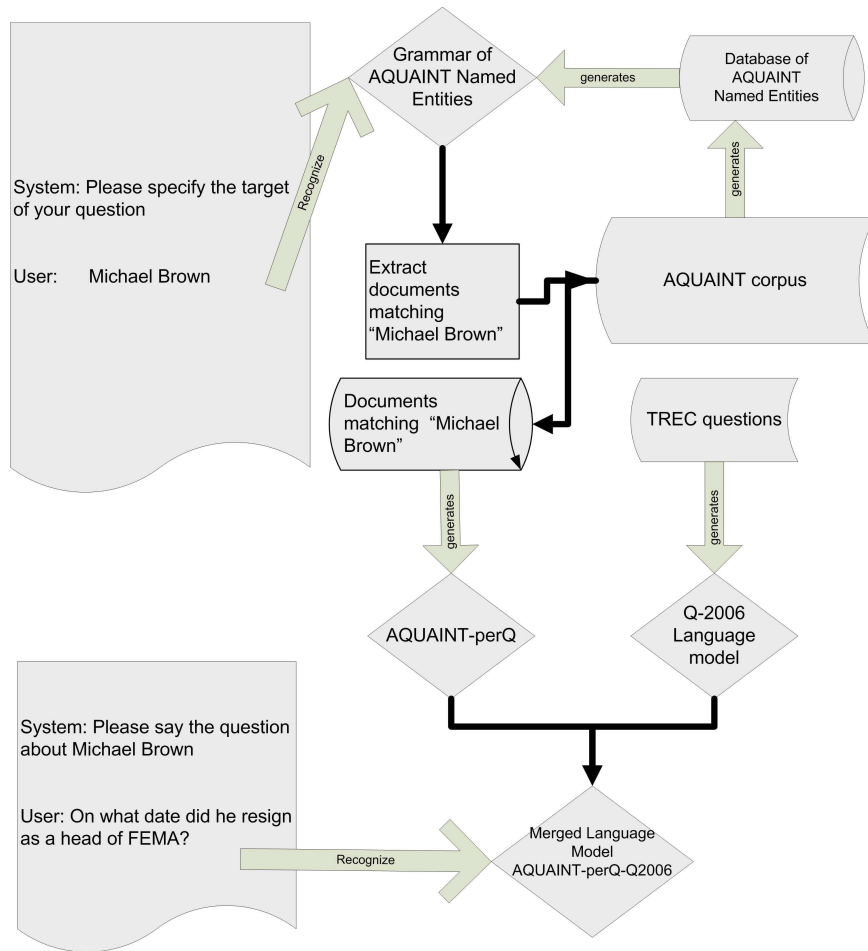


Figure 6: Dialog flow example.

Model	type	vocab size	description
Q-2007	general	5,337	TREC questions containing test set (total 4158 questions)
Q-2006	general	5,012	TREC questions <b>not</b> containing test set (total 3713 questions)
AQUAINT	general	3,000	all AQUAINT documents
AQUAINT-Q2006	general	6,344	all AQUAINT documents merged with the TREC questions
AQUAINT-perQ	per target name	7,211	up-to-100 top matches for the target of the question
AQUAINT-perQ-Q2006	per target name	10,210	up-to-100 top matches for the target of the question merged with the TREC questions

Table 26: Models used in the experiment

sition. So we train a separate model using these questions, named  $LM_{TREC}$ . These two models are then merged using linear interpolation. The interpolation weight,  $\lambda$ , is kept constant as optimized on a couple held-out spoken questions.

$$P_{LM}(W) = \lambda \times P_{LM_{AperQ}}(W) + (1 - \lambda) \times P_{LM_{TREC}}(W)$$

Note that this approach is only for using name-specific language models, and in all experiments we kept the acoustic model fixed. Using this approach while we have a better language model, its size is also smaller than the one obtained using the whole target corpus. This is very important for efficiency of a real-time recognizer.

### 6.1.3 Experiments and Results

In this study we use the TREC [Dang et al.2006] annual benchmark evaluation questions targeting the AQUAINT corpus consisting of 3 GB of written



Training/testing	num ne missed	40 unres set 3 avg	40 res set 4 avg	% err reduction set3	% err reduction set4
Q-2007	14.67	19.77	17.27		
Q-2006	35.67	45.65	32.12	35.3	19.0
AQUAINT	37.33	58.36	46.64	49.4	44.2
AQUAINT-Q2006	36	43.55	28.49	32.2	8.7
AQUAINT-perQ	14.67	42.51	42.59	30.5	38.9
<b>AQUAINT-perQ-Q2006</b>	14.67	<b>29.55</b>	<b>26.02</b>		

Table 27: Results averaged between the 3 speakers: number of named entities missed, error rate, relative error reduction for the AQUAINT-perQ-Q2006 model

news. The corpus is indexed using the Lucene [Luc] information retrieval engine.

We have selected 40 questions from the TREC 2007 evaluations. For 18 of the selected questions in the test set the target is a person, for 17 of the questions the target is an organization, and 5 of the questions have other type of target.

The questions are modified for the experiments. In *set3*, all questions are modified to contain the named entity. That is, if the original question contains a pronoun referring to the target, it is replaced with an appropriate form of the target. In *set4*, all questions are modified to *not* contain the named entity by replacing it with an appropriate pronoun. Table 25 describes how the questions are modified for the experiment. 40 questions with resolved and 40 questions with unresolved named entities are read and recorded by three subjects.

We compare using target-specific language models and a generic language model for the recognition of questions. All models in this experiment

are built using the the SRILM language modeling toolkit [Stolcke2002]. The speech recognition experiments are performed using SRI's Dynaspeak<sup>TM</sup> [Franco et al.2002] speech recognition system.

The models used in the experiment are summarized in Table 26. We report the average number of named entities missed from *set3*, the average word error rate over 40 questions, and the relative error rate reduction for the test model AQUAINT-perQ-Q2006 from each other model in Table 27. The TREC-2007 model is a “cheating model” that contains the questions used in the experiment. This model expectedly achieves the lowest error rate of 19.77% on *set3* (with named entities) and 17.27% on *set4* (without named entities).

The TREC-2006 model is the first baseline model built from 3713 TREC questions not containing the test set. This model has a relatively high error rate of 45.65% on the *set3* and 32.13% on the *set4*. Notice that although *set4* does not contain the target named entities, its error rate on the TREC-2006 model is relatively 45.9% higher than on the TREC-2007 “cheating” set. This shows the importance of the content words associated with the target names.

The second baseline model is built using the 3 GB AQUAINT corpus, pruning the vocabulary to 3000 words (guided by a system constraint). The AQUAINT model has the highest error rate of 58.36% on *set3* and 46.64% on *set4*. Although the AQUAINT corpus has large vocabulary coverage, the form of the questions differs from the form of the sentences in the corpus (such as sentence starting with Wh- words). We merge the AQUAINT model with the model trained with only questions, reducing the error rate by 25.4% on *set3* and 38.9% relatively on *set4*. The higher error rate reduction on the *set4* shows that the recognition improvement is not due to the better

recognition of named entities.

Next, we create a per-question model. The AQUAINT corpus is indexed with Lucene and queried using Lucene search API to extract as many as 100 documents matching the target named entity. These documents are used to build a question-specific language model AQUAINT-perQ. This model achieves 42.51% on *set3* and 42.59% on *set4*. Our final model AQUAINT-perQ-Q2006 is a merger of the AQUAINT model with the Q-2006 model. This model achieves the lowest WER among all tested models (except the “cheating” Q-2007 model) of 32.4% on *set3* and 28.65% on *set4*. This is a relative reduction of 32.2% on *set3* compared to the best generic model performance.

Note that, in addition to the dramatic reduction in word error rate, the ratio of missed named entity recognitions are halved coming down to levels which can be obtained using the cheating experiment.

We would like to point out the difference in speech recognition results between the speakers. Speakers 1 and 2 have higher WER on *set3* than on *set4* for all the models; however, speaker 4 achieves higher WER on *set4* for the AQUAINT-perQ model. It is possible that speaker 4 was very clear in pronouncing the target named entities and was able to achieve lower WER on the models that contain target named entities. We would like to perform this experiment on a larger subject pool to study the variability between the subjects.

#### **6.1.4 Future Work**

Future work includes the grounding process of the named entity. It may involve asking a user for the type of the named entity (e.g. person, organization, location, movie) and associations and building a focused grammar.

For example, a user may be asking a question about *Orhan Pamuk*, a Turkish writer. If the user specifies that the target is a person, a writer, and of Turkish descent, a more focused grammar may be built that would allow the recognition of the named entity.

## 6.2 Proposed Experiment: Adapt ASR Language Model to the Correction State

A very basic way in which a dialog system can responsively adapt is to notice and adapt to whether the dialog is in a 'normal' or 'misrecognition' state. This distinction is important, because as previous work has shown [P. Jordan1996, J. Shin2002, D. Bohus2005], user behavior and the very structure of the dialog change in 'misrecognition' states. There has been considerable work on identifying and characterizing misrecognitions in human-computer dialog [Young1994, Lemon and Gruenstein2004, Walker et al.2000b, Eskin2000, Hakkani-Tür et al.2005]. Some of the most interesting recent work is that by [Litman et al.2006]. I will use their characterization of misrecognitions. In this chapter I describe a planned experiment with the *Let's Go* system to examine the effect of ASR adaptation in response to identification of different types of misrecognition.

### 6.2.1 Motivation

Speech recognition remains the largest cause of misunderstandings in human-computer dialog systems. The language model of automatic speech recognition (ASR) component of a dialog system may be statistical or grammar-based. In a statistical ASR, an n-gram language model is trained from a large corpus of utterances. In a grammar-based ASR, a dialog designer

specifies a (usually domain-specific) recognition grammar by hand. Generally grammar-based ASR is more robust than statistical ASR on in-grammar utterances; however, grammar-based ASR fails more often and more comprehensively on out-of-grammar utterances. Larger grammars increase coverage but decrease recognition performance on in-grammar utterances. The performance of a statistical ASR also depends on the size of the model (the value of  $n$  in  $n$ -grams, or amount of context used), the amount of data used for training, and the goodness of fit of the training data to the domain of the dialog system. Many industrial dialog systems use grammar-based ASR because they prefer robustness in performance to coverage. Also, the design of a grammar-based ASR is more straightforward and does not require initial training data. By contrast, most research spoken dialog systems use statistical ASR as it allows researchers to experiment with larger vocabularies.

A common practice by system developers is to bootstrap a language model for a system by starting with a smaller, possibly hand-generated model, collecting data and generating a larger model from this data. Adaptation of the statistical language model to the dialog state is a common practice in many dialog systems [Riccardi and Gorin2000, Xu and Rudnicky2000]. Dialog state in this context means the state where a system asks a user for a particular information, for example the *Let's Go* system has five states:

1. *first\_query*: the system expects initial question of the user
2. *yes-no*: the system expects a yes/no answer
3. *place*: the system expects a location
4. *time*: the system expects an input of time

5. *next\_query*: the system expects a general question

I hypothesize that adaptation of the statistical language model to a particular type of utterance in a statistical ASR can help maximize the performance of a system's ASR component. I will evaluate the system performance first by running utterances from past dialogs through with the new language models and then in the case of improvement I will plug the new models into the live *Let's Go* system and evaluate overall speech recognition.

I classify errors in a dialog as misrecognitions and non-recognitions as in [Bohus et al.2006]. Errors in a human-computer dialog are inevitable and system performance can be judged by how efficiently the system recovers from errors. An initial study I conducted on the Communicator corpus shows that 39% of system rejections (*non-understandings*) are followed by another system rejection. This shows that the Communicator systems on average are not very good at recovering from errors. The initial analysis of the Communicator data, described in section 4, suggests that user paraphrases after rejections have a low chance of being recognized. In this experiment I will dynamically adapt the system's language model to the predicted type of the user's utterance: *correction* or *non-correction*. I hypothesize that this adaptation will improve ASR performance.

The rejection in this example may be caused either by a ASR rejection or by NLU rejection. The solution for the NLU rejections in a system using agent-based dialog manager architecture is a better design of a dialog manager which should handle the correction utterance. In this project I would like to address the cases when errors are caused by an ASR component.

System	User
Leaving Chicago on what date	Correct arrival city
OK, from Zurich to Denver. What date will you be traveling What date will you be traveling	Zurich to Kansas city

Table 28: Example of a non-recognition

### 6.2.2 Method

My hypothesis is that the vocabulary and grammar for correction utterances is different from that for non-correction utterances. So this experiment has two stages. In the first stage, I will build a model to classify user utterances as corrections or non-corrections. In the second stage, I will use this model’s classifications to adapt the statistical language model used in the ASR for the *Let’s Go* system. I will then evaluate the effect of this type of responsive adaptation on system performance.

**User utterance classification** I propose to apply the model for recognizing user corrections described in [Litman et al.2006]. Their model was trained on utterances collected using the TOOT dialog system in the domain of train information. Using prosodic, lexical, and dialog state features, the model classifies utterances as either corrections or non-corrections. I will use features similar to those used in [Litman et al.2006] to build a statistical model to classify user utterances from the *Let’s Go* corpus as corrections or non-corrections.

**Language model adaptation** The current version of the *Let’s Go* system switches between five ASR language models for different states of the dialog: *first\_query*, *yes-no*, *place*, *time*, and *next\_query*. For each of these five dialog states I will train two language models: one for correction utter-

ances and another for non-corrections. For the compatibility of comparison I will use the same dataset as the one used for training currently: the first two months of the *Let's Go* dialog data. I will classify the user's utterance as correction or non-correction and use the output of the appropriate ASR. To evaluate the result, I will compare speech recognition result with and without classification. The result without classification should correspond to ASR of the current system.

I will try several methods for the classification of utterances. As a baseline experiment I will split users' utterances into two classes using unsupervised clustering technique based on lexical and duration features. In the next approach I will use clustering including prosodic features. In the third method I will use classification learned from boosting technique reported in [Litman et al.2006]. The classification results on TOOT system in train information domain report F-measure in classifying corrections/non-corrections of .72/.89 . As my experiment ports the method to a new domain, we may need to adjust parameters of the classification model (rules learned by boosting).

**Evaluation of ASR performance** First, I will evaluate ASR performance offline. I will run the recognizer in "batch" mode of Let's Go data. I will classify each utterance as a correction or a non-correction using my classifier, and process it using the corresponding statistical language model. As a baseline, I will use the basic dialog state language models. This classification will be done separately for each of the five current states.

If the results of my offline experiment are promising, I will try it in the live *Let's Go* system. This will require folding my classifiers and language models into the system.



A new classification module for classifying user utterances as correction and non-correction will be required. It may be implemented as an additional Olympus server that receives input of lexical and prosodic features from the utterance and outputs a classification. Depending on which classifier performs the best, a prosodic feature extractor may be required to be added to the system.

The system will need to run two ASR modules in parallel and depending on the classification of the utterance a speech recognition result from appropriate ASR module should be used.

## 7 Schedule

I plan to defend my thesis at in May 2009. I have performed most of the corpus studies (studies on dialog data sets) for my dissertation, and taken a leadership role in the implementation of three spoken language interfaces: The Stony Brook Rate-A-Course System, the Stony Brook RavenCalendar System, and the Stony Brook Question Answering System. These systems will be the testbeds for the user studies I need to do to complete my dissertation. I have already completed two of my system experiments, one with the Rate-A-Course System and one with the Question Answering system. I have also formed a valuable research partnership with Carnegie Mellon University. A schedule for my experiments is outlined in table 29.

In February - March 2008 I will perform a corpus study and the responsive adaptation in ASR experiment on the *Let's Go* corpus (described in sections 4.1.2 and

In Spring 2008, I will also distribute the RavenCalendar dialog system to initial test users. I will insure that the system is fully functional and that

Time	Task1	Task2
Feb-March 08	Corpus study for adaptation in the forms of concepts and ASR experiment on Let's Go corpus	Insure that Calendar is ready for the experiments
April-May 08	Collaborate with CMU to set up prompt variation experiment on the Let's Go system	Design a long-term adaptation experiment for the Calendar system (optional experiment, will work on it if there is extra time)
June-Aug 08	Run prompt variation experiment on Let's Go	
Sep-Dec 08	Analyze the results of the prompt variation experiment on Let's Go	Collaborate with CMU to incorporate ASR adaptation to correction state (if the the results on data batch tests were successful)
2009	Thesis writing	

Table 29: Schedule of work

I record all of the required data.

I plan to collect the data from the directive adaptation experiment on Let's Go over Summer 2008.

In the Fall 2008, I will analyze the results of the prompt variation experiment. If the results from responsive adaptation experiment show a sufficient improvement in speech recognition, I will collaborate with CMU to implement the changes into the live Let's Go system for the ASR improvement.

In Spring 2009 I will focus on writing my thesis and present it in May 2009.

During my work I plan to submit the following publications:

- Exploring Directive Adaptation in Spoken Dialog Systems (Submitted to Student Workshop at ACL 2008)
- Analysis of rejections in Communicator (to submit for SIGDIAL 2008, March 14)
- Adaptation in the rate-a-course system experiment (to submit a short paper to HLT 2008, March 14)
- Responsive adaptation ASR experiments on Let's Go Corpus (to submit to GOTAL, April 4)
- Directive adaptation using prompt variation experiment on Let's Go (or Calendar)

## References

[Bechet et al.2004] F. Bechet, G. Riccardi, and D. Hakkani-Tür. 2004. Mining spoken dialog corpora for system evaluation and modeling. In *Proceedings of EMNLP*, Barcelona, Spain, July.

- [Bell et al.2000] L. Bell, J. Boye, J. Gustafson, and M. Wiren. 2000. Modality convergence in a multimodal dialogue system. In *Proceedings of GO-TALOG*.
- [BeVocal] BeVocal. BeVocal Cafe. <http://cafe.bevocal.com>.
- [Bohus and Rudnicky2002] D. Bohus and A. Rudnicky. 2002. LARRI: A language-based maintenance and repair assistant. In *Proceedings of IDS*.
- [Bohus and Rudnicky2003] D. Bohus and A. Rudnicky. 2003. Ravenclaw: Dialog management using hierarchical task decomposition and an expectation agenda. In *Proceedings of Eurospeech*.
- [Bohus et al.2006] D. Bohus, B. Langner, and M. Eskenazi and A. Rudnicky A. Rayx, and A. Black. 2006. Online supervised learning of non-understanding recovery policies. In *SLT*.
- [Brennan and Clark1996] S. Brennan and H. Clark. 1996. Conceptual pacts and lexical choice in conversation. *Journal of Experimental Psychology*, 22(6):1482–1493.
- [Brennan et al.2004] S. Brennan, M. Hoffman, and A. Stent. 2004. Adaptive spoken dialog with human and computer partners.
- [Brennan1990] S. Brennan. 1990. Conversation as direct manipulation: An iconoclastic view. *The Art of Human-Computer Interface Design*. for thesis, psych,.
- [Brennan1996] S. Brennan. 1996. Lexical entrainment in spontaneous dialog. In *Proceedings of ISSD*, pages 41–44.

- [Brennan1998] S. Brennan. 1998. The vocabulary problem in spoken dialogue systems. *MIT press*. for thesis, psych,.
- [Brown and Dell1987] P. Brown and G. Dell. 1987. Adapting production to comprehension: The explicit mention of instruments. *Cognitive Psychology*, 19:441–472.
- [Chartrand and Bargh1999] T. Chartrand and J. Bargh. 1999. The chameleon effect: The perception-behavior link and social interaction. *Journal of Personality and Social Psychology*, 76:893–910.
- [Church2000] K. Church. 2000. Empirical estimates of adaptation: the chance of two noriegas is closer to  $p/2$  than  $p^2$ . In *Proceedings of the ACL*.
- [Constantinides and others1998] P. Constantinides et al. 1998. A schema based approach to dialog control. In *Proceedings of ICSLP*.
- [D. Bohus2005] A. Rudnicky D. Bohus. 2005. Sorry, i didn't catch that! - an investigation of non-understanding errors and recovery strategies. In *SIGdial*. non-understandings.
- [Dang et al.2006] H. T. Dang, J. Lin, and D. Kelly. 2006. Overview of the TREC 2006 question answering track. In *NIST Special Publication: SP 500-272*.
- [Dubey et al.2006a] A. Dubey, P. Sturt, and F. Keller. 2006a. Parallelism in coordination as an instance of syntactic priming: evidence from corpus-based modeling. In *Proceedings of EMNLP*.

- [Dubey et al.2006b] A. Dubey, P. Sturt, and F. Keller. 2006b. Parallelism in coordination as an instance of syntactic priming: evidence from corpus-based modeling. In *Proceedings of EMNLP*.
- [Eskin2000] E. Eskin. 2000. Detecting errors within a corpus using anomaly detection. In *Proceedings of the NAACL*, Seattle, WA, April.
- [Franco et al.2002] Horacio Franco, J. Zheng, J. Butzberger, F. Cesari, M. Frandse, J. Arnold, V. Ramana, R. Gadde, A. Stolcke, and V. Abrash. 2002. Dynaspeak: Sri’s scalable speech recognizer for embedded and mobile systems. In *Proceedings of HLT*, San Diego, CA.
- [Fukubayashi et al.2006] Y. Fukubayashi, K. Komatani, T. Ogata, and H. Okuno. 2006. Dynamic help generation by estimating users mental model in spoken dialogue systems. adaptation.
- [Furnas et al.1987] G. Furnas, T. Landauer, L. Gomez, and S. Dumais. 1987. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971.
- [G. Riccardi2000] A.L. Gorin G. Riccardi. 2000. Stochastic language adaptation over time and state in natural spoken dialog systems. *IEEE*, 8(1):3–10. for thesis NLU.
- [Garner1995] S. Garner. 1995. Weka: The waikato environment for knowledge analysis.
- [Garrod and Anderson1987] S. Garrod and A. Anderson. 1987. Saying what you mean in dialogue: A study of conceptual and semantic coordination. *Cognition*. for thesis, psych,.

- [Garrod and Doherty1994] S. Garrod and G. Doherty. 1994. Conversation, coordination and convention: An empirical investigation of how groups establish linguistic conventions. *Cognition*, 53:181–215.
- [Gieselmann and Stenneken2006] P. Gieselmann and P. Stenneken. 2006. Communication with robots: Evidence from a web-based experiment on human-computer interaction. In *SLT*.
- [Gildea and Hofmann1999] D. Gildea and T. Hofmann. 1999. Topic-based language models using em. In *Proceedings of Eurospeech*, Budapest, Hungary, September.
- [Godfrey et al.1992] J.J. Godfrey, E.C. Holliman, and J. McDaniel. 1992. Switchboard: telephone speech corpus for research and development. In *Proceedings of ICASSP*, San Francisco, CA.
- [Gorrell et al.2002] G. Gorrell, I. Lewin, and M. Rayner. 2002. Adding intelligent help to mixed initiative spoken dialogue systems. In *ICSLP*. user modelling, adaptive system.
- [Gruenstein et al.2006] A. Gruenstein, S. Seneff, and C. Wang. 2006. Scalable and portable web-based multimodal dialogue interaction with geographical databases. In *Interspeech*. TOREAD.
- [Gustafson et al.1997] J. Gustafson, A. Larsson, R. Carlson, and K. Hellman. 1997. How do system questions influence lexical choices in user answers? In *Proceedings of Eurospeech*.
- [H. Branigan2004] A. Cleland H. Branigan, M. Pickering. 2004. Syntactic coordination in dialogue. *Cognition*, 75:B13–B25. Psych Syntax TOREAD.

- [Hakkani-Tür et al.2005] D. Hakkani-Tür, G. Tur, G. Riccardi, and H. K. Kim. 2005. Error detection in spoken dialog: From signal-to-noise ratio to semantic confidence scores. In *Proceedings of the ICASSP*, Philadelphia, PA, March.
- [Heidorn1978] G. Heidorn. 1978. Natural language dialogue for managing an on-line calendar. In *Proceedings of ACM/CSC-ER*.
- [Hockey et al.2003] B. Hockey, O. Lemon, E. Campana, L. Hiatt, G. Aist, J. Hieronymus, A. Gruenstein, and J. Dowding. 2003. Targeted help for spoken dialogue systems: intelligent feedback improves naive users performance. In *EACL*. for thesis.
- [Horton and Gerrig2002] W. Horton and R. Gerrig. 2002. Speakers' experiences and audience design: knowing when and knowing how to adjust utterances to addressees. *Journal of Memory and Language*, 47:589–606.
- [Horvitz and Paek2000] E. Horvitz and T. Paek. 2000. DeepListener: Harnessing expected utility to guide clarification dialog in spoken language systems. In *Proceedings of ICSLP*.
- [Huang and others2001] X. Huang et al. 2001. MIPAD: A next generation PDA prototype. In *Proceedings of ICSLP*.
- [Huang1991] X. D. Huang. 1991. A study on speaker-adaptive speech recognition. In *HLT*. for thesis.
- [Iyer and Ostendorf1999] R. Iyer and M. Ostendorf. 1999. Modeling long distance dependencies in language: Topic mixtures versus dynamic cache model. *IEEE Transactions on Speech and Audio Processing*, 7(1):30–39.



- [J. Shin2002] L. Gerber A. Kzetzadeh D. Byrd J. Shin, S. Narayanan. 2002. Analysis of user behavior under error conditions in spoken dialogs. In *ICSLP*. Error Corpus study.
- [Kellner et al.1998] A. Kellner, B. Rueber, and H. Schramm. 1998. Strategies for name recognition in automatic directory assistance systems. In *Proceedings of IEEE IVTTA*, Torino, Italy.
- [Komatani et al.2003] K. Komatani, F. Adachi, S. Ueno, T. Kawahara, and H. Okuno. 2003. Flexible spoken dialogue system based on user models and dynamic generation of voicexml scripts. In *SIGdial*. generation, user modeling.
- [Komatani et al.2007] K. Komatani, Y. Fukubayashi, T. Ogata, and H. Okuno. 2007. Introducing utterance verification in spoken dialogue system to improve dynamic help generation for novice users. In *SigDIAL*.
- [Komatani2005] K. Komatani. 2005. User modeling in spoken dialog systems to generate flexible guidelines. generation.
- [Kraljic and Brennan2005] T. Kraljic and S. Brennan. 2005. Prosodic disambiguation of syntactic structure: For the speaker or for the addressee? *Cognitive Psychology*, 50:194–231.
- [Kraut et al.2003] R. Kraut, S. Fussell, and J. Siegel. 2003. Visual information as a conversational resource in collaborative physical tasks. *Human-Computer Interaction*, 18(1–2).
- [Lemon and Gruenstein2004] O. Lemon and A. Gruenstein. 2004. Multi-threaded context sensitive speech recognition and interpretation of cor-

rective fragments. *ACM Transactions on Computer-Human Interaction*.  
TOREAD for thesis or for ASR-DM.

[Litman et al.2006] D. Litman, J.Hirschberg, and M. Swerts. 2006. Characterizing and predicting corrections in spoken dialogue systems. *Computational Linguistics*, 32:417–438.

[Lloyd et al.2005] Levon Lloyd, Dimitrios Kechagias, and Steven Skiena. 2005. Lydia: A system for large-scale news analysis. In *SPIRE*, pages 161–166.

[Lockridge and Brennan2002] C. Lockridge and S. Brennan. 2002. Addressees’ needs influence speakers’ early syntactic choices. *Psychonomics Bulletin and Review*. Psych.

[Luc] Lucene search engine. <http://lucene.apache.org/>.

[Merriam-Webster] Merriam-Webster. <http://www.m-w.com/dictionary/adapt>.

[Oviatt1995] S. Oviatt. 1995. Predicting and managing spoken disfluencies during human-computer interaction. *Computer Speech and Language*, 9:19–35.

[P. Jordan1996] R. Thomason P. Jordan. 1996. Refining the categories of miscommunications. In *AAAI Workshop on Detecting, Repairing, and Preventing Human-machine Miscommunication*, page 8. Dialog mis (1 citation) psycholing.

[Parthasarathy2004] S. Parthasarathy. 2004. Experiments in keypad-aided spelling recognition. In *Proceedings of ICASSP*, Montreal.

- [Pickering and Garrod2004] M. Pickering and S. Garrod. 2004. Toward a mechanistic psychology of dialog. *Behavioral and Brain Sciences*, 27:169–190. Psych.
- [Pickering et al.2000] M. Pickering, H. Branigan, A. Cleland, and A. Stewart. 2000. Activation of syntactic priming during language production. *Journal of Psycholinguistic Research*, 29(2):205–216.
- [Purver and Kempson2004] M. Purver and R. Kempson. 2004. Context-based incremental generation for dialogue. In *INLG*. generation.
- [Rabiner and Juang1986] L. R. Rabiner and B. H. Juang. 1986. An introduction to hidden Markov models. *IEEE ASSP Magazine*, pages 4–15, January.
- [Raux et al.2005] A. Raux, B. Langner, A. Black, and M Eskenazi. 2005. Let’s go public! taking a spoken dialog system to the real world. In *Proceedings of Eurospeech*.
- [Reitter and Moore2007] E. Reitter and J. Moore. 2007. Predicting success in dialogue. In *ACL*. dialog success.
- [Reitter et al.2006a] E. Reitter, F. Keller, and J. Moore. 2006a. Computational modelling of structural priming in dialogue. In *HLT/NAACL*. syntactic priming.
- [Reitter et al.2006b] E. Reitter, J. Moore, and F. Keller. 2006b. Priming of syntactic rules in task-oriented dialogue and spontaneous conversation. In *CogSci*. syntactic priming.

- [Riccardi and Bangalore1998] G. Riccardi and S. Bangalore. 1998. Automatic acquisition of phrase grammars for stochastic language modeling. In ? for thesis.
- [Riccardi and Gorin2000] G. Riccardi and A.L. Gorin. 2000. Stochastic language adaptation over time and state in natural spoken dialog systems. *IEEE*, 8(1):3–10. for thesis NLU.
- [Riccardi and Hakkani-Tür2003] G. Riccardi and D. Hakkani-Tür. 2003. Active and unsupervised learning for automatic speech recognition. In *Proceedings of the EUROSPEECH*, Geneva, Switzerland, September.
- [Schapire and Singer1999] Robert E. Schapire and Yoram Singer. 1999. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336.
- [Shi and Chang2004] Y. Shi and E. Chang. 2004. Studies in massively speaker-specific speech recognition. In *ICASSP*. for thesis.
- [Skantze2002] G. Skantze. 2002. Coordination of referring expressions in multimodal human-computer dialogue. In *Proceedings of ICSLP*.
- [Stenchikova and Stent2007] S. Stenchikova and A. Stent. 2007. Measuring adaptation between dialogs. In *SIGDIAL*. for thesis.
- [Stenchikova et al.2007] S. Stenchikova, B. Mucha, S. Hoffman, and A. Stent. 2007. Ravenscalendar: A multimodal dialog system for managing a personal calendar. In *HLT*. for thesis.
- [Stenchikova2007] S. Stenchikova. 2007. <http://www.ewh.ieee.org/soc/sps/stc/News/NL0711/NL0711-411.htm>.

- [Stent et al.2004] A. Stent, R. Prasad, and M. Walker. 2004. Trainable sentence planning for complex information presentations in spoken dialog systems. In *ACL*. generation.
- [Stent et al.2006] A. Stent, S. Stenchikova, and M. Marge. 2006. Dialog systems for surveys: The rate-a-course system. In *SLT*. for thesis.
- [Stolcke2002] A. Stolcke. 2002. SRILM – An Extensible Language Modeling Toolkit. In *Proceedings of the ICSLP*, Denver, CO, September.
- [Stoyanchev(Stenchikova) et al.2008] S. Stoyanchev(Stenchikova), D. Hakkani-Tur, and G. Tur. 2008. Name-aware speech recognition for interactive question answering. In *ICASSP*.
- [Tashev et al.2007] I. Tashev, M. Seltzer, Y. Ju, D. Yu, and A. Acero. 2007. Commute ux: Telephone dialog system for location-based services. In *SigDial*. dialog.
- [Thompson et al.2004] C. Thompson, M. Goker, and P. Langley. 2004. A personalized system for conversational recommendations. *Artificial Intelligence*. user modelling, adaptive system.
- [Tomko and Rosenfeld2004] S. Tomko and R. Rosenfeld. 2004. Speech graffiti vs. natural language: Assessing the user experience.
- [Tomko and Rosenfeld2006] S. Tomko and R. Rosenfeld. 2006. Shaping user input in speech graffiti: a first pass. In *CHI*. TOREAD.
- [Vo and Wood1996] M. Tue Vo and C. Wood. 1996. Building an application framework for speech and pen input integration in multimodal learning interfaces. In *Proceedings of ICASSP*.

- [Walker and Passonneau2001] M. Walker and R. Passonneau. 2001. Date: A dialogue act tagging scheme for evaluation of spoken dialogue systems.
- [Walker et al.2000a] M. Walker, J. Aberdeen, J. Boland, E. Bratt, J. Garofolo, L. Hirschman, A. Le, and S. Narayanan S. Lee, K. Papineni, B. Pellom, J. Polifroni, A. Potamianos, P. Prabhu, A. Rudnicky, and G. Sanders et al. 2000?a. Darpa communicator dialog travel planning systems: The june 2000 data collection. ?
- [Walker et al.2000b] Marilyn Walker, Jerry Wright, and Irene Langkilde. 2000b. Using natural language processing and discourse features to identify understanding errors in a spoken dialogue system. In *Proc. 17th International Conf. on Machine Learning*, pages 1111–1118. Morgan Kaufmann, San Francisco, CA.
- [Walker et al.2004] M. Walker, S. Whittaker, A. Stent, P. Maloor, J. Moore, M. Johnston, and G. Vasireddy. 2004. Generation and evaluation of user tailored responses in multimodal dialogue. *Cognitive Science*, 28. user modelling in generation.
- [Ward and Issar1994] W. Ward and S. Issar. 1994. Recent improvements in the cmu spoken language understanding system,. In *HLT*. for thesis.
- [W.Walker and et al.2004] W.Walker and et al. 2004. Sphinx-4: A flexible open source framework for speech recognition. Technical report, IBM.
- [Xu and Rudnicky2000] W. Xu and A. Rudnicky. 2000. Language modeling for dialog system. In *ICSLP*. Read in October 2007.

- [Yankelovich1994] N. Yankelovich. 1994. Talking vs taking: Speech access to remote computers. In *Proceedings of the Conference on Human Factors in Computing Systems*.
- [Young1994] S. Young. 1994. Detecting misrecognitions and out-of-vocabulary words. *IEEE*, ii:II/21–II/24. for thesis ASR NLU.
- [Yu et al.2000] Hua Yu, Takashi Tomokiyo, Zhirong Wang, and Alex Waibel. 2000. New developments in automatic meeting transcription. In *ICSLP*. for thesis OOV ASR.
- [Zue et al.2000] V. Zue, S. Seneff, J. R. Glass, J. Polifroni, C. Pao, T. J. Hazen, and L. Hetherington. 2000. Juplter: a telephone-based conversational interface for weatherinformation. *IEEE Transactions on Speech and Audio Processing*, 8(1):85–96.