Knowledge Discovery Through Spoken Dialog

Aasish Pappu

CMU-LTI-14-004

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Language and Information Technologies

Abstract

People can acquire knowledge not only from media such as books, but also through interactions with other people. Automated dialog agents, however, typically limit their learning from labeled data, and programming.

This thesis describes an approach that enables such agents to actively acquire new knowledge through spoken dialog interaction. To acquire knowledge in different situations, we propose several key dialog-driven strategies that include user-initiated, system-detected and system-initiated processes. Using these techniques, an agent can acquire domain-specific knowledge both from the public domain (through open domain knowledge bases) and in the informal space (through human users). Our approach incorporates two sets of techniques:

First, we design techniques that allow a spoken dialog agent to detect when an interaction contains unknown entities. These unknowns can manifest as unseen words/phrases, or unseen references to known entities. We use various spoken dialog features to detect and handle unknown entities. We then use open-domain knowledge bases and appropriate semantic-relatedness measures to deal with unknown references to known entities.

Second, we show that our novel conversational strategies allow our agent to acquire additional information about detected unknowns, which can improve the execution of appropriate dialog tasks. These strategies are designed to support two primary goals: (i) Validate the genuineness of detected unknowns, (ii) Elicit relevant information about these unknowns. In this thesis we show that the proposed strategies are effective, and useful. We demonstrate their effectiveness through a system that solicits situational information to augment its knowledge base from its users in a domain that provides information on events. We find that this knowledge is consistent and useful and that it provides reliable information to users.

Together these techniques allow agents to use spoken dialog capabilities to acquire and extend their knowledge bases of language and world knowledge without the need for expert intervention.

Acknowledgements

I am indebted to Carnegie Mellon University for introducing me to a world of brilliant and humble researchers. I am fortunate to have had the opportunity to listen to their monologues and discover valuable knowledge over these years. Dialogs with some of them have had even more significant impact in my knowledge discoveries.

My thesis advisor, Alex Rudnicky, has introduced me to the world of conversational systems. His *informs* and *requests*, *queries* and *reqalts* have shaped the structure and state of this thesis. He has taught me how to conduct research and look at the bigger picture of a problem. He has given me the freedom to digress from research and allowed me to work on things that eventually made an impact on my research.

Alan Black has been a great source of inspiration for all these years. His enthusiasm for writing code and doing research is contagious. His riveting presentation style has taught me fundamentals on how to give a technical talk to a broad audience.

My thesis committee members, Emma Brunskill and Antoine Raux, have been quite supportive throughout my thesis work. Their insightful questions, and thorough feedback was invaluable. Particularly I would like to thank Antoine for his significant contributions to Olympus/Ravenclaw dialog framework, platform for this thesis. I am very grateful to my internship mentors at Honda Research Institute, Rakesh Gupta and Teruhisa Misu. Their advice and suggestions was extremely valuable for one of the core components of this thesis.

I am very grateful to Thomas K. Harris, Satanjeev Banerjee, David Huggins-Daines, and Long Qin for providing me an excellent launchpad during the initial years of my research. Alok Parlikar and Ming Sun were great colleagues with awesome sense of humor. They have always been my sounding board for new ideas. I was very fortunate to collaborate with

Seshadri Sridharan, Matthew Marge, Peng Li, Benjamin Frisch and Gopala Anumanchipalli on different projects through these years. It was a great learning experience working with you guys. I would like to thank my friends in and around LTI for their help: Prasanna Muthukumar, Sunayana Sitaram, Udhyakumar Nallasamy, Harshit Surana, Nathan Schneider, Sidharth Gopal, Zhou Yu, Justin Chiu, Vivian Chen, Troy Hua, Pallavi Baljekar, Bhavana Dalvi, and Meghana Kshirsagar. Special thanks to *every one* of you for being participants in one or more of my user studies. Your feedback and support has helped this thesis in several ways. Thank you.

Most of all, my parents: they let me go down the rabbit hole and explore this wonderland. Their constant love and support was my source of courage to face the jabberwocky. They inspired me to work harder and be curiouser. They gave me permission to take this road when I didn't know whether it could bring me here. Thank you for everything!

Contents

Chapter 1

Introduction

An interactive agent should aim for efficient interaction with a user and for effective completion of tasks. An interaction is efficient, when the agent and the user exchange their intentions with few or no misunderstandings (dialog-costs). An agent is effective, when it finishes a task with few or no failures (task success). Therefore, dialog agents are primarily evaluated in terms of their dialog-costs and task success (Walker et al., 1997). However, in real life, an agent is highly vulnerable to errors. The user's input goes through several layers of processing — from capturing the speech, converting the speech to text, parsing the text, and finally, understanding intention of the user. Errors cascade through this pipeline. Thus, it is difficult for an agent to achieve high efficiency and effectiveness. Previous work in spoken dialog research (Hazen et al., 2002; Williams and Young, 2004; Skantze, 2007a; Bohus, 2007; Raux, 2008) has focused on errors due to capturing the input speech, misrecognition of the speech, errors in spoken language understanding and non-topical / incomplete input from the user. Besides these issues, an agent can also suffer from insufficient domain knowledge to understand intention of the user. For example, the user needs to know how to get to a new or less-known landmark, but the agent does not have any information including the name of the landmark. Consider the dialog in Figure 1.1. The agent attempts to resolve this problem by refering to other places that are acoustically similar to the landmark's name, after few turns it picks up that user is refering to an indian restaurant. This results in a large list of candidate landmarks and finally the user gives up on their query. It not only leads to long and ineffective interaction, but also hurts effectiveness of the agent. In such situations, if the agent were able to learn the name of the landmark from the user, then it will learn more information about the landmark from alternative sources and provide information to the user in future. In this work, we focus on how an agent might fill the gaps in its knowledge to help improve its effectiveness and efficiency.

Unknown Information

U: Where is shiva's restaurant located?
S: Sorry. Did you mean chivas?
U: No. shivas
S: Did you mean kiva han cafe?
U: shivas indian restaurant
S: Did you mean indian restaurant?
U: Yes
S: There are 56 indian restaurants in the database? Which one.
U: [Hangup]

Figure 1.1: Fictitious Dialog in unknown information scenario

In general, incomplete knowledge is a common problem for artificial agents — both for the interactive and for the non-interactive ones. For example, in a new environment, a mobile robot cannot navigate to a place without an annotated map of the environment. Creating an annotated map is laborious and time-consuming task for people. On the other hand, a robot, with basic sensory mechanism and interactive skills, can ask and learn about the places in the environment. This will not only reduce the annotation effort, but also allows the agent to quickly adapt to new environments. Similarly, a product recommendation agent may interact (implicitly through user behavior/explicitly through surveys) with its users to learn what they like and dislike to help make better recommendations. Whatever the domain may be, knowledge acquisition is essential for the agents to accomplish their tasks.

New knowledge can manifest itself in different forms e.g., as unknown words, as unknown facts, as unknown topics, as novel/indirect references to existing items in the knowledge etc. Detecting the unknown words in speech is popularly known as out-of-vocabulary (OOV) detection. Detecting indirect references to stored knowledge can be understood as a concept detection problem. Many researchers have developed algorithms to detect the OOV words (Schaaf, 2001; Chung et al., 2003; Qin and Rudnicky, 2012) and the topics/concepts (Cristianini et al., 2002; Blei et al., 2003; Carlson et al., 2010) from natural language. Most of these algorithms were developed in standalone or batch setting (as opposed to interactive setting). For a spoken dialog agent, detecting the OOV words and the topics/concepts is important to perform tasks. All processes in a dialog system pipeline, such as recognition, parsing and dialog management, depend on the words in an input. Therefore, it is critical to have a tighter integration of knowledge detection methods with an interactive system.

The problem with most of the spoken dialog frameworks (McTear, 1998; Bohus et al., 2007b; Young et al., 2010) is that they make no allowance for knowledge acquisition. A typical dialog agent is equipped with a static knowledge base — under an assumption that the agent knows everything that it needs to know about a domain to function properly. Although it is a reasonable assumption to make, it is not always true. (Bohus and Rudnicky, 2005c) found that a flight information dialog agent missed considerable portion of utterances due to novel words or phrases in those utterances. In such situations, an agent typically asks the person to rephrase or repeat the utterance. After a threshold, agent gives up on such utterances. Learning unknown information from these utterances can make the agent more efficient and effective than earlier.

Knowledge acquisition requires the agent to have conversational strategies. A conversational strategy determines what the agent needs to ask the user, how it should ask them, and when does it need to stop asking them. Different strategies need to be used in different acquisition scenarios. But existing conversational strategies are primarily targetted at helping a new user, providing context to the user and recovering from misrecognitions in user's speech. It is useful for an agent to have these strategies, but it is also important to have additional strategies to recover from inadequate knowledge.

1.1 Handling New Information

Spoken Dialog Systems are challenged by misunderstandings and non-understanding errors. Bohus and Rudnicky (2005c) found that some of the non-understanding errors are caused due to novel information in an such as, out-of-vocabulary words, out-of-grammar utterances. The novel information can manifest in different forms out-of-domain utterances or out-of-topic discourse. In dialog systems, the size of speech recognition vocabulary is limited. As a consequence of this limitation, a recognition engine can encounter out-of-vocabulary words. A spoken language parser can be challenged by phrases not covered in the domain grammar. Novice users who are not familiar with system's domain limitations, may request queries that might be part of the system's domain of operation. Off-topic discourse can occur when the dialog manager is not expecting a particular user input although it is a well-parsed utterance.

In most spoken dialog systems, non-understanding errors are either rejected or attempted to recover using certain dialog strategies. Bohus and Rudnicky (2005a) proposed a systematic rejection of non-understanding utterances by learning state-specific rejection thresholds. Filisko and Seneff (2006) developed a user-simulation driven approach to ignore non-understanding utterances and recover relatively easier ones by implicit or explicit confirmations and asking the user to spell the concept value. It is important to note that the rejected utterances may have novel information, if detected the system could learn this information.

1.2 Recovering and Discovering New Information

Acquiring new information depends on the nature of the information. According to Waxman and Booth (2000), infants learn words differently from the facts . This is against the popular argument that cognitive processes involved in learning words and facts are quite same (Markson and Bloom, 1997). However, Waxman and Booth (2000) found that children extend the usage of newly learned words but do not over-generalize the facts. Although, it is well studied that cognitive processes involved in memorizing words could be similar to memorizing phrases. Pinker (1984) theorizes that children memorize utterances i.e., string of words like auditory templates and later learn the meaning of words/phrases. The auditory templates are later segmented using stochastic techniques employing syllable-transition probabilities. A similar theory Peters (1983) claims that children employ different acquisition strategies to learn different pieces of information. From a dialog agent's perspective, it is necessary to employ different strategies to acquire words, meaning of the words and facts.

Typically, spoken dialog research (Lee et al., 2007; Filisko and Seneff, 2004b; Bohus and Rudnicky, 2005c) has focused on recovering from misunderstanding and non-understanding errors by asking the user to rephrase, repeat their utterance or suggesting system understandable prompts. Bohus and Rudnicky (2005c) has analyzed that 62% of the non-understanding errors and 72% of the misunderstanding errors are caused due to speech recognition. It is understandable that the previous research has focused on devising recovery strategies. Nevertheless, one should not overlook that 38% of the non-understanding errors are caused due to a system's inability to acquire new information presented in a user utterance. Attempts have been made to design dialog systems to acquire new information such as out-of-vocabulary (OOV) words. (Chung et al., 2003; Filisko and Seneff, 2005) have developed speak-spell strategy to acquire spellings of city names through interaction. The user is asked to say the city name and then spell or type the word through a dialog interface.

1.3 Learning New Information Through Interaction

Learning through interaction has been studied, but there was not much focus on spoken dialog strategies for learning. The popular spoken dialog strategy was speak-spell and variants of it to learn OOV words. Chung et al. (2003); Holzapfel et al. (2007) have looked at dialog systems that can detect OOV words using head-tail approach then ask the user to spell the word. Standard spelling language models were used for the spelling recognition.

Learning new words and their semantics has been the focus of early approaches in interactive learning. Haas and Hendrix (1980) has developed a system that not only learns new words and concepts, but also linguistic constructions used to express those concepts. This system has a first-order-logic KB, an initial set of concepts and words in the KB, an NLP parser based on pragmatic grammar, and a text-based dialog interface. This was one of the first attempts to acquire domain knowledge from non-expert users to improve a system. In their Nanoklaus system, Haas and Hendrix (1980) used an exploratory strategy to ask questions about a concept introduced by the user. This system is akin to popular twenty-question game where a player asks different questions to guess the concept or word. This strategy could exhaust a user, thus impractical for a longterm knowledge acquisition process. (Holzapfel et al., 2008) employed a similar approach to acquire category and properties of a physical object. Their system either asks the user to provide a one word/phrase description for a new object or browses through its knowledge base (KB) to ask yes/no questions about the object. The one-shot approach is efficient but not effective because the user description can be verbose and unfit for a category label. The browsing strategy can exhaust the interacting user, thus impractical.

Marcus and McDermott (1989) has proposed a knowledge-acquisition language to support an interface between the system and the domain experts. The system interrogates an expert about inconsistencies in the KB and help the expert revise the knowledge. Later, Witbrock et al. (2003) have developed a text-dialog based knowledge acquisition system that uses Cyc, a commonsense KB. This system asks the user about their favorite topics and then prepares an interaction agenda based on these topics. The agenda contains a wish-list knowledge items that system wants the user to amend. The user is allowed to add more items to this agenda. There is another line of work ((Kim and Gil, 2007)) that treated knowledge acquisition problem as analogous to a tutorial dialog system, with roles reversed. The system becomes student and the user becomes the tutor. They presented an analysis of principles followed in tutorial dialog and how these strategies can be used in an acquisition system.

Interactive learning in the context of human-robot interaction have been studied broadly under two settings: object learning and plan learning. In a broader context, object or plan learning involves multi-disciplinary areas of research such as vision and robotics. In the current context, however, we will focus on learning semantic information of an entity, grounding a physical object and learning a navigation plan between two points in an environment. A popular spoken dialog strategy is to use a trigger-phrase to initiate learning. Spexard et al. (2006) created an approach that makes the user say a trigger phrase "This is the *object_name*" to initiate the learning. Teaching new objects and their spatial orientation in an environment to a newcomer robot has been studied by Ghidary et al. (2002); Kruijff et al. (2007); Araki et al. (2011); Wei et al. (2009); Chai et al. (2014). This helps the robot to generate a topological map for the new environment. Griffith et al. (2009); Holzapfel et al. (2008); Thomaz and Cakmak (2009) have shown that physical properties of an object can be taught interactively. This allows an agent to associate visual characteristics of an object with the physical properties described in spoken utterances. Object recognition in bad-vision conditions is a challenging task. Kurnia et al. (2004); Holzapfel et al. (2008); Griffith et al. (2009); Ros et al. (2010) have demonstrated that an interactive agent can overcome that challenge by taking assistance from a human. Instruction-based learning is another popular approach for teaching things to the robot. Perzanowski et al. (1998); Theobalt et al. (2002); Bugmann et al. (2004b); Skubic et al. (2004); Doshi and Roy (2008); Rudnicky et al. (2010) have shown that the robot can be taught locations and navigation plans between locations through spoken instructions. Allen et al. (2007) have proposed a more generic approach to teach new tasks to a web-based agent in interactive fashion. This work is similar to (Witbrock et al., 2003), but the difference being task learning as opposed to knowledge base population.

Much of the above work did not focus on spoken dialog strategies, because vision is the primary input modality in most of the situated interactions. However, many of the spoken dialog systems operate in non-situated domains such as movie-recommendation, bus-schedules, and other non-visual entities. Although, some of the work attempted to devise strategies for such domains, they have employed browsing strategies.

My research is about enabling such agents to acquire new knowledge through dialog. We posit three situations in which an agent would interactively learn from users (1) user guided learning (2) system detected learning (3) system initiated learning.

1.3.1 User Guided Learning

A user can guide the agent when they are aware of limitations of system's knowledge about the domain. For example, in command-and-control domains, the user can teach the agent on how to navigate from point A to point B in an environment. This mechanism enables the agent to store new locations and new plans in a knowledge base and later use this information to perform navigational tasks.

This form of learning is potentially vulnerable to non-understandable instructions i.e., utterances with out-of-vocabulary (OOV) words and unseen references to known entities. In such situations, most systems ask the user to repeat their instruction until they succeed. We propose that system could automatically detect such utterances and learn new knowledge from them.

1.3.2 System Detected Learning

To detect and learn new knowledge from utterances with OOV words, we propose a sequence labeling approach. This approach looks for major or critical errors in machine's understanding of the user's utterance, since most OOV words manifest as errors in a non-understanding utterance.

Sometimes entities are part of the recognition vocabulary but they are semantically unknown to the agent. In such situations, the agent may fail to understand the user goal, since many agents typically use a semantic parser to understand user goals. We propose a robust method for user goal prediction that uses open-domain knowledge bases viz., NELL, Wordnet, Freebase.com to expand the semantic context of the user utterance. Open-domain knowledge bases are vast and can provide useful contextual information for common words in an utterance. However, they may not contain entities specific to the agent's domain. We address this issue through system-driven learning process that allows the agent to augment its knowledge base through interactions with its users.

1.3.3 System Initiated Learning

To compensate for domain knowledge that may not be available on the web, we propose that the agent solicit knowledge from its users through dialog. Our hypothesis is that these strategies are practical (in terms of time consumed), provide reliable knowledge about the domain. Such a knowledge can potentially improve the task performance of a dialog system.

1.4 Thesis Statement

Knowledge Discovery Through Spoken Dialog is feasible, useful and portable across domains with the help of data-driven techniques and conversational strategies.

1.5 Thesis Contributions

This thesis contributes following to our understanding of Spoken Dialog Research:

1. User Guided Learning Framework for Dialog Agents. This framework as a part of Spoken Dialog Architecture allows systems that can learn from humans in the course of their activities. See Chapter 2, (Rudnicky et al., 2010), Chapter 3, and (Pappu and Rudnicky, 2012).

2. System Detected Learning mechanism that identifies whether a user input speech has new information to learn. It will help the system identify new words and new facts using different information sources such as speech recognizers, dialog manager, parsers, and knowledge bases. See Chapter 4, (Pappu et al., 2014) and Chapter 5, (Pappu and Rudnicky, 2013).

3. System Initiated Knowledge Acquisition Strategies: to allow the system to acquire new information in any particular dialog state. These system initiated strategies are guided by quality of interactions with users. See Chapter 6, (Pappu and Rudnicky, 2014a) and Chapter 7, (Pappu and Rudnicky, 2014b).

Part I
User Initiated Learning

Chapter 2

Expert User Initiated Learning

Dialog systems are often implemented as static information access systems. Such systems are not able to process information outside of the domain, as defined by the developer. A dialog agent such as a speech-enabled domestic robot operating in a new home will encounter new information. For example, the name of a furniture not previously known to the robot. This can lead to failure in tasks that require to understand instructions with this name. With ability to absorb new information and store it, the agent can succeed in tasks related to this information.

Interactive learning, a process of absorbing knowledge through interaction, becomes necessary for systems to adapt to new environments and people. When a robot enters a new environment and interacts with a user it can learn either by detecting novel information from the user or detect gaps in its existing knowledge, then ask relevant questions to seek knowledge. Earlier works Perzanowski et al. (1998); Billard and Dautenhahn (1999); Lauria et al. (2001) have shown that this process can help the user to train the agent seamlessly and improve the task success of the agent.

Spoken dialog frameworks need to include the learning mechanism as an essential part of their architecture. State-of-the-art dialog frameworks such as Ravenclaw-Olympus (RAVEOLY) (Bohus et al., 2007b), CLSU toolkit (Sutton et al., 1998), AT&T SDT (Williams, 2010), HIS Dialog Framework (Young et al., 2010) and others (e.g., VoiceXML) only allow a static knowledge base (KB) for the applications built using them. A dialog application typically recognizes a user's utterance, parses it, if valid reads a static KB for the user query and responds back with an answer. The agent either correctly interprets the user's query, misinterprets it ("misunderstanding"), or does not interpret it at all ("non-understanding"). In order to verify whether the user's input is "non-understanding" or "misunderstanding", dialog systems typically use error-recovery dialog strategies. However, they do not verify if the user's input has novel information in it. Dialog systems are often built for a specific domain. In some domains (e.g., new-comer robot in a household setting), the items in the KB cannot be listed exhaustively. This leads to system failures, when dealing with the user's input that is not coherent with the KB. Such systems can benefit from a learning mechanism and fill the gaps in their KB through interaction.

We present a interactive learning mechanism integrated into RavenClaw/Olympus dialog framework. This allows us to build dialog systems that stay longer in an environment and incrementally improve their knowledge through long-term interaction with the users.

2.1 Interactive Learning Framework

We have proposed a framework for knowledge acquisition through spoken dialog interaction. Our proposed framework lies within the Ravenclaw-Olympus (RAVEOLY) Spoken Dialog Architecture. As shown in Figure 2.1, in a RAVEOLY based dialog system, Audio Server receives audio signal from the audio device and sends it to the Automatic Speech Recognizer (ASR). Then ASR decodes the speech signal into text and sends back the text to Audio Server. Then the text is transmitted to the Parser to extract the semantics, i.e., concepts. Confidence Annotator (CA) assigns confidence scores for the input concepts. Based on user's input and the context, Dialog Management (DM) component decides what to do next. This may involve communicating with the Domain Reasoner (DR), particularly with a Knowledge Base (KB). A natural language response is generated by Natural Language Generation (NLG) and sent to Output Manager (OM). The OM requests a Text-to-Speech Synthesizer (TTS) to synthesize the system's response. Finally, the synthesized speech is played back to the user by the OM.

Typically, DM interacts with DR to fetch information from the KB, analogous to a database query. In the proposed framework, the DR encompasses two new components to perform the learning mechanism. These components are 1) Knowledge Manager and 2) Knowledge Acquisition

Interface (dark blocks in the Figure 2.1).

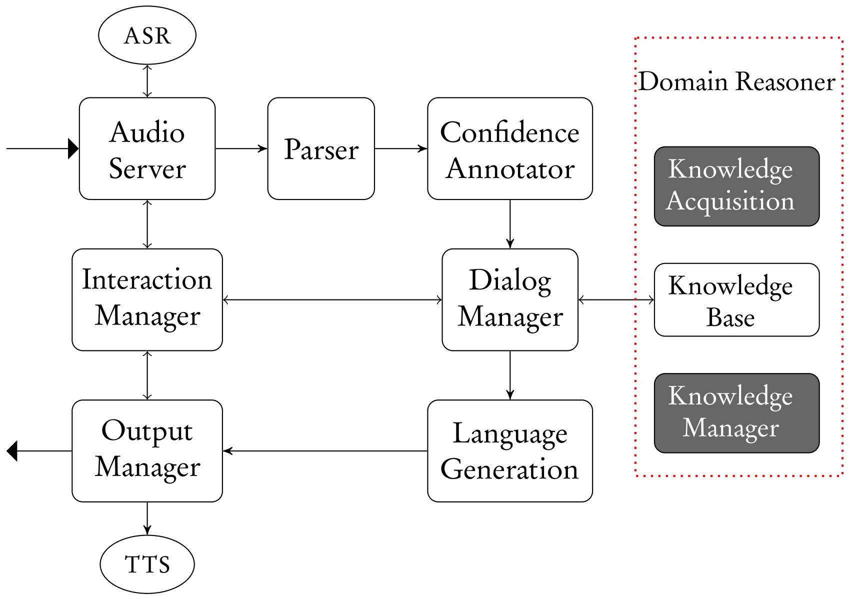


Figure 2.1: Typical Dialog System Components (white) with Learning Components (dark)

An ontology captures the knowledge about a domain. It describes concepts and their relationships in that domain. In our work, we use Protégé, a knowledge development toolkit as the platform. Protégé provides a knowledge editing tool that allows a domain expert to specify seed concepts and relations for a domain.

An ontology mainly constitutes of individuals, classes object-properties and datatype-properties. An individual or instance represents an object in an environment or a domain. A class is a collection of individuals. A class can have two types of properties: object and datatype. Object properties bear relation between two classes, whereas datatype properties are attributes of a class. Examples for each constituent in an ontology are given in the Table 2.1.

Classes in an ontology are arranged in a hierarchical taxonomy. An example taxonomy from navigation domain is shown in the Table 2.2. In Protégé every user-defined class is a subclass of a superclass called *Thing*. Every user-defined class should be specified as disjoint in order to avoid an individual to be part of more than one class (if intented). Properties of a class can be transitive, inverse, symmetric, and/or functional. For a detailed explanation of Protégé OWL ontology refer to its documentation.

We discussed the building blocks for the KB. In the following sections we discuss how different components interact with each other and the KB.

Table 2.1: Examples of Individuals, Classes and Properties

Ontology Component	**Navigation Domain**	**Hobby Domain**
Individual	Instance of hallway	Chess
Class	hallway	Board Game
Object Property	hallway connected to stairs	Person likes a Board Game
Datatype Property	hallway has a carpet	Board Game has number of players

2.1.1 Knowledge Manager Interface

A knowledge manager (KM) mediates between the KB and rest of the dialog system. KM provides different operations on the KB such as inserting a new concept, modifying a property of a concept, or deleting a concept from the KB. In Protégé a concept or individual must be named. However, there is a need for anonymous concepts or individuals during a discourse. In a typical discourse some of the discourse items mentioned are used in a short-term context. Thus, we create anonymous concepts with a temporary name in the KB to include short-term items.

Operations on a KB can be initiated either explicitly or implicitly. Example of an explicit operation is that the dialog system asks the KM to create or read a particular individual's value. An implicit operation would be that KM records every dialog act/action (along with a timestamp) in the form of history in the KB. Every dialog act/event is treated as an anonymous individual with corresponding properties and values. For example, if system performs a *request* act, then the *object* requested and *information* provided in the *request* prompt are recorded as properties of this particular *request* individual. Similarly, in navigation domain, one can record events performed by a robot. This information can help the agent give a critique about its status at a particular time. Another important application of implicit operation is to defer complex processes to later

Category	SubCategory	Example
Imperative	Leave-Location	Exit the building; Come out of the room
	Follow-Path	Walk along the corridor; go across the bridge
	Floor-Transition	Take the elevator to fourth floor; Take the stairs to the fifth
	Turn	Turn left
	Go-To	Walk to the elevators
	Continue	Keep going straight for few steps
Advisory	Floor-Level	You will see fourth floor of other building
	Floor-Transition	You will see elevators
	Compound-Location	You will see a hallway to the right of elevators
	End-of-Pathway	You will see end of the hallway
	Landmark	You will see a TV screen
Grounding	Compound-Location	You are on a hallway right next to the elevators
	End-of-Pathway	You are on the bridge leading to other building
	Floor-Level	You are on fourth floor of the building
	Landmark	You are on standing near TV screen
Meta Comments	Caution	You can find it immediately; Don't go that side
	Miscellaneous	Let me guide you through it; I guess a simpler way would be
	Preface	I will guide you to the cafe in that building

Table 2.2: Example Taxonomy in the Navigation Domain

time. When an utterance is spoken, a parser may not be able extract the concepts requested by the system. In a typical dialog system, the user is either asked to rephrase or repeat the utterance and finally the system gives up. To perform a diagnosis on "give up" situations, we let the KM record the utterance-id associated with the *object* requested by the system. This allows the system to represent the unknown *object* value in the form of an utterance-id which can be recovered after the parser is re-trained.

Besides operations on KB, we use KM to instruct actions to the DM. The KB represents present state of the environment/domain. In a navigation example, a robot may have moved from place *A* to *B* during the interaction with the user. Assume that the user has earlier commanded an action after moving to *B*. Such instructions needs to be recorded and later executed when the condition is satisfied. The role of KM is to record conditional commands and later notify the DM to acknowledge the command on behalf of the agent.

2.1.2 Knowledge Acquisition Interface

The gaps in the knowledge can be identified either by the system or detected while interacting with a user. Therefore, the gap needs to be filled proactively or contextually. KA interface interacts with the DM to ask user about the unknown information.

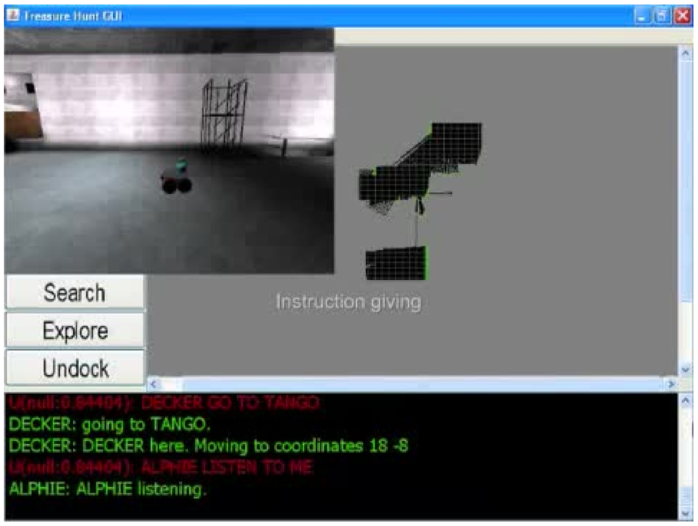
KA interface is responsible for retrieving knowledge gaps that the system would like validate with a user. We allow the domain expert to specify the criteria for the unknown information a system should clarify in a particular dialog state. For example, in a hobby domain, the expert wants the system to know a user's hobby and also inquire information related to the user's friend. The KA interface fetches gaps (if any) related to the user's friend and sends them to DM. This criteria can be specified in the Ravenclaw dialog task tree of the application.

KA is responsible for initiating learning related conversation with the user. The conversational behaviors/strategies are implemented as library agents. These library agents are domain-independent discourse agents part of the Ravenclaw library. We created learning related library agents in order to acquire information such as spelling of a word, concept of a word and plans.

We discuss an application of this framework in the following section.

2.2 Teamtalk: Search and Rescue Domain

Teamtalk, the navigation dialog system provides a command-and-control interface for a mobile robot (illustrated in Figure 2.2). This system is based on RAVEOLY dialog architecture and was originally developed by (Harris and Rudnicky, 2007). System is capable of understanding and executing navigation commands such as “go forward”, “turn right” , “stop” etc.



```
U(null:0.84404): DECKER GO TO TANGO
DECKER: going to TANGO.
DECKER: DECKER here. Moving to coordinates 18 -8
U(null:0.84404): ALPHIE LISTEN TO ME
ALPHIE: ALPHIE listening.
```

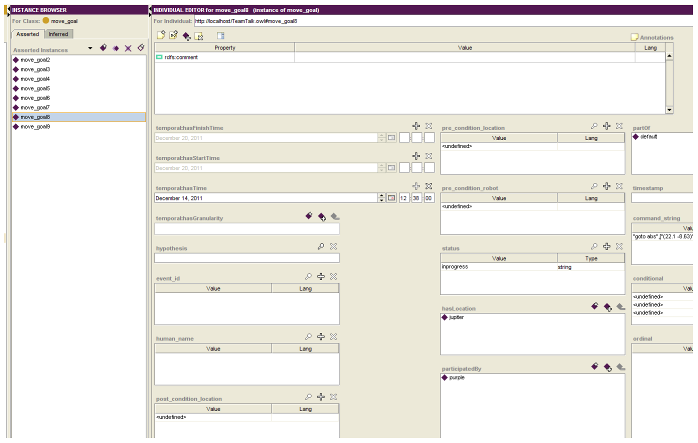
Figure 2.2: Screenshot of the TeamTalk interface with 3D virtual environment, 2D map and the dialog interface.

This system was further extended to learn plans and locations in the environment using the learning framework (Rudnicky et al., 2010). Recording plans can be helpful when the system needs to navigate from one place to another regularly. Recording landmarks is essential because a user can easily refer to location with a name instead of specifying coordinates to ask the robot to go to a location. The system operates in two modes: a) command mode and b) learning mode. In command mode, system only understands and executes an instruction. In learning mode, system can save a set of instructions as a plan for immediate and future execution of a plan. Similarly, it can remember a location with a user-specified name for the location. User can trigger the learning phase by using key-phrases such as “start learning *plan_name*”, to initiate plan learning and “stop learning” to end the learning phase. Similarly, the user can initiate location learning with a trigger phrase e.g., “remember this location as *location_name*”.

Besides learning plans and locations, this system also records history of events (commands issued) for diagnosis in later dialog sessions. These events are organized as episodes and stored in the KB. An episode is a temporally-related set of events e.g., an episode may constitute a spoken instruction event followed by the execution of an action event.

2.2.1 Ontology

The TeamTalk ontology keeps track of relevant entities in the search domain. The ontology is based on the OWL representation using the Protege environment (as shown in Figure 2.3). The ontology component responds to queries from the Domain Reasoner and accepts dynamic updates (e.g., the identity of newly labeled locations). The ontology also maintains a complete history of events that have occurred and stores this information for use in later sessions. Events are organized into episodes representing a coherent, temporally-related set of events. TeamTalk is able to process simple factual queries about the past. The system currently maintains a single knowledge base; once a new instance is created it is immediately transparently available to all robots. While we understand that private knowledge may have a role in team activities it does not appear critical for our current scenarios.


INSTANCE BROWSER
For Class: move_goal
Asserted Instances: move_goal2, move_goal3, move_goal4, move_goal5, move_goal6, move_goal7, move_goal8, move_goal9

INDIVIDUAL EDITOR for move_goal8 (instance of move_goal)
For Individual: http://localhost/TeamTalk.owl#move_goal8

Property	Value	Lang
rdfs:comment		

temporal:hasFinishTime: December 20, 2011
temporal:hasStartTime: December 20, 2011
temporal:hasTime: December 14, 2011 12:38:00
temporal:hasGranularity:
hypothesis:
event_id:
human_name:
post_condition_location: <undefined>
pre_condition_location: <undefined>
pre_condition_robot: <undefined>
status: inprogress (Type: string)
hasLocation: jupiter
participatedBy: purple
partOf: default
timestamp:
command_string: "goto abs",["(22.1 -8.63)"
conditional: <undefined>, <undefined>, <undefined>
ordinal:

Figure 2.3: Screenshot of the TeamTalk Ontology in the Protege Editor.

Learning Locations

User: Alphie Go forward Five meters
System: going forward five meters
U: Alphie call it as "jupiter"
S: ok, this is "jupiter"

Figure 2.4: Example Dialog for Location Learning

Learning Plans

U: Alphie [system] learn zigzag
S: start learning zigzag
U: go forward four meters
S: okay, adding forward four meters to zigzag
U: turn right forty five degrees
S: adding turn right forty five degrees to zigzag
U: go forward six meters
S: adding go forward six meters to zigzag
U: turn left forty five degrees
S: adding turn left forty degrees to zigzag
U: go forward four meters
S: adding go forward four meters to zigzag
U: stop learning zigzag
S: I have learned "zigzag".

Figure 2.5: Example Dialog for Play Learning

2.2.2 Plans

TeamTalk is able to learn and represent action sequences consisting of individual commands and aggregates. For example, the current system can learn a zig-zag behavior by listening to a series of movement instructions (illustrated in Figure 2.5). We refer to these as plays, and the implementation makes use of the Play Manager (PM) mechanism described in (Dias

Editing Plans

U: alphie edit zigzag
S: there are 2 moves and 2 turns in zigzag, what do you want to change?
U: change the distance of move from four to two
S: okay, changed the distance from four to two
U: change the distance of move from three to two
S: no operation found with distance two
U: stop editing
S: ok. Storing procedure.
U: Do a zigzag

Figure 2.6: Example Dialog for Plan Editing

et al., 2006). In TeamTalk, we add a Play Dispatcher to allow multiple plays to execute simultaneously. Plays are implemented as dynamic Ruby programs. Each play consists of steps corresponding to individual commands as well as standard programming constructs such as blocks and iteration. The human specifies a new play interactively while the robot manages the recording process and performs consistency checks. This ensures reasonable behaviors, although these may not necessarily abstract to reasonable interaction with the environment. Plays are stored in the ontology and subsequently become available to all robots. Plan provide functional descriptions of new robot behavior, therefore they can be customized to new situations.

2.2.3 Editing Plans for a New Environment

The purpose of abstract representation of a plan is allow the agent to adapt a plan to a new environment. We have designed dialog strategies to allow an expert user to modify an existing plan to suit the dynamics of a new environment. As illustrated in in the Figure 2.6, a user can trigger plan editing behavior and change the command values through dialog. First the user initiates plan editing dialog, then the system gives an overview of the instructions involved in this plan. Then the user requests the system to modify certain attribute of an instruction category. The system is also capable of handling exceptions when a requested attribute or instruction-type is not available in the plan.

2.3 Chapter Summary

In this chapter, we have presented a learning framework integrated into the RAVEOLY dialog architecture. We have incorporated new constituents into the framework namely Knowledge Manager and Knowledge Acquisition Interface. We have demonstrated that this framework can support user-initiated learning in a search-and-rescue application. We show that an agent can learn plans and locations in the given environment and store them in a knowledge base. The gathered information can be later used to perform tasks in the environment. Now that the agent can understand instructions and also learn plans composed of these instructions, we would like to know if we can use the same machinery in different environments. In the next chapter we are going to discuss how can we adapt learning mechanism to new environments given what the agent has learned in its initial environment.

Chapter 3

Adapting User-Initiated Learning to New Situations

In the previous chapter, we have discussed how can an expert user can teach the agent about its situation. In this chapter, we are going to look at how can we adapt the learnings of one situation to new situations. This is an important problem because a mobile robot or an agent is expected to move from one situation to the other. It can be challenging if an expert user has to teach the agent for every new environment that the agent visits. Hence it is essential to adapt the learning machinery to new situations.

Generating and interpreting instructions is a topic of enduring interest. Cognitive psychologists have examined how people perceive spatial entities and structure route instructions (Daniel and Denis, 1998; Allen, 1997). Linguists and others have investigated how people articulate route instructions in conversation with people or agents (Eberhard et al., 2010; Gargett et al., 2010; Stoia et al., 2008; Marge and Rudnicky, 2010). Artificial intelligence researchers have shown that under supervised conditions autonomous agents can learn to interpret route instructions (Kollar et al., 2010; MacMahon et al., 2006; Matuszek et al., 2010; Bugmann et al., 2004a; Chen and Mooney, 2010).

The language of instructions contains a variety of relevant propositions: a preface to a route, an imperative statement, or a description of a landmark. Following example illustrates instructions of different types.

PREFACE: You are going to La Prima Cafe in Wean Hall it is to our west
IMPERATIVE: You want to take a right then walk straight
DESCRIPTION: then you will see green wall with a tv screen

Previous work has proposed both coarse and fine-grained instruction taxonomies. Bugmann et al. (2004a) proposed a taxonomy of 15 primitive categories in a concrete "action" framework. In contrast, Daniel and Denis (1998) suggested a five-way categorization based on cognitive properties of instructions.

Instructions vary greatly and can include superfluous detail. (Denis et al., 1999) found that when people were asked to read and assess a set of instructions some of the instructions were deemed unnecessary and could be discarded. There is some evidence (Lovelace et al., 1999; Caduff and Timpf, 2008) that only the mention of significant landmarks along the route leads to better-quality instructions. Computational (rather than descriptive) approaches to this problem include: using sequence labeling approach to capture spatial relations, landmarks, and action verbs (Kollar et al., 2010; Wei et al., 2009), generating a frame structure for an instruction (MacMahon et al., 2006), or using statistical machine translation techniques to translate instructions into actions (Matuszek et al., 2010).

In order to adapt the instruction understanding system to new situations, first we need to analyze how route instructions are given in various scenarios. First we will describe a user study where we have collected a corpus of route instructions in different scenarios. Then we present its analysis in terms of a taxonomy suitable for automated understanding and a verification that the instructions are in fact usable by humans. With a view to automating understanding, we also constructed a grammar capable of processing this language, and show that it provides good coverage for both our corpus and corpora from three other situations (Kollar et al., 2010; Marge and Rudnicky, 2010; Bugmann et al., 2004a).

3.1 The Navagati Corpus

We collected a corpus of spoken instructions describing how to get from one part of a large building complex to another. To ensure consistency we recruited individuals who were familiar with the situation and consequently could formulate such instructions without reference to maps or other materials. Since we are also interested in how such instructions are edited, we also included conditions in which subjects were asked to modify their instructions in several ways. The corpus is publicly available[1].

3.1.1 Participants and Procedure

We recruited subjects who were both fluent English speakers and were also familiar with the situation (a university building complex). Subjects were told to imagine that they had encountered a visitor, not familiar with the campus, at a specific location (in front of elevators on a particular floor) who needed instructions to a specific location, a café two buildings away.

For each set of instructions, subjects were asked to think about the route and their instructions, then record them as a single monologue. Subjects sat in front of a computer and wore a close-talking microphone. Initially no map was provided and they were expected to rely on their memory. In subsequent tasks they were shown a floor-plan indicating a specific location of the visitor and asked to modify their instructions. Speech was transcribed using Amazon Mechanical Turk. Transcriptions were normalized to standardize spellings (e.g., building names).

3.1.2 Design

We were interested in two general cases: normal instructions (**Simple** scenario) and repairing existing instructions (**Repair** scenario). Each scenario included three tasks, as described below.

We selected two locations that could be walked between without necessarily going outside. The first location (A) was in front of an elevator on the seventh floor of Gates Hillman Center, the second location (B) was a cafe on the fifth floor of Wean Hall at CMU campus. The expected pathway included changes in floor, direction and passing through a different building. It required reasonably detailed instructions.

In the **Simple** scenario, subjects were asked to generate three variants, as follows: (1) instructions for $A \rightarrow B$; (2) for $B \rightarrow A$; and (3) a simplified version of (2). For the latter, we were interested in the degree of instruction reuse and the condensation strategy.

The **Repair** scenario was designed to probe how a subject would alter their instructions in response to complications. Subjects were asked to modify their initial Simple instructions ($A \rightarrow B$) to cope with: (1) visitor

[1]http://tts.speech.cs.cmu.edu/apappu/navagati/

missing a landmark and takes a wrong turn; (2) an obstruction (construction) blocking the original path; and (3) the visitor getting lost and ends up in an unknown part of the (middle) building. For each case, the subject was given a map that marked the visitor's location and had to get the visitor back on track.

3.1.3 Analysis

Nine subjects performed 6 tasks each, producing 54 sets of instructions, for a total of 65 minutes of speech. The transcriptions were segmented semi-automatically into atomic units corresponding to instruction steps. For example, the instruction "Go left, then turn right" was segmented into: "go left", and "then turn right" based on bigram heuristics. We compiled a list of most frequent bigrams and trigrams in the corpus e.g., "and then", "after that" etc. The transcriptions were segmented at the bigram/trigram boundaries and were manually verified for the correctness of a segment. The Simple scenario generated 552 instructions, the Repair part contained 382 instructions, a total of 934. The vocabulary has 508 types and 7937 tokens. Table 3.1 summarizes the factors measured in both the scenarios. Only two (marked by *) differed between scenarios (t-test at $p < 0.05$). We examined acoustic properties (for example mean pitch) but did not find any significant differences across scenario type. Figures 3.1 and 3.2 show excerpts of route instructions in simple and repair scenarios. Primary difference between them is within instructions 3 and 5. Here we see that the direction-giver has changed the path when they understood that there is an obstruction on the hallway.

Table 3.1: Simple vs Repair Scenario

Factors	Simple	Repair
# Tokens	4461	3476
# Types	351	375
# Instructions	552	382
# Words-per-Instruction*	7.5	8.0
# Landmarks	450	314
# Motion Verbs*	775	506
# Spatial Prepositions	61	60
# Filler Phrases	414	380

Simple Scenario

1. then you go the end of the hallway
2. then you turn right
3. you are going to see an elevator
4. if you take the elevator
5. and go to the fifth floor you get to your destination

Figure 3.1: Excerpt of Instructions in Simple Scenario

Simple Scenario

3. since the hallway is impassable
4. you want to take the stairs instead of the elevators
5. then you will turn right and see the cafe in front of you

Figure 3.2: Excerpt of Instructions in Repair Scenario

We can compare language similarity across scenarios by comparing the perplexity of text in the two scenarios. If the instructions and repairs are similar, we would expect that a model built from one scenario should be able to capture data from the other scenario. We randomly divided data from each scenario into training (70%) and testing data (30%). We built a trigram language model (LM) smoothed with absolute discounting using the CMU-SLM toolkit (Rosenfield, 1995). Then, we computed the perplexity on testing data from each scenario against each model. From Table 3.2, Simple-LM has lower perplexity compared to Repair-LM on the test sets. The perplexity of Simple-LM on Repair-Test is slightly higher when compared to Simple-Test. This could be due to the lexical diversity of the Repair scenario or simply to the smaller sample size. Table 3.1(row 1) indicates that the data in Repair scenario is smaller than data in Simple scenario. To explore the lexical diversity of these two scenarios we conducted a qualitative analysis of the instructions from both the scenarios.

In Task 1 of the Simple scenario, we only observed a sequence of instructions. However in Task 2 of Simple Scenario, we noticed references to instructions from Task 1 via words like "remember", "same route", etc. This suggests that instructions may be considered in context of previous exchanges and that this history should normally be available for interpretation purposes. In Task 3 of the Simple scenario, 7 out of 9 subjects simply repeated the instructions from Task 2 while the rest provided a different version of the same instructions. We did not observe any other qualitative differences across three tasks in the Simple scenario.

In Task 1 of the Repair scenario, all but one subject gave instructions that returned the visitor to the missed landmark, instead of bypassing the landmark. In Task 2, the obstruction on the path could be negotiated through a shorter or longer detour. But only 4 out of 9 participants suggested the shorter detour. In Task 3, we did not observe anything different from Task 2. Despite the difference in the situations, the language of repair was found to be quite similar. The structure of the delivery was organized as follows: (1) Subjects introduced the situation of the visitor; (2) then modified the instructions according to the situation. Introduction of the situation was different in each task, (e.g., "you are facing the workers" vs "looks like you are near office spaces" vs "if you have missed the atrium you took a wrong turn"). But the modification or repair of the instructions was similar across the situations. The repaired instructions are sequences of instructions with a few cautionary statements inserted between instructions. We believe that subjects added cautionary statements in order to warn the visitor from going off-the-route. We observed that 6.3% of the repaired instructions were *cautionary* statements; we did not observe cautionary statements in the original Simple scenario. In order to see the effect of these cautionary statements we removed them from both training and testing sets of the Repair scenario, then built a trigram LM using this condensed training data (Repair–w/o-cautionLM). Table 3.2 shows that perplexity drops when cautionary statements are excluded from the repair scenario, indicating that Simple and Repair scenarios are similar except for these cautionary statements.

3.2 Taxonomy of Route Instructions

Taxonomies have been proposed in the past. Daniel and Denis (1998) proposed a taxonomy that reflected attributes of spatial cognition and included 5 classes: (1) Imperatives; (2) Imperatives referring a landmark; (3)

Table 3.2: Perplexity of Simple/Repair Language Models

LM/Test	Simple-Test	Repair-Test	Repair w/o caution
Simple-LM	29.6	36.5	30.3
Repair-LM	37.4	37.3	35.6
Repair w/o cautionLM	31.9	37.6	26.8

Introduction of a landmark without an action; (4) Non-spatial description of landmarks and (5) Meta comments. Bugmann et al. (2004a) suggested 15 primitive (robot-executable) actions. We present a hierarchical instruction taxonomy that takes into account both cognitive properties and the needs of robot navigation.

Category	**SubCategory**	**Distribution**
Imperative	`Leave-Location`	2.3%
	`Follow-Path`	7.0%
	`Floor-Transition`	11.2%
	`Turn`	24.2%
	`Go-To`	27.2%
	`Continue`	28.0%
Advisory	`Floor-Level`	5.4%
	`Floor-Transition`	12.2%
	`Compound-Location`	13.4%
	`End-of-Pathway`	21.5%
	`Landmark`	47.5%
Grounding	`Compound-Location`	5.9%
	`End-of-Pathway`	8.2%
	`Floor-Level`	42.4%
	`Landmark`	43.5%
Meta Comments	`Caution`	14.7%
	`Miscellaneous`	36.0%
	`Preface`	49.3%

Table 3.3: Taxonomy of Categories

3.2.1 Categories

We segmented the spoken instructions using a criterion that split individual actions and observations. Our taxonomy is roughly comparable to that of (Daniel and Denis, 1998) but differs in the treatment of landmarks, which we place into their own category. The taxonomy has four major categories that subsume 18 sub-categories; these are given in Table 3.3. We now describe them in detail.

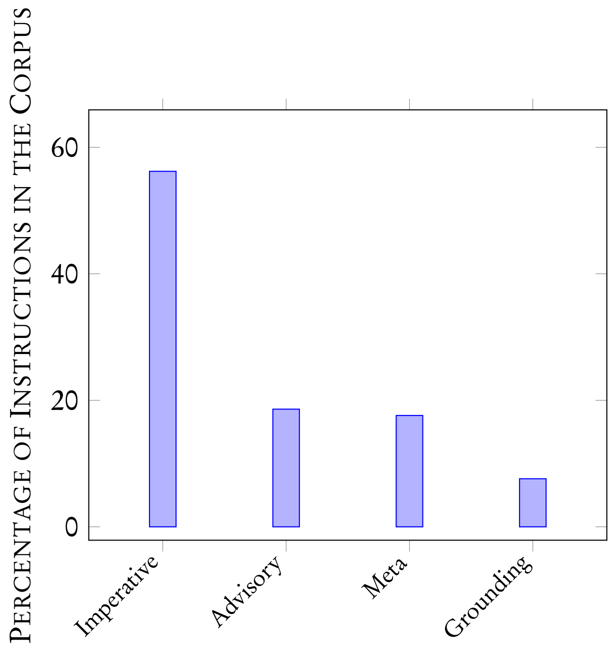


Figure 3.3: First Tier Instruction Categories

- Imperative Instructions
  Imperative instructions are executable and can result in physical displacement. We identified seven subcategories of Imperatives that distinguish different contexts (e.g., going along a corridor, changing floors via elevator or stairs, or going to a specific location).

  Imperative instructions can also include *preconditions* or *postconditions*. The order of their execution varies based on the directionality of the condition between two instructions. `Continue` is interesting because it can have *travel-distance* and *travel-direction* arguments, or even no arguments. In the latter case the follower continues an action (e.g., "keep walking"), until some unspecified condition ends it.

- Advisory Instructions
  While giving route instructions people mention landmarks along the route as feedback to the direction-follower. Some of these landmarks are not part of the path but do serve as waypoints for the follower (e.g., "you will see a hallway right there"). We observe that landmarks are distinct either functionally and/or physically. For example, a hallway is similar to a door as it connects two waypoints on the route that are located at same level. However it is different from elevators or stairs because they connect waypoints that are located on different levels of a building. Based on this distinction, we divided advisory instructions into five sub-categories depending on the type of landmark mentioned in the instruction (see Table 3.3).

  Compound locations (see Table 3.3) are closely located but physically distinct. They may constitute part-whole relationships e.g., "TV screen with a motion sensor". We observed that compound locations are used to disambiguate when multiple instances of a landmark type are present e.g., "chair near the elevator vs "chair near the hallway".

- Grounding Instructions
  Grounding instructions report absolute position. These instructions indicate current view or location as opposed to future view or location (indicated through advisory instructions). These instructions constitute a landmark name similar to advisory instructions and also follow the distinction between the type of landmark mentioned in the instruction (see Table 3.3).

- Meta Comments
  Meta comments are non-executable instructions added to route instructions. People often make these comments at the beginning of instructions and sometimes in between two imperative statements e.g., a precautionary statement. In our corpus we found meta-comments in two situations: (1) Preface or introduction of the

route; (2) Caution against a (metaphorical) pitfall in the route. Both the example instructions and the distribution of the subcategories are given in Table 3.3.

The language of meta comments is more diverse than that of the other three categories. If we build trigram language models for each category and measure the perplexity on a held-out set from same category the perplexity is relatively high for Meta (49.6) compared to other categories (Advisory: 19.5; Imperative: 18.5; Grounding: 11.4). This suggests that automatic understanding of meta comments might be problematic, consequently it would be useful to determine the relative utility of different instruction categories. The next section describes an attempt to do this.

3.3 Instructions Relevant for Navigation

In order to find that out we asked people to follow instruction sets selected from our corpus. Daniel and Denis (1998) conducted a similar study where they asked subjects to read a set of instructions and strike-off instructions with too much or too little information. However, people may or may not feel the same when they follow (physically navigate) these instructions. Therefore, in our study the experimenter read instructions (of varying amount of detail) to the subjects while they physically navigated through the situation.

3.3.1 Participants and Procedure

We chose 5 out of the 9 instruction sets, spoken by different subjects (of average length 26.8 instructions per set) from Task 1 of the Simple scenario discussed above. We did not use the others because they contained few instructions (average of 13.5) and provided fewer instances of instructions in different categories.

Our set of instructions included the full set, a set with only imperatives and additional sets adding only one of the remaining categories to the imperative set (see Table 3.4), producing 25 distinct sets of instructions. Additionally, building names and the destination name (transcribed in the instructions) were anonymized to avoid revealing the destination or the "heading" at the early stage of the route.

We recruited 25 subjects, each doing one variant of the instructions. In the session, the experimenter read one instruction at a time to the subject and walked behind the subject as they proceeded. Subjects were asked to say "done" when ready for the next instruction; they were allowed to ask the experimenter to repeat instructions but otherwise were on their own. The experimenter kept track of how and where a subject got lost on their way to destination. (No systematic effects were observed, but see below.) At the end subjects were handed the entire set of instructions and were asked to mark which instructions were difficult to follow or were redundant. Remaining instructions were deemed to be useful and interpretable.

Table 3.4: Variants of an Instruction Set

Variant	Imperative	Advisory	Grounding	Meta
Imp	✓			
Imp+Adv	✓	✓		
Imp+Grnd	✓		✓	
Imp+Meta	✓			✓
Entire Set	✓	✓	✓	✓

3.3.2 Analysis

Except for one subject, everybody reached the destination. Subjects found Imperative and Advisory instructions more useful compared to Grounding instructions and Meta comments, irrespective of the instruction-set they followed (see Figure 3.5). However, they also found a few of the imperative and advisory instructions difficult to follow as shown in Figure 3.4. While following these difficult instructions, people realized that they got lost and asked the experimenter to repeat the instructions. Examples of difficult instructions and the people's complaint on that instruction are as follows:

- *So you kind of cross the atrium* **Complaint:** participants reported that they were not sure how far they had to walk across the atrium.
- *Go beside the handrails till the other end of this building* **Complaint:** no absolute destination, multiple hallways at the end of handrails

- *Just walk down the hallway exit the building* **Complaint:** multiple exits to the building
- *After you get off the elevator, take a left and then left again* **Complaint:** more than one left confused the subjects
- *You can see the building just in front of you* **Complaint:** there were three buildings standing in front and the target building was slightly to the left.
- *You will see the corridor that you want to take* **Complaint:** there were two corridors and the orientation was unspecified in the instruction.

Category/Variant	Imp	Imp+Grnd	Imp+Meta	Imp+Adv	Entire Set
Diff-Imp	11	10	12	9	12
Diff-Adv	0	10	5	10	10
Diff-Grnd	0	0	13	0	0
Diff-Meta	4	15	12	4	4
Diff-All	6	9	11	7	9

Figure 3.4: What percent of instructions are Difficult (Diff)? Darker=Difficult

3.4 Adapting to New Situations

The Navagati (NAV) corpus instructions were divided into training set (henceforth abbreviated as NAV-train) and testing set (abbreviated as NAV-test) of size 654 (of 6 subjects) and 280 (of 3 subjects). The training set was used to create a grammar based on the taxonomy described in the earlier section.

3.4.1 Grammar

A domain-specific grammar was written to cover most frequent phrases from the training set using the Phoenix (Ward, 1991) format. Phoenix

Category/Variant	Imp	Imp+Grnd	Imp+Meta	Imp+Adv	Entire Set
Redun-Imp	5	8	12	11	8
Redun-Adv	5	10	19	10	29
Redun-Grnd	20	13	47	53	27
Redun-Meta	19	31	65	23	50
Redun-All	9	13	26	17	21

Figure 3.5: What percent of instructions are Redundant (Redun)? Darker=More Redundant Instructions

Corpus	#Instr	Words/Instr	Environmnt	Modality	H/R-H/R
NAV	934	9	UnivCampus	Speech	Humn-Humn
MIT	684	15	UnivCampus	Written	Humn-Humn
IBL	769	8	ModelCity	Speech	Human-Robot
TTALK	1619	7	OpenSpace	Speech	Human-Robot

Figure 3.6: Nature of the Corpora

Corpus	Lifting	PathWays	Landmarks	Adjectiv
NAV	0.029	0.046	0.169	0.13
MIT	0.045	0.016	0.163	0.062
IBL	*n.a.*	0.039	0.076	0.13
TTALK	*n.a.*	0.027	0.01	0.039

Figure 3.7: Type-Token Ratio of Concepts across Corpora

grammars specify a hierarchy of target concepts and is suited to parsing spontaneous speech. The resulting grammar produced correct and complete parses on 78% of the training data (NAV-train). The remaining training instances were not included due to unusual phrasing and disfluencies. The concepts in the grammar are listed in the Table 3.5.

Table 3.5: Higher level and Leaf node Concepts in Grammar

Category Concepts	**Examples**
Imperative	GoToPlace, Turn, etc
Conditional Imperative	Move_Until_X where X is a condition
Advisory Instructions	You_Will_See_Location
Grounding Instructions	You_are_at_Location
Auxillary Concepts	**Examples**
Locations	buildings, other landmarks on the route
Adjectives-of-Locations	large, open, black, small etc.
Pathways	hallway, corridor, bridge, doors, etc.
LiftingDevice	elevator, staircase, stairwell, etc.
Spatial Relations	behind, above, on right, on left, etc.
Numbers	turn-angles, distance, etc.
Ordinals	first, second as in floor numbers
Filler phrases	you may want to; you are gonna; etc.

- Vocabulary from New Situations
  Concepts such as Locations, Pathways and Adjectives-of-Location use vocabulary that is specific to an situation, and the vocabulary of these concepts will change with surroundings. We used an off-the-shelf part-of-speech tagger (Toutanova et al., 2003) on NAV-train to identify "location-based" nouns and adjectives. These were added to the grammar as instances of their respective concepts.

3.4.2 Parsing Instructions

A parse can fall into one of the following categories: 1) *Complete*: clean and correct parse with all concepts and actions mentioned in the instruction. 2) *Incomplete*: If some arguments for an action are missing. 3) *Misparse*: no usable parse produced for an instruction.

Table 3.6 shows that 87% of the instructions from the NAV corpus (excluding meta comments) are parsed correctly. Correct parses were produced for 89% of Imperatives, 87% of Advisory and 73% of Grounding instructions. Meta comments were excluded because they do not constitute any valid actions and can be ignored. Nevertheless 20% of the meta comments produced a valid parse (i.e. unintended action).

3.4.3 Evaluating Adaptability

The results for the NAV corpus seem encouraging but it would be useful to know whether the NAV grammar generalizes to other directions scenarios. We selected three corpora to examine this question: MIT (Kollar et al., 2010), IBL[2] (Bugmann et al., 2004a) and TTALK[3] (Marge and Rudnicky, 2010). All were navigation scenarios but were collected in a variety of settings (see Figure 3(a)). Corpus vocabularies were normalized using the process described in 5.1.1 and location specific nouns and adjectives added to the grammar. Punctuation was removed. Figure 3(b) shows the type-token ratios for "variable" concepts. There are more landmarks and adjectives (that tag along landmarks) in NAV and MIT compared to IBL and fewest in TTALK corpus (a closed space with two robots). Since, IBL and TTALK do not involve extensive navigation inside the buildings there are no instances of the elevator concept. However, IBL corpus has "exits, roads, streets" in the city environment which were included in the PathWay concept.

3.4.4 Performance across Corpora

We randomly sampled 300 instructions from each of the three corpora (MIT, IBL and TTALK) and evaluated their parses against manually-created parses. Table 3.6) shows results for each type of parse (Complete, Incomplete, or Misparse). Meta comments were excluded, as discussed earlier. The NAV grammar appears portable to three other corpora. As shown in Category-Accuracy of Table 3.6, Imperatives and Advisory instructions are well-parsed by the grammar. In TTALK corpus, there are very few landmark names but there are certain unusual sentences e.g., "she to the rear left hand wall of the room" causing lower accuracy in Advisory instructions. We noticed that MIT corpus had longer description of the landmarks, leading to lower accuracy for Grounding. From Table 3.6 11% to 16% of Imperative instructions fail to get parsed across the corpora. We consider these failures/errors below.

[2]http://www.tech.plym.ac.uk/soc/staff/guidbugm/ibl/readme1.html
[3]http://www.cs.cmu.edu/~robotnavcps/

Table 3.6: Parse Results

Parse Results	**NAV**	**MIT**	**IBL**	**TTALK**
# Instructions	280	300	300	300
% Complete	87%	78.8%	83.8%	83.4%
% Incomplete	3.1%	17%	6.6%	3.7%
% Misparse	9.8%	4.1%	9.5%	13%
Category Accuracy				
Imperative	89%	89.4%	86.5%	84.7%
Advisory	87%	93.4%	87.4%	60%
Grounding	73%	62%	100%	100%

3.4.5 Error Analysis

We found six situations that produced incomplete and misparsed instructions: (1) Underspecified arguments; (2) Unusual or unobserved phrases; (2) False-starts and ungrammatical language; (3) Uncovered words; (4) Prolonged description of landmarks within an instruction; (5) Coreferences; 6) Non-specific instructions (eg. either take the right hallway or the left hallway).

- Incomplete and Misparsed Instructions
  Out-of-Vocabulary (OOV) words were responsible for the majority of incomplete parses across all the corpora; many were singletons. Unusual phrases such as "as if you are doubling back on yourself" caused incomplete parses. We also observed lengthy descriptions in instructions in the MIT corpus, leading to incomplete parses. This corpus was unusual in that it is composed of written, as opposed to spoken, instructions.

  Misparsed instructions were caused by both ungrammatical phrases and OOV words. Ungrammatical instructions contained either missed key content words like verbs or false starts. These instructions did contain meaningful fragments but they did not form a coherent utterance e.g., "onto a roundabout". We note that incomplete or otherwise non-understandable utterances can in principle be recovered through clarification dialog.

Table 3.7: Error Analysis for Incomplete and Misparsed instructions

Incomplete	**NAV**	**MIT**	**IBL**	**TTALK**
# Incomplete Instructions	8	49	19	10
MissingArgs	50%	8%	0%	0%
UnusualPhrases	0%	28%	35%	60%
Lengthy Descriptions	0%	20.4%	0%	0%
Coreferences	0%	0%	20.2%	0%
Non-concrete phrases	3%	2%	5%	0%
OOVs	47%	41.6%	39.8%	40%
Misparse				
# Misparse Instructions	25	12	27	39
Ungrammatical phrases	24%	44%	16%	10%
OOVs	76%	66%	84%	90%

3.5 Chapter Summary

In this chapter, we wanted to address the problem of adapting learning mechanism and the underlying instruction understanding process to new situations. In order to have a better understanding of the structure of instructions and to investigate how these might be automatically processed, we collected a corpus of spoken instructions. We found that instructions can be organized in terms of a straighforward two-level taxonomy. We examined the information contents of different components and found that that the Imperative and Advisory categories appear to be the most relevant, though our subjects had little difficulty dealing with instructions composed of only Imperatives; physical context would seem to matter.

We found that it was possible to design a grammar that reasonably covered the information-carrying instructions in a set of instructions. And that a grammar built from our corpus generalized quite well to corpora collected under different situations.

This chapter shows that robust instruction-understanding systems can be implemented and, other than the challenge of dealing with location-specific data, can be deployed in different situations. We believe that this study also highlights the importance of dialog-based clarification and the need for strategies that can recognize and capture out-of-vocabulary words. In the next chapter, we will discuss how can we handle such errors and recover useful information from these errors.

Part II
System Detected Learning

Chapter 4

Detecting and Learning from Critical Errors in Dialog

In the previous chapter, we have seen that user-guided learning is challenged by out-of-vocabulary words that cause misunderstanding or non-understanding of the instructions. Bohus and Rudnicky (2005c) found that some of the non-understanding errors are caused due to novel information in an such as, out-of-vocabulary words, out-of-grammar language, utterance. The novel information can manifest in different forms out-of-domain utterances or out-of-topic discourse. In dialog systems, the size of speech recognition vocabulary is limited. As a consequence of this limitation, a recognition engine can encounter out-of-vocabulary words. A spoken language parser can be challenged by phrases not covered in the domain grammar. Novice users who are not aware of system's boundaries of operation, may request queries that might be part of the system's domain of operation. Off-topic discourse can occur when the dialog manager is not expecting a particular user input although it is a well-parsed utterance.

In a slot-filling setting (e.g., flight reservation domain), a spoken dialog system (SDS) can use the domain-knowledge to verify the slots (e.g., city names) and their values (e.g., Florence) in the utterance. In SMS setting, there is no such domain-knowledge available to verify the errors. Consider the following example:

REFERENCE: We are planning to meet at Madras Cafe

HYPOTHESIS-1: We are planning to meet at MY ADDRESS COFFEE
HYPOTHESIS-2: We are planning _ meet to madras cafe

While HYPOTHESIS-1's error lead to miscommunication, the HYPOTHESIS-2's error usually does not affect communication and can be tolerated [1]. This leads to the question: *How do we detect significant errors that lead to miscommunication?* In a slot-filling SDS, the error detection module labels the slot-values as error or correct, and assigns a confidence score to the label. Then, the dialog manager looks at these labels and makes a decision whether to use the ASR-result, or reject it. In this pipeline, the error detection typically has no access to the dialog decisions made, thus no feedback on the detected errors. This leads to another question: *Can we optimize the major error detection to maximize the correction of major errors?* This work investigates the two questions posed in this paragraph.

Conventional error detection, both in slot-filling (Filisko and Seneff, 2004a; Pincus et al., 2013; Bechet and Favre, 2013) and in dictation (Shi and Zhou, 2005; Ogawa et al., 2012), does not discriminate errors based on severity. Studies that discriminate between errors (Bohus and Rudnicky, 2001; Prasad et al., 2012; Dufour et al., 2012), do not optimize the detection process to ensure that the errors are worth clarifying. Bohus and Rudnicky (2005b); Skantze (2007a) optimize confidence thresholds to task success for slot-filling. In this work, we present an error detection technique, as well as dialog-action optimized error detection. In particular, we investigate:

1. *Which errors are more important and how do we detect them?* We propose a 3-way labeling method that labels words as either correct, MAJOR errors or MINOR errors. We also show that our method outperforms 2-way labeling methods such as (Ogawa et al., 2012; Pincus et al., 2013; Bechet and Favre, 2013) that only distinguish between correct and error words.

2. *Can we optimize the detection of major errors to maximize their correction?* To this end, we optimize the detection step using metacosts that update word error labels and show that it performs better at selecting corrective-actions. We simulate the dialog using a hand-crafted dialog policy in our experiments.

[1]Especially when texting is a secondary task, such as while driving

In this chapter, first we will analyze speech recognition errors based on their severity. Then we will present dialog strategies that can help us recover information from such errors. We will describe our sequence labeling approach for error detection that optimizes towards the dialog action selection. We show that optimized approach outperforms the unoptimized version significantly, thus helps in recovering information from errors. We will discuss the interactive version of the error detection system. We have successfully applied our approach for spoken short messages and for speech-to-speech translation application. In the speech-to-speech translation application, we show that we can recover OOV words and learn their concept category through dialog.

4.1 Analysis of Recognition Errors

4.1.1 Data Collection

We collect the spoken SMS corpus in two phases [2]. (1) Subjects are given a set of scenarios and asked to respond with an SMS in their own words. (2) A different set of subjects are asked to read the messages that are collected in the previous step.

The scenarios are associated with daily activities that subjects might be familiar with. An example scenario is *You are scheduling when to meet with your friend. Send a message on it*. In both phases, subjects are provided with a web interface to type in their text messages or to read the prompted text-message. Subjects are recruited from Amazon Mechanical Turk.

In the first phase of the data collection, we have 66 scenarios with about 5K subjects typing in 40K text messages based on these scenarios. In the second phase, we have 2746 spoken messages from 238 subjects. These messages are transcribed using Google's Speech Recognition Engine. The recognizer's output contains a list of n-best hypotheses and confidence score associated with the 1-best hypothesis. Some of these utterances are discarded by the recognizer and only 2692 utterances are decoded. We observe that the discarded utterances are either badly recorded or contain heavy background noise. We also observe that such observations are specific to a particular set of subjects. In order to get a fair notion of the

[2]Since coming up a text message is cognitively-intense process for subjects, we separate this step from speaking the message

Table 4.1: Corpus Statistics

	#Utts	#Errors	Avg Errors/Utt	Avg Tokens/Utt	WER
Correct Utts	1028	-NA-	-	10.41	0.0%
Error Utts	1127	4472(27%)	3.96	14.59	24.62%
All Utts	2155	4472(16%)	2.07	13.38	15.48%

errors in this corpus, we discard sessions recorded by these subjects. After discarding those sessions, we are left with 2155 utterances to analyze.

4.2 Corpus Analysis

To get an insight into the errors in text messages, we first analyze the corpus. As shown in the Table 4.1, in this corpus, a little over 50% utterances have recognition errors. Most of the erroneous utterances are longer due to false-starts and insertion errors. On average there are 3.96 erroneous words per utterance in the utterances with errors. We did not observe a strong correlation (kendall's τ = -0.01) between length of the utterance and number of errors in an utterance.

First, we would like to know the error-breakdown by part-of-speech. We observe that majority of the tokens belong to open-class of words i.e., verbs, and nouns. We also find that errors occur in prepositions, determiners, and the other closed-class of words. Adjectives and adverbs belong to open class of words but with fewer errors. One explanation is that colloquial expressions typically use fewer modifiers when compared to written text. We would like to highlight that the ratio of errors:correct-instances is higher for verbs, nouns and proper names compared to closed-class words.

Second, we look at severity of the errors. Misrecognizing a closed-class word is typically less severe. Misrecognizing a sequence of words can change the meaning of a message. Also, function words can contain important temporal/locative information (e.g., in vs on) and those errors can affect the meaning of the message.

Third, we want to inspect number of these errors that are part of a continguous sequence of errors. Figure 4.1 shows length of the error sequence and its frequency in the corpus. It is obvious that there are a lot more isolated errors (length = 1), but it is interesting to note that there are almost as many errors in two-word error sequences as in isolated errors.

Sequential errors are distinct from isolated ones mainly because most of erroneous words are commonly found bigrams, trigrams in spoken language e.g., “you are”, “they are going” etc. Errors in such sequences may affect the meaning of the message.

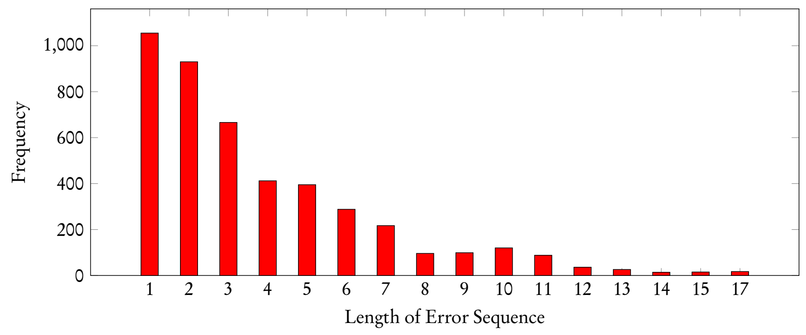


Figure 4.1: Length of sequence of errors vs frequency of errors

4.2.1 Major Errors and Minor Errors

Based on our pilot analysis of the corpus, we find that some errors impact the interpretation of a message than the others. To get an empirical estimate of the error severity, we annotate the errors in the 1-best hypotheses of the utterances. Our objective is to label isolated errors as MAJOR, or MINOR. If an error is part of a sequence of errors then all these errors are labeled as MAJOR errors.

We have following suggestions for the annotators:

- If the error leads to misunderstanding the message then it is a major error
- Otherwise it is a minor error

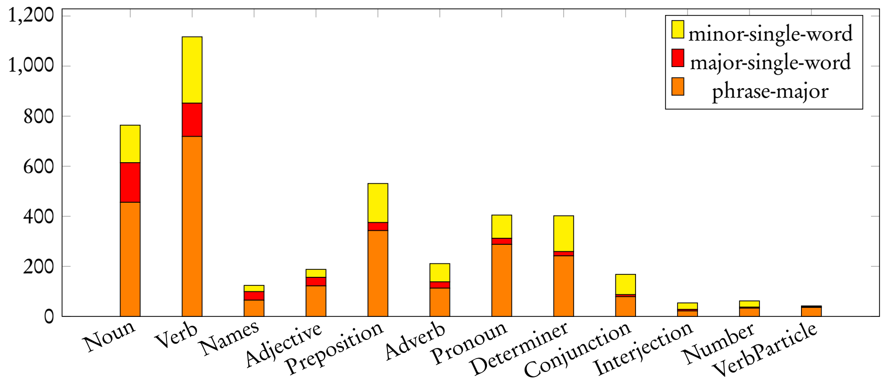
Example scenarios for the major and minor errors:

- If pronouns, prepositions, adjectives or adverbs doesn’t change the meaning, then the error is minor.
- If the error is due spelling-mistake (baby vs babie) or word-compounds (highway vs high way), then it is minor.

- If the error deletes or substitutes a content word like noun, verb or a proper name then it is major.
- If the error occurs in a sequence of errors then it is major.

We divide the erroneous utterances into three equal parts. Three annotators together have labeled the errors occurred in the 1-best hypotheses 100 of utterances. The inter-annotator agreement on these utterances has a high kappa score ($\varkappa = 0.74$) overall and on single-word errors the agreement is $\varkappa = 0.66$. In these 100 utterances, there are 208 errors in sequence and 195 (˜95%) of them are critical for understanding the message. Figure 4.2 shows that majority of the errors are part of a sequence of errors. This is in agreement with our observations from the pilot analysis. Most of the major errors are nouns, verbs and proper-names. Occassionally, some instances of closed-class words are labeled major because they affect the meaning of the utterance. For example, "your" instead of "our" changes the meaning of "when is your appointment". We observe that majority of the open-class word-errors i.e., nouns, verbs and proper-nouns are part of a phrase. We also observe deletions are mostly part of phrase errors. Therefore we need to treat the phrase errors as critical as the isolated ones. In the following two sections, we discuss our dialog policy to handle erroneous utterances and approach to detect major and minor errors in an utterance.

Figure 4.2: Major and Minor Errors Breakdown by POS



4.3 Dialog-Actions to Recover New Information

To recover information from "major" errors, we simulate dialog-actions based on the errors in the message. The defined dialog actions are naive [3], since the current focus is not error correction, but to improve error detection to accurately select an action. The policy is as follows:

- **Confirm:** Playback (sliced from user's speech) the error segment of the utterance and seek for correction, if there is a major error. Similar to (Prasad et al., 2012; Stoyanchev et al., 2013).
- **Repeat:** Discard the current ASR result and ask to repeat the message, if there are more than two major errors. This means that false alarm does not matter when there are more than two major errors that are correctly predicted.
- **Continue:** Confirm implicitly and send the message, if there are no errors or only minor errors.

Based on these actions and the annotated errors: Out of 1127 error utterances, 694 require **Confirm** action, 80 require **Repeat**, and 353 have only minor errors and do not require corrections (**Continue**). Note that, accurate action-selection of **Confirm** and **Repeat** is important in this task to avoid miscommunication of messages i.e., false acceptances. In speech-based applications, typically false acceptances are not desirable as they increase the interaction overhead (Komatani and Kawahara, 2000). Next, we discuss our approach to detect major and minor errors.

4.4 Multi-Level Error Detection

In the last section, we discussed the nature of the errors in the text messages and observed that some errors are more critical than the others. In this section, we present our approach to detect both major and minor errors in an utterance. We cast this problem as a sequence labeling problem where each word in an utterance is labeled as correct, major-error or minor error. Our goal is to detect as many errors as possible, particularly the major ones that are critical for understanding the text message.

[3] Ideally action selection is done considering the likelihood of error recovery.

Table 4.2: features used for error detection

Feature Type	feature
LEX	word itself word position BEG, MID, END
ASR	log posterior confidence duration of word (in sec) presence of ngram for current word 3gram,2gram,1gram,OOV number of alternative nodes in the confusion network
SYN	POS tag log posterior of POS confidence Chunk label for the word
SUB	presence of subword in time frame of hybrid decoder output

4.4.1 Method

To train an error detection model, we use off-the-shelf Linear-Chain Conditional-Random-Fields (CRF) toolkit (Kudo, 2013). We use lexical, acoustic, syntactic and other features to train this model. Table 4.2 gives an overview of features in this model. Previous work (Pincus et al., 2013) found that word position has predictive power to determine the errors. ASR features such as posterior probability and duration of the word-segment are standard features to determine errors.

In our case, the recognizer does not provide word-level confidence scores. Therefore, we estimate these scores through an alternative process. First, we force-align each of the hypotheses in the n-best list with the speech using Sphinx 3 (Placeway et al., 1997) and obtain the word-level time and acoustic model (AM) score. Then, we obtain language model (LM) scores by measuring the perplexity of the hypotheses over a trigram language model using SRI-LM toolkit (Stolcke, 2002). The language model is trained on a dataset of tweets [4]. Finally, we generate a confusion network from the n-best hypotheses along with the word-level AM and LM scores obtained in the previous steps using SRI-LM. The resultant confusion network contains word-level posterior probabilities. We use the log-posterior probabilites of words in the 1-best hypothesis as features in our model.

We also use presence of an ngram for the word in the language model as a feature. Since speech recognition is a generative process based on an

[4]We collected conversational tweets from 04/13 to 07/13. We exempt retweets and status tweets because they do not suit our domain

underlying distribution, every word in the recognition output can be either generated from a trigram, bigram or unigram with different probabilities. For example, if the candidate word in the output is "boy", it can be because the history (context before that word) was either "the young boy" (trigram), OR "young boy" (bigram), OR "boy" (unigram). Given a language model and a word output, we can generate this "categorical" feature for the CRF [unigram, bigram, trigram, oov]. We use "oov" category If the word doesn't exist in the language model.

Another important feature is *ratio of alternative nodes to a current node in a word confusion network*. Figure 4.3 shows a portion of a word confusion network. For word positions 1 and 3, there is only one candidate word and ratio would be 1/1. For word position 2 (highlighted in gray), there are 3 candidates and the ratio of nodes to that position would be 1/3.

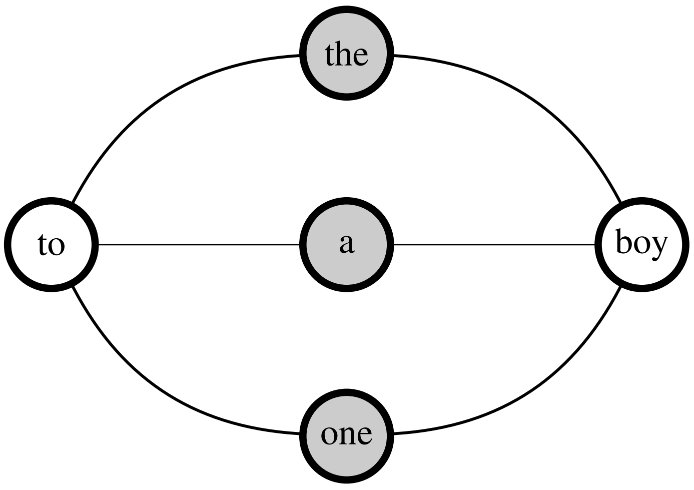


Figure 4.3: a snapshot of the confusion network

We also use part-of-speech tag as a feature. We use the twitter pos tagger (Owoputi et al., 2013) to obtain the pos-tags for each word in the 1-best hypothesis. In addition to pos-tag, we use the chunk-label (NP, VP, PP, ADJP etc.) associated with each word using OPEN-NLP chunker (OpenNLP). We believe that disfluent phrases lead to abrupt chunking when compared to fluent phrases, helping us discriminate between errors and correct words.

Utterances with major errors typically have open-class words and some of them are out-of-vocabulary(OOV) words for the recognizer. We detect their presence using a hybrid decoder (Qin and Rudnicky, 2012). A hybrid decoder is a standard speech recognition engine (e.g. sphinx) that takes a hybrid language model, hybrid dictionary and a standard acoustic model (e.g., WSJ acoustic model) as its input. Hybrid language model is a combination of subword language model and a conventional word language model, where subwords are phone sequences of variable length such as "AA_SH", "IY_SH", "P_AH_P" etc. Hybrid dictionary has pronunciations (represented as a phoneme sequence) for both regular words and subwords. Example entries in hybrid dictionary:

TEST	T EH S T
AA_SH	AA SH
JOHN	JH AA N
IY_SH	IY SH

We train a hybrid language model from a corpus of tweets with 84,867 tokens and 812,722 utterances. While training the hybrid model, we consider the nouns and proper-nouns (39,451 out of 84,867) appearing in tweets as OOV tokens. We use sphinx3 decoder with HUB4 acoustic model and the hybrid language model.

Figure 4.4 shows temporally aligned outputs of regular decoder ("hyp"), and hybrid decoder ("hybrid"). We also show truth ("ref") of the actual speech only for the reader's understanding. Within the time frame of a word in the regular output ("hyp"), we note whether it is aligned with a subword (e.g., AY*T*M*AY) in the "hybrid" output. If it is aligned then we assign "1" as the feature value of sub_feat, otherwise "0". In the example below, last three words of the "hyp" output (excluding the trailing silence) have subwords in their time frame. This approach is similar to a previous work (Burget et al., 2008) that uses a combination of multiple recognizers to improve the OOV detection.

4.4.2 Optimizing Error Detection

The error detection approach mentioned above is designed to optimize word-level prediction. We wish to optimize the error prediction to maximize the action selection performance. For example, the detection method should know that an utterance is rejected beyond a certain threshold of major errors, so classifying multiple major errors correctly does not im-

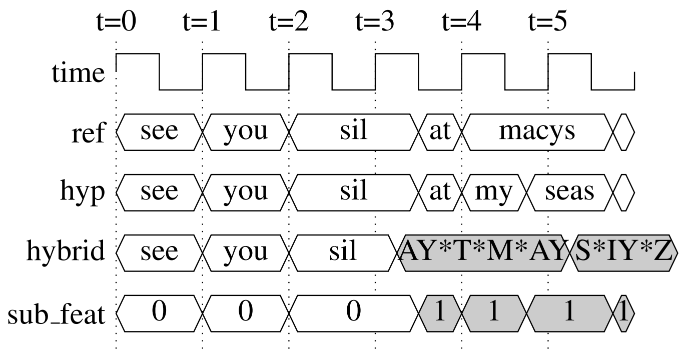


Figure 4.4: illustration of subword feature extraction for error detection

pact the action. Similarly action is same for correct utterance and minor errors, so it is not necessary to distinguish them to select the dialog-action.

We define this problem as re-scoring problem from N-best hypotheses of error prediction. The action-optimized approach is two-phased and illustrated in Figure 4.5.

(1) Optimization Phase: We apply metacosts (Domingos, 1999) to the detection output and optimize the costs with randomized grid-search that selects optimal action. Metacosts can turn any arbitrary error-based classifier into a cost-sensitive classifier and found to perform well in label-imbalance setting such as ours (Wu et al., 2008). We divide the data into validation data and testing data. Then we run the error detection method discussed earlier over the validation data to optimize the metacosts during gridsearch. We manually picked a range of costs based on our pilot experiments. We could derive these costs from data if we have had longer interactions instead of one-shot interaction. Our method, a CRF model, produces marginal probabilities for all the candidate labels for a token in the sequence. The 1-best output of the model may not be optimal for action-selection, therefore we apply metacost matrix to revise the labels in the sequence. The revised label is chosen as follows:

$$\text{revised}_t = \arg\max(\text{cost}_{t,i} \times P_i). \qquad (4.1)$$

Where t is the predicted label of a word,
$i \in [\text{Correct}, \text{Major}, \text{Minor}]$, $cost$ is $n \times n$, metacost matrix (n is number of labels) and P_i is marginal probability of a candidate label. Once we revise the label sequence, we repeat this process for the n-best label sequences

produced by the model. Then, we apply ROVER (Fiscus, 1997), a voting algorithm on the revised error sequences. ROVER was found to be useful in minimizing the error in ASR, thus we apply it to the revised error sequences to get a voted error sequence.

ROVER uses a two step approach. In the first step, it takes multiple label sequences as its input and combines them into a single label transition network. This network is created by running a dynamic-time warping algorithm to align multiple label sequences. In the second step, this network is evaluated for candidate labels at each word position with respect to a voting criterion (e.g., confidence score of a label). Voting criterion determines which label should be selected at any given position in the sequence.

- By frequency: select most frequent label across different sequences as the best label for that word position.
- By average confidence score: Rank the candidate labels by taking the average of their confidence scores across sequences.
- By maximum confidence score: Just pick the label that got highest confidence score in any sequence for that word position.

In our experience, we found average confidence score often produces optimal output and we have used that as our voting criterion in all our reported results. Based on the voted error sequence (obtained from ROVER) and the dialog policy, we choose the dialog action, then evaluate if it matches the action based on the true-error-sequence. For randomized grid search (Bergstra and Bengio, 2012), we use the implementation available in scikit-learn (Pedregosa et al., 2011).

(2) Evaluation Phase: We apply the optimized costs to the error detection output. We detect errors in a test utterance, apply the metacosts on the nbest-list of error sequences, pass the list through ROVER and obtain voted error sequence to select the target dialog-action for the utterance.

4.5 Detection Experiments

Our objective is to evaluate how these features fare in major error detection, i.e., accurately label whether a word is "correct", "major-error" or "minor-error". We have two baselines for critical error detection (a) MAJORITY baseline: simply labels every word as "correct". (b) RULE BASED

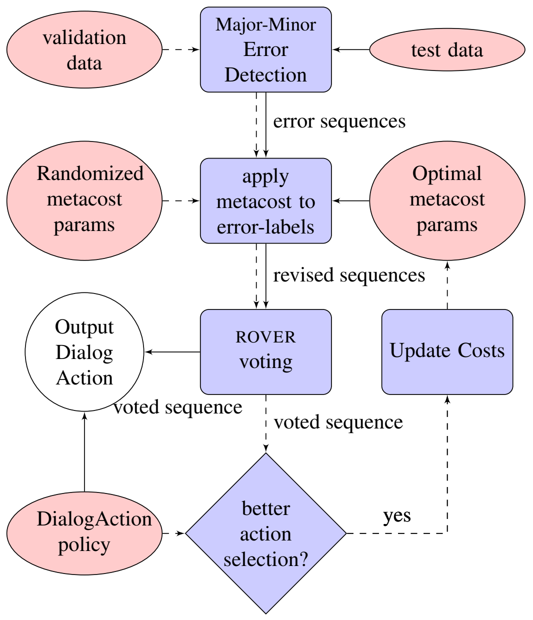


Figure 4.5: Workflow for Optimizing the Detection to Dialog-Action. Bold lines show evaluation phase and dotted lines shows the metacost-optimization.

major/minor error detection: first we measure ASR confidence scores for each word (estimated as mentioned in the Method section), label words as errors below a confidence threshold and then based on their POS tag all the nouns, proper-nouns, verbs are tagged as major errors otherwise minor. We compute the confidence threshold in a 10-fold cross validation setting. The threshold is measured as the average confidence of the correct words in the training data of that fold.

Deletion errors are often major and they are distinct from other major errors because there is no superficial evidence for the deleted words in a hypothesis. This leads to absence of token-related features such as pos-tag, chunk-label etc. Therefore, we handle deletions separately and the results are presented separately for the deletion errors.

4.5.1 Experiment Setup

We normalize the colloquial expressions (e.g., *I've* to *I have*) to their standardized version in the corpus. We extract features described above for each utterance in the corpus. We use these features to train a CRF model.

Since CRF models only allow binary features we postprocess the real-valued features by binning them into intervals. We bin them into 10 bins from 0.1 to 1.0. For example if the feature value is 0.4 then the feature vector looks like [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]. We perform 10-fold cross validation on the entire corpus of 2155 utterances.

4.6 Detection Results

We compare the baseline models discussed above against the sequence labeling models with features. Table 4.3 shows that the proposed model (ALLFEATS) in word-level classification of errors significantly ($p < 0.05$) outperforms both the baselines, MAJORITY and RULE BASED by 1.87% and 1.53% respectively. We observe from Table 4.4 that individual feature models do well in a particular class. Although we expected the SUB feature model to contribute to major error detection, the results show otherwise. One explanation to this result could be that we have used it as a binary feature instead of a real-valued feature. We have further analyzed the features and their predictive power in the next section.

We observe that consistent low-recall is due to the fact that the corpus is heavily imbalanced towards the correct-labels (only 16% of the words have errors in them). Up-sampling the corpus could improve the recall. Since our focus is to investigate the features that are promising for the detection task rather than handling the imbalance in the dataset, we will leave up-sampling for future work.

Table 4.3: Word level performance of the Multi-Level Error Detection Model against the Baselines in-terms of Precision(%), Recall(%) and F-score(%)

Class	MAJORITY			RULE BASED			ALLFEATS		
	P	R	F	P	R	F	P	R	F
Cor	91.03	100	95.30	93.32	91.71	92.50	92.66	98.07	95.28
Maj	0.00	0.00	0.00	21.18	21.18	21.18	40.15	16.83	**23.42**
Min	0.00	0.00	0.00	9.38	18.22	12.34	33.06	10.33	**15.54**
All	82.61	90.89	86.56	88.03	85.97	86.90	87.39	90.52	**88.43**

Detecting the type of error in an utterance is important for the dialog strategy. Therefore, we evaluate our model at the utterance level to see if

Table 4.4: Word level performance of the Isolated Feature Models in terms of Precision(%), Recall(%) and F-score(%)

Class	LEX			ASR			SYN			SUB		
	P	R	F	P	R	F	P	R	F	P	R	F
Cor	91.98	98.67	95.21	91.69	99.53	**95.44**	91.35	99.93	**95.44**	91.03	100.00	95.30
Maj	28.93	6.79	10.86	51.34	8.19	**13.82**	60.58	1.82	3.52	0.00	0.00	0.00
Min	33.88	8.83	**13.94**	0.00	0.00	0.00	14	0.23	0.46	0.00	0.00	0.00
All	86.82	90.71	88.03	86.77	91.27	87.99	87.26	91.35	87.53	82.61	90.89	86.56

Table 4.5: Utterance level Accuracy (in %) of Baselines and the Multi-level Error Detection Model

Class	MAJORITY	RULE BASED	ALLFEATS
Correct	100	100	**99.91**
Major	0	22.62	**26.23**
Minor	0	15.19	**15.92**
All	53.45	62.87	**64.1**

it can predict if an utterance has at least one "major error", only a "minor error" or none of them ("correct"). Table 4.5 shows that the proposed model can make a prediction about error occurrence with an overall accuracy of 64.1%.

As mentioned earlier, we train a separate model for deletion detection. The force-align algorithm inserts silence breaks in between the words and we use these breaks as surface tokens to train deletion detection model. While training the deletion detection model we use the reference-hypothesis alignment output to these label silence markers as deletion errors. We collapse sequence of deletions as a single deletion error. Although several features are absent for the silence segment, yet we have the context surrounding this segment to help the model predict deletion errors. We use all set of features that we described in the previous section. We compare this model against two baselines (a) MAJORITY baseline: labels every token as "correct" and (b) HEURISTIC baseline: labels a silence segment as deletion when the segment duration is greater than 0.39. This threshold is equal to average of the duration of all the silence segments in the corpus. The proposed ALLFEATS model significantly outperforms both baselines.

From Table 4.7 we see that 3-way method performs 18% better than baselines selecting actions (from 50.0/47.4 to 59.2). We believe that 3-WAY LABEL model can be optimized to improve on correcting major errors. In the subsequent sections, we present the action-optimized version.

4.7 Error Analysis of Error Detection

In a sequence labeling method, errors are often predicted in the context where features interact with each other. Some features, however, have bet-

Table 4.6: Deletions detection results in terms of Precision, Recall and F-score

Class	MAJORITY			HEURISTIC			ALLFEATS		
	P	R	F	P	R	F	P	R	F
Corr	91.03	100	95.30	99.37	95.71	**97.50**	92.84	97.85	95.27
Del	0.0	0.0	0.0	10.59	45.43	17.17	53.84	29.26	**37.51**
All	82.99	91.09	86.85	83.06	88.47	85.62	88.91	90.9	**89.48**

Table 4.7: Sentence level Action-Selection F-score(%) of the Proposed Model (3-WAY LABEL) against the Baselines. Our approach has 59.2% F1 score for action selection, significantly better than the baselines.

Action	MAJORITY	RULE BASED	3-WAY LABEL
Continue	78.1	65.0	80.2
Confirm	0.0	17.3	**22.4**
Repeat	0.0	2.2	**16.3**
All	50.0	47.4	59.2

ter predictive power in isolation. We inspected these features by measuring the χ^2 score (Chi-squared goodness of fit test) in different classification settings. Table 4.8 shows the features ranked by their scores in different classification settings. First row shows that the errors (irrespective of severity) are better discriminated from correct words by confusion-net feature, ASR confidence score, followed by other syntactic features. We observe that some features are better in context (e.g., subword) than in isolation. We observe similar trend in the second row of Table 4.8. Deletion errors are well discriminated by the position of the word segment, ASR confidence, duration, and subword features. This shows a striking contrast between the tasks of detecting deletion errors and detecting other critical errors.

We have further analyzed the predictions made by the model in comparison to the marginal probability associated with each prediction. Figure 4.6(a) shows the density of predictions against the intervals of margin probability associated with the prediction. Each line plot is associated with the

Table 4.8: Predictive power (measured in χ^2 test score) of isolated features in different settings

Type	1	2	3	4	5	6
Cor vs Err	confusion (2407)	asr-conf (2325)	ngram (735)	pos (585)	pos-conf (325)	chunk-label (140)
Cor vs Maj vs Min	confusion (2493)	asr-conf (2344)	pos-tag (2344)	ngram (771)	pos-conf (326)	chunk-label (168)
Cor vs Del	position (72.26)	asr-conf (64.55)	duration (46.19)	subword (14.31)	pos-tag (0.0)	pos-conf (0.0)

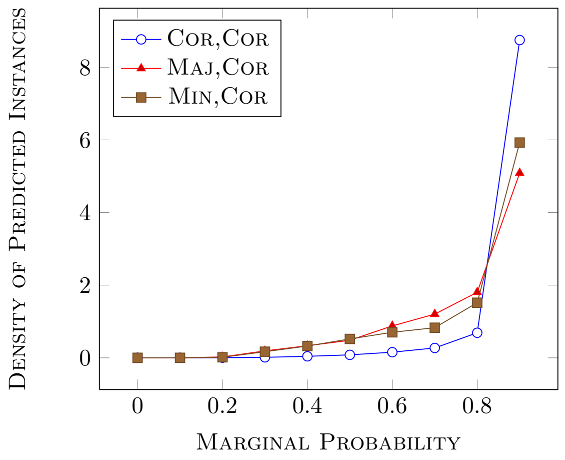
actual label when the predicted label is "correct". A similar trend can be noticed with respect to deletions Figure 4.6(b). We understand when the model predicts the word as correct with a probability less than 0.9, then its more likely to be a major or minor error. This reinforces the notion that the model is highly biased towards correct label and raises the need for error optimization.

To understand how the error detection performance varies by POS, we brokedown the F1-score of our proposed model by POS tag. This is presented in Figure 4.7. We see that the error detection is particularly good for Names, followed by adjectives, numbers and nouns. One explanation for this result is that open-class words are relatively more frequent class of words in a corpus compared to closed-class of words. Therefore it makes it easier for the error detection module to spot them if they ever occur in wrong context.

4.8 Optimized Detection Experiments

4.8.1 Optimized Detection Setup

We want to optimize the error detection process to improve action selection. This helps us to minimize the false-alarms by clarifying major errors. We compare optimized action selection results (VOTED ACTION OPTIMIZED) with the 3-WAY LABEL. We use 4-best error sequences throughout the experiments based on pilot experiments on a development set.



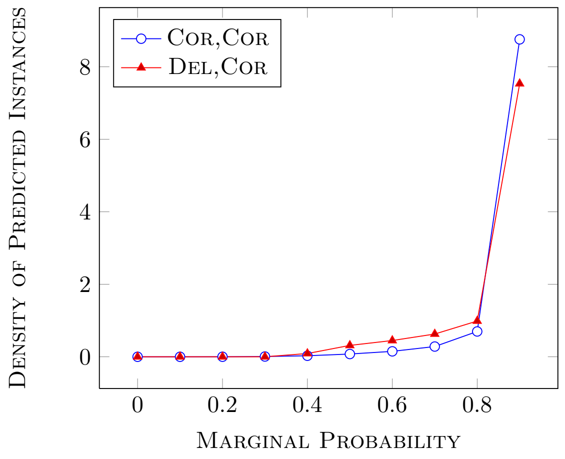
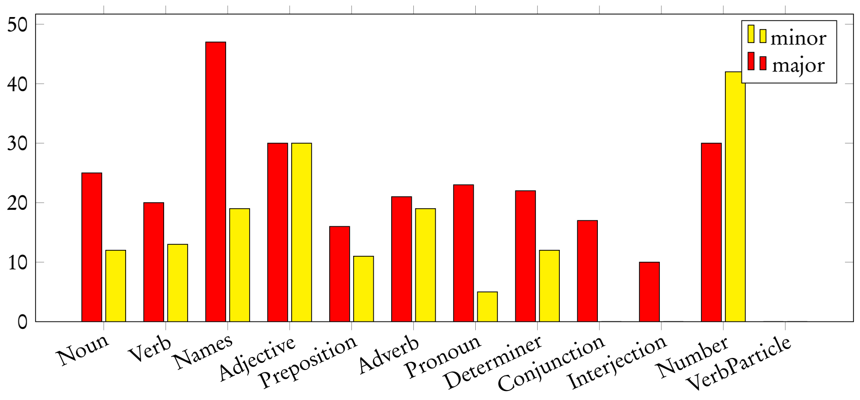


Figure 4.6: Histogram of the (actual, predicted) tuples against the marginal probability of the prediction

4.8.2 Optimized Detection Results

Table 4.9 shows significant improvement ($p < 0.01$) in predicting corrective-actions (Confirm and Repeat), when we use VOTED ACTION OPTIMIZED model without degrading overall performance measured by F1 score. The optimized model improves the selection of corrective-actions by weighted average of 31% (25% for Confirm and 85% for Repeat). We improve recall for error-correction, which is important in label imbalance tasks e.g., anomaly detection (Akoglu et al., 2012).

Figure 4.7: F1 score of Major and Minor Errors Breakdown by POS



Strategy	3-WAY LABEL			3-WAY VOTED ACTION OPTIMIZED		
	P	R	F	P	R	F
Continue	72.4	89.8	80.2	78.4	78.0	78.2
Confirm	37.6	15.9	**22.4**	33.5	24.3	**28.2**
Repeat	44.4	10.0	**16.3**	38.4	25.0	**30.3**
All	60.2	58.2	**59.2**	62.3	58.0	**60.1**

Table 4.9: Sentence level Action-Selection performance Action-Optimized Model (VOTED ACTION OPTIMIZED) against the unoptimized ones in terms of Precision, Recall and F-score. Optimization improves F1 for corrective actions by 31% (25% for confirm and 85% for repeat.)

We list the word-level precision and recall in Table 4.10 to investigate the reasons for this improvement. Recall of our method 3-WAY VOTED ACTION OPTIMIZED for major and minor is significantly higher than that of 3-WAY. This confirms that optimized weight lead to aggressive error detection because false alarm for "Repeat" is tolerated if at least two predicted errors are correct. When the predicted label is "Correct" an example of optimized costs is [1.1, 3.2, 2.1] for [Correct, Major, Minor] labels. This shows that missing major errors is more critical than missing minor errors. We believe that combination of voting and metacosts boosts the performance of the optimized model.

In summary, VOTED ACTION OPTIMIZED model improves the action

Class	3-WAY LABEL			3-WAY VOTED ACTION OPTIMIZED		
	P	R	F	P	R	F
Cor	92.6	98.0	95.2	94.3	95.4	94.8
Maj	40.1	16.8	23.4	28.8	27.9	28.3
Min	33.0	10.3	15.5	24.0	15.7	19.0
All	87.4	90.5	**88.4**	88.9	89.7	**89.3**

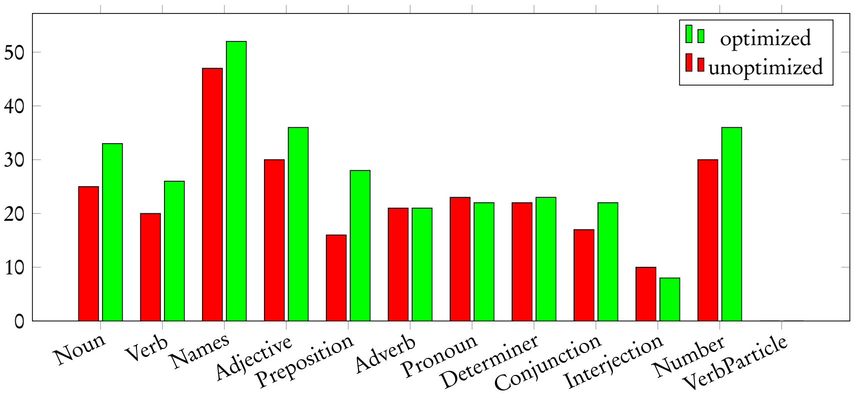
Table 4.10: Word level performance of Action-Optimized Model (VOTED ACTION OPTIMIZED) against the unoptimized ones in terms of Precision, Recall and F-score. Optimization improves error detection by an additional 1% (88.4 to 89.3)

selection by 1.5% (59.2 to 60.1) and error detection by an additional 1% (88.4 to 89.3), bringing the overall improvement in error detection to 2.7% (86.9 to 89.3).

Although our objective is to improve the performance on corrective-actions, Table 4.10 shows that the optimized model has also significantly ($p < 0.01$) improved at detecting major errors by 5% (from 23.4 to 28.3), minor errors by 4% (from 15.5 to 19.0) and overall 1% (from 88.4 to 89.3).

We have analyzed the major error detection result by breaking them down by their POS. In Figure 4.8, we see that our optimized approach consistently outperforms the unoptimized version across all the POS tags.

Figure 4.8: Breakdown by POS: F1 score of Major Error Prediction by unoptimized vs optimized



4.9 Interactive Detection and Recovery

Our goal is to develop an error detection and recovery application. Figure 4.9 shows example dialog of a system with the user to recover from errors in spoken short messages. Our application is based on a web-based dialog architecture that we have developed on top of Freeswitch [5].

4.9.1 Error Detection Module

This module implements the ideas discussed earlier in this chapter. We extract various features in this module. An example feature vector looks as follows. This feature vector is an input to the CRF module that performs error detection.

```
example_feature_dict = {
'hypothesis': 'good morning <sil> this is a test <sil>',
'confusion_net': '0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5',
'pos_tag_confidence': '0.9954 0.9947 1.0 0.9529 0.9983 0.9985 0.9989 1.0',
'begin_timestamp': '0.39 0.55 0.96 0.99 1.28 1.39 1.47 1.98',
'end_timestamp': '0.54 0.95 0.98 1.27 1.38 1.46 1.97 2.9',
'pos_tag': 'A N sil O V D N sil',
'ngram': '2gram 3gram sil 1gram 2gram 3gram 3gram sil',
'subword_feature': '0 0 0 0 0 0 0 1',
'asr_confidence': '1 1 1 1 1 1 1 0.99923',
'word_position': 'BEG BEG BEG MID MID END END END',
'duration': '0.15 0.4 0.02 0.28 0.1 0.07 0.5 0.92'
}
```

4.9.2 Error Recovery Module

We select a dialog action based on the error detection output as discussed earlier in the chapter. For *confirm* dialog action, we need to slice the audio corresponding to the error region and play it back to the user in the step 3 of dialog shown in Figure 4.9. To do that we need the begin and end timestamps of the error region. Then we use sox to slice the audio in the error region.

We want to fix the error region using the output from new fragment of user speech. In step 4 user repeats the error region and the speech

[5] http://www.freeswitch.org

gets transcribed. Then we take the n-best list produced by the ASR and search for phonetically closest hypothesis to the target error region. E.g., nbest list: *[madras, address, madras cafe]* and target error region: *mad address cafe*. In the example, third item in the nbest list is the closest match. To obtain closest match, we convert the n-best list into their phonetic equivalents using a g2p program [6]. Then compute levenshtein distance between phonetic transcriptions of the target error region and each item on the nbest-list. Then, system replaces the new candidate in the original utterance and confirms with the user if the updated message is correct.

We built this dialog system based on our web-based dialog architecture. We describe this architecture in the following section.

Error Clarification in SMS domain

S[1]: Speak your message now
U[2]: the applying to me [we are planning to meet] to [at] Madras Cafe
S[3]: please repeat the phrase <....>
U[4]: planning to meet
S[5]: did you say the planning to meet madras cafe
U[6]: yes
S[7]: sending the message "the planning to meet madras cafe"

Figure 4.9: Example Dialog for Error Clarification in SMS domain. Note: text in [ ] shows inline ground truth

4.9.3 Web-Based Dialog Architecture

Freeswitch is the backbone of this architecture. It supports VOIP, SIP based calling, call routing, and simultenous calls to a freeswitch application and provides boilerplate code to write an application in many of the popular computer languages. This allows easy integration of existing code as library calls from the application code. Figure 4.10 shows the flow chart of the architecture. A call is initiated via front-end such as, phone-line, SIP, web-based RTMP connection or VOIP. Then the audio

[6]http://www-i6.informatik.rwth-aachen.de/web/Software/g2p.html

from the user's channel is streamed to a freeswitch server and recorded to a unix named pipe. A voice-activity-detection component reads the audio stream from the named-pipe to detect speech. This component uses Sphinx's VAD algorithm (available in sphinxbase library [7] to segment the audio into speech and silence chunks. When a speech segment is detected, it is sent to an automatic speech recognizer (ASR) engine via TCP socket. Then the recognition hypothesis is sent to dialog application. The dialog application processes the hypothesis and sends a response back to the user via text-to-speech engine and the application front-end (e.g., GUI). In the current configuration, two speech recognizers are plugged into the system — a local recognizer, Pocketsphinx (Huggins-Daines et al., 2006) and a commercial cloud-based recognizer. Any recognition engine can be incorporated (given an appropriate interface), using a simple TCP connection, allowing the use of different recognition schemes. Additionally, developers are able to evaluate different (acoustic and language) modelsin the context of the same dialog application. The source code for VAD, ASR interfaces (C/C++) and example configuration for a freeswitch application is available as tar ball [8].

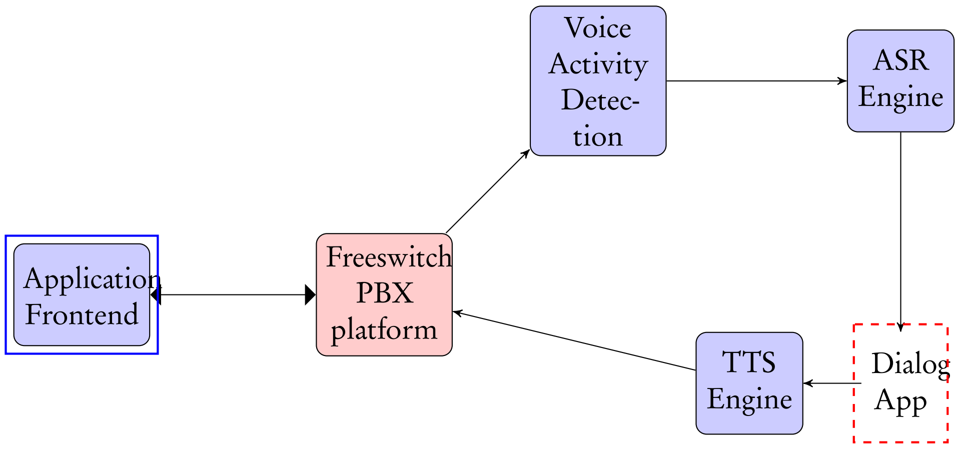


Figure 4.10: Architecture Flowchart

[7]http://sourceforge.net/projects/cmusphinx/files/sphinxbase
[8]http://www.speech.cs.cmu.edu/apappu/pubdl/fs-dialog-interface.tar.gz

4.10 Recovering OOVs in Speech-To-Speech MT

Besides spoken short messages domain, we have conducted preliminary experiments to detect and recover new information in the context of a speech-to-speech translation system. Our task is to recognize a user's English utterance, detect potential errors in the recognition, report to the user and finally translate corrected recognition into target language (Iraqi). The dialog system reports to the user about the low-confident region and asks them to spell the OOV phrase, rephrase their original sentence or inform if they are satisfied with recognition. Six users participated in this experiment, total 81 dialog sessions were conducted with 179 utterances spoken by the users. Some of these utterances are clarification replies to the dialog system. In this experiment, we are only interested in the utterances intended for translation because we want to know whether confidence scores are useful features to determine OOV regions in such utterances. After the user speaks, there can be more than one possible action as shown in Table 4.11. If the user is not satisfied with the response they can restart the session manually otherwise the system can either approve the recognition or pose a clarification to the user.

Table 4.11: Possible actions after user utterances

Action on User utterances	Frequency
System requests correction	68
System approves utterance	23
User suspends session	8
Total domain utterances	99

In response to system's clarification question, the user can take one of the actions listed in Table 4.12. Remember, we are interested in finding out what is the actual WER of utterances that the system predicted as low confident input. Table 4.13 shows that the difference between the WER of user approved and that of system approved utterances is quite small. In contrast, utterances that required clarification have relatively higher WER. Therefore, it can be understood that confidence scores obtained from a ROVER mechanism can be reliable for detecting badly recognized inputs, in turn potential OOV regions.

Table 4.12: User Action on System's Clarification

User Action	Frequency
suspend the session	9
spells OOV phrase	22
rephrased	22
approved	15
Clarification Requested	68

Table 4.13: User Action on System's Clarification

Utterance type	WER %
clarification requested	43.4
system approved	29.7
user approved	27.1

4.10.1 *OOV* Learning Strategy

We developed OOV learning this system. This strategy requires the speech recognizer to use a spelling language model. We evaluated this language model on two datasets, `eval1` and `eval2`. These two datasets were actual dialog sessions with the end-users using a dialog system for speech-to-speech machine translation. The spelling language model is trained using a list of 37K words available in the recognizer's vocabulary. Table 4.14 shows that the spelling recognition used with the OOV strategy is accurate on both datasets. We found that higher WER on the second dataset (`eval2`) is because of badly segmented speech utterances.

4.10.2 Category Learning Strategy

We developed a category learning strategy for newly acquired word in the same system as discussed above. After the system acquires the spelling for the new word, it asks the user if word belongs to a particular category and lists a set of categories as options to the user. This is another popular strategy to acquire category labels for a word. On both `eval1` (72 category

Table 4.14: WER for the spelling recognition of OOV words

DataSet	Number of Utterances	WER
`eval1`	88	8.7%
`eval2`	22	16.3%

state utterances) and `eval2` (11 category state utterances) datasets, we had 100% accuracy at recognizing user's input for the target category. Figure 4.11 shows an example dialog of the system while it handles the OOV errors.

OOV Dialog

U: I'm looking for big [KAHUNA]
S: I'm looking for big. Either spell <> in the NATO alphabet. say Rephrase or say Continue
U: Kilo Bravo[Alpha] Hotel Uniform November Alpha
S: Is KBHUNA, person, location or other?
U: person
S: Adding KBHUNA and translating
S: Translating: "I'm looking for big @kbhuna@"

Figure 4.11: Example Dialog for Error Clarification in S2SMT. Note: text in [] shows inline ground truth

An alternative approach to category learning is to ask the user to suggest a category word/phrase to label newly acquired word. For example, system asks the user to give a synonym for the newly acquired word. Then the synonym word/phrase can be associated with the newly acquired word. We have used this strategy but it comes with obvious limitations. There were only 7 instances of this strategy overall and the WER is 77%. There are couple of reasons for this bad performance. Firstly, it assumes that synonym word or phrase is part of the recognition vocabulary which is not necessarily true. Secondly, it assumes that the user can come up with a simple and generic label. This assumption can fail if the newly acquired word cannot be described in one word or the user cannot think of it at that moment.

4.11 Discussion

Early work on error detection (Zhang and Rudnicky, 2001; Shi and Zhou, 2005) showed that combining multiple features, mitigates the drawbacks of individual features. Our approach uses lexical, syntactic, ASR and subword features to detect errors. Bechet and Favre (2013); Pincus et al. (2013) found that error detection in speech-to-speech translation can benefit from lexical and syntactic features. In news transcribing domain, dictation setting, Dufour et al. (2012) applied the sequence labeling approach using a CRF. The main difference between our work and their work is that we revise the error sequence by applying metacosts, improving the accuracy of error detection and the accuracy of action-selection. Shichiri et al. (2008) proposed minimum bayes-risk decoding as an alternative to standard decoding approach to better recognize the "significant" words in the spoken query. In this work, we show that assigning different costs to the errors and optimizing them, can improve the error detection accuracy.

Our approach is similar to decision-theoritic work on dialog planning and grounding problems. For example Wu (1991) has proposed a decision theoritic approach to generate navigation plans to points-of-interest. They have used multi-attribute utility metric to generate suitable plan. These attributes are based on difficulty of plan execution, degree of conflict with other goals, over specificness of the plan, productiveness and success likelihood of the plan. In our case, we do not directly use any of these attributes because metacost matrix is based on the risk of missing the major errors in the detection process. We could include success and difficultness attributes if we have realistic estimates from several live dialogs. Difficultness attribute measures the effort or time required to have the user help the system recover from errors. Success attribute would measure how likely is that a particular type of error can be recovered. If there are discontinuous errors then we could estimate which errors are recoverable.

Grounding is the problem of finding a suitable interpretation of a natural language instruction and converting it a system action. Some instructions can be ambiguous and may have different executional costs associated with different interpretations. Therefore systems employ a decision-theoritic approach to select a suitable action. Paek and Horvitz (2003); Skantze (2007b) have shown that grounding decisions can be made cost-effectively, both based on live interactions and collected dialog data. In (Skantze, 2007b), they have used different sets of handcrafted confidence thresholds for making grounding decisions, then find optimal thresholds based on user studies. In our work, we use different set of metacosts and determine optimal costs based on grid-search on the held-out data.

Decision-theoritic approach has been applied to other fundamental problems in conversational systems such as turn-taking. For example Raux and Eskenazi (2009) has applied cost matrix to floor-transition / turn-taking problem. They have presented a finite state turn-taking approach with non-deterministic transitions between the states. The cost matrix represents risk associated with actions that the system can take in each state. In our case metacost matrix represents risk associated with different label candidates with respect to top candidate label.

Potential future direction to our work would be deriving metacost matrix from scratch instead of optimizing over different sets of cost values. We could also reformulate this into a multi-attribute cost/utility problem by factoring in difficultness and success likelihood attributes.

Although dialogs are sequential in nature we did not consider POMDP based approach in this work, since we are primarily dealing with one-shot interactions in almost-open-domain scenario. Moreover, there are no finite set of concept-value pairs in this domain. This makes policy learning less feasible in a POMDP setting, that is otherwise feasible in standard spoken dialog systems (Thomson and Young, 2010; Liang et al., 2013). However, we could apply reinforcement learning (RL), if we could restrict the concept-value space to certain context e.g., people names in the sender's address book or activities/events related to sender's location (in the SMS domain). Daubigney et al. (2011) have shown that Gaussian Processes with RL can help in policy-learning for large-scale dialog systems. In a POMDP setting, it is possible to have different rewards for different actions in a state. Similarly, we can derive different metacosts for *Repeat*, *Confirm* and *Continue* actions by replacing the optimization function i.e., average F1-score metric with weighted average F1-score metric.

4.12 Chapter Summary

In this chapter, we presented an approach to detect critical errors and learn new information from those errors. First we presented an analysis on errors that occur in spoken SMS messages and proposed an approach to detect these errors. We found that some errors are more important than the others. We observed that major errors often occur in a sequence of errors, and they also occur in content words such as verbs, nouns and proper-nouns.

In order to detect the errors, we use a CRF model with lexical, syntactic, ASR and subword features. Our method shows good improvements over a majority-baseline and a rule-based baseline. Our analysis shows that although features in-combination give best results, some features have better predictive power than others in insolation. Features such as ASR posterior score, confusion-score, ngram-label and part-of-speech are better at discriminating between non-deletion errors and correct words. However, word position, ASR posterior score, segment duration, and subword can discriminate well between deletion errors and correct words.

Since major errors cause misinterpretation in text messages, it is useful to optimize error detection to maximize major-error correction. We achieve this by revising the error-sequence produced by the error detection model. We use metacosts to revise the error-sequence and optimize these costs using gridsearch. We show that our optimized model improves the selection of corrective-actions by weighted average of 31% (25% for Confirm and 85% for Repeat) and the overall error detection is improved by 2.7% (86.9% to 89.3%) over the rule-based baseline. We have also shown that we can recover OOV words in speech-to-speech translation applications to help translate a sentence accurately.

The focus of this chapter was restricted to learning information through recognition errors. However, errors could occur at semantic level as well. Sometimes entities are part of the recognition vocabulary but they are semantically unknown to the agent. In such situations, the agent may fail to understand the user goal, since many agents typically use a semantic parser to understand user goals. In the next chapter we will focus on this issue.

Chapter 5

Learning Through Open-Domain Semantic Knowledge Bases

Spoken dialog agents are designed with particular tasks in mind. These agents could provide information or make reservations, or other such tasks. Many dialog agents often can perform multiple tasks: think of a customer service kiosk system at a bank. The system has to decide which task it has to perform by talking to its user. This problem of identifying what to do based on what a user has said is called task prediction. Consider the following example:

PHRASE-1: go to the table
PARSE-1: [GotoLocation] (go to the [location] table)

PHRASE-2: go to the desk
PARSE-2: NO PARSE

Both Phrase-1 and Phrase-2 refer to same task "[GotoLocation]". However, system understands Phrase-1 but fails to understand Phrase-2. This happens because task prediction is typically framed as a parsing problem: A grammar is written to semantically parse the input utterance from users, and these semantic labels in combination are used to decide what the intended task is. However, this method is less robust to errors in user-input. A dialog system consists of a pipeline of cascaded modules, such as speech recognition, parsing, dialog management. Any errors made by these modules propogate and accumulate through the pipeline. Bohus and Rudnicky (2005c) have shown that this cascade of errors, coupled with users employing out-of-grammar phrases results in many "non-understanding" and "misunderstanding" errors.

There have been other approaches to perform dialog task prediction. Gorin et al. (1997) has proposed a salience-phrase detection technique that maps phrases to their corresponding tasks. Chu-Carroll and Carpenter (1999) casted the task detection as an information retrieval — detect tasks by measuring the distance between the query vector and representative text for each task. Bui (2003) and Blaylock and Allen (2006) have cast it as a hierarchical sequence labeling problem using Hidden Markov Models (HMM). More recently, Bangalore and Stent (2009) built an incremental parser that gradually determines the task based on the incoming dialog utterances. Chen and Mooney (2010) have developed a route instructions frame parser to determine the task in the context of a mobile dialog robot. These approaches mainly use local features such as dialog context, speech features and grammar-based-semantic features to determine the task. However grammar-based-semantic features would be insufficient if an utterance uses semantically similar phrases that are not in the system's domain or semantics. If the system could explore semantic information beyond the scope of its local knowledge and use external knowledge sources then they will help improve the task prediction.

Cristianini et al. (2002); Wang and Domeniconi (2008); Moschitti (2009) found that open-domain semantic knowledge resources are useful for text classification problems. Their success in limited data scenario is an attractive prospect, since most dialog agents operate in scarce training data scenarios. Bloehdorn et al. (2006) has proposed a semantic smoothing kernel based approach for text classification. The intuition behind their approach is that terms (particularly content words) of two similar sentences or documents share superconcepts (e.g., hypernyms) in a knowledge base. Semantic Similarity between two terms can be computed using different metrics (Pedersen et al., 2004) based on resources like WordNet.

Open domain resources such as world-wide-web, had been used in the context of speech recognition. Misu and Kawahara (2006) and Creutz et al. (2009) used web-texts to improve the language models for speech recognition in a target domain. They have used a dialog corpus in order to query relevant web-texts to build the target domain models. Although Araki (2012) did not conduct empirical experiments, yet they have presented an interesting architecture that exploits an open-domain resource like Freebase.com to build spoken dialog systems.

In this work, we have framed the task prediction problem as a classification problem. We use the user's utterances to extract lexical semantic features and classify it into being one of the many tasks the system was designed to perform. We harness the power of semantic knowledge bases by bootstraping an utterance with semantic concepts related to the tokens in the utterance. The semantic distance/similarity between concepts in the knowledge base is incorporated into the model using a kernel. We show that our approach improves the task prediction accuracy over a grammar-based approach on two spoken corpora (1) Navagati (Pappu and Rudnicky, 2012): a corpus of spoken route instructions, and (2) Roomline (Bohus, 2003): a corpus of spoken dialog sessions in room-reservation domain.

This chapter is organized as following: First we describe the problem of dialog task prediction and the standard grammar based approach to predict the dialog task. Then, we describe the open-domain knowledge resources that were used in our approach and their advantages/disadvantages. We will discuss our semantic kernel based approach after that. Finally, We report our experiment results on task prediction and we will analyze the errors that occur in our approach, followed by concluding remarks and possible directions to this work.

5.1 Parser based Dialog Task Prediction

In a dialog system, there are two functions of a semantic grammar — encode linguistic constructs used during the interactions and represent the domain knowledge in-terms of concepts and their instances. Table 5.1 illustrates the tasks and the concepts used in a navigation domain grammar. The linguistic constructions help the parser to segment an utterance into meaningful chunks. The domain knowledge helps in labeling the tokens/phrases with concepts. The parser uses the labeled tokens and the chunked form of the utterance, to classify the utterance into one of the tasks.

The dialog agent uses the root node of a parser output as the task. Figure 5.1 illustrates a semantic parser output for a fictitious utterance in the navigation domain. The dialog manager would consider the utterance as an “Imperative” for this example.

Table 5.1: Tasks and Concepts in Grammar

Tasks	**Examples**
Imperative	GoToPlace, Turn, etc
Advisory Instructions	You_Will_See_Location
Grounding Instructions	You_are_at_Location
Concepts	**Examples**
Locations	buildings, other landmarks
Adjectives-of-Locations	large, open, black, small etc.
Pathways	hallway, corridor, bridge, etc.
LiftingDevice	elevator, staircase, etc.
Spatial Relations	behind, above, on left, etc.
Numbers	turn-angles, distance, etc.
Ordinals	first, second, etc. floor numbers

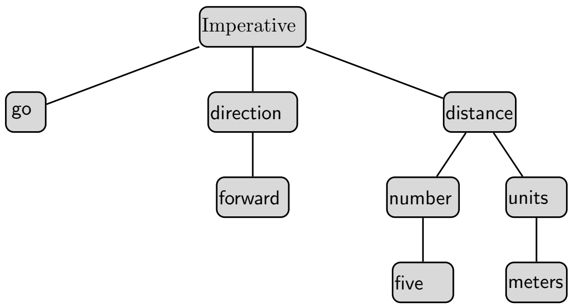


Figure 5.1: Illustration of Semantic Parse Tree used in a Dialog System

5.1.1 Grammar: A Knowledge Resource

Grammar is a very useful resource for a dialog system because it could potentially represent an expert's view of the domain. Since knowledge engineering requires time and effort, very few dialog systems can afford to have grammars that are expert-crafted and robust to various artefacts of spoken language. This becomes a major challenge for real world dialog systems. If the system's grammar or the domain knowledge does not conform to its users and their utterances, the parser will fail to produce a correct parse, if the parse is incorrect and/or the concept labeling is incorrect. Lack of comprehensive semantic knowledge is the cause of this problem. An open-domain knowledge base like Wordnet (Miller, 1995), Freebase (Bollacker et al., 2008) or NELL (Carlson et al., 2010) contains comprehensive information about concepts and their relationships present in the world. If used appropriately, open-domain knowledge resources can help compensate for incomplete semantic knowledge of the system.

5.2 Open-Domain Semantic Knowledge Bases

Like grammars, open-domain knowledge resources contain concepts, instances and relations. The purpose of these resources is to organize common sense and factoid information known to the mankind in a machine-understandable form. These resources, if filtered appropriately, contain valuable domain-specific information for a dialog agent. To this end, we propose to use three knowledge resources along with the domain grammar for the task prediction. A brief overview of each of the knowledge resources is given below:

5.2.1 Wordnet: Expert Knowledge Base

Wordnet(Miller, 1995) is an online lexical database of words and their semantics curated by language experts. It organizes the words and their morphological variants in a hierarchical fashion. Every word has at least one synset i.e., sense and a synset has definite meaning and a gloss to illustrate the usage. Synsets are connected through relationships such as hypernyms, hyponyms, meronyms, antonyms etc. Each synset can be considered as an instance and their parent synsets as concepts. Although Wordnet contains several (120,000) word forms, some of our domain-specific word forms (e.g., locations in a navigation domain) will not be present. Therefore, we would like to use other open-domain knowledge bases to augment the agent's knowledge.

5.2.2 Freebase: Community Knowledge Base

Freebase.com (Bollacker et al., 2008) is a collaboratively evolving knowledge base with the effort of volunteers. It organizes the facts based on types/concepts along with several predicates/properties and their values for each fact. The types are arranged in a hierarchy and the hierarchy is rooted at "domain". Freebase facts are constantly updated by the volunteers. Therefore, it is a good resource to help bootstrap the domain knowledge of a dialog agent.

5.2.3 *NELL*: Automated Knowledge Base

Never-Ending Language Learner(NELL) (Carlson et al., 2010) is a program that learns and organizes the facts from the web in an unsupervised fashion. NELL is on the other end of the knowledge base spectrum which is not curated either by experts or by volunteers. NELL uses a two-step approach to learn new facts: (1) extract information from the text using pattern-based, semi-structured relation extractors (2) improve the learning for next iteration based on the evidence from previous iteration. Every belief/fact in its knowledge base has concepts, source urls, extraction patterns, predicate, the surface forms of the facts and a confidence score for the belief. Although the facts could be noisy in comparison to ones in other knowledge bases, NELL continually adds and improves the facts without much human effort.

5.3 Semantic Kernel based Dialog Task Prediction

We would like to use this apriori knowledge about the world and the domain to help us predict the dialog task. The task prediction problem can be treated as a classification problem. Classification algorithms typically use bag-of-words representation that converts a document or sentence into a vector with terms as components of the vector. This representation produces very good results in scenarios with sufficient training data. However in a limited training data or extreme sparseness scenario such as ours, Siolas and D'Alché-Buc (2000) has shown that Semantic Smoothing Kernel technique is a promising approach. The major advantage of this approach is that they can incorporate apriori knowledge from existing knowledge bases. The semantic dependencies between terms, dependencies between concepts and instances, can be encoded in these kernels. The semantic kernels can be easily plugged into a kernel based classifier help us predict the task from the goal-oriented dialog utterances.

In our experiments, we used an implementation of Semantic Kernel from (Bloehdorn et al., 2006) and plugged it into a Support Vector Machine (SVM) classifier (SVM$^{\text{light}}$) (Joachims, 1999). As a part of experimental setup, we will describe the details of how did we extract the semantic dependencies from each knowledge base and encoded them into the kernel.

5.4 Experiments

Our goal is to improve the task prediction for a given spoken dialog utterance by providing additional semantic context to the utterance with the help of relevant semantic concepts from the semantic knowledge bases. The baseline approach would use the Phoenix parser's output to determine the intended task for an utterance. From our experiments, we show that our knowledge-driven approach will improve upon the baseline performance on two corpora (1) Navagati Corpus: a navigation directions corpus (2) Roomline Corpus: a room reservation dialog corpus.

5.4.1 Setup

We have divided each corpus into training and testing datasets. We train our task classification models on the manual transcriptions of the training data and evaluated the models on the ASR output of the testing data. Both Navagati and Roomline corpora came with manually annotated task labels and manual transcriptions for the utterances. We filtered out the non-task utterances such as "yes", "no" and other clarifications from the Roomline corpus. We obtained the ASR output for the Navagati corpus by running the test utterances through PocketSphinx (Huggins-Daines et al., 2006). The Roomline corpus already had the ASR output for the utterances. Table 5.2 illustrates some of the statistics for each corpus.

Our baseline model for the task detection is the Phoenix (Ward, 1991) parser output, which is the default method used in the Ravenclaw/Olympus dialog systems (Bohus et al., 2007a). For the Navagati Corpus we have obtained the parser output using the grammar and method described in (Pappu and Rudnicky, 2012). For the Roomline corpus, we extracted the parser output from the session logs from the the corpus distribution.

Semantic Facts to Semantic Kernel

The semantic kernel takes a term proximity matrix as an input, then produces a positive semidefinite matrix which can be used inside the kernel function. In our case, we build a term proximity matrix for the words in the recognition vocabulary, as shown in the Figure 5.2. Bloehdorn et al. (2006) found that using the term-concept pairs in the proximity matrix is more meaningful following the intuition that terms that share

[1]Originally has 10356 utts; filtered out non-task utts.

Corpus-Stats	**Navagati**	**RoomLine**
Tasks	4	7
Words	503	498
Word-Error-rate	46.3%	25.6%
Task Utts	934	1891[1]
Task Training-Utts	654	1324
Task Testing-Utts	280	567
Tasks		
	N1. Meta N2. Advisory N3. Imperative N4. Grounding	R1. NeedRoom R2. ChooseRoom R3. QueryFeatures R4. ListRooms R5. Identification R6. CancelReservation R7. RejectRooms

Table 5.2: Corpus Statistics

more number of concepts are similar as opposed to terms that share fewer concepts. We have used following measures to compute the proximity value P and some of them are specific to respective knowledge bases:

- `gra`: No weighting for term-concept pairs in the Grammar, i.e., $P = 1$, for all concepts c_i of t, $P = 0$ otherwise. Here c_i is i^{th} concept associated with the term t.

- `fb`: Weighting using normalized Freebase.com relevance score, i.e.,

$$P = \frac{\text{fbscore}(t, c_i) - \text{fbscore}(t, c_{min})}{\text{fbscore}(t, c_{max}) - \text{fbscore}(t, c_{min})} \tag{5.1}$$

- `nell`: Weighting for the NELL term-concept pairs using the probability for a belief i.e.,

$$P = \text{nellprob}(t, c_i) \tag{5.2}$$

, for all concepts c_i of t, $P = 0$ otherwise.

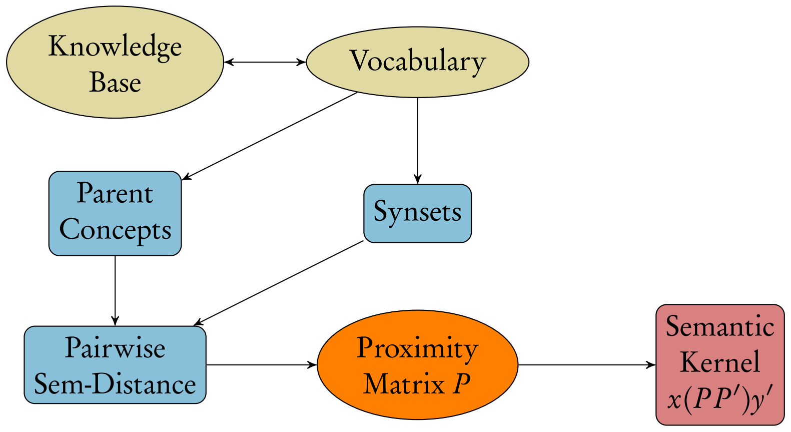


Figure 5.2: flowchart of building a semantic kernel

- `wnpath`: Weighting for the term-concept pairs in the Wordnet based on the shortest path, i.e.,

$$P = \mathrm{wn}_{PATH}(t, c_i) \tag{5.3}$$

  for all concepts c_i of t, $P = 0$ otherwise.

- `wnlch`: Weighting for the term-concept pairs in the Wordnet based on the Leacock-Chodorow Similiarity, i.e.,

$$P = \mathrm{wn}_{LCH}(t, c_i) \tag{5.4}$$

  for all concepts c_i of t, $P = 0$ otherwise.

- `wnwup`: Weighting for the term-concept pairs in the Wordnet based on the Wu-Palmer Similarity, i.e.,

$$P = \mathrm{wn}_{WUP}(t, c_i) \tag{5.5}$$

  for all concepts c_i of t, $P = 0$ otherwise.

- `wnres`: Weighting for the term-concept pairs in the Wordnet based on the Resnik Similarity using Information Content, i.e.,

$$P = \mathrm{wn}_{RES}(t, c_i) \tag{5.6}$$

  for all concepts c_i of t, $P = 0$ otherwise.

To create a grammar-based proximity matrix, we extracted the concept-token pairs from the parser output on the reference transcriptions in both corpora. In order to create a wordnet-based proximity matrix, we retrieve the hypernyms for the corresponding from Wordnet using the Wordnet 3.0 database[2]. For the freebase concept-token pairs, we query tokens for a list of types with the help of the MQL query interface[3] to the freebase. To retrieve beliefs from NELL we downloaded a tsv formatted database called every-belief-in-the-KB[4] and then queried for facts using unix `grep` command.

5.4.2 Results

Our objective is to evalute the effect of augmented semantic features on the task detection. As noted earlier, we divided both corpora into training and testing datasets. We build our models on the manual transcriptions from the training data and evaluate on the ASR hypotheses of the testing data. For the Navagati corpus, we use the same training-testing split that we used in our previous work because the grammar was developed based on the training data. For the Roomline corpus, we randomly sample 30% of the testing data from the entire corpus. First we will look at the performance of bag-of-words (BOW) based classifiers in comparison to semantic parser (semparser) baseline. Then we will compare BOW baselines with Latent-Semantic-Indexing (LSI) based models. Finally, we will discuss how semantic models are built and how they compare to these baselines in terms of F1-score.

We built BOW baselines using following classifiers: perceptron, KNN, naive-bayes and a linear SVM classifier. These results are reported in Table 5.3. On Navagati corpus, linear SVM outperforms the semparser baseline and on Roomline corpus, none of the BOW classifiers outperform the semparser baseline. The language used in navagati corpus was more verbose since it was collected in the context of a monologue setting as opposed to the roomline corpus that is collected through woz/system driven dialog. Roomline corpus has more tasks compare to Navagati corpus (in proportion to the actual data available). Although both corpora has similar sized vocabulary, Roomline corpus has more tasks compare to

[2]http://www.princeton.edu/wordnet/download/
[3]https://www.googleapis.com/freebase/v1/search
[4]http://rtw.ml.cmu.edu/rtw/resources

Navagati corpus. That might explain poorer performance of BOW models on Roomline.

Table 5.3: F1 (in %) comparison of semparser baseline against bag-of-words baselines

Corpus	semparser	perceptron	KNN	naive-bayes	SVM
Navagati	40.1	58.0	57.5	56.9	58.5
Roomline	54.3	38.9	43.5	47.3	46.4

We have seen that BOW models may fall short in performance because of data sparsity of the represenative features (words). In order to see if we can alleviate this problem by reduce the dimensionality, we propose to compare our approach with a Latent Semantic Indexing (LSI) based model. For this purpose, we use LSI method as implemented in the GENSIM toolkit (Řehůřek and Sojka, 2010). First we create LSI model from a large corpus (brown corpus (Francis and Kucera, 1979)). Then we use this model to transform input BOW vectors in Navagati and Roomline corpora into their corresponding latent dimensional vectors. Then we build an SVM classification model using the training split on these latent dimensional vectors. We present the results of this approach in Table 5.4. We varied the number of dimensions while building the LSI model. We saw that model built with 400 dimensions performs best on Navagati corpus and model with 500 dimensions performs best on Roomline corpus. These models perform better than than best BOW models. Our hypothesis is that semantic knowledge base based models can perform better than these models. Intuitively, concepts in KB are equivalent to latent dimensions and they are "verified" by experts and humans as opposed to latent dimensions that were derived in an unsupervised fashion.

Our first semantic-kernel based model SEM-GRA uses the domain grammar as a "knowledge base". This is a two step process: (1) we extract the concept-token pairs from the parse output of the training data. (2) Then, assign a uniform proximity score (1.0) for all pairs of words that appear under a particular concept otherwise 0.0 (`gra` as mentioned in the previous section). We augment the grammar concepts to the utterances in the datasets, learn SEM-GRA model and classify the test-hypotheses. For all our models we use a fixed $C = 0.07$ value (soft-margin parameter) for

Table 5.4: F1 (in %) of LSI based Model with different latent dimensions on corpora

#Dimensions	Navagati	Roomline
100	64.9	47.2
200	66.8	46.7
300	66.6	46.5
400	**68.8**	46.0
500	67.9	**47.9**

the SVM classifiers. This model achieved highest performance at this value during a parameter-sweep. SEM-GRA model outperformed the semparser baseline on both datasets (see Table 5.5). It clearly takes advantage of the domain knowledge encoded in the form of semantic-relatedness between concepts and token pairs.

Table 5.5: F1 (in %) comparison of parse baseline against semantic-kernel models with their corresponding similarity metrics

Corpus	semparser	SEMGRA	SEMFBASE	SEMNELL
Navagati	40.1	65.8	68.7	66.2
Roomline	54.3	79.7	83.3	81.1

What if a dialog system does not have grammar to begin with? We use the same two step process to build semantic-kernel based models using one open-domain knowledge base at a time. We built Wordnet based models (SEM-WNWUP, SEM-WNPATH, SEM-WNLCH, SEM-WNRES) using different proximity measures described in the previous section. From Table 5.6 SEM-WNRES model, one that uses information content performs the best among all wordnet based models. In order to compute the information content we used the pair-wise mutual information scores available for brown-corpus.dat (Francis and Kucera, 1979) in the NLTK (Bird et al., 2009) distribution. Other path based scores were also computed using NLTK API for Wordnet.

We observed that our wordnet-based models capture relatedness be-

Table 5.6: F1 (in %) comparison of wordnet semantic-kernel models with different similarity metrics

Corpus	SEMWNWUP	SEMWNPATH	SEMWNLCH	SEMWNRES
Navagati	67.1	67.7	66.4	**69**
Roomline	77.3	79.5	79.6	**80.6**

tween most-common nouns (e.g., room numbers) and their concepts but not for some of the less-common ones (e.g., location names). To compensate this imbalance, we use larger knowledge resources freebase.com and NELL. First we searched for the facts in each of these knowledge bases using every token in the vocabulary of both corpora. We pick the top concept for each token based on the score provided by the respective search interfaces. In freebase we have about 100 concepts that are relevant to the vocabulary and in the NELL model we have about 250 concepts that are relevant to the vocabulary in each of the corpora. The models based on NELL (SEM-NELL) and Freebase (SEM-FBASE) capture relatedness between less-common nouns and their concepts. We can see that both of these models perform comparable to the domain grammar model SEM-GRA which also captures the relatedness between less-common nouns and their concepts. We believe that both freebase and NELL has a superior performance because of wider-range of concept coverage and non-uniform proximity measures used in the semantic kernel, which gives a better judgement of relatedness than a uniform measure used in the SEM-GRA model.

Since we observed that an individual model is good at capturing a particular aspect of an utterance, we extended the individual semantic models by combining the proximity matrices from each of them and augmenting their semantic concepts to the training and testing datasets. We built four combined models as shown in Table 5.7 by varying the wordnet's proximity metric to identify which one of them works best in combination with other semantic metrics. The `wnres` metric performs the best both in standalone and combination settings. Bloehdorn et al. (2006) also found that `wnres` particularly performs well for lower values of the soft-margin parameter in their experiments.

Table 5.7: F1-Score (in %): Models with semantics combined from different KBs (ALL-KB)

Model	Navagati	Roomline
GRA+WNWUP+FBASE+NELL	70.8	82.2
GRA+WNPATH+FBASE+NELL	70.1	81.4
GRA+WNLCH+FBASE+NELL	70.8	81.3
GRA+WNRES+FBASE+NELL	**73.4**	**83.7**

5.5 Discussion

We have built a model that exploits different semantic knowledge bases and predicts the task on both corpora with high accuracy. But how is it affected by factors like misrecognition and context ambiguity?

5.5.1 Influence of Recognition Errors

When the recognition is bad, it is obvious that the accuracy would go down. We would like to know which of these knowledge resources can augment useful semantics despite misrecognitions. Table 5.2 shows that WER on the Navagati corpus is about 46% and the Roomline corpus is about 25%. We compared the F1-score of different models on utterances for different ranges of WER as shown in the Figure 5.3 on the Navagati Corpus. We notice that the model built using all knowledge bases is more robust even at higher WER. We did similar analysis on the Roomline corpus and did not notice any differences across models due to relatively lower WER (25.6%).

Table 5.8: Most confusable pairs of tasks (confusability in %) in Navagati Corpus for KB based classification models

KBType	ALL-KB		WNRES		NELL		FBASE		
ActualTask	N2	N4	N2	N4	N2	N4	N1	N2	N4
Predicted	N3	N1	N3	N3	N3	N3	N3	N3	N3
Confusion	10	27	26	33	26	39	22	29	44

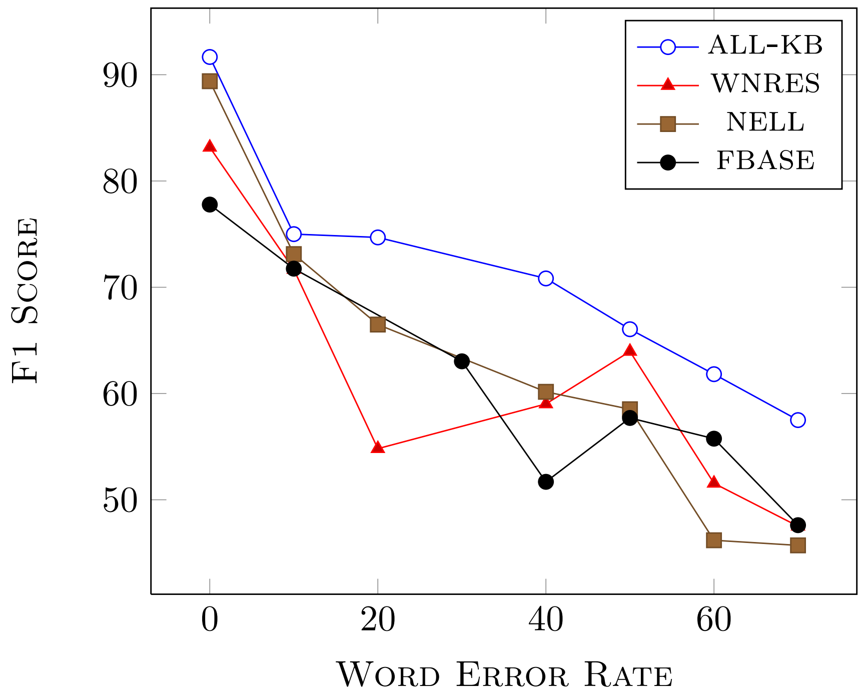


Figure 5.3: Word Error Rate vs F1-Score for KB-based Models on Navagati Corpus

Table 5.9: Most confusable pairs of tasks (confusability in %) in Roomline Corpus for KB based classification models

KBType	ALL-KB	WNRES		NELL		FBASE			
ActualTask	R4	R4	R6	R4	R6	R3	R4	R5	R6
Predicted	R3	R5	R5	R1	R1	R1	R3	R1	R1
Confusion	36	49	44	25	44	32	23	53	55

5.5.2 Confusion among Tasks

We found that particular pairs of tasks are more confusing than others. Here we present an analysis of such confusion pairs for both corpora for different classification models. Table 5.8 and Table 5.9 show the pairs of tasks that are most confused in the experiments. The ALL-KB model (a combination of all knowledge bases) has least number of confusion pairs among all the models. This is due to more relevant concepts are augmented to an utterance compared to fewer relevant concepts that augmented while using individual models.

We inspected the confused tasks by examining the feature vectors of misclassified examples. While using the ALL-KB model 10% of the utterances from N2 (Advisory) were confused for N3 (Imperative) because of phrases like "your left", "your right". These phrases were often associated with N3 utterances. To recover from such ambiguities, the agent could ask a clarification question e.g., "are we talking about going there or find it on the way?" to resolve the differences between these tasks. The system could not only get clarification but also bootstrap the original utterance of the user with the clarification to gather additional context to retrain the task detection models. The individual models were also confused about N2 and N3 tasks, where we could use similar clarification strategies to improve the task prediction. 27% of the N4 (grounding about current robot's position) utterances were confused for N1 (meta comments about the robot's navigation route) because these utterances shared more number of freebase concepts with the N1 model. The system could resolve such utterances by asking a clarification question "are we talking about the current position?". Individual models i.e., SEM-WNRES, SEM-FBASE and SEM-NELL suffered mostly from the lack of concepts capturing semantics related to all types of entities (e.g., most common nouns, less common entities etc.,) found in an utterance.

We examined the confusion pairs in the Roomline corpus and observed that R4 (ListRooms) and R3 (Queries) tasks were most confused in the ALL-KB model. On closer inspection, we found that R4 utterances are about listing the rooms that are retrieved by the system. Whereas, R3 utterances are about asking whether a room has a facility (e.g., projector availability). In the ambiguous utterances, often the R4 utterances were about filtering the list of rooms by a facility type.

5.6 Chapter Summary

In this chapter, we proposed framing the dialog task prediction problem as a classification problem. We used an SVM classifier with semantic smoothing kernels that incorporate information from external knowledge bases such as Wordnet, NELL, Freebase. Our method shows good improvements over three types of baselines: (i) parser-based baseline (ii) BOW baseline and (iii) LSI based classifier. Our analysis also shows that our proposed method makes task prediction be more robust to moderate ASR errors.

We presented an analysis on task ambiguity and found that these models can confuse one task for another. We believe that this analysis highlights the need for dialog based clarification strategies that cannot only help the system for that instance but also help the system improve its task prediction accuracy in future dialog sessions. Open-domain knowledge bases are vast and can provide useful contextual information for common words in an utterance. However, they may not contain entities specific to the agent's domain. We address this issue through system-initiated learning process (as discussed in Chapter 6 and 7) that allows the agent to augment its knowledge base through interactions with its users.

Part III
System Initiated Learning

Chapter 6

System-Initiated Knowledge Acquisition Strategies

Many spoken dialog agents are designed to perform specific tasks in a specified domain e.g., information about public events in a city. To carry out its task, an agent parses an input utterance, fills in slot-value pairs, then completes the task. Sometimes, information on these slot-value pairs may not be available in its knowledge base. In such cases, typically the agent categorizes utterances as non-understanding errors. Ideally the incident is recorded and the missing knowledge is incorporated into the system with a developer's assistance — a slow offline process.

There are other sources of knowledge: automatically crawling the web, as done by NELL (Carlson et al., 2010), and community knowledge bases such as Freebase (Bollacker et al., 2008). These approaches provide globally popular slot-values (Araki, 2012) and high-level semantic contexts (Pappu and Rudnicky, 2013). Despite their size, these knowledge bases may not contain information about the entities in a specific target domain. West et al. (2014) have shown in their study that Freebase.com has attribute-value information available for the most popular (by query frequency) person entities. There is a significant amount of attribute information missing for 90% of the person-type entities. To compensate for domain knowledge that may not be available on the web, we propose that the agent solicit knowledge from its users through dialog.

Users in the agent's domain can potentially provide specific information on slot/values that are unavailable on the web, e.g., regarding a recent interest/hobby of the user's friend. Lasecki et al. (2013) have elicited natural language dialogs from humans to build NLU models for the agent and Bigham et al. (2010) have elicited answers to visual questions by integrating users into the system. One observation from this work is that both users and non-users can impart useful knowledge to system. In this chapter, we propose spoken language strategies that allow an agent to elicit new slot-value pairs from its own user population to extend its knowledge base. Open-domain knowledge may be elicited through text-based questionnaires from non-users of the system, but in a situated interaction scenario spoken strategies may be more effective. We address the following research questions:

1. *Can an agent elicit reliable knowledge about its domain from users?* Particularly knowledge it cannot locate elsewhere (e.g., on-line knowledge bases). Is the collective knowledge of the users sufficient to allow the agent to augment its knowledge through interactive means?

2. *What strategies elicit useful knowledge from users?* Based on previous work in common sense knowledge acquisition(Von Ahn, 2006; Singh et al., 2002; ?), we devise spoken language strategies that allow the system to solicit information by presenting concrete situations and by asking user-centric questions.

We address these questions in the context of the EventSpeak dialog system, an agent that provides information about seminars and talks in an academic environment. This chapter is organized as follows. First we discuss knowledge acquisition strategies. After that, we describe a user study on these strategies. Then, we present an evaluation on system acquired knowledge and finally we summarize the discussion made in this chapter.

6.1 Knowledge Acquisition Strategies

We posit three different circumstances that can trigger knowledge acquisition behavior: (1) initiated by expert users of the system(Holzapfel et al., 2008; Spexard et al., 2006; Lütkebohle et al., 2009; Rudnicky et al., 2010), (2) triggered by "misunderstanding" of the user's input(Chung et al., 2003; Filisko and Seneff, 2005; Prasad et al., 2012; Pappu et al., 2014), or (3) triggered by the system. They are described below:

Table 6.1: System initiated strategies used by the agent for knowledge acquisition in the EventSpeak system.

StrategyType	**Strategy**	**Example Prompt**
QueryDriven	QueryEvent	I know events on campus. What do you want to know?
	QueryPerson	I know some of the researchers on campus. Whom do you want to know about?
Personal	Buzzwords	What are some of the popular phrases in *your* research?
	FamousPeople	Tell me some well-known people in *your* research area
Show&Ask	Tweet	How would you describe this talk in a sentence, say a tweet.
	Keywords	Give keywords for this talk in your own words.
	People	Do you know anyone who might be interested in this talk?

QueryDriven. The system prompts a user with an open-ended question akin to "how-may-I-help-you" to learn what "values" of a slot are of interest to the user. This strategy does not ground user about system's knowledge limitations. However, it allows the system to acquire information (slot-value pairs) from user's input. The system can choose to respond to the input or ignore the input depending on its knowledge about the slot-value pairs in the input. Table 6.1 shows strategies of this kind i.e., QueryEvent and QueryPerson.

Personal. The system asks a user about their own interests and people who may share those interests. This is an open-ended request as well, but the system expects the response to be confined to the user's knowledge about specific entities in the environment. Buzzwords and FamousPeople expects the user to provide values for the slots.

Show&Ask. The system provides a description of an event and asks questions to ground user's responses in relation to that event. E.g., given the title and abstract of a technical talk, the system asks the user questions about the talk.

Tweet strategy is expected to elicit a concise description of the event, which eventually may help the agent to both summarize events for other users and identify keywords for an event. Keywords strategy expects the user to explicitly supply keywords for an event. People strategy expects the user to provide names of likely event participants.

We hypothesized that these strategies may allow the agent to learn new slot-value pairs that may help towards better task performance.

6.2 Knowledge Acquisition Study

We conducted a user study to determine reliability of the information acquired by the system. We performed this study using the EventSpeak [1] dialog system, which provides information about upcoming talks and other events that might be of interest, and about ongoing research on campus. The system presents material on a screen and accepts spoken input, in a context similar to a kiosk.

The study evaluated performance of the seven strategies described above. For Show&Ask strategies, we had users respond regarding a specific event. We used descriptions of research talks collected from the university's website. We used a web-based interface for data collection;

[1]http://www.speech.cs.cmu.edu/apappu/kacq

the interface presented the prompt material and recorded the subject's voice response. Testvox[2] was used to setup the experiments and Wami[3] for audio recording.

6.2.1 User Study Design

We recruited 40 researchers (graduate students) from the School of Computer Science, at Carnegie Mellon, representative of the user population for the EVENTSPEAK dialog system. Each subject responded to prompts from the QUERYDRIVEN, PERSONAL and SHOW&ASK strategies.

In the QUERYDRIVEN tasks, the QUERYEVENT strategy, the system responds to the user's query with a list of talks. The user's response is recorded, then sent to an open-vocabulary speech recognizer; the result is used as a query to a database of talks. The results are then displayed on the screen. The system applies the QUERYPERSON strategy in a similar way. Example responses from the user study for these strategies are given below:

QueryEvent
Are there any talks on Spoken Language Processing

QueryPerson
Are there any talks by Alex Rudnicky

In the PERSONAL tasks, the system applies the BUZZWORDS strategy to ask the user about popular keyphrases in their research area. The system then asks about well-known researchers (FAMOUSPEOPLE) in the user's area. Example responses from the user study for these strategies are given below:

Buzzwords
Prosody
Parametric Speech Synthesis
Intonation Modeling

[2]https://bitbucket.org/happyalu/testvox/wiki/Home
[3]https://code.google.com/p/wami-recorder/

FamousPeople
Kishore Prahallad
Alan Black

In the SHOW&ASK tasks, we use two seminar descriptions per subject (in our pilot study, we found that people provide more diverse responses (in term of entities) in the SHOW&ASK based on the event abstract, compared to PERSONAL, QUERYDRIVEN . We used a set of 80 research talk announcements (consisting of a title, abstract and other information). Example talk information given below:

TITLE: Efficient and Effective Large-scale Search
ABSTRACT: Efficient and Effective Large-scale Search Search engine indexes for large document collections are often divided into "shards" that are distributed across multiple computers and searched in parallel to provide rapid interactive search. Typically, "all" index shards are searched for each query. For organizations with modest computing resources the high query processing cost of this exhaustive search setup can be a deterrent to working with large collections. This thesis questions the necessity of exhaustive search and investigates "distributed selective search" as an alternative where only a few shards are searched for each query.For selective search to be as effective as exhaustive search it is important for the chosen shards to contain the majority of the relevant documents...

For each talk, the system used all three strategies viz., TWEET, KEYWORDS and PEOPLE. For the TWEET tasks, subjects were asked to provide a one sentence description. They were allowed to give a non-technical/high-level description if they were unfamiliar with the topic. For the PEOPLE task, subjects had to give names of colleagues who might be interested in the talk. For the KEYWORDS task, subjects provided keywords, either their own words or ones selected from the abstract. Example responses from the user study for these strategies are given below:

Tweet
so this talk is about trying to replace exhaustive search with a more efficient search which is distributed selective search which basically tries to search in a distributed environment

Keywords
information retrieval
search
distributed systems
large scale search

People
bhavana dalvi
derry wijaya
jamie callan

Since the material is highly technical, we were interested whether the tasks are cognitively demanding for people who are less familiar with the subject of a talk. Therefore, users were asked to indicate their familiarity with a particular talk (research area in general) using a scale of 1–4: 4 being more familiar and 1 being less familiar.

6.2.2 Corpus Description

This user study produced 64 minutes of audio data, on average 1.6 minutes per subject. We transcribed the speech then annotated the corpus for people names, and for research interests. Table 6.2 shows the number of unique slot-values found in the corpus. We observe that the number of unique research interests produced during SHOW&ASK is higher than for other strategies. This confirms our initial observations that this strategy elicits diverse responses. The PERSONAL task produced a relatively higher number of researcher names (FAMOUSPEOPLE strategy) than other tasks. One explanation might be that people may find it easier to recall names in their own research area, as compared to other areas. Overall, we identified 139 unique researcher names and 485 interests.

6.2.3 Analysis of Dialog Strategies

One of the objectives of this work is to determine *What strategies can the agent use to elicit knowledge from users?* Although, time-cost will vary with task and domain, a usable strategy should, in general, be less demanding. We analyzed the time-per-task for each strategy, shown in Figure 6.1. We found that the TWEET strategy is not only more demanding, it has higher

Table 6.2: Corpus Statistics

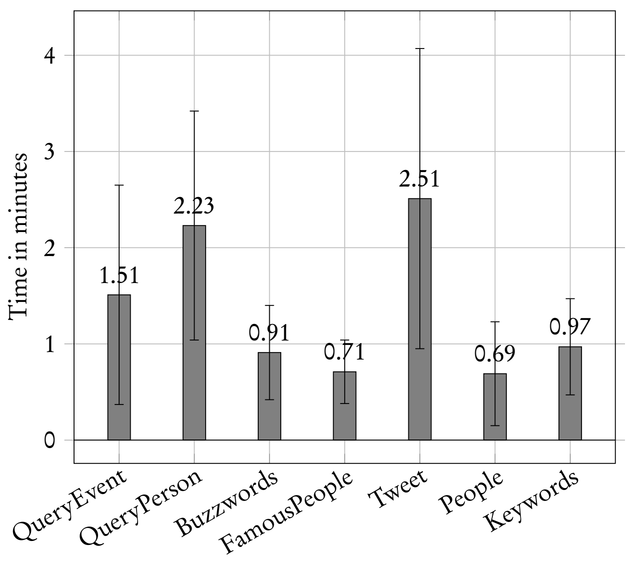
StrategyType	Unique Researcher Names	Unique Research Interests
QUERYDRIVEN	21	30
PERSONAL	77	107
SHOW&ASK	76	390
Overall	139	485

variance than other tasks. One explanation is that people would attempt to summarize the entire abstract including technical details, despite the instructions indicated that a non-technical description was acceptable. We can see a similar trend in Figure 6.2 that irrespective of expertise-level, subjects take more time to give one sentence descriptions. We also observe high variance and higher time-per-task for QUERYPERSON; this is due to the system deliberately not returning any results for this task. This was done to find out whether subjects would repeat the task on failure. Ideally the system needs to only rarely use this strategy to not lose user's trust and solicit multiple values for a given slot (e.g., person name) as opposed to requesting list of values as in FAMOUSPEOPLE and PEOPLE strategies. We find that PEOPLE, KEYWORDS, FAMOUSPEOPLE and BUZZWORDS strategies are efficient with a time-per-task of less than one minute. As shown in Figure 6.2, subjects do not take much time to speak a list of names or keywords.

6.3 Evaluation of Acquired Knowledge

To answer *Can an agent elicit reliable knowledge about its domain from users?* we analyzed the relevance of acquired knowledge. We have two disjoint list of entities, (a) researchers and (b) research interests; in addition we have speaker names from the talk descriptions. Our goal is to implicitly infer a list of interests for each researcher without soliciting the user for the interests of every researcher exhaustively. To each researcher in the list, we attribute list of interests that were mentioned in the same context as researcher was mentioned. We tag list of names acquired from the FAMOUSPEOPLE strategy with list of keywords acquired from

Figure 6.1: Time per Task for all strategies



the BUZZWORDS strategy — both lists acquired from same user. We repeat this process for each name mentioned in relation to a talk in the SHOW&ASK strategy. We tag keywords mentioned in the KEYWORDS strategy to researchers mentioned in the PEOPLE strategy.

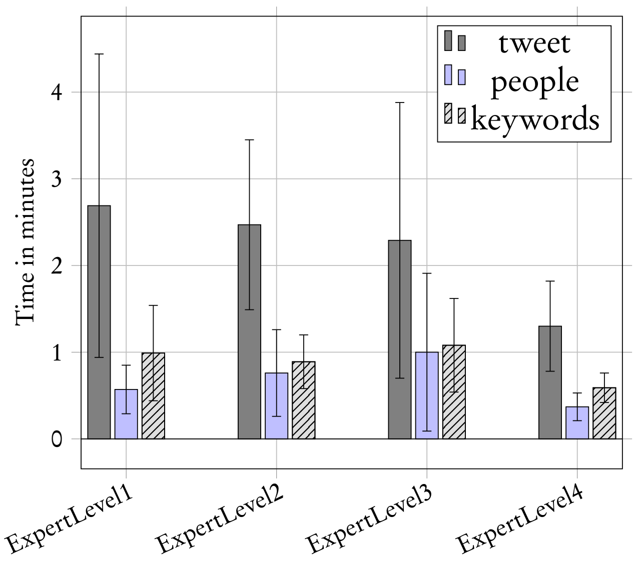
6.3.1 Analysis

We produced 200 entries for researchers and their set of interests. We then had two annotators (senior graduate students) mark whether the system-predicted interests were relevant/accurate. The annotators were allowed to use information found on researchers' home pages and Google Scholar[4] to evaluate the system-predicted interests.

This can be seen as an information retrieval (IR) problem, where researcher is "query" and interests are "documents". So, we use *Mean Precision*, a common metric in IR, to evaluate retrieval. In our case, the ground truth for relevant interests comes from the annotators. The results are shown in Figure 6.3. Our approach has high precision, 90.5%, for all 200 researchers. We see that irrespective of the strategy used to acquire

[4]scholar.google.com

Figure 6.2: Time per Task vs Expertise



entities, precision is good. We also compared our predicted interests with interests listed by researchers themselves on Google Scholar. There are only 85 researchers from our list with a Google Scholar page; for these our accuracy is 80%, again good. Moreover, significant knowledge is absent from the web (at least in our domain) yet can be elicited from users familiar with the domain.

6.4 Appendix: Example Derived Interests

Here we show a list of example derived interests for some of the researchers.

System Predicted Researcher-Interests 1

rich stern deep neural networks, speech recognition, signal processing, neural networks, machine learning, speech synthesis

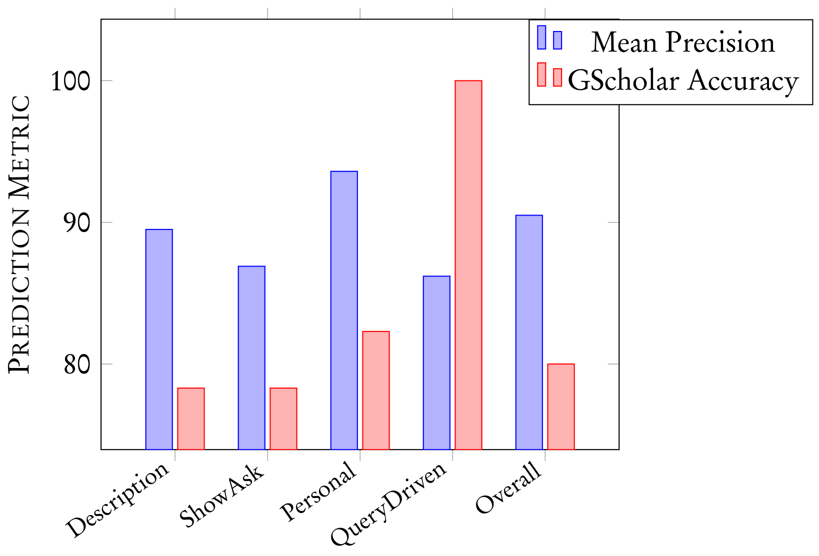


Figure 6.3: Mean Precision for 200 researchers, broken down by the "source" strategy used to acquire their name
Note: Only 85 of 200 researchers had Google Scholar pages, GScholar Accuracy is computed for only those 85.

System Predicted Researcher-Interests 2
kishore prahallad dialogue systems, prosody, speech synthesis, text to speech, pronunciation modeling, low resource languages

System Predicted Researcher-Interests 3
carolyn rose crowdsourcing, meta discourse classification, statistical analysis, presentation skills instruction, man made system, education models, human learning

System Predicted Researcher-Interests 4
florian metze dialogue systems, speech recognition, nlp, prosody, speech synthesis, text to speech, pronunciation modeling, low re-

source languages, automatic accent identification

System Predicted Researcher-Interests 5

madhavi ganapathiraju protein structure, continuous graphical models, generative models, structural biology, protein structure dynamics, molecular dynamics

System Predicted Researcher-Interests 6

alexander hauptmann discriminatively trained models, deep learning, computer vision, big data

System Predicted Researcher-Interests 7

jamie callan learning to rank, search, large scale search, web search, click prediction, information retrieval, web mining, user activity, recommendation, relevance, machine learning, web crawling, distributed systems, structural similarity

System Predicted Researcher-Interests 8

lori levin natural language understanding, knowledge reasoning, construction grammar, knowledge bases, natural language processing

6.5 Chapter Summary

We describe a set of knowledge acquisition strategies that allow a system to solicit novel information from users in a situated environment. To investigate the usability of these strategies, we conducted a user study in the domain of research talks. We analyzed a corpus of system-acquired knowledge and have made the material available[5]. Our data show that users on average take less than a minute to provide new information using the proposed elicitation strategies. The reliability of acquired knowledge

[5] www.speech.cs.cmu.edu/apappu/pubdl/eventspeak_corpus.zip

in predicting relationships between researchers and interests is quite good, with a mean precision of 90.5%. We note that the PERSONAL strategy, which tries to tap personal knowledge, appears to be particularly effective. More generally, automated elicitation appears to be a promising technique for continuous learning in spoken dialog systems. In this chapter, we have only focused on how can we elicit useful information. In the next chapter, we are going to discuss how can we exploit this knowledge to improve task performance of the agent.

Chapter 7

Learning Situated Knowledge Graphs

In many information access applications, systems need to update their domain knowledge over time to maintain accuracy. For example, an event-recommendation agent would need to know when new events appear in its domain. General information is available from the web. For example commonsense knowledge bases such as NELL (Carlson et al., 2010), Freebase (Bollacker et al., 2008). The agent can find information in these knowledge bases, but only about popular entities. Other types of information are also available, for example about popular concerts, but only those with a web presence. We are interested in acquisition of the latter type of knowledge, not always present on-line but shared on an informal basis within groups, and how it can be obtained through interaction with people.

Knowledge bases such as NELL, Freebase, and Wordnet can help in expanding semantic context, thus improve text classification in low training data scenarios (Cristianini et al., 2002; Wang and Domeniconi, 2008; Moschitti, 2009). In our previous work (Pappu and Rudnicky, 2013), we found that these knowledge bases are useful in improving dialog task prediction by expanding a user query with additional semantic context. However, the semantic context is only applicable to common content words as opposed to specific entities in the domain e.g., names of specific events. On the other hand, people that might interact with a dialog system providing information access in a domain, can provide knowledge which is useful. The dialog agent could therefore proactively seek information from its users and build a *knowledge base* or a *folksnomy* for its domain. This process is sometimes known as collaborative tagging (Golder and Huberman, 2006). As a result, the agent gains access to ontologies of information that are present in the users' minds.

Building knowledge bases is essential for a system that needs to answer user queries with situated and contextual information. Systems such as Google Now (goo), Apple Siri (app) use a variety of secondary knowledge sources about the user to answer a query. This knowledge might include the user's previous search behavior, emails and other information about the user [1]. Such knowledge is not readily available in a general commonsense knowledge base or a domain knowledge base for a typical dialog agent. To this end, we aim to define dialog strategies that an agent can use to build up a *knowledge base* based on information obtained through interaction with users.

Collaborative effort from people can help an agent solve complex problems. For example (Von Ahn, 2006) have shown that people can build a commonsense knowledge base by playing games with a computer. Our work aims at building a knowledge base through purposeful dialog between a system and its users. (Bigham et al., 2010) have shown that a variety of information can be obtained: for example, people can provide answers to visual questions and aid physically-disabled users. In our work, we seek to acquire knowledge available only through users to help the system provide better quality responses to subsequent user queries. We address the following questions:

1. *Can dialog-driven acquisition capture domain knowledge?* The agent solicits information from its users according to dialog strategies. This information is used to augment a knowledge base for the domain. We evaluate the coverage provided by this knowledge base both qualitatively and quantitatively.

2. *Is the acquired knowledge useful to the system?* The knowledge acquisition process should aid the system to improve its task success rate. To this end, we evaluate the system's performance in its event information-access domain, before and after knowledge acquisition. We show that acquired knowledge significantly improves the system's performance, as assessed by independent judges.

[1]http://googleblog.blogspot.com/2013/08/just-ask-google-for-your-flights.html

3. *Can we time the solicitation to reduce user annoyance?* The knowledge acquisition process may annoy the user if the system decides to seek information randomly. If the system can assess the quality of an ongoing interaction, then it could decide whether to ask a question or not. The quality of interaction can rely on factors such as task completion, number of turns, speech recognition quality etc. Previous work Ultes and Minker (2014) has shown interaction quality can be reliably measured using statistical methods. In this chapter, we build an interaction quality measurement model using various dialog features. Then we conduct a user study to evaluate whether a human judge would agree with the system's decision to solicit information or not. Finally, we show that the system can select questions based on certain utility factor such that it improves the quality of the knowledge graph (KG).

This chapter is organized as follows. First we present a qualitative and quantitative analysis of the knowledge base. Then, we evaluate the performance of the system on an information-access task, to show that acquired knowledge is indeed useful for system performance. Then we will discuss when to solicit information and how solicitation questions can be selected.

7.0.1 Knowledge Acquisition Study Corpus

The user study described in the previous chapter, yielded 64 minutes of audio data, with on average 1.6 minutes per participant. We have orthographically transcribed the user utterances. Then annotated the corpus [2] for people names, and research interests. Table 7.1 shows the number of unique slot-values found in the corpus. We observe that *Personal* task yielded relatively higher number of researcher names (the *FamousPeople* strategy) than other tasks. This may have happened due to people finding it easier to recall people names from their own research area, compared to names in other areas. Overall, the user study yielded 139 unique researcher names and 485 research interests.

[2] Corpus is available for download http://www.speech.cs.cmu.edu/apappu/pubdl/eventspeak_corpus.zip

Table 7.1: Breakdown of unique number of researcher names and researcher interests elicited/acquired by strategy type

StrategyType	Researcher Names	Research Interests
Query Driven	21	29
Personal	77	107
Show & Ask	76	390
Overall	139	485

7.1 Acquired Situated Knowledge Base

In this section, we address our first question: *Can the dialog-driven acquisition capture domain knowledge?* From the corpus collection, we have a list of researcher names and a list of research interests. To address our question, the system should infer a list of interests for each researcher i.e, link each researcher to a set of interests. In short, the system creates a bipartite graph with links between two disjoint sets: researchers and interests. We quantitatively analyze the consistency of this bipartite graph with respect to domain. We analyze this graph qualitatively by creating a network of blocks/communities of researchers based on their mutual interests. Details given below.

7.2 Entities and Relations

We have a disjoint list of entities: (a) researchers and (b) research interests. Our goal is to infer a list of interests for each researcher. For each researcher that was co-mentioned with a research interest, we create a link between researcher and that interest. For example, in a given dialog session with a user, researcher names mentioned in *FamousPeople* strategy are linked to interests mentioned in *Buzzwords* strategy. We repeat this process for researcher names and interests mentioned with respect to a talk i.e., *Keywords* associated with a particular talk are linked to *People* mentioned with that talk. This process produces a bipartite graph with connections between researchers and research interests.

We have 200 researchers (including the ones listed on the talk description), each mapped to a subset of interests from 485 unique interests. On average a researcher has 7.8 interests, with a standard deviation of 7.6 (this is because some researchers got more mentions across talks than others).

Table 7.2: Mean Precision for 200 researchers breakdown by the "source" strategy used to acquire their name.

Source-of-Instance	Researchers	Mean Precision
Query Driven	21	86.2%
Personal	77	93.6%
Show & Ask	76	86.9%
Talk Description	61	89.5%
Overall	200	90.5%

We observed that some interests are linked to researchers more often than others — *machine learning*, *information retrieval* and *big data* are top-3 interests, linked with 49% of the researchers. To assess the quality of predicted interests, we asked two senior Carnegie Mellon graduate students to label whether a predicted interest of a researcher is accurate. Table 7.2 shows the mean precision [3] for the predicted interests with a breakdown by source of researcher name and has good accuracy irrespective of the source.

To better understand how researchers are linked to interests and in general how researchers are aligned to each other, we construct an adjacency matrix of researchers. The details are described in the next subsection.

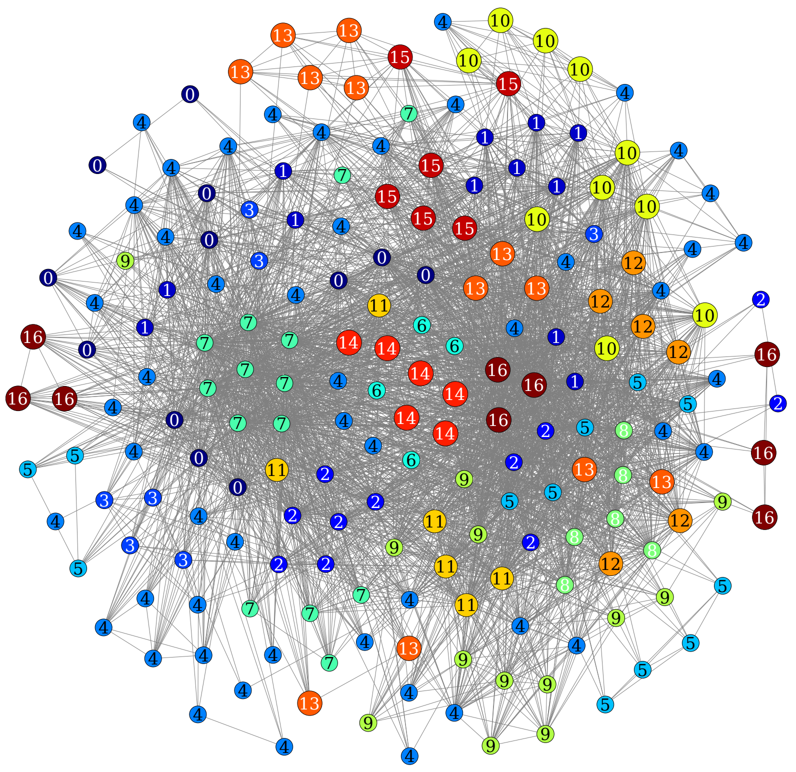
7.3 Analysis of Entity Network

To create an adjacency matrix of entities, i.e., the researchers, we compute the Jaccard index (Jaccard, 1901) for each pair of researchers based on their interests. The Jaccard index compares similarity of sample sets (say A and B) and is computed as: We use the Jaccard index of two researchers as the weight of their connecting edge. We then convert the adjacency matrix to a network (an undirected graph) using a graph tool package [4]. To find communities or blocks in the resultant network, we use a stochastic block inference algorithm (Peixoto, 2013). This algorithm tries to minimize description length (MDL) of the network (measured in nats-per-edge) to produce a block-partitioned version of the network. Intuitively, a block

[3]We use only precision because we do not have exhaustive list of relevant interests to measure the recall

[4]http://graph-tool.skewed.de

Figure 7.1: Network of researchers colored by their community

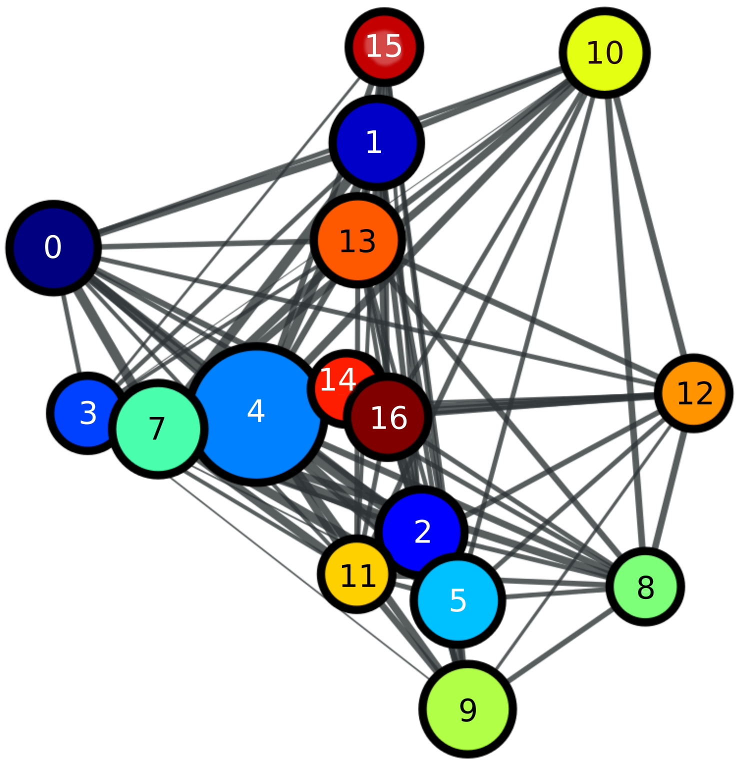


represent a set of nodes that more often interact within each other than with rest of the network; in our case, blocks are research communities. An illustration of a full-blown network is shown in Figure 7.1 and a condensed version of the network is shown in Figure 7.2 with research interests associated with each block shown in Figure 7.3.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{7.1}$$

Since the block inference algorithm optimizes the MDL, it may overfit the number of blocks needed to represent the network. Although in certain graphs (e.g., citation network) it is impossible to have modular or well-separated blocks, it is desirable to have reasonably separated blocks.

Figure 7.2: Condensed version of the block partitioned network of researchers. Each vertex represents a block of researchers.



Newman's modularity (Newman, 2006) is a metric, typically used to measure the strength of division in a graph Modularity cannot capture blocks in smaller graphs, hence we do not use it directly for block partition. To achieve a reasonable separation, we ran several iterations of the block inference algorithm, varying the minimum number of blocks required for the network. In Figure 7.4, we see that MDL is lowest (8.8) and MODULARITY is highest (0.08) when we set the minimum number of blocks to 16. This yielded a network of 17 blocks, as shown in Figure 7.1.

Figure 7.3: Research Interests associated with each block in the network. Blocks with similar interests have thicker edges.

0. speech synthesis, crowdsourcing
1. neural networks, graphical models
2. natural language processing, active learning
3. machine translation, social media
4. big data, active learning
5. information retrieval, distributed systems
6. high dimensional problems, sample complexity
7. speech recognition, human-computer interaction
8. clustering, applied machine learning
9. scalable optimization, structured sparse learning
10. protein structure, graphical models
11. information extraction
12. search, learning to rank
13. crowdsourcing, deep learning
14. community detection
15. deep learning, computer vision
16. information extraction, neuro science

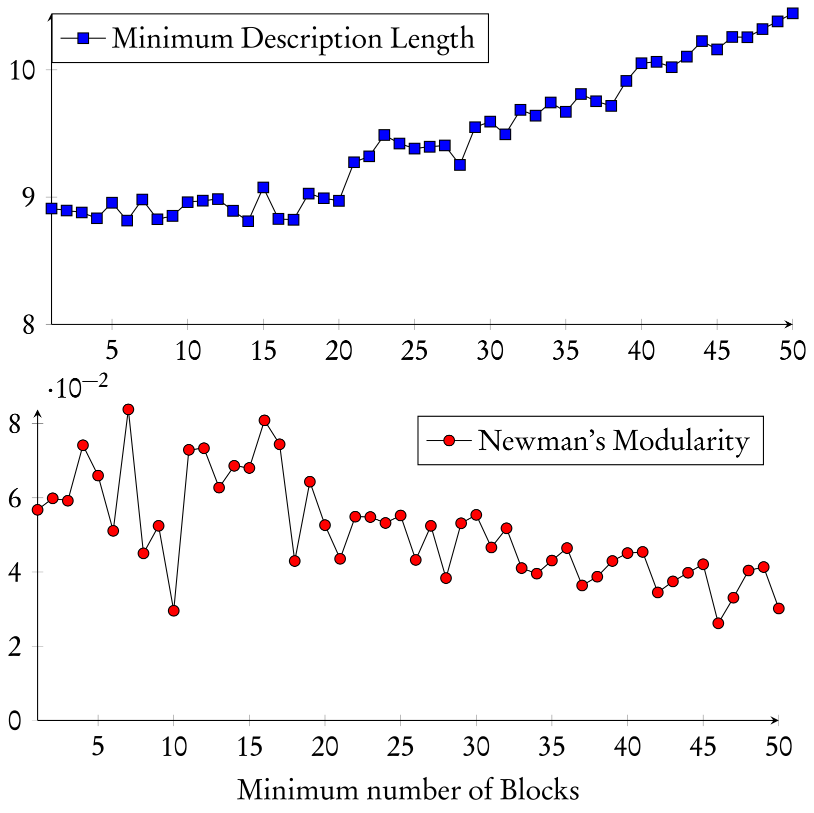
7.4 Using Knowledge: Query Expansion

Now, we want to know *whether the acquired knowledge is useful to the system* This can be demonstrated by showing that the EventSpeak System can respond more precisely to user queries by returning more relevant talks, after knowledge acquisition than before. Previous work (Navigli and Velardi, 2003) has shown that query expansion based on semantic networks can improve retrieval performance. We compare performance by expanding user query with acquired knowledge against the unexpanded queries.

7.4.1 Query Expansion Setup

We built a database of 160 research talks using abstracts, titles and speaker names as indices. For this purpose, we use Latent Semantic Indexing

Figure 7.4: MDL and MODULARITY of network against minimum number of blocks at initialization



method as implemented in the GENSIM toolkit (Řehůřek and Sojka, 2010). In the user study described earlier, we collected 40 queries (29 unique) about research areas and 40 (21 unique) about researchers — a total of 50 unique queries. We use these 50 queries to evaluate retrieval performance. Research area queries were expanded with their top-3 co-occurring research interests in the corpus. For researcher queries, we used the top-3 researcher's interests. Ten results with and without query expansion were retrieved. We then asked human judges (senior graduate students) to assess the relevance of each result on a scale of 1-4 (higher the better) with respect to a query. Our hypothesis is that results based on query expansion will have higher relevance compared to the results without query expansion.

7.4.2 Query Expansion Results

We asked 5 human judges to rate the relevance of results from 50 queries with and without query expansion. We measured inter-annotator agreement by having a pair of judges annotate the same set of 20 queries. Overall we obtained ratings for 100 (5x10x2) queries: 5 sets of 10 queries each, rated by 2 judges per set — with good agreement (Cohen's $\kappa = 0.41$).

Table 7.3 shows the mean relevance, as rated by the annotators, for the retrieved talks with respect to the query. The query based expansion system outperforms the baseline, as observed in previous work (Navigli and Velardi, 2003). Expansion works particularly well for "researcher" queries. One reason may be that person names may not have appeared in the talk description (and not indexed), but the research interests used to expand the query may appear in the talk descriptions (and indexed). We show that using this knowledge in expanding user queries can result in significantly [5] ($p < 0.01$) more relevant results (2.5/4 vs 1.8/4) than before acquisition.

Table 7.3: Mean relevance-per-query on scale of 1-4 (higher the better). Knowledge-based query expansion results are statistically ($p < 0.01$) more relevant than those without expansion.

QueryType	**Without Expansion**	**With Expansion**
Researcher	1.1 (stdev=0.8)	2.4 (stdev=0.6)
ResearchArea	2.2 (stdev=0.6)	2.5 (stdev=0.6)
Overall	1.8 (stdev=0.9)	**2.5** (stdev=0.6)

7.5 Timing the Knowledge Solicitation

In the previous sections we discussed how can we solicit information through system initiated dialog. The acquisition process was untimed i.e., the system would seek information after every interaction. In a real interaction, it is undesirable to solicit information anytime or every time. Often interactions are solely measured in terms of objective measures such as task completion and turns taken. However, it is important to assess the overall experience of an interaction. One way to measure the overall

[5] using unpaired t-test

experience is by measuring the quality of interaction on a likert scale. In this section we will discuss how can we build a statistical model to measure interaction quality and then ask a question that could potentially improve the quality of KG.

7.5.1 Interaction Quality Measurement

To measure the interaction quality (IQ) we use LEGO corpus (Schmitt et al., 2011) with features described in that work and we apply sequence labeling techniques that are similar to the ones proposed in their work. Note that we are not particularly interested in the novelty of the measuring IQ, rather interested to find out whether IQ can be a deciding factor to solicit information. LEGO corpus contains interactions of the Lets Go! Bus information system from 2006. Every interaction is divided into exchanges where an exchange includes system's turn and user's turn. We show statistics of this corpus in Table 7.4. Originally there are 348 interactions in the corpus but we only 237 have been labeled with interaction quality. There are 6379 exchanges and on average 27 exchanges per interaction.

Table 7.4: Data Statistics

Metric	Statistics
No. Interactions	237
No. Exchanges	6379
Avg. Exchanges	26.9
Avg. Interaction Quality of an Interaction	3.4 (stdev=1.4)
Avg. IQ per Exchange	3.8 (stdev=0.9)

7.5.2 Experiment Setup

To measure the interaction quality we divide the corpus into training (187 interactions) and testing (50 interactions) data. We cast the interaction quality measurement as a sequence labeling problem with exchange labeled with one of the 5 classes $\{1,2,3,4,5\}$ as proposed in (Ultes and Minker, 2014). We use two statistical approaches (i) a linear chain CRF (mallet implementation (McCallum, 2002)) and (ii) a sequence labeling SVM (SVM struct implementation (Joachims, 2008)). We compare models built using these approaches with a rule-based baseline. Often turn count is considered as an indicator of efficient interaction. Therefore, we construct our baseline based on the turn count. We will rate each exchange on the scale of 1-5 (higher the better) based on the turn counter. Thus, lower the turn counter higher the interaction quality. Intuitively this is a strong baseline. Table 7.5 shows an analysis of turn count by interaction quality for exchanges in the training data. We see that the median number of turns is quite distinct for each quality. We can reliably use the turn counter range to determine the quality of exchange.

Table 7.5: Number of User Turns in Training Interactions w.r.t. Quality of Exchange

Quality	Mean Turns	Median Turns	Min	Max
1	65.2	37	7	257
2	31.4	24	6	125
3	21.2	17.5	4	72
4	16.6	12	3	74
5	10.3	6	1	78

Schmitt et al. (2011) have proposed large set of features and presented an analysis of features and their correlation with interaction quality. We performed a χ^2 (chi-squared) test on the features and observe that features shown in Table 7.6 highly correlate with interaction quality. This observation agrees with analysis presented in (Schmitt et al., 2011). #System Questions, RoleIndex, Turn Counter (#UserTurns and #SystemTurns) are highly ranked features with good predictive power.

7.5.3 Evaluation

In the table 7.7 we present evaluation of the baseline and the statistical methods to predict interaction quality sequences. We see that CRF model outperforms both baseline and SVM_HMM with respect to F-score. Whereas, when we measure the Root Mean Square Error (RMSE) of the interaction quality with respect to the ground truth, both SVM_HMM and CRF perform similarly (1.26). We should remember that measuring interaction quality is a difficult task, even for humans.

Table 7.6: χ^2 statistic between features and IQ

Feature	Chi-val	Rank
#SystemQuestions	1413.66	1
RoleIndex	1205.3	2
#UserTurns	1085.1	3
#SystemTurns	1085.1	3
%RePrompts	904.5	5
#TimeOuts_ASRRej	849.9	6
#ASRRejections	822.4	7
ASRConfidence	402.0	8
#RePrompts	373.1	9
MeanASRConfidence	267.4	10
#TimeOutPrompts	255.7	11
#HelpRequests	0	12

Table 7.7: F-score in % (higher the better) and RMSE (lower the better) of Interaction Quality using Sequence Labeling Models

Model	RMSE	F-score					
		1	2	3	4	5	*OverAll*
RuleBased	2.34	24.2	23.1	21.6	28.8	57.4	37.2
CRF	1.27	41.1	31.8	34.8	40.4	72.3	50.4
SVM_HMM	1.26	53.9	29.0	14.6	39.1	71.0	47.0

7.5.4 Interaction Quality Based Solicitation

To evaluate how well humans can predict the interaction quality and how do they judge its effect on knowledge solicitation, we have conducted a user study. In this study, we presented an interaction to a human judge and asked them to rate that interaction on the scale of 1-5. An excerpt of one of the interaction is illustrated in the Figure 7.5. LEGO corpus contains interactions with system prompt and ASR hypothesis of the user's response. We use these interactions in our user study. In the study, we asked human judges to assess whether the user in that interaction would be annoyed if the system would ask a survey question at the end of the interaction. We have used 50 test interactions in this user study and divided them into 5 sets with 10 per set. 5 human judges were presented with two sets each and the inter-annotator agreement on average is 0.64.

Excerpt from Let's Go! corpus

S: Welcome to the CMU Let"s Go bus information system.
S: To get help at any time, just say Help or press zero.
S: What can I do for you?
U: 54C
S: The 54C . Is this correct?
U: YES
S: Right
S: Where do you wanna leave from?
U: PENN AND MAIN
S: Leaving from PENN AND MAIN . Did I get that right?
U: YES
S: Alright
S: What is your destination?
U: OAKLAND
S: Going to OAKLAND . Is this correct?

Figure 7.5: Excerpt of an Interaction from Let's Go! corpus

Table 7.8 shows that there is a good correlation between interaction quality and opportunity to solicit information. In other words, higher the quality of interaction stronger the opportunity to solicit information. Therefore, we can reliably use IQ to trigger knowledge acquisition.

Table 7.8: Correlation between Possibility of Solicitation and IQ obtained from the study and average IQ computed from the LEGO corpus

Correlation	IQ (from the study)	IQ (from the corpus)
pearson	0.73 ($p < 0.01$)	0.57 ($p < 0.01$)
spearman	0.72 ($p < 0.01$)	0.56 ($p < 0.01$)
kendall	0.65 ($p < 0.01$)	0.46 ($p < 0.01$)

Once the system decides to acquire new knowledge, it can either elicit list of instances or it can validate existing information in the KG. As the time progresses, there will be fewer new instances to elicit and the system accumulates noisy information in the KG. Therefore, it makes more sense to validate information that can improve the quality of the KG. In the following section, we shall look at how can a system can validate existing information by asking questions and can rank those questions when there are several questions to ask.

7.6 Validating Knowledge Graphs

To build a KG for the Let's Go! domain, we need to identify entities in this domain. We identify these entities from the interaction log files available in the LEGO corpus. First we filter out the semantic parser's output from the log files using unix `grep` command. Then we select semantic slot-value pairs that map to neighborhoods, bus stops and bus routes. Remember in the earlier sections we have discussed semantic slots in the academia domain. Researchers are analogous to neighborhoods/bus stops and research interests are analogous to bus routes.

A KG for the Let's Go! domain, intuitively, should connect neighborhoods through bus routes. From the log files we obtained several entities of the domain as shown in the Table 7.9. We can derive potential routes associated with each neighborhood by observing their co-occurrence. First we list out all the dialog sessions that an entity belongs to. Then we compute jaccard similarity (Jaccard, 1901) between a pair of entities. In this case, we compute similarity between a bus route and a neighborhood based on their dialog session sets. If the similarity is greater than 0, we create a link between that pair of entities. Through this process, we obtained 635 relations between neighborhoods and routes, and 2289 relations between stops and routes.

We organize the entities and their relations in the form of an affinity matrix to build a graph of entities. The matrix contains neighborhoods connected to other neighborhoods that are similar to themselves. We compute cosine similarity between vectors of bus routes associated with a pair of neighborhoods. We build two such matrices — one for neighborhoods and another for bus stops. We compute cosine similarity between entities as follows:

Table 7.9: Statistics of Let's Go! Entities and Relations

Entity	**Frequency**
Neighborhood	74
Routes	63
Stops (Intersections)	188
Relations	
Neighborhood <–> Route	635
Stops <–> Routes	2289

$$\cos(\theta) = \sum_{i=1}^{n} \frac{\text{neigh\_A}_i \times \text{neigh\_B}_i}{\sqrt{\sum_{i=1}^{n}(\text{neigh\_A}_i)^2} \times \sqrt{\sum_{i=1}^{n}(\text{neigh\_B}_i)^2}} \tag{7.2}$$

neigh_A is a vector with bus routes as dimensions and jaccard index scores as values. Once we construct a graph from this affinity matrix, we run the block partition algorithm on this graph. This yields communities in the graph. One can use modularity of this graph to assess the quality of the network. Since this is a bus route network, ideally, it should be well partitioned network with more connections among a region of neighborhoods and fewer connections outside the region. However, our graph may have invalid edges due to the cascade effect of ASR errors that have trickled into the dialog session logs. Therefore, we want to remove potentially weak or invalid edges. To this end, we can use edge weight as a criterion to select weaker edges and validate them. We ask users to verify whether neighborhoods connected through these edges are indeed connected via bus routes associated with the edges.

We select candidate edges for validation if their edge weight is in 2nd quartile, when we sort all the edges in increasing order of their edge weight. This is a reasonable criteria because edges with lowest weights are mostly invalid and edges in the upper quartile are mostly valid. We only want to validate ambiguous edges. We have randomly selected 50 edges in the 2nd quartile. We frame a validation question based on the edge attributes i.e., *neighborhood A*, *neighborhood B* and *bus routes*. For example, S: Could you tell me whether DOWNTOWN and WILKINSBURG neighborhoods are connected by anyone of these routes [61C,61A]?

If the answer is *No* then we eliminate that edge, otherwise we leave it intact. These questions are asked in the user study described above and the judges have answered *Yes*, *No* or *I don't know* to those 50 questions. We had two judges answer every question with an average inter-annotator agreement of 0.72. We have automatically eliminated the edges in the 1st quartile. This increased the modularity of the network from 0.05 to 0.1. From the user study, we have identifed 33 out of 50 edges are invalid. After we eliminate them the modularity has increased to 0.13.

7.6.1 Qualitative Analysis of Neighborhood Graph

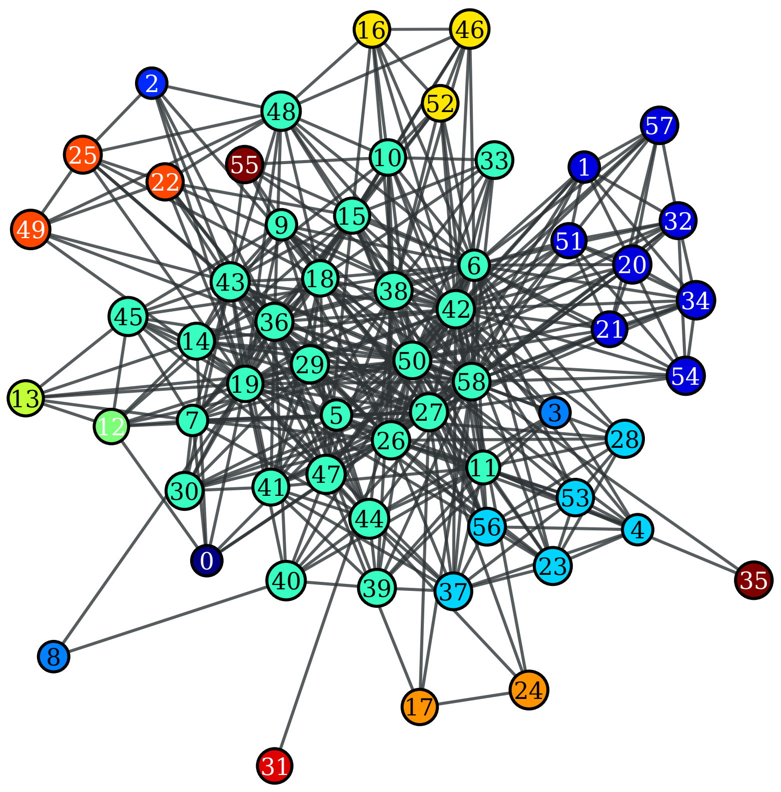
Figure 7.6 shows a graph of neighborhoods where each node is colored by its community/cluster color. There are 12 clusters in the graph. To provide a qualitative analysis of the graph, we have tagged these neighborhoods in google maps with their respective colors (cluster color) as shown in the Figure 7.7.

We observe that most of the clusters are geographically aligned. Remember that these clusters were originally derived directly from the dialog session logs with noisy ASR output. There are no geo-tags related to these sessions which makes this graph more interesting. Most of the downtown and uptown neighborhoods viz., SouthSide, Squirrel Hill, Downtown, North Side, Sheraden etc. are part of central cluster. Interestingly, Collier and West Miffin are also part of this central cluster. Although they are not geographically close to other neighborhoods in that cluster, they are port authority garages and share the bus routes with neighborhoods in that cluster.

Most of the north western neighborhoods viz., Pittsburgh International Airport, Moon Valley, Robinson Township are part of one cluster. There is another cluster in the north with Indiana township, McCandles Township, Zoo area.

There are two isolated clusters with geographically distant members. One with Bellevue and Sharpsburg, another with Crafton and North Versailles. They are not directly connected to each other but they share most bus routes with neighborhoods in the downtown cluster. That could explain their membership in their respective clusters.

Figure 7.6: Network of bus neighborhoods colored by their cluster



Since we construct our graph based on noisy entities, there are outliers in all of these clusters. We could improve the quality of this graph, by observing the system initiated explicit confirmations in the dialog exchanges. In addition, we can use interaction quality of an exchange to assign confidence scores to the discovered entities. In this work, we show that we can use noisy dialog logs to reliably reconstruct a KG.

7.7 Discussion

In this work, we have only shown that KG can be useful for improving information relevance through query expansion. We believe that there are other applications of KG for a conversational agent such as:

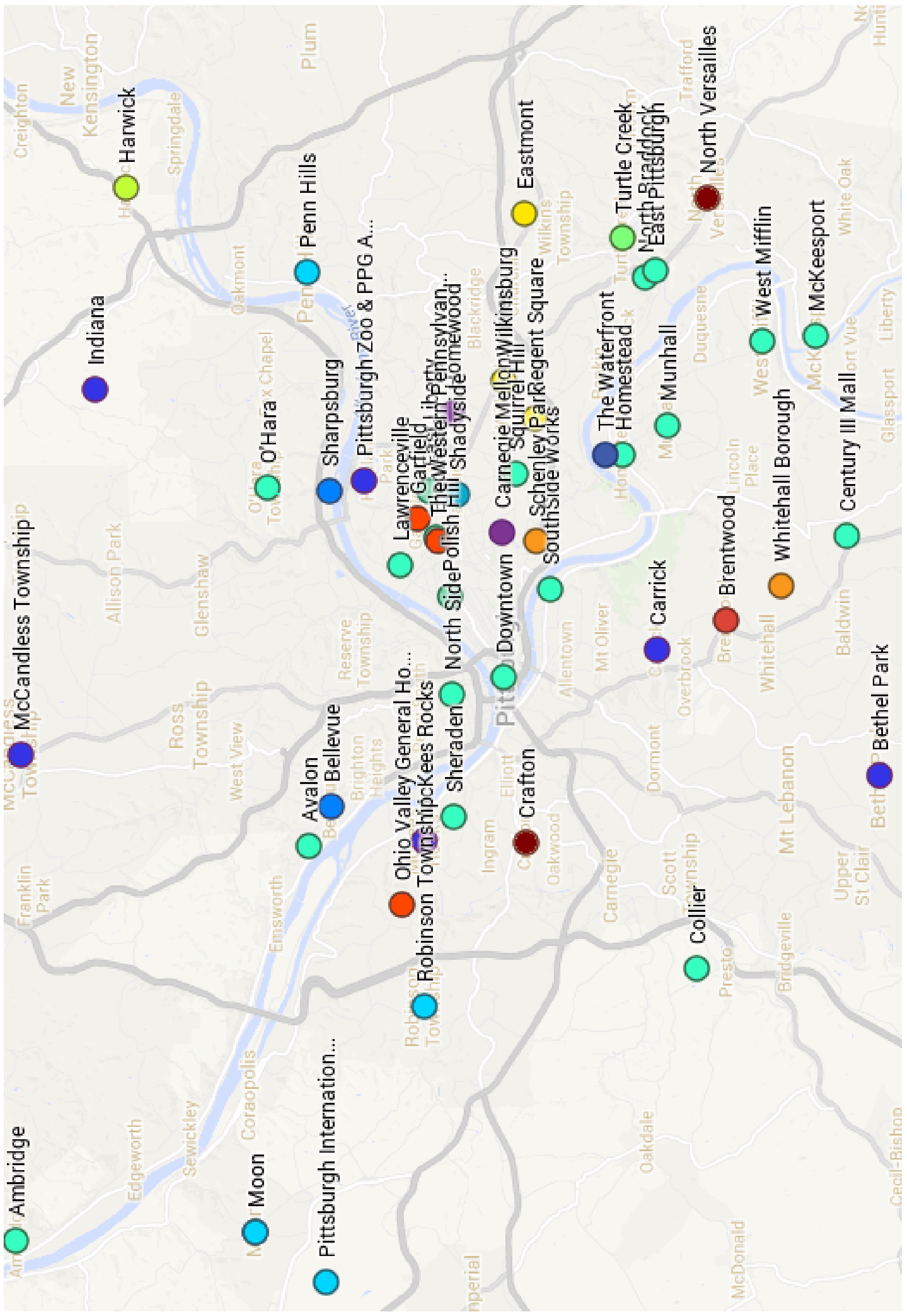


Figure 7.7: Automatically clustered Bus Neighborhoods in Pittsburgh

- to improve NLU component,
- to disambiguate entities in a dialog,
- to improve state-specific language modeling
- to improve belief tracking in a dialog.

Heck and Hakkani-Tur (2012) has shown that language understanding component can benefit from an open-domain KG. They proposed an unsupervised approach for training NLU component. They show their approach approximately matches supervised NLU method. We believe that a domain-dependent KG (such as the one discussed in this chapter) can improve NLU even more. It could potentially reduce the search space while parsing an utterance with relevant entity labels from KG. (Heck and Hakkani-Tur, 2012) also discusses how out-of-domain (textsc{ood}) utterances can be detected using KG. They show that with almost 90% accuracy one can detect OOD queries. In certain domains e.g., movies and books, different entities may have similar/same surface form. It can be challenging for an NLU component to have them disambiguate through user. Since it may not be possible to do that when there are several candidate entities. Mika et al. (2008) has proposed an entity linking method that makes uses of wikipedia and a domain corpus to improve entity disambiguation for entities in that domain. We can employ such an approach that makes use of graph properties (centrality, betweenness, degree etc.) to disambiguate the slot-types.

For recognition of utterances in a certain dialog state, typically dialog systems use state-specific language models for a better accuracy over a generic language model. Raux et al. (2010) has shown that the recognition can be even better if a bayesian network is included into this setup. This bayesian network encodes dependencies between user intent and the words used to express this intent. One of the benefits this approach is that it can capture various surface realizations and map them to a certain concept/intent. An example from the Let's Go domain, *airport* can be referred as *international airport* or *pittsburgh airport*. The bayesian network encodes this information by marginalizing the distribution of words that could potentially refer to same neighborhood. In our KG analysis, we have observed that *airport* and *pittsburgh airport* nodes have stronger edge weight and both belong to same community in the graph. Similarly *pittsburgh downtown* and *downtown* nodes belong to same community with a stronger edge. It shows that KG implicitly captures this prior knowledge and it can be exploited in the same way as bayesian networks were used in (Raux et al., 2010). Similar ideas can be applied to belief tracking. Raux and Ma (2011) have extended their previous work on language modeling to belief tracking. In this work, they use two types of networks: a static network that represents the user goal and a set of dynamic networks that represent observations (ASR / NLU) in the dialog. The static network is a form of KG discussed in this work. We could their approach to incorporate dynamic edges in a subsection of our graph that capture beliefs in an ongoing dialog.

In addition to belief tracking, we can borrow ideas from query-chaining approaches used in the information retrieval community to improve task success of a dialog. (Boldi et al., 2008) have proposed a graph-based representation to improve the understanding of user intent. They connect potential “followup” queries with a directed edge, since intuitively they belong to the same search intent. Thus, any path over a query graph represents overall search behavior. They show that such a query-flow graph can be obtained from a large scale query log. They demonstrate its usefulness through query recommendation. We could use such a graph for prompting a user with potential query phrases. This is similar to YOU-CAN-SAY error-recovery strategy proposed in (Bohus and Rudnicky, 2005c). Key difference is that we would use dynamically generated YOU-CAN-SAY prompt from the query-flow-graph.

7.8 Chapter Summary

In this chapter, we discussed a method that helps an agent to acquire domain knowledge through dialog with users and uses it to build a semantic representation of an academic field. The system uses a set of strategies to collect entities (researchers and research interests). These entities are linked by their co-occurrence to produce a bipartite graph linking researchers and research interests.

To verify that the acquired knowledge is consistent, we asked human annotators to judge whether the interests predicted by the system were accurate. We found that the predicted interests for researchers have a high mean precision of 90.5%, i.e., annotators agree with the system’s predictions in most cases. To analyze this knowledge qualitatively, we build a network of researchers connected through their mutual interests and divide this network into blocks using a block inference algorithm. This results in a set of blocks/communities of researchers (like a citation network) that covers the original academic field. We found that acquired knowledge used for query expansion provides more relevant results (2.5/4 vs 1.8/4), according to human judges, than without this acquired knowledge.

We can reliably construct KGs both from knowledge elicitation dialogs and existing dialog logs. We reliably constructed KGs for two domains: academic domain and bus information domain. Knowledge acquisition involves cost (user's time) and incentive (new knowledge for system). We have shown that interaction quality is a reliable measure to judge whether to solicit information from a user or not. To this end, we conducted a user study and observed that there is a strong correlation (pearson $= 0.7$) between interaction quality and the possibility of knowledge acquisition. We have also shown that quality of a KG can be improved by validating ambiguous edges through dialog with users.

Chapter 8

Conclusion and Future Work

This thesis has examined dialog-driven techniques for domain knowledge population. These techniques lay down a foundation of automated domain learning for conversational systems. This can potentially reduce human effort required to deploy a dialog system for new domains and improve user experience during interaction.

We have shown that a conversational agent can learn about its domain under three different settings viz., user initiated learning, system detected learning and system initiated learning. We have also shown that a system can improve its task performance by learning in each of these settings. Learning through dialog is therefore important for a dialog system. We have shown that proposed knowledge detection techniques and conversational strategies allow us to capture domain knowledge while interacting with a user. We have also shown that these techniques are usable across different domains such as navigation domain, speech to speech translation, spoken short messages, and information-access domains. Traditionally, conversational agents have been equipped with a static domain knowledge base which is based on domain expert's knowledge of the domain. The proposed dialog-driven techniques can manipulate an existing domain knowledge base and improve the agent's awareness of its domain.

This thesis proposed knowledge detection techniques and show that it can recover useful information from non-understood user input utterances. However, the techniques themselves, along with their domain-independent extensions are generic and can be applied to problems other than knowledge detection. We cast the out-of-vocabulary detection in dialog as an error detection problem. We can extend this approach to perform named entity detection for out-of-vocabulary phrases. The proposed semantic smoothing kernel approach can be extended to learn new concepts during the interaction. The proposed system initiated knowledge acquisition strategies are also extensible to new domains. While we have only seen its impact in information access domain in this thesis, the strategies can be useful for discovering entities in command-and-control domains (e.g., navigation) and open-domain interactions (e.g., spoken short messages).

LiteSF, a lightweight speech interface architecture built on top of Freeswitch, an open source softswitch platform is one of the contributions of this thesis work. A softswitch enables us to provide users with access over several types of channels (phone, VOIP, etc.) as well as support multiple users at the same time. We demonstrated this architecture through the interactive error detection system discussed in Chapter 4.

8.1 Future Directions

This thesis has only looked at learning new instances and relations between those instances for a given set of concepts. We have not directly addressed the problem of learning new concepts and dialog management for new domains. Learning new instances for a new domain itself has several interesting challenges that are not addressed in this thesis. If we were to build dialog systems for a wide range of users, we may encounter issues with multilingualism and accented speech. We could adapt a dialog agent to multilingualism by discovering new instances across several languages using a set expansion method as proposed in (Wang and Cohen, 2007). Set expansion methods are commonly used to bootstrap lexicons with the help of large amount of unlabeled and unstructured data. For example (Pantel et al., 2009) has proposed a set expansion technique based on distributional similarity statistics. Their approach uses the world-wide web as a source of information to grow a list of instances of a certain concept.

Open-domain ontologies often surrogate for lack of domain-specific ontology. Open-domain ontologies could create conceptual and terminological confusion. (Navigli and Velardi, 2004) has proposed a method and a tool *OntoLearn* that could learn domain-specific ontologies from raw text such as domain web sites, manuals and documents shared among a community. Their method extracts domain-specific keywords from the documents and organizes them in a hierarchical fashion. Then these keywords are semantically mapped to concepts in Wordnet. Mapping step involves computing distributional similarity statistics for domain-specific keywords and Wordnet concepts.

Many domains lack sufficient documents or web sites to apply ontology learning techniques as the one discussed above. We could perform domain adaptation to mine domain-specific relations and concepts from an open-domain ontology. Plank and Moschitti (2013) have proposed a tree-kernel based domain adaptation technique for relation extraction (RE). Their approach combines word clustering techniques and kernel-based RE techniques. Such approaches are often promising for dialog systems that suffer from data scarcity.

As the domain knowledge base evolves, it is important to validate whether it has valid information. One way to attest this is by asking a user explicitly but this may not be possible everytime. Another way to attest or validate new information is by using it understand user input. An utterance is typically parsed for semantic slot value pairs and one can use knowledge-based approach for semantic role labeling. Thus validating the knowledge implicitly. Pradet et al. (2013) has proposed a semantic role labeling approach that is facilitated by a knowledge base. They use English Verb Net as their knowledge base to transform syntactically-analyzed sentences into semantically-analyzed ones. One benefit of such an approach is that it does not require a priori training data. Such a knowledge base based approach is extensible to a larger scale semantic role labeling as proposed in (Heck et al., 2013). Such unsupervised methods can tremendously improve intent understanding for conversational agents.

Learning domain ontologies present one set of challenges and learning dialog task structure presents a different set of them. Often dialog task structure are hand crafted to restrict errors in an interaction. However that would curb adaptability of an agent. Gasic et al. have proposed POMDP based domain adaptation techniques for dialog management. They use a transfer learning approach to select a candidate source domain from a pool of domains to adapt to a target domain. They show that their adaptation technique guarantees good performance even in the initial stages of adaptation.

This work stands as a platform to make a spoken dialog system learn relevant semantic information both from humans and external knowledge sources. We would able to extend this paradigm to let the system elicit new references to a known semantic concept. For example, a navigation agent knows a task called "GoToRestaurant" but the user-utterance had the word “diner” and it was not seen in the context of “restaurant”. The agent somewhat predicts this utterance is related to “GoToRestaurant” using the approach described in this dissertation. It could ask the user an elicitation question: “You used diner in the context of a restaurant, is diner really a restaurant?”. The answer to this question will help the system gradually understand what parts of an open-domain knowledge base can be added into its own domain knowledge base. We believe that the holistic approach of learning from automated processes and learning through dialog, will help the dialog systems get better interaction by interaction.

Bibliography

Apple siri.
URL http://www.apple.com/ios/siri/.

Google now.
URL http://www.google.com/landing/now/.

Leman Akoglu, Hanghang Tong, Jilles Vreeken, and Christos Faloutsos.
Fast and reliable anomaly detection in categorical data.
In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 415–424. ACM, 2012.

G. Allen.
From knowledge to words to wayfinding: Issues in the production and comprehension of route directions.
Spatial Information Theory A Theoretical Basis for GIS, pages 363–372, 1997.

James Allen, Nathanael Chambers, George Ferguson, Lucian Galescu, Hyuckchul Jung, Mary Swift, and William Taysom.
Plow: A collaborative task learning agent.
In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 1514. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

Masahiro Araki.
Rapid development process of spoken dialogue systems using collaboratively constructed semantic resources.
In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 70–73, Seoul, South Korea, July 2012. Association for Computational Linguistics.
URL http://aclweb.org/anthology-new/W/W12/W12-1608.

Takaya Araki, Tomoaki Nakamura, Takayuki Nagai, Kotaro Funakoshi, Mikio Nakano, and Naoto Iwahashi.

Autonomous acquisition of multimodal information for online object concept formation by robots.
In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1540–1547. IEEE, 2011.

Srinivas Bangalore and Amanda J Stent.
Incremental parsing models for dialog task structure.
In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 94–102. Association for Computational Linguistics, 2009.

Frederic Bechet and Benoit Favre.
Asr error segment localization for spoken recovery strategy.
In *Proceedings of the ICASSP*. IEEE, 2013.

James Bergstra and Yoshua Bengio.
Random search for hyper-parameter optimization.
volume 13, pages 281–305. JMLR.org, 2012.

Jeffrey P Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, et al.
Vizwiz: nearly real-time answers to visual questions.
In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 333–342. ACM, 2010.

Aude Billard and Kerstin Dautenhahn.
Experiments in Learning by Imitation - Grounding and Use of Communication in Robotic Agents.
Adaptive Behavior, 7(3):411–434, 1999.

Steven Bird, Ewan Klein, and Edward Loper.
Natural language processing with Python.
O'Reilly Media, 2009.

Nate Blaylock and James Allen.
Hierarchical instantiated goal recognition.
In *Proceedings of the AAAI Workshop on Modeling Others from Observations*, 2006.

David M Blei, Andrew Y Ng, and Michael I Jordan.
Latent Dirichlet Allocation.
Journal of Machine Learning Research, 3(4-5):993–1022, 2003.

Stephan Bloehdorn, Roberto Basili, Marco Cammisa, and Alessandro Moschitti.
Semantic kernels for text classification based on topological measures of feature similarity.
In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 808–812. IEEE, 2006.

Dan Bohus.
Roomline.
`http://www.cs.cmu.edu/~dbohus/RoomLine`, 2003.

Dan Bohus.
Error awareness and recovery in conversational spoken language interfaces.
PhD thesis, Carnegie Mellon University, 2007.

Dan Bohus and Alexander I Rudnicky.
Modeling the cost of misunderstanding errors in the cmu communicator dialog system.
In *Proceedings of ASRU*, pages 252–255. IEEE, 2001.

Dan Bohus and Alexander I Rudnicky.
A principled approach for rejection threshold optimization in spoken dialog systems.
2005a.

Dan Bohus and Alexander I. Rudnicky.
A principled approach for rejection threshold optimization in spoken dialog systems.
In *Proceedings of Interspeech*, pages 2781–2784. ISCA, 2005b.

Dan Bohus and Alexander I Rudnicky.
Sorry, I didn't catch that!-an investigation of non-understanding errors and recovery strategies.
In *6th SIGdial Workshop on Discourse and Dialogue*, 2005c.

Dan Bohus, Antoine Raux, Thomas K Harris, Maxine Eskenazi, and Alexander I Rudnicky.
Olympus: an open-source framework for conversational spoken language interface research.
In *Proceedings of the workshop on bridging the gap Academic and industrial research in dialog technologies*, number April, pages 32–39. Association for Computational Linguistics, 2007a.

Dan Bohus, Antoine Raux, Thomas K Harris, Maxine Eskenazi, and Alexander I Rudnicky.
Olympus: an open-source framework for conversational spoken language interface research.
In *Proceedings of the workshop on bridging the gap Academic and industrial research in dialog technologies*, number April, pages 32–39. Association for Computational Linguistics, 2007b.

Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna.
The query-flow graph: model and applications.
In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 609–618. ACM, 2008.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor.
Freebase: a collaboratively created graph database for structuring human knowledge.
SIGMOD 08 Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pages 1247–1249, 2008.

G. Bugmann, E. Klein, S. Lauria, and T. Kyriacou.
Corpus-based robotics: A route instruction example.
Intelligent Autonomous Systems 8, 2004a.

Guido Bugmann, Ewan Klein, Stanislao Lauria, and Theocharis Kyriacou.
Corpus-based robotics: A route instruction example.
In *Proceedings of Intelligent Autonomous Systems*, pages 96–103, 2004b.

Hung H Bui.
A general model for online probabilistic plan recognition.
In *International Joint Conference on Artificial Intelligence*, volume 18, pages 1309–1318. Citeseer, 2003.

Lukas Burget, Petr Schwarz, Pavel Matejka, Mirko Hannemann, Ariya Rastrow, Christopher White, Sanjeev Khudanpur, Hynek Hermansky, and Jan Cernocky.
Combination of strongly and weakly constrained recognizers for reliable detection of oovs.
In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 4081–4084. IEEE, 2008.

D. Caduff and S. Timpf.
On the assessment of landmark salience for human navigation.
Cognitive processing, 9(4):249–267, 2008.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr., and Tom M Mitchell.
Toward an Architecture for Never-Ending Language Learning.
Artificial Intelligence, 2(4):1306–1313, 2010.

Joyce Y Chai, Lanbo She, Rui Fang, Spencer Ottarson, Cody Littley, Changsong Liu, and Kenneth Hanson.
Collaborative effort towards common ground in situated human-robot dialogue.
In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pages 33–40. ACM, 2014.

D.L. Chen and R.J. Mooney.
Learning to interpret natural language navigation instructions from observations.
Journal of Artificial Intelligence Research, 37:397–435, 2010.

Jennifer Chu-Carroll and Bob Carpenter.
Vector-based natural language call routing.
Computational linguistics, 25(3):361–388, 1999.

Grace Chung, Stephanie Seneff, and Chao Wang.
Automatic acquisition of names using speak and spell mode in spoken dialogue systems.
In *Proceedings of the 2003 Conference of the iNorth American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 32–39. Association for Computational Linguistics, 2003.

Mathias Creutz, Sami Virpioja, and Anna Kovaleva.
Web augmentation of language models for continuous speech recognition of sms text messages.
In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 157–165. Association for Computational Linguistics, 2009.

Nello Cristianini, John Shawe-Taylor, and Huma Lodhi.
Latent semantic kernels.
Journal of Intelligent Information Systems, 18(2):127–152, 2002.

M.P. Daniel and M. Denis.
Spatial descriptions as navigational aids: A cognitive analysis of route directions.
Kognitionswissenschaft, 7(1):45–52, 1998.
ISSN 0938-7986.

Lucie Daubigney, Milica Gašić, Senthilkumar Chandramohan, Matthieu Geist, Olivier Pietquin, Steve Young, et al.
Uncertainty management for on-line optimisation of a pomdp-based large-scale spoken dialogue system.
In *Proceedings of 12th Annual Conference of the International Speech Communication Association*, pages 1301–1304, 2011.

M. Denis, F. Pazzaglia, C. Cornoldi, and L. Bertolo.
Spatial discourse and navigation: An analysis of route directions in the city of venice.
Applied Cognitive Psychology, 13(2):145–174, 1999.

M Bernardine Dias, Thomas K Harris, Brett Browning, Edward Gil Jones, Brenna Argall, Manuela M Veloso, Anthony Stentz, and Alexander I Rudnicky.
Dynamically formed human-robot teams performing coordinated tasks.
In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, pages 30–38, 2006.

Pedro Domingos.
Metacost: a general method for making classifiers cost-sensitive.
In *Proceedings of the SIGKDD*, pages 155–164. ACM, 1999.

Finale Doshi and Nicholas Roy.
Spoken language interaction with model uncertainty: an adaptive human-robot interaction system.
Connection Science, 20(4):299–318, 2008.

Richard Dufour, Géraldine Damnati, and Delphine Charlet.
Automatic error region detection and characterization in lvcsr transcriptions of tv news shows.
In *Proceedings of the ICASSP*, pages 4445–4448. IEEE, 2012.

K. Eberhard, H. Nicholson, S. Kubler, S. Gundersen, and M. Scheutz.
The indiana .cooperative remote search task.(crest) corpus.
In *Proc. of LREC*, volume 10, 2010.

Edward Filisko and Stephanie Seneff.
Error detection and recovery in spoken dialogue systems.
In *Proceedings of the HLT-NAACL*, pages 31–38. ACL, 2004a.

Edward Filisko and Stephanie Seneff.
Error detection and recovery in spoken dialogue systems.
In *Proceedings of HLT-NAACL 2004 Workshop on Spoken Language Understanding for Conversational Systems*, pages 31–38, 2004b.

Edward Filisko and Stephanie Seneff.
Developing city name acquisition strategies in spoken dialogue systems via user simulation.
In *6th SIGdial Workshop on Discourse and Dialogue*, 2005.

Edward Filisko and Stephanie Seneff.
Learning decision models in spoken dialogue systems via user simulation.
In *Proceedings of AAAI Workshop on Statistical and Empirical Approaches for Spoken Dialog Systems. Boston, Massachusetts*, 2006.

Jonathan G Fiscus.
A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover).
In *Proceedings of the ASRU*, pages 347–354. IEEE, 1997.

W Nelson Francis and Henry Kucera.
Brown corpus manual.
Brown University Department of Linguistics, 1979.

A. Gargett, K. Garoufi, A. Koller, and K. Striegnitz.
The give-2 corpus of giving instructions in virtual environments.
In *Proc. of LREC*, 2010.

Milica Gasic, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young.
Pomdp-based dialogue manager adaptation to extended domains.
In *Proceedings of the SIGDIAL 2013 Conference*, pages 214–222. Association for Computational Linguistics.

Saeed Shiry Ghidary, Yasushi Nakata, Hiroshi Saito, Motofumi Hattori, and Toshi Takamori.
Multi-Modal Interaction of Human and Home Robot in the Context of Room Map Generation.
Autonomous Robots, 13(2):169–184, 2002.

Scott A Golder and Bernardo A Huberman.
Usage patterns of collaborative tagging systems.
Journal of information science, 32(2):198–208, 2006.

Allen L Gorin, Giuseppe Riccardi, and Jeremy H Wright.
How may i help you?
Speech communication, 23(1-2):113–127, 1997.

Shane Griffith, Jivko Sinapov, Matthew Miller, and Alexander Stoytchev.
Toward interactive learning of object categories by a robot: A case study with container and non-container objects.
2009 IEEE 8th International Conference on Development and Learning, pages 1–6, 2009.

Norman Haas and GG Hendrix.
An approach to acquiring and applying knowledge.
Defense Technical Information Center, 1980.

Thomas K Harris and Alexander I Rudnicky.
TeamTalk: A platform for multi-human-robot dialog research in coherent real and virtual spaces.
In *Proceedings of the National Conference on Artificial Intelligence*, page 1864, 2007.

Timothy J Hazen, Stephanie Seneff, and Joseph Polifroni.
Recognition confidence scoring and its use in speech understanding systems.
Computer Speech & Language, 16(1):49–67, 2002.

Larry Heck and Dilek Hakkani-Tur.
Exploiting the semantic web for unsupervised spoken language understanding.
In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 228–233. IEEE, 2012.

Larry P Heck, Dilek Hakkani-Tür, and Gökhan Tür.
Leveraging knowledge graphs for web-scale unsupervised semantic parsing.
In *INTERSPEECH*, pages 1594–1598, 2013.

Hartwig Holzapfel, Thomas Schaaf, Hazım Kemal Ekenel, Christoph Schaa, and Alex Waibel.
A robot learns to know people—first contacts of a robot.
KI 2006: Advances in Artificial Intelligence, pages 302–316, 2007.

Hartwig Holzapfel, Daniel Neubig, and Alex Waibel.
A dialogue approach to learning object descriptions and semantic categories.
Robotics and Autonomous Systems, 56(11):1004–1013, November 2008.

D. Huggins-Daines, M. Kumar, A. Chan, A.W. Black, M. Ravishankar, and A.I. Rudnicky.
Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices.
In *ICASSP*, volume 1. IEEE, 2006.

Paul Jaccard.
Étude comparative de la distribution florale dans une portion des alpes et des jura.
Bulletin del la Société Vaudoise des Sciences Naturelles, 37:547–579, 1901.

Thorsten Joachims.
Making large-scale support vector machine learning practical.
In *Advances in kernel methods*, pages 169–184. MIT Press, 1999.

Thorsten Joachims.
Sequence tagging with structural support vector machines.
2008.
URL http://www.cs.cornell.edu/people/tj/svm_light/svm_hmm.html.

Jihie Kim and Yolanda Gil.
Incorporating tutoring principles into interactive knowledge acquisition.
International Journal of Human-Computer Studies, 65(10):852–872, October 2007.

T. Kollar, S. Tellex, D. Roy, and N. Roy.
Toward understanding natural language directions.
In *Proceeding of the 5th ACM/IEEE HRI*. ACM, 2010.

Kazunori Komatani and Tatsuya Kawahara.
Generating effective confirmation and guidance using two-level confidence measures for dialogue systems.
In *Proceedings of the ICSLP*. ISCA, 2000.

Geert-jan M Kruijff, Hendrik Zender, Patric Jensfelt, and Henrik I Christensen.
Situated dialogue and spatial organization: What, where... and why.
International Journal of Advanced Robotic Systems, 4(2):125–138, 2007.

Taku Kudo.
crfpp.sourceforge.net, 2013.

Rahmadi Kurnia, Altab Hossain, Akio Nakamura, and Yoshinori Kuno.
Object Recognition Through Human Robot Interaction by Speech.
In *Procedings of the 2004 IEEE International Workshop on Robot and Human Interactive Communication*, pages 619–624, 2004.

Walter Stephen Lasecki, Ece Kamar, and Dan Bohus.
Conversations in the crowd: Collecting data for task-oriented dialog learning.
In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013.

Stanislao Lauria, Guido Bugmann, Theocharis Kyriacou, Johan Bos, and Ewan Klein.
Training personal robots using natural language instruction.
In *IEEE Intelligent Systems*, volume 16, pages 38–45, 2001.

Cheongjae Lee, Sangkeun Jung, Donghyeon Lee, and Gary Guenbae Lee.
Example-based error recovery strategy for spoken dialog system.
In *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, pages 538–543. IEEE, 2007.

Wei-Bin Liang, Chung-Hsien Wu, and Meng-Hsiu Sheng.
Affective-cognitive dialogue act detection in an error-aware spoken dialogue system.
In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*, pages 1–6. IEEE, 2013.

K. Lovelace, M. Hegarty, and D. Montello.
Elements of good route directions in familiar and unfamiliar environments.
Spatial information theory. Cognitive and computational foundations of geographic information science, pages 751–751, 1999.

Ingo Lütkebohle, Julia Peltason, Lars Schillingmann, Christof Elbrechter, Britta Wrede, Sven Wachsmuth, and Robert Haschke.
The Curious Robot – Structuring Interactive Robot Learning.
In *International Conference on Robotics and Automation*, ICRA'09, pages 4156–4162. IEEE, IEEE, 2009.

M. MacMahon, B. Stankiewicz, and B. Kuipers.
Walk the talk: Connecting language, knowledge, and action in route instructions.
Def, 2(6):4, 2006.

Sandra Marcus and John McDermott.
SALT: A knowledge acquisition language for propose-and-revise systems.
Artificial Intelligence, 39(1):1–37, 1989.

M. Marge and A.I. Rudnicky.
Comparing spoken language route instructions for robots across environment representations.
In *SIGDIAL*, 2010.

L Markson and P Bloom.
Evidence against a dedicated system for word learning in children.
Nature, 385(6619):813–815, 1997.

C. Matuszek, D. Fox, and K. Koscher.
Following directions using statistical machine translation.
In *Proceeding of the 5th ACM/IEEE international conference on Human-robot interaction*, pages 251–258. ACM, 2010.

Andrew Kachites McCallum.
Mallet: A machine learning for language toolkit.
2002.
URL http://mallet.cs.umass.edu/.

Michael F McTear.
Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit.
In *Fifth International Conference on Spoken Language Processing*, volume 5, pages 1223–1226. ISCA, Citeseer, 1998.

Peter Mika, Massimiliano Ciaramita, Hugo Zaragoza, and Jordi Atserias.
Learning to tag and tagging to learn: A case study on wikipedia.
IEEE Intelligent Systems, 23(5):26–33, 2008.

George A Miller.
WordNet: a lexical database for English.
Communications of the ACM, 38(11):39–41, 1995.

Teruhisa Misu and Tatsuya Kawahara.
A bootstrapping approach for developing language model of new spoken dialogue systems by selecting web texts.
In *Proc. Interspeech*, pages 9–12, 2006.

Alessandro Moschitti.
Syntactic and semantic kernels for short text pair categorization.
In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 576–584, 2009.

Roberto Navigli and Paola Velardi.
An analysis of ontology-based query expansion strategies.
In *Proceedings of the 14th European Conference on Machine Learning, Workshop on Adaptive Text Extraction and Mining, Cavtat-Dubrovnik, Croatia*, pages 42–49, 2003.

Roberto Navigli and Paola Velardi.
Learning domain ontologies from document warehouses and dedicated web sites.
Computational Linguistics, 30(2):151–179, 2004.

Mark EJ Newman.
Modularity and community structure in networks.
Proceedings of the National Academy of Sciences, 103(23):8577–8582, 2006.

Atsunori Ogawa, Takaaki Hori, and Atsushi Nakamura.
Recognition rate estimation based on word alignment network and discriminative error type classification.
In *Proceedings of Spoken Language Technology Workshop*, pages 113–118. IEEE, 2012.

OpenNLP.
http://opennlp.apache.org/.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith.
Improved part-of-speech tagging for online conversational text with word clusters.
In *Proceedings of the NAACL-HLT*, pages 380–390, 2013.

Tim Paek and Eric Horvitz.
On the utility of decision-theoretic hidden subdialog.
In *ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems*, 2003.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas.
Web-scale distributional similarity and entity set expansion.
In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 938–947. Association for Computational Linguistics, 2009.

Aasish Pappu and Alexander Rudnicky.
Predicting tasks in goal-oriented spoken dialog systems using semantic knowledge bases.
In *Proceedings of the SIGDIAL 2013 Conference*, pages 242–250. ACL, 2013.

Aasish Pappu and Alexander I Rudnicky.
The Structure and Generality of Spoken Route Instructions.
Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue, pages 99–107, 2012.

Aasish Pappu and Alexander I Rudnicky.
Knowledge acquisition strategies for goal-oriented dialog systems.
In *Proceedings of the 15th SIGDIAL Conference*, pages 194–198, 2014a.

Aasish Pappu and Alexander I Rudnicky.
Learning situated knowledge bases through dialog.
In *Proceedings of the 15th Interspeech*, 2014b.

Aasish Pappu, Teruhisa Misu, and Rakesh Gupta.
Investigating critical speech recognition errors in spoken short messages.
In *Proceedings of IWSDS*, 2014.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi.
WordNet:: Similarity: measuring the relatedness of concepts.
In *Demonstration Papers at HLT-NAACL 2004*, pages 38–41. Association for Computational Linguistics, 2004.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay.
Scikit-learn: Machine learning in Python.
volume 12, pages 2825–2830, 2011.

Tiago P Peixoto.
Parsimonious module inference in large networks.
Physical Review Letters, 110(14):148701, 2013.

Dennis Perzanowski, Alan C Schultz, and William Adams.
Integrating natural language and gesture in a robotics domain.
Proceedings of the 1998 IEEE International Symposium on Intelligent Control ISIC held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA Intelligent Systems and Semiotics ISAS Cat No98CH36262, pages 247–252, 1998.

Ann M Peters.
The units of language acquisition, volume 1.
Cambridge Univ Press, 1983.

Eli Pincus, Svetlana Stoyanchev, and Julia Hirschberg.
Exploring features for localized detection of speech recognition errors.
In *Proceedings of the SIGDIAL*, pages 132–136. ACL, 2013.

Steven Pinker.
Language Learnability and Language Development, volume 7.
Harvard University Press, 1984.

P Placeway, S Chen, Maxine Eskenazi, U Jain, V Parikh, Bhiksha Raj, Mosur Ravishankar, Ronald Rosenfeld, K Seymore, M Siegler, R Stern, and E Thayer.
The 1996 Hub-4 Sphinx-3 System.
Proceedings of DARPA Speech Recognition Workshop, pages 85–89, 1997.

Barbara Plank and Alessandro Moschitti.
Embedding semantic similarity in tree kernels for domain adaptation of relation extraction.
In *ACL (1)*, pages 1498–1507, 2013.

Quentin Pradet, Gaël De Chalendar, Guilhem Pujol, et al.
Revisiting knowledge-based semantic role labeling.
LTC'13 Proceedings, 2013.

Rohit Prasad, Rohit Kumar, Sankaranarayanan Ananthakrishnan, Wei Chen, Sanjika Hewavitharana, Matthew Roy, Frederick Choi, Aaron Challenner, Enoch Kan, Arvind Neelakantan, et al.
Active error detection and resolution for speech-to-speech translation.
In *Proceedings of IWSLT*, 2012.

Long Qin and Alexander I Rudnicky.
OOV Word Detection using Hybrid Models with Mixed Types of Fragments.
Interspeech-2012, 2012.

Antoine Raux.
Flexible turn-taking for spoken dialog systems.
PhD thesis, Carnegie Mellon University, 2008.

Antoine Raux and Maxine Eskenazi.
A finite-state turn-taking model for spoken dialog systems.
In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 629–637. Association for Computational Linguistics, 2009.

Antoine Raux and Yi Ma.
Efficient probabilistic tracking of user goal and dialog history for spoken dialog systems.
In *INTERSPEECH*, pages 801–804, 2011.

Antoine Raux, Neville Mehta, Deepak Ramachandran, and Rakesh Gupta.
Dynamic language modeling using bayesian networks for spoken dialog systems.
In *INTERSPEECH*, pages 3030–3033, 2010.

Radim Řehůřek and Petr Sojka.
Software Framework for Topic Modelling with Large Corpora.
In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50. ELRA, 2010.

Raquel Ros, Akin E. Sisbot, Séverin Lemaignan, Amit Pandey, and Rachid Alami.
Robot, tell me what you know about...?: Expressing robot's knowledge through interaction.

Proceedings of the ICRA 2010 Workshop on Interactive Communication for Autonomous Intelligent Robots ICAIR Making robots articulate what they understand intend and do, 2010.

R. Rosenfield.
The cmu statistical language modeling toolkit and its use in the 1994 arpa csr evaluation. 1995.

Alexander I Rudnicky, Aasish Pappu, Peng Li, and Matthew Marge.
Instruction Taking in the TeamTalk System.
In *Proceedings of the AAAI Fall Symposium on Dialog with Robots*, number Dm, pages 173–174, 2010.

Thomas Schaaf.
Detection of OOV words using generalized word models and a semantic class language model.
In *Proc. Eurospeech*, pages 2581–2584. Citeseer, 2001.

Alexander Schmitt, Benjamin Schatz, and Wolfgang Minker.
Modeling and predicting quality in spoken human-computer interaction.
In *Proceedings of the SIGDIAL 2011 Conference*, pages 173–184. Association for Computational Linguistics, 2011.

Yongmei Shi and Lina Zhou.
Error detection using linguistic features.
In *Proceedings of the HLT-EMNLP*, pages 41–48. ACL, 2005.

Takashi Shichiri, Hiroaki Nanjo, and Takehiko Yoshimi.
Minimum bayes-risk decoding with presumedword significance for speech based information retrieval.
In *Proceedings of the ICASSP*, pages 1557–1560. IEEE, 2008.

Push Singh, Thomas Lin, Erik T Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu.
Open mind common sense: Knowledge acquisition from the general public.
In *CoopIS, DOA, and ODBASE*, pages 1223–1237. Springer, 2002.

Georges Siolas and Florence D'Alché-Buc.
Support vector machines based on a semantic kernel for text categorization.
In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 5, pages 205–209. IEEE, 2000.

Gabriel Skantze.
Error handling in spoken dialogue systems: Managing uncertainty, grounding and miscommunication.
Gabriel Skantze, 2007a.

Gabriel Skantze.
Making grounding decisions: Data-driven estimation of dialogue costs and confidence thresholds, 2007b.

Marjorie Skubic, Dennis Perzanowski, Samuel Blisard, Alan Schultz, William Adams, Magda Bugajska, and Derek Brock.
Spatial language for human-robot dialogs, 2004.

Thorsten Spexard, Shuyin Li, Britta Wrede, Jannik Fritsch, Gerhard Sagerer, Olaf Booij, Zoran Zivkovic, Bas Terwijn, and Ben Krose.
BIRON, where are you? Enabling a robot to learn new places in a real home environment by integrating spoken dialog and visual localization.
Integration The Vlsi Journal, (section II):934–940, 2006.

L. Stoia, D.M. Shockley, D.K. Byron, and E. Fosler-Lussier.
Scare: A situated corpus with annotated referring expressions.
In *LREC 2008*, 2008.

Andreas Stolcke.
SRILM-an extensible language modeling toolkit.
System, 3:901–904, 2002.

Svetlana Stoyanchev, Alex Liu, and Julia Hirschberg.
Modelling human clarification strategies.
In *Proceedings of the SIGDIAL*, pages 137–141. ACL, 2013.

Stephen Sutton, Ronald A Cole, Jacques De Villiers, Johan Schalkwyk, Pieter JE Vermeulen, Michael W Macon, Yonghong Yan, Edward C Kaiser, Brian Rundle, Khaldoun Shobaki, et al.
Universal speech tools: the cslu toolkit.
In *ICSLP*, volume 98, pages 3221–3224, 1998.

Christian Theobalt, Johan Bos, Tim Chapman, Arturo Espinosa-Romero, Mark Fraser, Gillian Hayes, Ewan Klein, Tetsushi Oka, and Richard Reeve.
Talking to Godot: Dialogue with a Mobile Robot.
In *Proceedings of IEEERSJ International Conference on Intelligent Robots and Systems IROS 2002 PG 1338*, volume 2, pages 1338–1343 vol.2. Citeseer, 2002.

Andrea L Thomaz and Maya Cakmak.
Learning about objects with human teachers.
In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 15–22. ACM, 2009.

Blaise Thomson and Steve Young.
Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems.
Computer Speech & Language, 24(4):562–588, 2010.

K. Toutanova, D. Klein, C.D. Manning, and Y. Singer.
Feature-rich part-of-speech tagging with a cyclic dependency network.
In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

Stefan Ultes and Wolfgang Minker.
Interaction quality estimation in spoken dialogue systems using hybrid-hmms.
In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 208–217, Philadelphia, PA, U.S.A., 2014. Association for Computational Linguistics.

Luis Von Ahn.
Games with a purpose.
Computer, 39(6):92–94, 2006.

Marilyn A Walker, Diane J Litman, Candace A Kamm, and Alicia Abella.
PARADISE: A framework for evaluating spoken dialogue agents.
In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 271–280. Association for Computational Linguistics, 1997.

Pu Wang and Carlotta Domeniconi.
Building semantic kernels for text classification using wikipedia.
In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 713–721. ACM, 2008.

Richard C Wang and William W Cohen.
Language-independent set expansion of named entities using the web.
In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 342–350. IEEE, 2007.

W. Ward.
Understanding spontaneous speech: the phoenix system.
In *ICASSP*. IEEE, 1991.

Sandra R. Waxman and Amy E. Booth.
Principles that are invoked in the acquisition of words, but not facts.
Cognition, 77(2):B33–43, 2000.

Yuan Wei, Emma Brunskill, Thomas Kollar, and Nicholas Roy.
Where to go: Interpreting natural directions using global inference.
In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3761–3767. IEEE, 2009.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin.
Knowledge base completion via search-based question answering.
In *Proceedings of the 23rd international conference on World wide web*, pages 515–526. International World Wide Web Conferences Steering Committee, 2014.

J D Williams.
Incremental partition recombination for efficient tracking of multiple dialog states.
In *Acoustics Speech and Signal Processing ICASSP 2010 IEEE International Conference on*, pages 5382–5385, 2010.

Jason D Williams and Steve Young.
Characterizing task-oriented dialog using a simulated ASR channel.
In *Proceedings of the ICSLP*. Citeseer, 2004.

Michael Witbrock, David Baxter, and Jon Curtis.
An interactive dialogue system for knowledge acquisition in cyc.
Proceedings of the workshop on mixed-initiative intelligent systems, pages 138–145, 2003.

Dekai Wu.
Active acquisition of user models: implications for decision-theoretic dialog planning and plan recognition.
User Modeling and User-Adapted Interaction, 1(2):149–172, 1991.

Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al.
Top 10 algorithms in data mining.
volume 14, pages 1–37. Springer, 2008.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu.
The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management.
Computer Speech Language, 24(2):150–174, 2010.

Rong Zhang and Alexander I Rudnicky.
Word level confidence annotation using combinations of features.
In *Proceedings of Eurospeech*, 2001.