Addis Ababa University

College of Humanities, Language Studies, Journalism and

Communications Studies

Department of Linguistics

Developing Amharic Spoken Dialogue System: A Hybrid Approach

Fitsum Seyoum

March, 2015

Addis Ababa University

College of Humanities, Language Studies, and Journalism and Communication studies

Department of Linguistics

A Thesis Submitted to the School of Graduate Studies of

Addis Ababa University in

Partial Fulfillment of the Requirements for the

Degree of Master of Science in

Computational Linguistics

DECLARATION

This thesis is my original work, has not been presented for a degree in any university and all sources of material used for the thesis have been duly acknowledged.

_____

Fitsum Seyoum

This thesis has been submitted for examination with my approval as university advised.

_____ _____

_____ _____

_____ _____

ACKNOWLEDGEMENT

I want to thank God for His goodness in every walk of my life.

I would like to express my appreciation and gratitude to my advisor, Dr. Million Meshesha, for his encouragement and support. He has provided me great insight in tackling problems. I have learnt how to think critically from him and benefitted from his wealth of wisdom.

I also would like to thank my second advisor Ato Biniyam Ephrem for his help and valuable suggestions in my writing.

Thank you to the Addis Ababa University, Computational Linguistics Department Students for being good friends and for the support in participating in data collection and testing. Tafesse Walea, Biniyam Getahun, Abiy Desta and Dionasios Deressa, thank you very much.

I am truly grateful to the people at Adrasha.net and Afalagi.com for their cooperation and for allowing me to use the data from their websites.

I want to thank my family for their support for the entire duration of this study. Bilen with her husband Bereket, Belyou with her husband Mesfin and Zebib, thank you very much. I want to thank my mother Aslefech Mulat for her unceasing support and encouragement.

Lastly, but certainly not the least, I would like to thank my wife, Tsion Mulugeta. Without her love, support and patience, I would not have been able to pursue my dreams.

Table of contents

LIST OF FIGURES

TABLE

ALGORITHM

ACRONYMS

ASDT	AT&T Statistical Dialogue Toolkit
ASR	Automatic Speech Recognition
DTMF	Dual Tone Multi-Frequency
FSA	Finite State Automaton
IVR	Interactive Voice Response
MDP	Markov Decision Process
NLU	Natural Language Understanding
NLG	Natural Language Generation
POMDP	Partially Observable Markov Decision Process
SDS	Spoken Dialog System
TTS	Text-to-speech
VoiceXML	A standard XML format for specifying interactive voice dialogues between a human and a computer
WER	Word Error Rate

ABSTRACT

This research attemps to propose an approache to make deciscions under uncertainity for designing a dialogue manager for Amharic spoken dialogue system, Amharic as under-resourced language. A prototype Amharic Spoken Dialogue System was implemented on hotel and restaurant address information domain for experimentation. Data for this research was collected through a method called Wizard of Oz and domain knowledge is prepared using address search websites. How to design a dialogue manager which is robust for languages especially with low performing Automatic Speech Recognition unit is a fundamental question of this research. Previous studies on designing of spoken dialogue system for under resourced languages focused mainly on Automatic Speech Recognizer. We reviewed methods and frameworks on dialogue management. Design of Partially Observable Markov Decision Processes (POMDP) based dialogue manager, which provides a principled framework to plan under uncertainity, yields robustness. Low perfoming Spoken Dilaougue System (SDS) componenets, especially the Automatic Speech Recognizer (ASR) were considered the causes of uncertainity. Maintaining multiple hypotheses (evidences) improves the correctness of the dialogue manager. We conducted experiments to test correctness score, error rate and robustness. With a maximum of 6 N-best list and 20 partitions the correctness score grow by 14.19%. Increasing the number of n-best list of 6 reduced the error rate by 5.78% and with 6 n-best list. The belief updated below 0.12 seconds with 20 partitions and 6-nbest list. And the dialogue manager was able to complete a task with an average 8.75 turns by 50% Word Error Rate. The finding from the research illustrates that POMDP-based design approach to dialogue management is robust and possible to develop an improving spoken dialogue system for under-resourced languages.

CHAPTER ONE

INTRODUCTION

Spoken dialogue system (SDS) is one of the technologies that can be used to provide important information on a variety of tasks. According to Jurafsky and Martin (2006), Spoken Dialogue System can be defined as a computer program or an agent that converses with human users using natural language to perform a task. The tasks can be making travel arrangements, answering questions about weather or sports, routing telephone calls, acting as a general telephone assistant.

1.1. Background

In the recent years, SDS has been one of the growing research and development interest area due to various motivations such as its ease of use, its favorability for different users and its capacity to help developing countries to growth.

The technology of SDS complies with the general software interface principle that they should be simple to use for anyone. As indicated by Gould and Lewis (1985), any software interface designed for people should not pose any difficulty to learn and should contain features users really need in their task and should be easy to use. From the users' perspective, these interfaces are the points of interaction with the machine for some useful purpose and therefore should be intuitive and simple to use. In other words, to benefit from the machine's storing and processing information capabilities, interfaces should be simple enough. As Ahmed et al. (2013) describes, one approach to achieve the effort of making software interface simple, is through speech interface where users simply speak to a machine to get information, or to get some task done for them. Ahmed et al. (2013) argue, speech interface applies spoken language which is the most

natural, efficient, and flexible way of conveying information among human beings. For example, if we consider a keyboard input, it requires a typing skill. The Graphical User Interfaces (GUI) needs reading and appropriate actions that require a prior training. To communicate through speech, a user doesn't require training like the keyboard or should be acquainted with the GUI. Most software interfaces require learning the interface before using them. Speech interfaces enable easy human-machine interaction and thereby assist the efforts to make the software interface to be easy.

Hence, SDS can be designed for expert and non-expert computer users (Ahmed et al., 2013). There are conditions and potential situations which SDS is suitable for expert users with an experience of interacting with machines as well as for novices and non-expert users. One example situations for advanced users is hands free driving, which enables drivers to have their emails read, make a reservation for a hotel or find the nearest hotel or restaurant through speech while driving. Expert users can be relieved from routine tasks while engaged in some other tasks. Bell (2014) also points out, non-experts, computer novices and aged people are potential user groups for the present and future generation of spoken dialogue systems, since little or no training is needed to use speech based software. Spoken dialogue systems designed to provide information on current market pricing and health related request can work for users of all levels of expertise. Applications of Spoken Dialogue Systems can also benefit system designers for educational purpose, assistive technologies for the physically disabled and could be beneficial in oral societies (Bell, 2014; Gover et al., 2008).

As reported by Grover et al. (2008), there is a strong belief that SDS will have a considerable impact in the developing world. This is because firstly, speech based access to information may

enable illiterate or semiliterate people to take part in the information age. Second, telephone networks (especially mobile/cellular networks) are spreading rapidly. This can serve as infrastructure for SDS, and third, the strong oral culture that exists in many traditional societies is likely to yield more acceptable than text-based or GUI based sources. Bhagavan (1990) also explains that if a developing nation leapfrogged[1] to a newly emerged ICT, it would then be exposed to great potential in alleviating poverty and securing economic growth, as well as the possibility of surpassing developed and industrialized countries in economic development. One example of such technological leapfrogging is the shift of people in developing world from no land line phone to mobile phone (Fong, 2009). Mobiles are becoming ubiquitous. According to the figures on UN data, Mobile cellular subscriptions per 100 inhabitants in Ethiopia, grow from 2.37 in 2008 to 27.25 in 2013. A technology already available in the hands of millions of people can serve as an infrastructure to implement SDS and thereby quality information can be accessed and may facilitate technological leapfrogging.

1.2. Statement of the Problem

Spoken dialogue is easy and effortless for humans, but modeling it in artificial way is difficult (Bell, 2014). The motivations for building a spoken dialogue system for resourceful languages is the advances made in language technology such as Automatic Speech Recognizers, however the advance is far from solving the problems in building SDS (Lee et al. 2010). Current automatic speech recognition is an error prone technology and the fact that many of the commercial software are constructed to be mobile and used in noisy surroundings, aggravate this problem. In addition, spoken language is naturally ambiguous and uncertain (Chinael and Hamidreza, 2013) and therefore the state of the conversation can never be known with certainty (Williams and

[1] The use and utilization of advanced and state-of-the-art technology without using the technology before it can be referred to as

Young, 2005), controlling the dialogue flow (Cuayahuitl, 2005) and the problem of scalability (Thomson, 2009) are among the difficulties to resourceful languages.

An attempt to design an SDS for under-resourced languages like Amharic makes it more challenging. According to Solomon et al. (2012) this is because human language technology in Ethiopia is characterized by, among other things, lack of language resources. Furthermore, most researches in these areas are done only for academic achievement of individual students and hence there is a lack of integration. As indicated by Ege et al. (2009) the available resources are far from complete which makes developing SDS more difficult.

Due to the lack of language-resource, there is also a problem with the availability of off-the-shelf language technology components that a spoken dialogue system requires (except the text-to-speech unit for this prototype). Unlike the resourceful languages like English where commercial or open source language technology components which can be readily plugged into a dialogue system (Qiao et al., 2010) are available, important components such as Amharic ASR are not available. Moreover, spoken dialogue system needs other components for the task of natural language understanding, natural language generation, output generation and dialogue management (Jurafsky and Martin, 2006).

To satisfy timely and relevant information need of the society given that a language is under-resourced, one possible solution is to use key based dialogue system using mobile phones. According to Zegeye (2007) and Zewdu (2010), the Ethiopian Commodity Exchange (ECX) has launched a Short Message Service (SMS) and Interactive Voice Receiver (IVR) system that allows customers to acquire information about general and their personal accounts on the exchange's floor, from anywhere in the country. It works by calling a number and choosing the

appropriate menu using number keys to get the information desired. It is designed to address the problem of insufficient market information on the agriculture sector. Another encouraging example of such implementation is the 8028 IVR/SMS System by Ethiopian Agricultural Transformation Agency (ATA)[2] that provides information on high-value crops and a wide range of agricultural activities. According to a fact sheet published (ATA, 2014), an average of 500 calls per day, 7,700 unique callers, and a total of 57,400 calls into the system has been reported. Most of today's IVR and transaction-processing applications employ a touch-tone or dual-tone multi frequency (DTMF) user interface. However, key based interactions using mobile phones are generally considered as cumbersome from the users' perspective (Grover et al. 2008). And from development perspective, as the system becomes more complicated, constructing them is time consuming and they are difficult to extend (Lison, 2013).

The other possibility is to design spoken dialogue systems employing handcrafted methods. As indicated by Grover et al. (2008), these systems require high quality speech recognizers. In addition, Grover et al. (2008) showed that users prefer to use SDS than DTMF. However, handcrafted SDS lacks robustness.

According to Taguchi (1993), a system is said to be robust if the design or control variable of the system are configured in such a way that requirements of the system can be fulfilled with variations in the noise or uncontrolled factors. As W. Eckert (1994) explained, robustness of a dialogue system is how to be tolerable of the system for the error such as speech misrecognition or noisy surroundings. Independent modules cooperate: one module might correct the other modules errors.

[2] www.ata.gov.et checked on 9/16/2014

Literatures show that, statistical methods are state of the art approaches for dialogue system in general and dialogue management in particular (Lee et al., 2010). In recent years, statistical methods are gaining much attention due to their robustness (Williams et al. 2005) (Young et al., 2014) (Li et al., 2013). However, there is a disparity among researchers that in order to build statistical spoken dialogue system, large amount of data is needed (Minker and Bennacef, 2004) (Ahmed et al, 2013) which is expensive to collect. On the other hand a newly emerging paradigm of "Deploy, collect and improve" (Young, 2014), allows researchers and system developers to collect data after deploying an initial SDS (preferably domain specific) and improve the system. This approach enables to collect data by recording real dialogues between the user and the system to improve the speech recognizer, the natural language understanding and also the dialogue manager. This method allows users to be in the loop for collecting data and evaluating the system.

In addition, studies such as (Young, Breslin et al. 2014 and Young, Gasic et al. 2012), showed that in a quite environment POMDP-based and state of the art handcrafted dialogue managers yielded comparable results. However, in noisy environments POMDP-based dialogue managers outperform handcrafted ones. The results of these researches showed that POMDP-based SDS are more robust than handcrafted systems. The similarity of an ASR in a noisy environments and ASR for under-resourced languages is a relatively higher Word Error Rate (WER). Therefore, it can be hypothesized that if POMDP-based dialogue manager showed robustness in noisy environment, it could also show the same behavior for low performing ASR unit.

The main goal of this research is to propose methods and approaches to design a dialogue manager for a less resourced language like Amharic. To the best of our knowledge, there are no researches conducted on Spoken Dialogue System for Amharic either handcrafted or statistical.

With this in view, this research more specifically aspires to answer the following research questions:

- Is it possible to design a dialogue manager which is robust for languages with low performing Automatic Speech Recognition and Natural Language Understanding units?
- How to design and model an improving and extensible dialogue manager?

1.3. Objective of the study

1.3.1. General objective

The general objective of this research is to propose methods of development of a task-oriented Amharic Spoken Dialogue System.

1.3.2. Specific Objective

To achieve the above mentioned general objective, the following specific objectives have been considered:

- To investigate suitable methods and algorithms of designing dialogue manager robust to ASR, NLU and user error
- To identify users' and system's action for the prototype
- To evaluate the performance of the dialogue manager

- To build a prototype end-to-end spoken dialogue system for address searching and for collecting data.

1.4. Methodology

This section describes the design adopted by this research to achieve the objective of proposing methods, algorithms and frameworks of modeling a dialogue manager, which is robust to ASR errors and evaluate performance of the dialogue manger. More specifically, the methods used to collect data, train ASR, determine user and system action, integrate system components and finally evaluating the dialogue manager are described. This research employs experimental research design.

1.4.1. Study Design

As a general framework, the user-centered design principles of (Gould et al. 1985) was used which includes studying the user and task, building simulation and prototypes and testing iteratively the design with users.

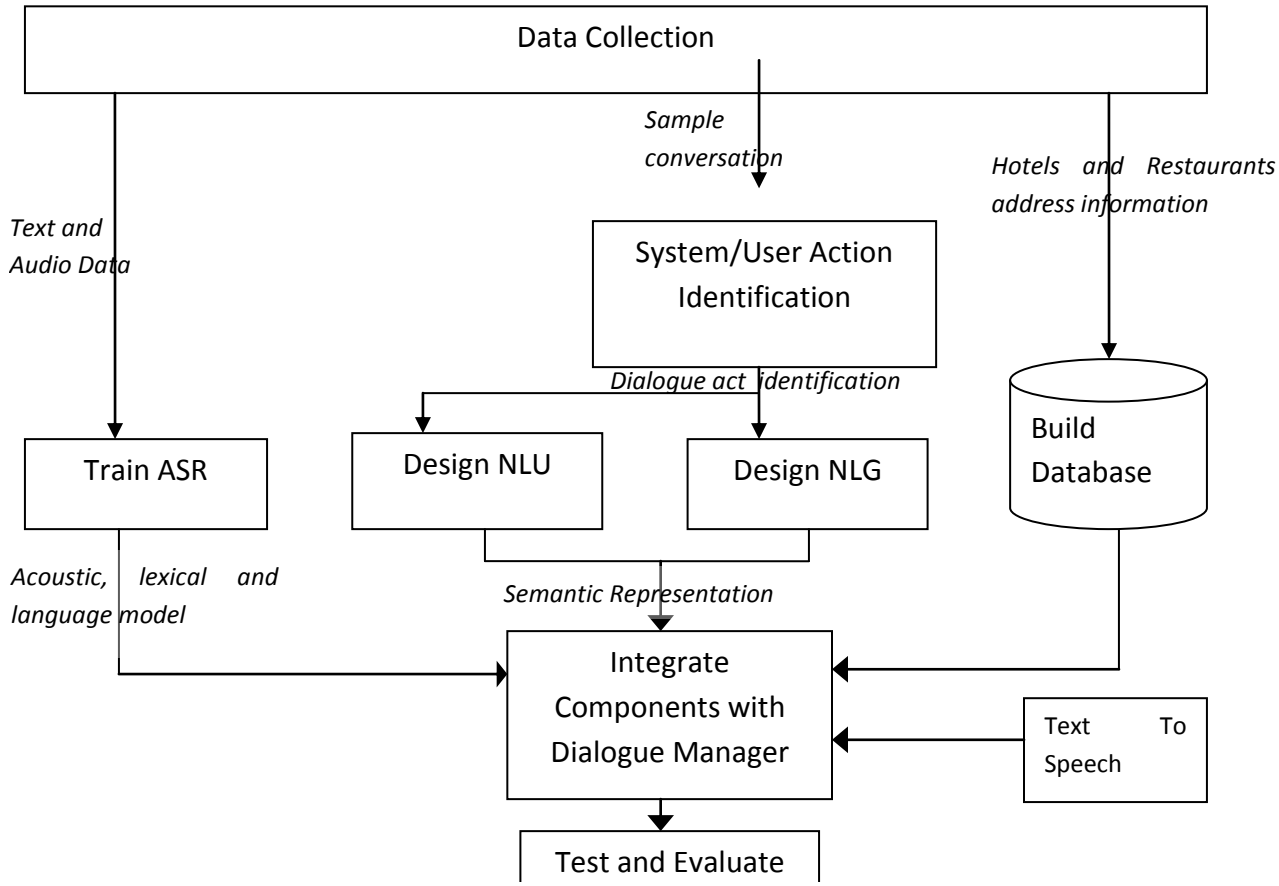


Figure 1. 1 Research Design

Figure 1.1 shows research design employed for this research. For the prototype of this research, data is required for identification of user/system action, for the training of Amharic ASR and for the domain knowledge. To identify system and user action a Wizard of Oz technique is used. Proceeding from identification of system and user actions, NLU and NLG are designed. A domain specific Amharic ASR is trained and adopted. For the prototype to work, each SDS component is integrated in a single frame to work with the dialogue manager. Lastly, the dialogue manager is evaluated for robustness, action completion and speed.

1.4.2. Data Collection and Preparation

For the experiment and for prototype development, domain specific data have been collected. The data was collected from two address search websites called Ethiopian Business Directory[3] and Afalagi Search Engine Plc[4]. For the prototype 156 address records of hotels, restaurants and cafes in Addis Ababa have been collected. The selection was based on the availability of the resource and records free from discrepancy. This data was used as domain knowledge for the spoken dialogue system. After interacting with the prototype, the final result can be fetched from a database constructed to hold these data.

A technique called Wizard of Oz (WOz) was employed to extract parameters for identification of user and system action and record audio to train the ASR. Wizard of Oz refers to an initial experimentation where a spoken dialogue system is mimicked by a human operator at the back end who communicates with potential users through initiating recorded prompts (Scheffler, 2002). In this way, a corpus was collected to draw out features for use in SDS development. Through these interactions, what queries and responses is required to complete a task has been identified (Sec 2.8.1). To identify what the user tries to accomplish by speaking to the machine has to be identified beforehand for to model the language understanding unit. Using WOz, for the domain selected 7 system actions and 4 categories of user actions and have been identified.

After deploying the prototype, speech data was collected using the framework. The interaction between real users can be recorded to retrain the system. The new collected data will allow the system to improve by retraining the ASR and modifying the language understanding unit.

[3] www.adrasha.com
[4] www.afalagi.com

1.4.3. Training Amharic ASR

The Amharic Automatic Speech Recognizer is trained using 132 domain specific sentences recorded and 100 adaptation sentences are used. The basic phrases for the prototype were 119. Five people participated in the recoding with about 2 hours of recording.

1.4.4. Design of Natural Language Understanding and Natural Language Generation Units

The task of Natural Language Understanding (NLU) is to convert user utterances to semantic representation and Natural Language Generation (NLG). This converts the semantic representation or the system act to a surface form. For both tasks rule-based approach was utilized. These units were simplified to avoid complexity due to conceptual errors. A word based dialogue with single user and system action is utilized for the prototype.

1.4.5. Integration and Evaluation

The components of SDS had to be integrated with the dialogue manager to work as a system. The prototype in this research employed pipeline architecture (Section 2.6).

The dialogue manager was evaluated for robustness, action completion and speed. It was evaluated for its ability to recover from error, the rate to complete a task on different circumstances and how fast it is. Since pipleline architecture is imployed and the NLU is simplified, errors are only emited from the ASR unit for the purpose of this research.

1.4.6. Prototype Development Tools

The tools described below were used in the process of collecting data, training the ASR, as a dialogue management framework and text processing. These tools were selected based on their ease of access and researcher's prior experience.

SUEDE: As described by Landay (2000, p. 1) SUEDE is a prototyping tool for voice enabled interfaces that allow rapid and iterative creation of such interfaces after identifying user and system actions. It supports iterative design, allowing a designer to quickly create an interface prototype, conduct user studies, and analyze the test data in a single tool. However, since the user/system actions and tasks are simple, this research did not use much of the capabilities of this tool.

HTK: The Hidden Markov Model Toolkit (HTK[5]) is a portable toolkit for building and manipulating hidden Markov models. The tools within HTK provide facilities for speech analysis, HMM training, testing and results analysis. The software supports HMMs using both continuous density mixture Gaussians and discrete distributions and can be used to build complex HMM systems. This toolkit was used to train Amharic ASR for the SDS.

Julius: Julius[6] is a high performance, continuous speech recognition software based on word N-grams. It is able to perform recognition at the sentence level with a vocabulary in the tens of thousands. Julius realizes high-speed speech recognition on a typical desktop PC. It performs at near real time and has a recognition rate of above 90% for a 20,000-word vocabulary dictation task. The best feature of the Julius system is that it is multipurpose. By recombining the pronunciation dictionary, language and acoustic models one is able to build various task specific systems. Monophone, triphone, or tied-mixture triphone models can be used. Julius uses HTK

[5] http://htk.eng.cam.ac.uk/.
[6] http://julius.sourceforge.jp/en_index.php

(Hidden Markov Toolkit) HMM definition files. Tied-mixture models are recognized by the joins within the tied-mixture codebook. Julius automatically detects the model type at startup. This research employed Julius as a live speech recognition engine. Initially, this tool is selected for this research because of it is incorporated in a tool called WaveSurfer[7], an open source tool for sound visualization and manipulation (Salvi and Vanhainen, 2014). According to Salvi and Vanhainen, there are other free ASR development tools such as CMU Sphinx and Kaldi, however these tools usually require certain degree of expertise from the user for setting up and running it.

eSpeak: eSpeak[8] is a compact open source speech synthesizer software for several languages which Amharic is also included. It runs on Linux and Windows environments. eSpeak uses a "formant synthesis" method. This allows many languages to be provided in a small size. The speech is clear, and can be used at high speeds, but is not as natural or smooth as larger synthesizers which are based on human speech recordings. eSpeak was used as an Amharic text to speech unit for this research.

ASDT: ASDT (AT&T Statistical Dialogue Toolkit) is developed by Jason Williams. It is developed using a Python programming language. The package contains a major module called "Partition Distribution" to track multiple hypotheses for dialogue states, which can be used to build a POMDP-based Spoken Dialogue Systems. ASDT was the most important tool for this study.

An end-to-end spoken dialogue system is incorporated for English language based on cloud ASR and TTS which requires AT&T sign up. It has three types of dialogue managers implemented

[7] http://sourceforge.net/projects/wavesurfer
[8] http://espeak.sourceforge.net

based on inititiation (Section 4.1.2.4). The tool also includes a grammar for recognizing according to the type of dialogue manager used. The performance of the belief update is controlled by parameters such as number of partitions to track, number of N-best list and number of dialogue history to track. It is also possible to track dialogues offline.

Python: For text processing and to modify the main tool for this research i.e ASDT, Python 2.7 was used. Python is a simple but powerful programming language with functionalities for processing linguistic data (Bird et al., 2009).

1.5. Scope and Limitation of the Study

This study is more of designing the dialogue manger, not a system focused study. More attention was given to explore methods, approaches and algorithms to design a robust dialogue manager. A prototype has been implemented to a domain of searching addresses of restaurants and hotels in Addis Ababa for Amharic with a focus of testing robustness, speed and task completion of the dialogue manager. A Reinforcement Learning method was used to design the dialogue manager. More specifically, a type of Reinforcement Learning called Partially Observable Markov Decision Processes (POMDP) based dialogue manager with handcrafted action selection was employed (Section 2.7.2.2.2). The choice of this approach is based on its capacity to model uncertainty.

Concerning the evaluation of the dialogue manager, there was no base-line dialogue manager for to compare results. In addition, evaluation for user acceptance is not conducted since this study is not system focused.

As far as the study is concerned with extensibility and improvement, no other domain was tested other than hotels and restaurants to demonstrate the extensibility and the initial ASR is not retrained with the speech data collected from the framework due to time constraints. The ASR used was an HMM based, small vocabulary, user independent and spontaneous.

1.6. Significance of the study

The study will benefit and help the software developers as a starting point to understand the theoretical and practical framework in developing Spoken Dialogue Systems. It shows the current challenges and opportunities with respect to developing Spoken Dialogue Systems for local languages.

This study may serve researchers who have an interest in Spoken Dialogue Systems and other Human Language Technologies as a reference, guide and motivation for future researches. It may also serve as a framework with the new paradigm of "Deploy, Collect Data and Improve" to develop usable task oriented spoken dialogue systems to collect data and thereby improve components such as ASR and NLU and better understand user behavior.

Information on tourism-related services on the web is growing rapidly offering information on hotels, flights, tickets and the like (Malaka and Zipf, 2000). Each service uses different interface with different functionalities. Implementing SDS for address searching for hotels and restaurants will benefit customers due to simplicity of speech interface, hotels and restaurants providing information on their whereabouts.

1.7. Organization of the Paper

The first chapter introduces Spoken Dialogue Systems and the main challenge in developing SDS. It also presents the statement of the problem, objective, methodology, scope and limitation of the study and the significance of the study.

Chapter Two: discusses the theoretical background of spoken dialogue systems and previous research work in the area of Spoken Dialogue Systems and the different approaches attempted to develop an SDS with particular attention to dialogue management. Finally, it discusses about related works.

Chapter Three: describes in the methodology of data collecting, designing, architecture of SDS.

Chapter Four: describes about experimental setup and prototype implementation detail of hybrid of POMDP and handcrafted dialogue management algorithm. This chapter discusses experimental results followed by discussion and finding of the study.

Chapter Five: the final chapter provides the conclusion and recommendations drawn from the findings of the study.

CHAPTER TWO

LITERATURE REVIEW

In this chapter an attempt is made to introduce relevant concepts and approaches utilized in the field of Spoken Dialogue Systems with particular stress on dialogue management.

From users' point of view, spoken dialogue system can offer various advantages since these computer agents can be engaged in work 24 hours a day without getting tired. In addition, they could be adapted to provide personalized information or treat all people in the same way (Pérez-Marín, Pascual-Nieto 2011). The future dialogue agents are expected to learn and adapt to new domains and languages easily (Jokinen, 2000).

2.1. Overview of Spoken Dialogue System

Spoken dialogue system is a system that detects speech input, provides meaning in the current context, select appropriate action and finally generates an answer by means of a text-to-speech synthesis or pre-recorded prompt (Bell, 2014).

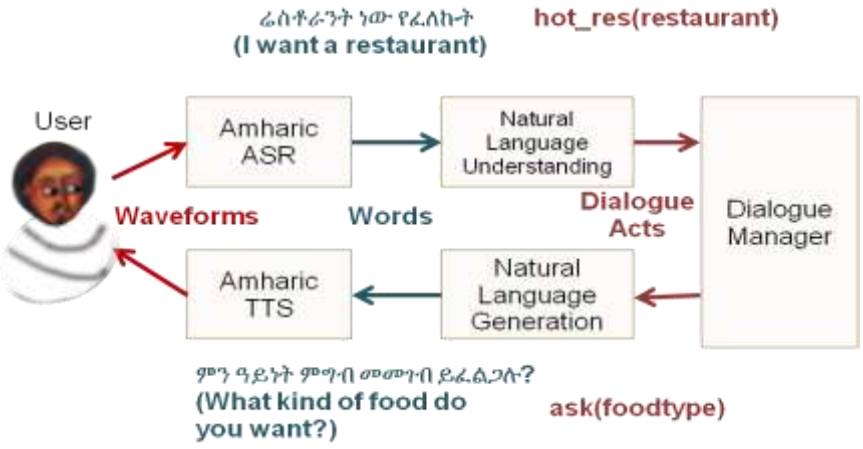


Figure 2. 4 Typical Structure Of A Spoken Dialogue System

Figure 2.1 provides a typical structure of a spoken dialogue system with its process. The wave form of the utterance of the user is picked up by the recognizer and converted into a string of words. The natural understanding unit (NLU) assigns a semantic representation for the string which is a dialogue act. For example the string "I want a restaurant" is changed to an already defined dialogue act that determines whether the request is hotel or restaurant i.e. "hot_res(restaurant)". The dialogue manager accepts a dialogue act (user act) and selects an appropriate action (system act) in this case it selected an action for asking the food type "ask(foodtype)". The system act has to be converted to a surface form or into words for the user to understand. This is a task of the natural language generation unit. Finally the text-to-speech renders the generated text to an audible format.

According to Lee et al. (2010) SDS can be viewed as advanced application of spoken language technology. SDS comprises of other natural language processing components such as Text to Speech (TTS) and Natural Language Understanding (NLU) in an integrated manner making its architecture complicated.

However, for many resourceful languages, SDSs are becoming ubiquitous due to their rapid improvement in performance and decrease in cost (Lee et al. 2010). Some examples are Apple's Siri[9], Google's Google now[10] and Nuance's Dragon Go[11]. There are also attempts to develop SDS for under-resourced languages for the purpose accessing information (Plauché et al. 2006) (Grover et al. 2008).

[9] http://www.apple.com/ios/siri/
[10] http://www.google.com/landing/now/
[11] http://www.nuance.com/for-individuals/mobile-applications/dragon-go/

Thomson (2009) stresses, to look at the dialogue system in a relatively simple way some assumptions should be considered. First, only dialogues with two participants are considered. Second, all interactions between the system and the user are in the form of turns and third, task-oriented dialogue systems which are designed to accomplish a well defined task are considered.

2.2. Human to Human Conversation

In order to model human-machine interaction it is imperative to understand some properties of human-human conversation.

2.2.1. Turn-taking

In human-human conversation people take turns to talk; they know who should talk next and when they should talk (Jurafsky and Martin, 2006). A turn in a dialogue is a period in which one of the participants has a chance to say something to the other participant and turn taking means to one participant taking over or giving up the turn in a dialogue (Bühler, Minker 2011, p. 16). Turn-taking is often hurried and might seem to be entirely irregular. However, they are highly predictable and at least to some extent rule-governed (Bell 2014).

Because of the limitations of existing technologies, conversations between humans and machines are uncomplicated and more controlled than human conversations (Jurafsky and Martin, 2006). In most current dialogue systems, waiting time (between ½ and 1.0 second) to detect for the machine end of a turn is manually predetermined. However, such constraints pay no attention to important conversational phenomena such as interruptions, speech overlap, backchannels and co-completion of utterances (Lison 2013, p. 25).

2.2.2. Speech Act

As explained by Austin (1962) and Searle (1969), there is more in communication than simply information transfer. In other words, utterances have goals and often do things besides conveying information. They are therefore best expressed as actions or speech acts rather than abstract statements about the world. For example, in a wedding ceremony when the officiators say "I pronounce you husband and wife", they are not communicating information, but rather changing the couples' states from singles to a married couple.

Austin (1962) maintains three kinds of speech acts: locution:- what is actually said by a speaker, illocution:- what is accomplished by what is said and perlocution:- what the hearer does in response to the utterance. Under illocutionary acts, Searle (1979) introduced a taxonomy of speech acts:

Assertive: these are information convey kind of utterances which statements tell how things are.

Directives: the speaker tries to get someone to do something through commands, persuasive speech or invitation

Commissive: the speaker commits himself/herself to do something.

Expressives: the speaker expresses feelings and attitudes

Declarations: bring about changes through the utterances (the wedding example above changes the reality where the state of the couple changes after the declaration).

As indicated by Bell (2014), understanding speech acts can help us learn the general organization of natural discourse. The specific speech act structure may vary depending on the task and domain of the dialogue, and its degree of formality. Speech acts are related to dialogue pairs or

adjacency pairs (Section 2.2.3) in the sense that commonly occurring pairs like question-answer make up two subgroups in the much larger array of speech act categories.

For spoken dialogue system, the original idea of "speech acts" has been modified to "dialogue act" tags to include actions relating to turn-taking, social attitudes and grounding (Traum, 1999). For example there is a dialogue act for turn taking, "take-turn" and "give-turn" can be the possible acts. For confirmations and affirmations, dialogue act tags allow actions such as "confirm" and "affirm". When the agent tries to "confirm" action to reach at some level of certainty, it might ask "Did you say you wanted traditional food?" and from the user side, an "affirm" act might be used to represent "Yes!".

In addition, according to Thomson (2009), in the traditional definitions of dialogue acts, the semantic information is separated from the act. For a dialogue system however, a simplified form of semantic information should be provided for the agent. For instance, if a user is asking for a hotel, it is important to the system that it is a hotel that is being requested and the dialogue act can be represented as "request(hotel)" attaching the act and the meaning together.

2.2.3. Dialogue Pairs

Adjacency pair or dialogue pair is the expected response for a given utterance. There are utterances which require certain expected response such as greeting should be followed by greeting and a question should be followed by an answer. To modeling dialogue, these pairs and the dialogue expectations should be identified (Jurafsky and Martin, 2006). For example as Yimam (1997) illustrates, in Amharic when two people meet for the first time expression and the response is as follows:

Expression: ጤና ይስጥልኝ

'May (God) give (you) health on my behalf

Response:አብሮ ይስጥልኝ

'May he give us together on my behalf

In human-human conversation, if the response to a question is silence, it could be interpreted as significant silence i.e. a refusal to respond. The silence due to slow operation of ASR unit for instance, could be interpreted as significant silence (Jurafsky and Martin, 2006).

2.2.4. Conversational Implicatures

As shown by Grice (1989), there are implied meanings from the context in speech acts. Grice formalized these ideas in terms of a cooperative principle composed of four conversational maxims that are assumed to be true in a natural conversation: the maxim of quality ("be truthful"), the maxim of quantity ("be exactly as informative as required"), the maxim of relation ("be relevant"), and the maxim of manner ("be clear"). Cooperation principle and these maxims enable the listeners to infer more than what is literally being said.

Benotti and Blackburn (2011) showed that the ability to reason by deduction of causal implicatures provide a room to make decisions about "how to determine the next upcoming actions " and "when to request for clarification"; which will help to model task-oriented dialogue as a sub-dialogue to negotiate of the underlying task. For example:

Mary: The chest is locked, the crown is inside

Bill: Give me the crown

Bill causally implicated: Open the chest

Benotti and Blackburn (2011)

Benotti and Blackburn (2011) used an Artificial intelligence classical planning algorithm called FastForward to handle the sub-dialogue for their dialogue manager Frolog. It requires the initial state, the goal and the available actions and outputs a sequence of actions.

2.2.5. Grounding

During conversations, participants in the conversation make sure there is mutual understanding of each others' contribution. In addition, they also show they are listening (Lison 2013, p. 28). Participants repair, extend or replace the noun phrase under discussion (Bell, 2014). Saying things like "Yes", "Uh-huh", etc. is natural in human-human conversation to indicate they are listening, sometimes finish other people's sentences and refuse to behave in a turn-taking manner (Thomson, 2009).

2.3. Dialogue System Components

Typical spoken dialogue system (Figure 2.1) consists of the Automatic Speech Recognizer (ASR), Natural Language Understanding (NLU), Dialogue Manager (DM), Natural Language Generation (NLU) and Text to Speech (TTS) units (Jurafsky and Martin, 2006).

2.3.1. Automatic Speech Recognition

Automatic Speech Recognition is a process of converting string of acoustic signal into a sequence of words (Ahmed et al. 2013). i.e. it finds the most likely word sequence $W*$ given an acoustic observation O and a probability model P(W|O) (Bühler, Minker 2011, p. 5). The observation is usually a sequence of acoustic features vectors calculated on a frame basis. Mathematically:

$$W* = \arg\max_w P(O|W)P(W) \qquad (2.1)$$

Equation (2.1) shows the commonly accepted factorization of the stochastic model into two distinct components. The formula shows the acoustic model P(O|W) determines a probability to an acoustic observation conditioned by a word sequence and the language model P(W) assigns the probability of a word string given the sequence of words. Most ASR use Hidden Markov Models (HMM) to represent the acoustic model (Bühler, Minker 2011, p. 5).

According to Ege et al. (2009, p. 1), ASR systems may be categorized as 1) speaker dependent or independent based on system's capability to recognize speech from new speaker or from people whose speech is used during the development of the recognizer. 2) Isolated or continuous speech based on whether the speaker must pause between words or speaks naturally. 3) Small, Medium or Large Vocabulary system based on vocabulary size as small (1 to 99), medium (100-999) and large (1000 or more) words. 4) Read or spontaneous speech based on the system's capacity to recognize read speech or spontaneous speech. 5) Noisy or Noise free speech based on system's capacity to recognize speech in noisy or only in noise free environment.

Morbini et al. (2013) presents some of the major criteria for selection of a speech recognizer for a dialogue system. These criteria include whether it is customizable (e.g. add vocabulary), output option (e.g. ranked n-best hypothesis, incremental results), performance (response speed) and output quality (accuracy). N-best list is, a list of speech recognizer's guesses for an utterance.

2.3.2. Natural Language Understanding (NLU)

The natural language understanding unit receives the word sequence recognized by the speech recognizer and gives semantic representation it (Ahmed et al. 2013). NLU may also comprise of

other Natural Language Processing modules like morphological analyzer, part of speech tagger and parser to analyze the user's utterance (Lee et al. 2010). For example NLU unit maps the pre-processed utterance to a meaning representation ባህላዊ ምግብ መመገብ ፈልጌ ነበር (I want to eat traditional food) is converted to foodtype(Traditional) in which the dialog act and user goal are extracted.

NLU can be designed using rule-based approaches based on frame-and-slot semantics (Jurafsky and Martins, 2007). Rule-based approaches are better for permitting to specify linguistic knowledge. However, they are fragile and rigid, especially with speech recognition errors. Statistical (data-driven) models, on the other hand, are usually more robust and efficient (Bühler, Minker 2011, p.5). For example unsupervised method is proposed by Chen et al. (2013) for automatic training and filling of semantic slots from unlabeled speech data and supervised approach by Moreira et al. (2011) by taking process of NLU as a classification problem given set of user acts. However, according to Bühler and Minker (2011, p. 5) data driven methods require collecting and preparing a realistic and extensive corpus of training data.

2.3.3. Natural Language Generation

Natural Language Generation (NLG) unit: chooses the concepts to express to the user, plans out how to express these concepts in words, and assigns any necessary prosody to the words (Jarafsky and Martin, 2006). Appropriate response to the user query is planned and executed with this NLG unit.

Language generation modules are implemented in one of two ways, templates and generators (Jarafsky and Martin, 2006). The simplest approach for natural language generation is to use

templates. As an example, a template might transform “ask(foodtype)” into “የምግቡ አይነት ባህላዊ ወይስ ዓለም አቀፍ ይሁን? (What kind of dish would you like? Traditional or international?). This method is known as template-based generation, and the sentences created by these templates are often called prompts. While most of the words in the template are fixed, templates can include some variables. A second method for language generation relies on techniques from the field natural language generation. Here the dialogue manager builds a representation of the meaning of the utterance to be expressed, and passes this meaning representation to a full generator. Such generators generally have three components, a sentence planner, surface realize, and prosody assigner.

2.3.4. System Output

The system output is usually done through a text-to-speech (TTS) module or pre-recorded audio (Ahmed et al. 2013). The surface form from the NLG unit is converted to an audible format using a TTS unit. According to Ahmed, TTS plays a very important role in a SDS because the quality of the synthesized speech represents the SDS from the users' perspective. Intelligibility and naturalness are the measures of quality of speech synthesis.

2.3.5. Dialogue Manager

Dialog Manager (DM) is the central unit of the spoken dialogue system because it coordinates the activity of all components, controls the dialog flow, and communicates with external applications (Lee et al., 2010). An appropriate action in terms of providing information, asking for clarification, or confirmation is selected by the Dialogue Manager. The dialogue management

module is also responsible for detecting and repairing errors in the dialogue through verifications, confirmations and corrections (Ahmed et al., 2013).

2.4. Types of Dialogue Systems

The types of dialogue system may be classified based on their modality and initiative of the dialogue manager.

Even though this work is on a system that uses audio as input and output, there are systems that are text based. A medium for what the system accepts as input or output (such as voice, gestures, touch screens or graphical user interfaces) is referred to as a modality and a system that uses more than one modality is called to as a multi-modal system (Scheffler, 2002).

“Initiative” of a dialogue system is concerned with who controls the dialogue. Accordingly there are three types of initiatives (Martin and Jurafsky, 2006): system-initiative, user initiative and mixed initiative. In the case of system initiative, the system has the initiative to guide the dialog at each step. In the user-initiative, the user takes control of the dialogs, and the system responds to whatever the user directs. The user initiative and system initiative can be labeled as single initiative because either the system or the user controls the conversation. If both user and system have a chance to control to some extent, this is called mixed-initiative. The fact is that the system has overall control of the dialogs. However, the users can barge in and change the dialog direction.

According to Bühler and Minker (2011, p. 26) the models have their own advantages and disadvantages. For example, novice users and in noisy environments, system initiative might be preferable because errors can be minimized since the system specifies what the user need to say

and strategy is more robust to progress in the dialogue. On the other hand, expert users prefer to assume the initiative in the dialogue. This is because they know what the system expects. From design perspective user initiative is more difficult than system initiative, because what the user might say is unpredictable, which increases the chances of misinterpreting the user.

2.5. Models of Dialogue

Plan-based Models: In this model, each dialogue participant is modeled as an agent having explicit goals, basing their acts on their respective goals and adapting their goals based on the inferred goals of the dialogue partner (Scheffler, 2002). Instead of only trying to determine what the user wants, one should aim to determine a shared plan. This is particularly relevant in collaborative dialogues, such as language learning, where both participants must work together to achieve a task. Dialogue systems based on this idea are said to use plan-based or agent-based dialogue managers (Thomson, 2009). An advantage of a plan based model is that general behavior rules may be stated which a dialogue system has to obey. Such rules of behavior may include cooperation, helpfulness, and sincerity. They may be, for instance, based on the Gricean maxims (Bühler and Minker 2011, p. 27).

Information State Model: A central idea in information state model is that dialogue acts correspond to dialogue moves and are used to update an information state subject to certain preconditions. The information state represents the accumulation of everything that has happened in the dialogue, and is used by the dialogue manager to choose its next action according to an update strategy. Logic programming is one possible method of implementing this update strategy (Thomason, 2009).

Information state-based models are a general way of implementing dialogue to implement finite-state, frame-based, or more complex models (section 2.7.1) (Bühler and Minker 2011, p. 27).

2.6. Architectures of Spoken Dialogue Systems

The common architectures to combine SDS components are the pipeline approach and blackboard approach. Lison (2013) describes a pipeline approach as simple and straight forward because the components are arranged sequentially from the speech recognition unit to the speech synthesis unit while blackboard architectures revolve around a central dialogue state and each component connected to it to change and monitor its value. Pipeline systems are criticized for the rigidity of information flow and poor turn-taking capabilities. While the blackboard approaches allow more flexible information flow and components can be designed to take advantage of contextual information.

2.7. Approaches to Dialogue Management

In this section, an attempt is made to review common approaches to dialogue management. As the major task of the dialogue manager is to control the flow of the dialogue (Gasic, 2011), handcrafted or data driven approaches may be employed.

2.7.1. Hand-crafted Dialogue Management

A simple way of designing a dialogue manager to play its role is to define a set of rules that the system follows during the course of the dialogue (Gasic, 2011). The dialogue manager asks questions until all necessary features are fulfilled. This approach follows a system-initiated dialogue discussed above.

2.7.1.1. Finite State Automata

According to Lison (2013), the simplest approach to dialogue management is based on finite-state automata (FSA) where FSA is defined by a collection of states and directed edges between them. Decision making is made possible by associating each state with a specific action to execute at that state. Each edge in the automaton is labeled with a condition on the user input that, if satisfied, will move the current state from the source of the edge to its target (Lison, 2013). When using a Finite State Automaton (FSA) model for dialogue management, each node of the FSA represents a dialogue state with the associated system questions and the arcs denote possible user answers. Current commercially deployed dialogue managers are based on a similar structure called call-flow, which makes use of high-level specification languages such as VoiceXML to implement the functional specification and the detailed design of the interaction.

As stated by Bühler and Minker (2011), finite-state models offer a high degree of system control, which is vital in many environments. However, as (Gasic, 2011) indicated, this approach has several limitations: 1) it is difficult to extend because system design depends heavily on the domain 2) it has no capacity to automatically improve the dialogue manager's behavior and 3) these systems are sensitive to speech understanding errors and require hand-crafted error handling schemes to deal with these errors, typically in the form of a dedicated error-handler.

2.7.1.2. Frame based or Form based

The other option using the approach called frame-based or form-filling to design dialogue manager (Jurafsky and Martin, 2006). Frame-based approaches expect a defined set of concepts that the user can speak about, called slots, which take on values from a pre-defined set. The filled

slots determine the state of the dialogue. The dialogue manager controls the flow by using a pre-specified action for each filled slots. This approach allows users to freely fill a slot in any order and fill extra slot at a time, therefore it supports mixed initiative (Thomson, 2009). Note that if these slots and fillers correspond to the attributes and values of the underlying database, the complexity of these systems can be expressed in terms of the number of slots and the number of fillers they support (Gasic, 2011).

2.7.1.3. Agenda based

The agenda based approach is motivated by sub-dialogues of frame based approaches (Thomson, 2009). The form-filling approach is further extended in the agenda-based dialogue management framework to support more complex dialogues. In the agenda-based approach, the dialogue consists of sub-dialogues and each sub-dialogue has an agenda that directs its flow. While this approach is more flexible, it does not support automatic optimization of the dialogue manager and does not replace the need for a separate error handler that deals with speech understanding errors (Gasic, 2011).

2.7.2. Data-driven dialogue management

Handcrafted systems offer a better way of understanding of how to structure, dialogue manager (Thomson, 2009). However, these approaches are time consuming to construct, difficult to extend and poor in handling uncertainties.

The basic goal of the data driven approach is to find a way of creating and optimizing dialogue policy automatically, achieving domain independent dialogue management and handling uncertainties (Lison 2013, p.39). As indicated by Ahmed et al. (2013), spoken dialogue systems

have to deal with the inherent uncertainties from ASR and NLU units. This is because ASR and NLU units make errors.

2.7.2.1. Supervised Learning

Supervised learning finds an approximate function given enough training examples (Mitchell, 1997, pp. 23). Dialogue management can be considered as a classification problem to utilize supervised techniques to create dialogue policy. According to Lison (2013), it is possible to train a classifier by providing dialogue examples which can be human-human or human machine interactions. A Wizard of Oz approach (Section 3.2.1) can also be employed to collect data. The dialogue policy can then be estimated by employing classifiers such as decision trees and logistic regression.

Unlike the handcrafted systems, these techniques provide a portable and extendable dialogue manager (Gasic, 2011). The policy constructed from supervised techniques allows dialogue manages to be used across different domains.

However, there are shortcomings to this approach. First, it requires large amount of training data. Even a very large dialogue corpus would represent only a small portion of the total set of reasonable dialogues. In addition distinct instances of state variables can grow at least exponentially therefore faces data scarcity issues. Second, it tends to select the same action to similar dialogue in the training. To put it differently, it neglects temporal connections between dialogue states (Lison, 2013) (Gasic, 2011).

2.7.2.2. Reinforcement Learning (RL)

As we have seen in the previous sections, handcrafted approaches are time consuming to construct, difficult to expand and there is no principled way of handling errors. On the other hand, supervised methods need large amounts of data and yet cannot represent all the dialogue. The action selection is also confined to the one that occurs in the collected corpus. A method that can explore all possible behavior of dialogue states and that can automatically create policy and optimize it is desirable.

According to Sutton (1998), reinforcement learning (RL) is a process of learning what action to take and how to map situations to actions with a goal of maximizing a numerical reward by trial and error. Using RL approach, dialogue can be modeled as a sequential decision process and the dialogue management behavior is optimized with respect to the reward (Gasic, 2011). RL provides a principled solution to the task of dialogue policy optimization under uncertainty.

RL methods involve an agent (machine), environment and actions. The agent interacts with the environment through its action resulting in a change of state. The environment rewards or penalizes the agent for its action. The goal of the agent is to find the best action through trial and error. The best action is the one that maximizes the agent's expected long-term reward. That means the agent learns from its environment by taking actions and receiving rewards (Lison, 2013) (Gasic, 2011).

2.7.2.2.1. Markov Decision Processes (MDP)

A Markov decision process (MDP) is a type of RL that starts with the assumption that dialogue operation consists of a set of states that a machine agent can be in at every particular time step,

S, and a set of actions, *A*, that it can take in these states. At each time step *t* the machine is in a state s_t and takes an action a_t. It will then make a transition to the next state s_{t+1} and receive a reward r_{t+1} (Zhang, 2013; Gasic, 2011). The transition probability T is defined as $P(s' \mid s,a)$; R defines the expected (immediate, real-valued) reward $r(s,a)$ (Zhang, 2013).

$$r_{t+1} = T(s_t, a_t, s_{t+1}) \qquad (2.2)$$

Policy π which is a mapping from state to action assumes that every state variable is fully observable (Blythe, 1999). Deterministic π can be used to select actions for the agent to take. As π is optimal, for state s_t it returns action a_t, (Zhang, 2013).

$$a_t = \pi(s_t) \qquad (2.3)$$

which maximizes long term return defined (Zhang, 2013).

$$\Theta = \sum_{k=0}^{n} \lambda^k r_{t+k+1} \qquad (2.4)$$

where r_i the reward defined in (2.2) (i=t+1, t+2,…) and λ is future reward discount factor ($0 \leq \lambda \leq 1$)

The deterministic π (s) is calculated (Zhang, 2013)

$$\pi(s) = \arg\max_a Q^{\pi}(s,a) \qquad (2.5)$$

$Q^{\pi}(s,a)$ is a value function

Whereas, stochastic π (s) where action is selected randomly, is defined (Zhang, 2013)

$$\pi(s, a_k) = \frac{Q^{\pi}(s,a_k)}{\sum_{i=1}^{n} Q^{\pi}(s,a_i)} \qquad (2.6)$$

Where n is the number of possible actions that can be taken in state s, and ($0 \leq k \leq n$). When π is optimal, π (s, a) returns the probability that a may maximize i.e. the long term benefit when it is taken in state s.

2.7.2.2.2. Partially Observable Markov Decision Processes (POMDP)

A limitation faced by MDP approaches is the assumption that the current state is fully observable. This assumption does not hold for most dialogue systems, because of the presence of multiple sources of uncertainties such as noisy or low performing ASR. A solution to this drawback is to extend the MDP framework by allowing the state to be a hidden variable that is indirectly inferred from observations. Such modification is called Partially Observable Markov Decision Process (POMDP) (Lison, 2013).

Partially Observable Markov Decision Processes (POMDPs) have gained recent popularity in the application of spoken dialog systems (Young et al., 2012). States are partially observed because of noise. Uncertainty in user speech and intention can be modeled using POMDP. In addition, POMDPs reduce number of parameters that need to be tuned to produce useful behavior in other

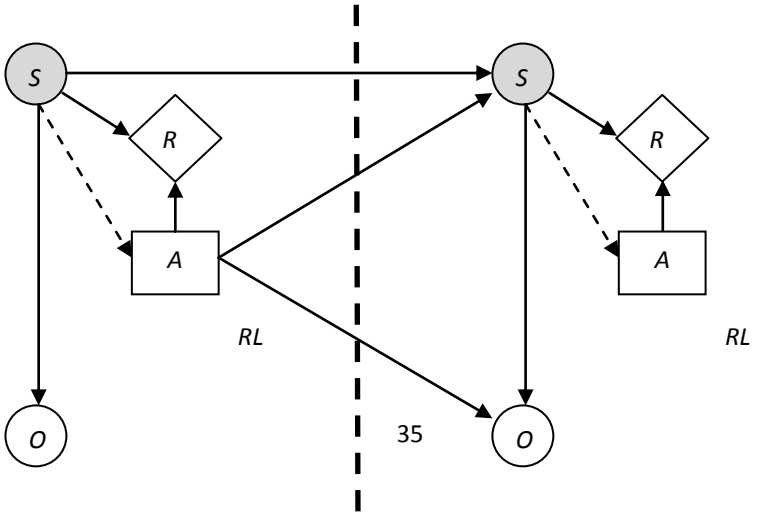


Figure 2. 5 Influence Diagram of Standard POMDP (Williams and Young, 2005)

models for example to handle errors.

POMDPs can be expressed as “planning under uncertainty”. This is an artificial intelligence concept to design agents that can plan in incomplete or faulty information where actions may not always yield the same results as well as where there may be tradeoffs between the different possible outcomes of a plan (Blythe, 1999).

As a mathematical framework formal definition of POMDP is provided as a tuple $\{S, A, T, R, O, Z, \lambda, b_0\}$ where S is a set of states where the agent can be in; A is a set of actions that an agent may take; T defines a transition probability $P(s' \mid s, a)$; R defines the expected (immediate, real-valued) reward $r(s, a)$; O is a set of observations the agent can receive about the world; and Z defines an observation probability, $P(o' \mid s', a)$; λ is a geometric discount factor which is between 0 and 1 i.e. $0 \leq \lambda \leq 1$; and b_0 is an initial belief state $b_0(s)$ (Williams and Young, 2005).

Figure 2.5 shows standard POMDP. Circles indicate random variables (unobserved if shaded and observed if un-shaded). Squares represent decision nodes; and diamonds represent utility (reward) nodes. Solid directed lines indicate causal effect and dashed directed lines indicate that a distribution is used (and not the actual unobserved value). The subscript RL indicates that actions are selected using Reinforcement learning.

According to Williams and Young (2005) at each time-step, the environment is in unobserved state $s \in S$. Since the agent cannot determine s directly, a distribution over states is maintained called a “belief state,” b, which a kind of inference with initial belief state b_0. To indicate the probability of being in a particular state s, the belief is expressed as $b(s)$. The action selection is based on the belief b, the machine selects an action $a \in A$, receives a reward $r(s, a)$ which means

the reward is determined by the action and the state, and transitions to state s', where s' is an unobserved state, depends only on s and a. The machine then receives an observation $o' \in O$ which is dependent on s' and a. In other words, the environment tells to the agent for example "you are in state 5 and you have four possible actions". When the agent takes an action based on its belief (understanding of the world state with uncertainty) for instance action 2, the environment tells to the agent "because you took that action, you will receive a reinforcement of 7 units" which can be a reward or penalty. At each time-step, the belief state distribution b is updated (Williams and Young, 2005).

$$b'(s') = p(s'|o', a, b)$$

$$= \frac{p(o'|s', a, b)p(s'|a, b)}{p(o'|a, b)}$$

$$= \frac{p(o'|s', a)\sum_{s \in S} p(s'|a, b, s)p(s|a, b)}{p(o'|a, b)}$$

$$= \frac{\mathrm{p(o'|s',a)}\sum_{\mathrm{s \in S}} \mathrm{p(s'|a,s)b(s)}}{\mathrm{p(o'|a,b)}} \qquad (2.7)$$

Equation 2.7 can further be simplified by considering the denominator as normalization constant k, since it is independent of s'. The numerator consists of the observation function Z, transition matrix T, and current belief state b. Hence, equation 2.7 can be re-written (Williams and Young, 2005).

$$b'(s') = k.p(o'|s', a)\sum_{s \in S} p(s'|a, s)b(s) \qquad (2.8)$$

Updating the value of *b* at each time-step is called "belief monitoring" (Williams and Young, 2005). The value *b* is a very complex number and has useful property since it is a complete summary of the dialog history. It provides a proper *sufficient statistic* given initial belief state b_0 and history $\{a_1, o_1, a_2, o_2,...\}$. The belief *b* is Markovian with respect to b_0 and $\{a_1, o_1, a_2, o_2,...\}$. When computing a new belief state, the update (Equation 2.8) takes into account all possible state transition histories. Planning algorithms need only consider *b* when selecting actions.

As mentioned above, at each time-step, the agent receives reward r_t. The cumulative, infinite-horizon, discounted reward is called the *return* (Williams and Young, 2005).

$$\Theta = \sum_{t=0}^{\infty} \lambda^t r_t \qquad (2.9)$$

where λ is the geometric discount factor, $0 \leq \lambda \leq 1$. The goal of the machine is to choose actions in such a way as to maximize the expected return $E[\Theta]$ – i.e., to construct a *plan* called a *policy* which indicates which actions to take at each turn. In general, a policy π can be viewed as a mapping from belief state to action $\pi(b) \in A$, and an *optimal policy* $\pi^*(b) \in A$ is one which maximizes $E[\Theta]$.

2.7.2.2.3. SDS-POMDP

The classical POMDP framework must be factored to allow the user's goal, the user's intention and dialogue history. In a classical POMDP (Figure 2.2), the POMDP state S expresses the unobserved state of the world. This state can naturally be factored into three different components: the user's goal S_u, the user's action A_u, and the dialog history S_d. Hence, the factored POMDP state S is defined (Williams and Young, 2005).

$$s = (s_u, a_u, s_d) \qquad (2.10)$$

Hence, the system state S_m that maintains the internal state, becomes the belief state b over s_u, a_u, and s_d (Williams and Young, 2005)

$$s_m = b(s) = b(s_u, a_u, s_d) \tag{2.11}$$

Recognition result $\widetilde{A_u}$ which is partially observable (noisy) and the confidence score C will then be cast as the SDS-POMDP observation O (Williams and Young, 2005). These are the evidences that aggregate to give observation.

$$o = (\widetilde{a_u}, c) \tag{2.12}$$

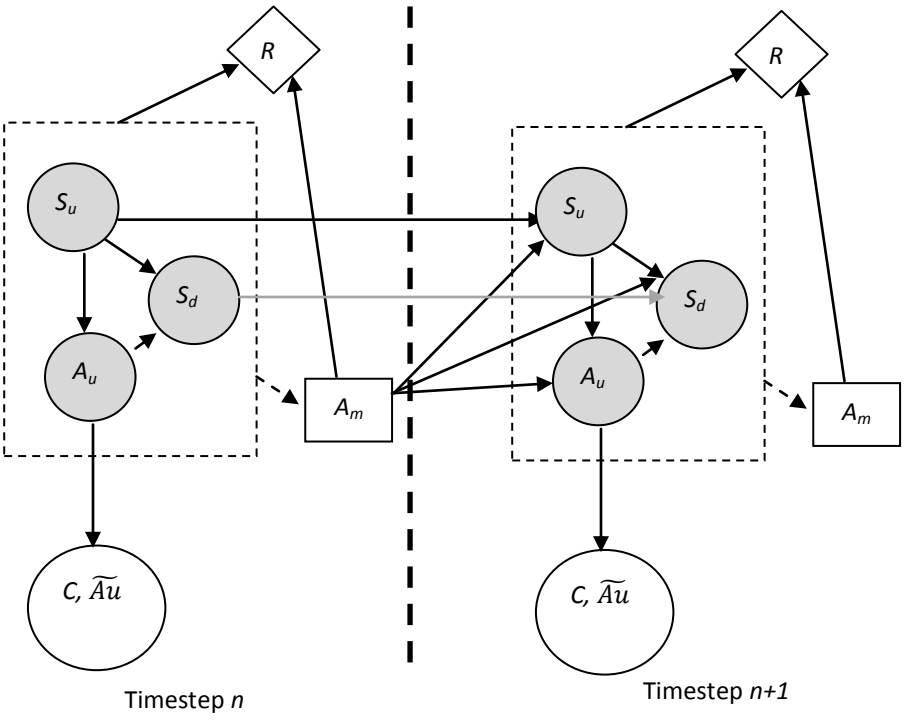


Figure 2. 6 Influence Diagram of SDS-POMDP (Williams and Young, 2005)

This factored form of POMDP is called SDS-POMDP. A summary of comparison between POMDP and SDS-POMDP components is given by the following table.

	Standard POMDP	SDS-POMDP
State set	S	(S_u, A_u, S_d)
Observation set	O	(A'_u, C)
Action set	A	A_m
Transition function	$p(s'/s,a)$	$p(S'_u\|S_u, a_m)p(a'_u\|s'_u, a_m)p(s'_d\|a'_u, s_d, a_m)$
Observation function	$p(o'/s')$	$p(\tilde{a}'_u, c'\|a'_u)$
Reward function	$r(s,a)$	$r(s_u, a_u, s_d, a_m)$
Belief state	$b(s)$	$b(s_u, a_u, s_d)$

Table 2. 2 Comparison between POMDP and SDS-POMDP (Williams and Young, 2005)

2.7.3. Problem of Tractability

Theoretically, every belief point *b* could map to an arbitrary action $\pi(b)$. However the states can grow exponentially and for this reason finding optimal policies for POMDPs is in general intractable. It is difficult to map $\pi*(b)$ to an arbitrary action. The belief space has to be partitioned to finite number of regions for optimal policy.

There are effective approximate solutions to this problem. In the following section, an approximation algorithm called "Incremental Partition Recombination" is described. The algorithm is developed by Williams (2010) and made available in the tool AT&T Statistical Dialogue Toolkit (ASDT) which this research is based.

Other approximation techniques used in SDS include the Recurrent Neural Network (RNN) and Bayesian update. According to Henderson et al., (2014), RNN is capable of generalizing to unseen dialogue state hypotheses, and requires little feature engineering. Thomson and Young (2009) showed Bayesian update based on the loopy belief propagation algorithm for tractable solution of POMDP based dialogue system.

2.7.3.1. Partition Recombination

The algorithm “Partition Recombination” helps not only to partition the belief space, but also prevent the number of partitions from growing too large Williams (2010). This is done by recombining or merging partitions with low belief (distribution over all states). That is, instead of trying to handle exponential number of user intentions and goals depending on the user action, it is better to consider only number of important partitions. For instance, in the restaurant information system, if the user is asked for type of food and responded as “Traditional”, the algorithm partitions only the probability of “Traditional” as the food type, the probability of “not Traditional” as the food type from the listing. The recombination further eliminates partitions with lowest belief.

When performing recombination, there is a trade-off between tracking more partitions and considering more N-Best list entries that both yield increases in accuracy at the expense of more computation Williams (2010). Algorithm 2.1. shows incremental partition recombination algorithm.

Data: Set of partitions **p**, their beliefs *b(p)*, ASR N-best list $\tilde{u}$, parameter p_{max}

Result: Updated set of partitions **p** and beliefs *b(p)*

1 $\forall_p \in \mathbf{p} : \hat{b}(p) \leftarrow 0$;

2 $\forall_p \in \mathbf{p} : P_{u^*}(p) \leftarrow 1$;

3 **for** $n \leftarrow 1$ **to** $|\tilde{u}|$ **do**

4 **for** $m \leftarrow 1$ **to** |p| **do**

5 If possible, split p_m on u_n; add children to p;

6 $\hat{b}(p_m) \leftarrow \hat{b}(p_m) + P(u_n|p_m, a)P(u_n|\tilde{u})b(p_m)$;

7 $P_{u^*}(p_m) \leftarrow P_{u^*}(p_m) - P(u_n|p_m, a)$;

8 $b'(p_m) \leftarrow \hat{b}(p_m) + P(u^*|\tilde{u}_1^n)Pu^*(p_m)b(p_m)$;

9 end

10 while |p| > p_{max} do

11 $p \leftarrow argmin_{p:p\ is\ leaf} b'(p)$;

12 Recombine p with parent; remove from **p**;

13 end

14 end

15 $k \leftarrow \sum_p b'(p)$;

16 $\forall p \in p: b(p) \leftarrow b'(p)/k$;

Algorithm 2. 2 Incremental Partition Recombination(Williams, 2010)

2.7.4. Hybrid Approaches

In the classical POMDP formulation, the optimization process is free to choose any action at any time (Lee et al., 2010) as discussed above. As a result, there is no obvious way to incorporate domain knowledge or constraints such as business rules. A combination of Handcrafted and POMDP method was proposed to constrain the set of possible actions based on conventional rules in the POMDP framework (Williams, 2008). In this approach, the optimization process runs faster and more reliably than in a classical POMDP because unimportant action choices are pruned by the conventional rules. Lison (2013) also proposes probabilistic rule which hybrid models of dialogue management that can account for both the complexity and uncertainty that characterize many dialogue domains. ASDT tool allows specifying rules after determining the belief of the state the agent is in. A maximum and minimum belief threshold can be set to select an action.

2.8. Related Works

2.8.1. Spoken Dialogue System for Under-resourced Languages

Plauché et al. (2006) attempted to explore the possibilities of designing a spoken dialogue system for under-resourced languages. The goal of this research was to provide practical guidelines for the development of low cost spoken dialogue system (SDS) in new languages and in contexts of limited resources and for oral communities. In oral communities, information is primarily disseminated by word-of-mouth.

The results obtained from this research suggest that it is possible to develop a low cost, rapidly deployable speech technologies for under resourced languages. The research was done in Tamil

Nadu (a state in southeastern India on the Bay of Bengal) for Tamil, Telugu, and Kannada languages.

The motivation of this research was, the less effectiveness of mass media in influencing people to improve practices in health, agriculture, or education than traditional, oral methods and content that stem from within a community.

The central focus of this work was on adapting the ASR unit of the Spoken Dialogue System with key features such as making it small, easily modifiable and that can be scaled to new domains and dialects and hence this can help oral communities with access to digital and local resources after building dialogue systems. It is argued that this is a better approach than to develop large vocabulary, continuous speech systems.

The research made a review of metrics to evaluate ASR systems such as accuracy, reliability and vocabulary size and adapting techniques of ASR for under-resourced languages. The adapting techniques discussed are cross language transfer (used when no existing data is available for a language), language adaptation (used to generate linguistic resources for a target language by initializing acoustic models on available source language data and adapting the models to the target language using a very limited amount of training data) and bootstrapping where acoustic models are initialized from a small amount of transcribed source data.

The Spoken Dialogue System is developed with under-resourced languages in focus.

It is a template based SDS which can run over the phone or on a kiosk. The ASR is based on a transcribed recoding of 80 speakers. 28 vocabulary options for the SDS have been selected. The speech recognizer used triphone HMM models (single Gaussian) and state-based parameterizing

for robust estimation. Decision tree based, state-tied, triphone models easily accommodate new words and contexts by traversing through the tree and synthesizing the triphones from a cluster of acoustically similar models. When evaluated the Monophone models yielded 73% accuracy, while triphone models performed at 97%.

The present work on Amharic SDS is related to the work of Plauché et al. (2006) in the attempt to resolve the challenges of under-resourced languages and hence to benefit from Spoken Dialogue Systems and have an easy and efficient way of accessing information. The solution for this research focuses on the ASR unit of the dialogue system by limiting the vocabulary to be recognized and utilizing adapting techniques which is very interesting and can be used as an input. However, robustness is still an open research question.

On the other hand, this research is different on the focus and emphasis where the problem lies, which is the dialogue management. ASRs are error prone and errors also are emitted from the NLU (Natural Language Understanding) unit as well and it seemed very difficult to add robustness to a template based dialogue manager. This research attempts to investigate an explicit way of modeling uncertainty from low performing ASR.

Barnard, et al. (2010) showed to resource development, automatic speech recognition, text-to-speech systems, and user-interface design for under-resourced languages. Chhetri, (2012) also attempts to present Voice User Interface Design framework for the illiterate and for languages with limited linguistic resources. Both employed handcrafted dialogue managers.

2.8.2. POMDP-based Spoken Dialogue Systems

2.8.2.1. POMDP-based Dialogue Systems in Noisy Environments

Young et al. (2014) show statistical POMDP-based dialogue managers offer the promise of increased robustness, reduced development and maintenance costs, and scalability to large open-domains compared to conventional hand-crafted rule-based dialogue management systems. The evaluation is made on real users instead of using simulators which are popular methods for their low cost.

A real-world restaurant information system is deployed and evaluated in a motor car using subjects. The results show that the statistical approach is indeed more robust. The other findings are that results from a user simulator significantly over-estimate performance both absolute and relative in addition performance results obtained over the telephone can provide useful predictors of performance in noisier environments such as the motor car.

This work is related to the underlying problem that is an increased ASR WER (Word Error Rate). Both noisy environments and ASR for under-resourced languages are characterized by relatively higher WER. The results of this research showed that POMDP-based SDS are more robust than handcrafted systems.

2.8.2.2. POMDP-based Dialogue Systems in Assistive Technology

Li et al. (2013), studied on SDS implemented in assistive technology. People with motor disabilities often face substantial challenges using interfaces designed for manual interaction. Such obstacles might be partially alleviated by automatic speech recognition. These individuals

may also have co-occurring speech-language challenges that result in high recognition error rates. Some of these challenges are fatigue, over-nasalization and vocal fry.

An end-to-end spoken dialogue system was constructed for these target users, adult wheelchair users with multiple sclerosis and other progressive neurological conditions in a specialized-care residence which is aimed to enable them to access information and communication services through speech.

The technique used is boosting to discriminatively learn meaningful confidence scores and ask confirmation questions within a partially observable Markov decision process (POMDP) framework.

The POMDP dialogue manager significantly increased the number of successfully completed dialogues compared to a baseline threshold-based strategy. The reduction in dialogue completion times was more pronounced among speakers with higher error rates, illustrating the benefits of probabilistic dialogue modeling for the target population.

As can be noted from the above studies, there is a commonality in the nature of the problem. Developing Spoken Dialogue Systems for under-resourced language, in a noisy environment and to people with motor disabilities with co-occurring speech-language challenges share the same problem of higher recognition error rates. Two of the researches utilized POMDPs (Partially Observable Markov Decision Processes) and yielded a better performance than a baseline SDS. However, for the research on development of an SDS for under-resourced languages the capabilities of POMDPs are not tested. Therefore, it is fair to argue that if POMDP-based spoken dialogue systems showed increase robustness in a noisy environment with people speech-

language challenges, it can be used to build Spoken Dialogue Systems of low performance components which can then be improved through the data collected.

This review covers the overview of Spoken Dialogue from a system perspective, components of dialogue system and dialogue management, which is the central focus of this research. To the best of the researcher's knowledge, the method planning under uncertainty for dialogue management has been applied to resourceful languages but scarcely investigated for under-resourced languages.

Handcrafted approaches for under-resourced language lacks robustness, and are generally time consuming. POMDP-based approaches are more robust in noisy environments than handcrafted and performed better when used for people with motor disabilities.

POMDP-based spoken dialogue systems showed robustness in a noisy environment such as in car systems and an elderly facility where higher ASR Word Error Rate (WER) is observed. Low performance ASRs are also characterized by higher WER. It argued that POMDP-based dialogue management enables to develop a robust Spoken Dialogue System for Amharic that can complete a task. The same framework can also be used to collect data which can be used for improving the system following the "Deploy, collect and improve" paradigm.

CHAPTER THREE

METHODOLOGY

This chapter explains the methodological approach employed in this research to collect data, design dialogue manager and evaluate it. This research specifically explores firstly, methods of designing a robust dialogue manager to ASR, NLU and user errors second, methods of designing an improving dialogue manager and third method of evaluating the dialogue manager.

3.1. Architecture of Amharic Spoken Dialogue System

This research employs a pipeline architecture. The main reason is that ASDT is designed based on a pipeline architecture (Williams, 2008). Figure 3.1 shows how the components of SDS are connected and interacts while it is operating.

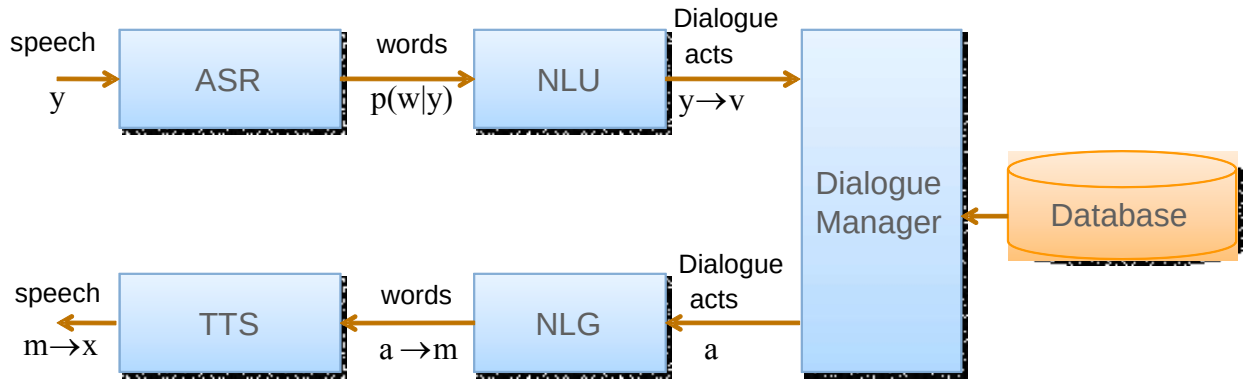


Figure 3. 2 Architecture of Amharic Spoken Dialogue System

The statistical ARS finds possible string **w** given the users' speech **y**. A simple keyword matching language understanding unit maps the string **w** to its corresponding dialogue act **v**. This research used signle word sentence to simplify the NLU unit. The POMDP-based dialogue manager selects appropriate system action **a** (dialogue act itself). Rule based natural language

generation unit assigns surface form **m** for the selected action **a** by the dialogue manager. For this research, the dialogue manager is set to output signle system act at a time. Finally the knowledge based text-to-speech unit converts the words **m** to speech **x**.

3.2. Dialogue Management Design

This research employed a POMDP-based dialogue manager. The shortcomings of handcrafted approaches, that is time consuming, unable to handle uncertainty in a principled way and their problem of extendibility and portability led to data driven methods. Among the data driven methods, supervised approaches require example training set which is not acquired and action selection is based on single instance not considering the whole dialogue history. The viable approach is based on reinforcement learning. However, among reinforcement learning MDP assumes complete observation of state where the agent is in. This is problematic for spoken dialogue system in general where there is uncertainty arises from ASR and NLU units and specifically for this research where errors from these units are expected to be severe at their initial states and hence there is more uncertainty.

The principled way of handling uncertainly is POMDP framework. A classical POMDP cannot be used since the observations factored to user goal, history, and user action which leads to SDS-POMDP.

One of known issue of reinforcement learning is intractability of states which requires approximation. Among the available methods of approximation such as Bayesian Network, Neural Recurrent Network and Partition Recombination, Partition Recombination was used for this research. This method is used the major algorithm of ASDT is Partition Redistribution.

Finally the action selection was done through handcrafted method. For a simple prototype system like the one employed for this research, once the system has a belief on a state, a rule can be to select a system action. Structured POMDP suggested by Lison (2013), and handcrafted heuristic approach is suggested by Williams (2008). ASDT tool employs the latter one.

3.2.1. Data Collection Methods

Data collection is an important research area and represents a significant portion of the work in developing a spoken language system (Lamel, 1998). Data can be collected in the following methods.

Application Specific: It is necessary to collect application-specific data, which is useful for accurate modeling at different levels (acoustic, lexical, syntactic and semantic) and as a knowledge base (Lamel, 1998).

Human-human or Human-machine Interaction: Corpora of human–human from call centers or human–machine dialogues from previously deployed SDS can be prepared. Human–machine dialogues are preferred because people behave differently when they talk to a machine (Lison, 2013).

Wizard of Oz: It is common practice to use a WOz setup or a bootstrap system to collect an initial corpus. Wizard of Oz can be explained as a simulation of a computer system by way of a hidden human operator, who is in this case a "wizard". The user is told to interact with the machine without knowing the presence of human operator behind. This will help to understand user behavior towards interacting with a machine and would be observable if there is awareness

of human operator behind. An initial corpus can also be collected using this technique (Petrik, 2004) (Lamel 1998).

Bootstrap System: The bootstrap system is often based on prior work: acoustic models or training data may be taken from a different task; an initial vocabulary can be obtained by considering the task and introspection; and a simple language model can be estimated on a set of typed queries. These queries can also be used to develop an initial set of rules for the semantic analyzer (Lee et al., 2010) (Lison, 2013).

Deploy, Collect and Improve: As introduced by Young (2014), it is suggested to deploy an end-to-end Spoken Dialogue System which enables the system to be trained online and enables to accumulate data from real users and expand its coverage thereby avoiding data scarcity problems. In other words this paradigm encourages deploying a system and letting it automatically learn to continually get better and expand its coverage.

For prototype development and experimentation, domain specific and representative conversation data are required.

Domain specific data is required as a knowledge base for the prototype and to train the ASR (Williams, 2008) (Qiao, et al. 2010). The knowledge for the prototype is represented in a database that interacts with the dialogue manager. This research used two address search websites, since the prototype was implemented on hotel and restaurant domain for domain specific data. Database enables to formulate different queries and keep the dialogue manager domain independent. ASDT provides a tool that converts plane, comma separated file to a database. In addition, instead of training a generic ASR, it is better to train or adapt the ASR for better performance of the SDS (Anderson, 2014).

To identify dialogue act and to collect initial speech data for training the ASR, a WOz approach was also employed. WOz is a common approach to capture user/system act and record initial voice corpus (Lamel, 1998). WOz better captures users' interaction behavior than intuition based design. This method was employed to some extent because the user act and system act are simple and straight forward.

Simulated user was also used in this research because simulated user can generate large amount of data for testing at the concept level (user act). Simulated user enables to explore every possible user's input, which cannot be achieved in a real user environment. ASDT is provided with simulated user generator. Given the database, it can interact with the dialogue system with possible user actions (Williams, 2008).

The deployed prototype is capable of recording the speech input online. This data can be used as for future improvement of the system.

Collecting data from human-human conversation is time consuming and expensive. Since there are no prior projects or researches on SDS, human-machine interaction data from implemented SDS is rarely available. The same reason also applies for bootstrapping method which also requires previously collected data.

3.2.2. Evaluation Methods

A spoken dialogue system is built of distinct modules and although for most of the components like the ASR, there are defined methods of evaluation, joint evaluation of the complete system is challenging (Gasic, 2011). In addition, evaluating the dialogue manager's performance is difficult by itself, because of the vast space of possible dialogues.

Gasic (2011) states, four evaluation methods for spoken dialogue system:

Evaluation with Human Users: The most forward approach to evaluation is to have the dialogue manager interact with humans, and let human users rate the dialogues. However, this makes it infeasible to evaluate all possible dialogues and is also costly and time-consuming to perform.

PARADISE Framework: The PARADISE requires a corpus of real users' dialogues annotated with user satisfactions and a set of objective measures. Measures such as number of turns and number of times the systems confirms determine the dialogue cost measures. This in turn used to construct a weighted function of the dialogue success which determines dialogue performance.

Reinforcement Learning Reward: Instead of human users to rate the dialogues, an alternative evaluation metric is to use the reinforcement learning reward function. This is done by employing simulator instead of human users interact with the dialogue manager. This in turn allows a wider coverage of dialogue space in evaluation because of large number of dialogues the simulator can generate. In addition, artificial noise levels can be introduced.

Task Completion: In order to evaluate effectiveness of a particular technique or feature inside the dialogue manager, success rate and average length of the dialogue can be employed.

As the main focus of this study was to show the capabilities of POMDP-based dialogue manager, evaluation was confined to this component. The purpose of evaluating the dialogue manager to determine to what extent the dialogue manager is robust. In other words, with how many turns does the dialogue manager complete a task given different level of errors from the ASR (Word Error Rate) and what type and amount of evidence (hypothesis) required to handle uncertainty

without degrading the speed of the dialogue manager should be explored. For this purpose among the available evaluation methods, task completion was selected using both simulated and human users. ASDT allows the dialogue manager to be evaluated for using varied amount of hypothesis, speed and accuracy.

CHAPTER FOUR

EXPERIMENTATION AND DISCUSSION

This study attempted to develop a prototype of POMDP-based Amharic spoken dialogue system. The goal of the experiment is to quantify the benefits of POMDPs modeling uncertainty in building SDS for under-resourced languages. It is designed to verify the advantages of maintaining multiple hypotheses used to model the uncertainty that arise mainly from the ASR. The experiments were limited to the dialogue manager and its effects on accuracy and error.

Experiments were performed on simulated and real user environments. Several experiments were carried out on a simulated environment. User simulation is done at dialogue act level by adding random noise to the system. It is designed in such a way that a simulated user is trying to find a hotel, restaurant or café in particular neighborhood in Addis Ababa. The system asks the user a series of questions until it reaches to the user's goal or until it fails, and then if successful, displays the results according to the query. The machine has six actions available, including greet, ask-hotres, ask-foodtype, ask-subcity, ask-neighborhood and display results. The user's goal specifies the user's desired location. From the address search websites, 128 user goals were used.

To evaluate how the POMDP framework handles uncertainties that emit from Amharic ASR, an end-to-end SDS prototype was built. In the following section, we describe the end-to-end SDS prototype prepared for the experimentation.

4.1. Implementation and Experimentation Setup

The prototype operates in the following way. It takes initiative by greeting and asking what the user is looking for, እንኳን ወደ ሆቴል አፈላላጊ ፕሮግራም በደህና መጡ፣ የፈለጉት ሆቴል፣ ሬስቶራንት ወይስ ካፌ ነው?. The text from the speech recognizer engine is given to the Natural Language Understanding unit to determine the user action for instance if they user says "hotel" and the corresponding recognition hypothesis with confidence score passed to the NLU to determine the user action as "hotres(hotel)". The dialogue manager uses rules to select appropriate system action according to its belief of the dialogue state. The belief is determined from the hypothesis. Then this system action interpreted to a surface form by NLG and sent to the text-to-speech unit for the user in audible format. Since the development has an iterative nature, after integrating the components revising the system and user action was required.

4.1.1. Amharic ASR

An ASR trained for this experiment was used and integrated with ASDT. ASDT is basically designed to use its own cloud-based[12] speech recognizer and text to speech for English language. The ASR used was an HMM based, small vocabulary, user independent and spontaneous. A monophone model was used and no attempt was made to improve the performance of the recognizer. The HTK toolkit was used to train and Julius was used as ASR Engine.

Julius should be given a grammar to constraint the recognition. For example, in "one word sentence" recognition, where the user is expected to respond with a single word, the grammar will have the following structure.

[12] https://service.research.att.com/smm

```
S: NS_B SENT NS_E
SENT: HOTERES
SENT: SUBCITY
SENT: NEIGH
SENT: FOODTYPE
```

(a)

```
% NS_B
<s>       sil

% NS_E
</s>      sil

%HOTERES
HOTEL         HO TE L
RESTORANT         RE S TO RA N T
KAFE         KA FE
HOTEL         HO TE L

%SUBCITY
LDETA         L DE TA
QIRQOS          QI R QO S
BOLE         BO LE
…
```

(b)

Figure 4. 1 Julius Grammar(a) and Vocabulary File(b) used for the Prototype

The sentence in this case is composed of silences (sil) at the beginning and the end of sentences NS_B and NS_E and the sentence category. This file has a file extension ".grammar". A separate vocabulary file with extension ".voca" contains all the words under each category to define candidate words in each category with its pronunciation information. When the system prompts request such as "Are you looking for a hotel or restaurant?", the user can respond "hotel".

A grammar compiler converts .grammar and ".voca" files to ".dfa" and ".dict" files. When executing Julius, ".dfa" and ".dict" files should be specified together with acoustic model or all these can be contained in a configuration file ".conf".

Julius provides different options. To run Julius speech engine the following command was used.

```
julius-4.3.1 -input mic -C amhsimple.conf -output 5 -n 5 –module  -record c:/recorded
```

Option `-input mic` will tell Julius to get the audio input from a raw audio device like microphone or line input. To load the configuration file, `-C amhsimple.conf` is used. `-output 5` and `-n 5` defines to output the top N sentence hypothesis to be output at the end of search and the number of candidates Julius tries to find respectively. The `-module` option runs Julius on "Server Module Mode". TCP/IP[13] protocol is used to interface with Julius speech recognizer engine. A python client script can interface with the engine to fetch the recognized output. And the `-record c:/recorded` is used to auto-save successively under the directory specified.

4.1.2. Integrating with ASDT

4.1.2.1. Building a Database

A script included in ASDT creates database given a comma separated text file. This file contains only the fields which are required to retrieve restaurant information. For the experiment four attributes are selected. The first column is used to register whether the listing is a hotel, restaurant or cafe. The second one holds food type, whether it is traditional, international dish or both. The remaining two holds the whereabouts information, i.e. the sub-city and neighborhood. The ASDT script converts this file to SQLITE3 database. SQLite is a software library that implements a self-contained SQL database engine that does not require server to run (Zhang 2013).

[13] a set of protocols developed for the internet to get data from one network device to another

Figure 4. 2 Comma-Separated File for the Portotype

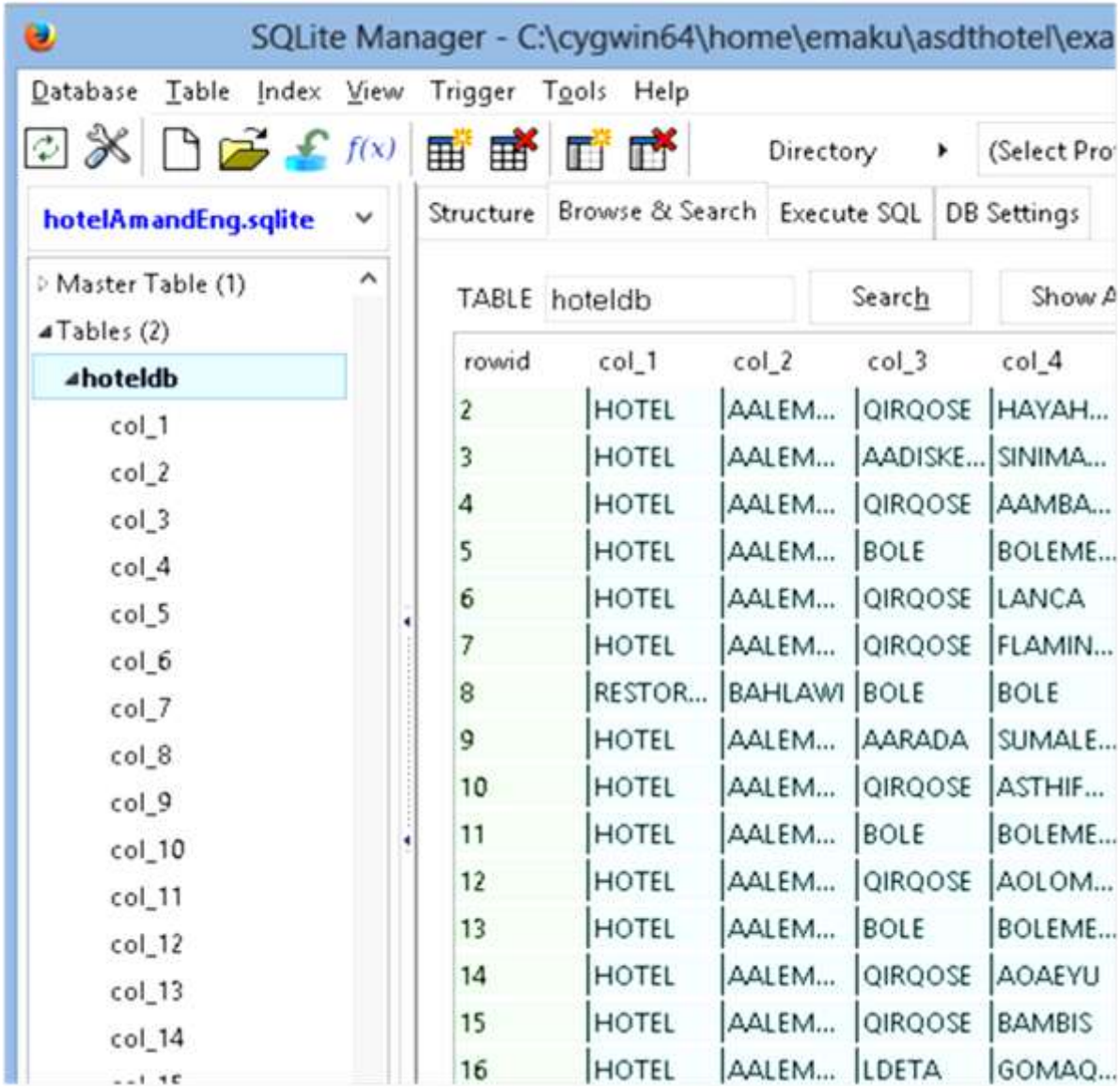


Figure 4. 3 Sqlite Database Managr used in the Prototype

4.1.2.2. Building Grammar

To build a grammar ASDT uses a script from the database. The script creates different grammars for the kinds of dialogue managers described above and create a grxml file format. Grammar for the open dialogue manager contains all the fields. If the grammar is changing one after the other the script creates one grammar file for each field. Example of such grammar created for the field "sub-city" would look like the following.

```
<grammar tag-format="semantics/1.0" version="1.0" root="all">
<rule id="all">
<item><one-of>
   <item>aadisketema<tag>out.subcity="AADISKETEMA"</tag></item>
   <item>bole<tag>out.subcity="BOLE"</tag></item>
   <item>yeka<tag>out.subcity="YEKA"</tag></item>
   <item>nefaseselklafto<tag>out.subcity="NEFASESELKLAFTO"</tag>
</item>
   <item>none<tag>out.subcity="NONE"</tag></item>
   <item>gulele<tag>out.subcity="GULELE"</tag></item>
   <item>nefase selk lafto<tag>out.subcity="NEFASE SELK
LAFTO"</tag></item>
   <item>ldeta<tag>out.subcity="LDETA"</tag></item>
   <item>aadis ketema<tag>out.subcity="AADIS KETEMA"</tag></item>
   <item>kolfeqeraniyo<tag>out.subcity="KOLFEQERANIYO"</tag>
</item>
   <item>qirqose<tag>out.subcity="QIRQOSE"</tag></item>
   <item>aaqaqiqaliti<tag>out.subcity="AAQAQIQALITI"</tag></item>
   <item>aarada<tag>out.subcity="AARADA"</tag></item>
</one-of></item>
</rule>
</grammar>
```

Figure 4. 4 ASDT Grammar

4.1.2.3. Natural Language Understanding Unit

For NLU the user action is determined using a simple string matching technique. This is a redundant unit for this particular setting since what is recognized is bound in grammar. However, to make the unit portable and extendable a separate unit is created. A rule based approach was implemented to determine the user act i.e. if what was recognized fall under one of the categories defined such as confirmation, or one of the fields, the user action determined accordingly. A python module was written and called from the dialogue manager to determine user action. For

example if the recognized term has a key word “Bole” and that falls under “subcity” returning it as the user act. This was done for all n-best list i.e. the top hypothesis of the ASR.

```
if hotres.__contains__(term):
    return 'hotres'
elif subcity.__contains__(term):
    return 'subcity'
elif neigh.__contains__(term):
    return 'neigh'
elif foodtype.__contains__(term):
    return 'foodtype'
…
```

Figure 4. 5 Code Fragment of NLG

4.1.2.4. Dialogue Managers

ASDT can be configured to use three different types of dialogue managers (Williams, 2008). The first one is a called rigid dialogue manager which asks for each slot in a predefined order. The second one is called Directed dialogue manager that asks for individual slots until there is a matching with a unique listing in the belief state and until the belief is above a threshold. The last dialogue manager called open dialogue manager, asks an open question until the lower threshold in some slots has been reached, then it asks for a specific slot with lower belief and continues until the upper threshold is reached.

Among the three dialogue managers ASDT has, the direct dialogue manager was selected for its simplicity for the experiment. Changing the dialogue manager to the remaining two types is a matter of changing the grammar and importing appropriate dialogue manager since components are modularized in the tool.

4.1.2.5. Spoken Language Generation

The natural language understanding converts the system action to a surface form. The prompts are selected after the dialogue manager selects an action. A rule based approach was utilized for this experiment. Figure 4.6 is the code segment that handled the system utterance.

```
elif (field == 'hotres'):

    readam.gettext('%sየፈለጉት ሆቴል፣ ሬስቶራንት ወይስ ካፌ ነው?\n' %(repeatmessage))

elif (field == 'foodtype'):

      readam.gettext('%sየምግቡ አይነት ባህላዊ ወይስ አለም አቀፍ ይሁን?\n' %(repeatmessage))
```

Figure 4. 6 Code Segment Of NLG

4.1.2.6. Integrating Text To Speech

For this experiment eSpeak was used as a Text To Speech engine. To integrate with ASDT, the surface form of the NLU unit was sent to eSpeak's through a command line option. eSpeak reads a temporary file created for a particular prompt according to the action selected. A python module is created to interact with the dialogue manager and eSpeak.

4.1.2.7. Fetching the desired information

The last step is to display the information to the user based on the user's utterance. This was done by running a python script to manipulate the SQLite database after changing user’s utterance to a database query. For this purpose a separate database which was created to display Amharic Unicode text was used.

```
sqlite_file = 'hotelAmandEng.sqlite'
  table_name = 'hotrescafe'
  h=querydetail['hotres']
  f=querydetail['foodtype']
  s=querydetail['subcity']
  n=querydetail['neigh']
  print h,f,s,n
  conn = sqlite3.connect(sqlite_file)
  c = conn.cursor()
  c.execute('SELECT
col_5,col_6,col_7,col_8,col_9,col_10,col_11,col_12,col_13,col_14,col_15,col_16,col_17 \
  FROM hotrescafe WHERE col_1=? and col_2=? and col_3=? and col_4=?',(h,f,s,n))
  all_rows = c.fetchall()
  resutlt = str(all_rows)
  numberoffetched = str(len(all_rows))
  tosay = 'በሰጡት መረጃ መሰረት '+numberoffetched+' ቦታ ተገኝቷል'
```

Figure 4. 7 Code Segment to Fetch data from the Database

4.1.2.8. Configuration Option

ASDT provides a configuration option for three variables. The first one is the maximum number of n-best entries to consider (maxN-best). Additional entries on the N-best list, if present, are ignored the second, is the maximum number of partitions to maintain (maxPartitions). After each N-best entry is incorporated, partitions are repeatedly recombined until they reach to this threshold. The third one is the maximum number of histories to maintain for each partition (maxHistories). At the end of each update, low probability histories are deleted until there are at most this maximum number is reached for each partition.

4.2. Illustration

To illustrate the properties of POMDP-based dialogue manager, we considered a spoken dialogue system with no confidence scoring and which makes speech recognition errors with a fixed error rate. The following example is in the restaurant information system domain. The left column shows the machine and user utterances, and the recognition results from the user's utterance (shown in brackets). The center column shows a portion of the POMDP belief state; b represents the belief over a component of the user's goal (venue hotel, restaurant or cafe). The right-hand column shows what a typical a handcrafted method would track this component of the user's goal.

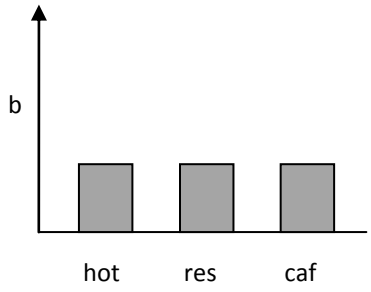
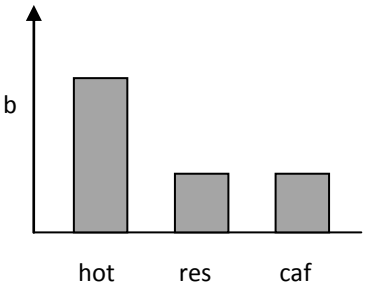
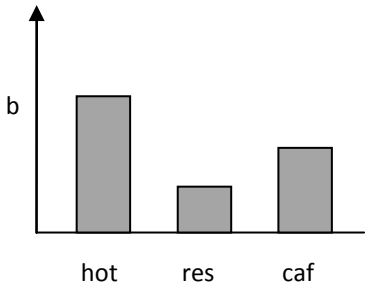
System /User /ASR	POMDP belief state	Handcrafted Methods
Prior to start dialogue		hot_res: <empty>
S: የፈለጉት ሆቴል፣ ሬስቶራንት ወይስ ካፌ ነው? (Are you looking for a hotel, restaurant or a café?) U: ሆቴል (Hotel) [Hotel]		hot_res: HOTEL
S: **የምግቡ አይነት ባህላዊ ወይስ ዓለም አቀፍ ይሁን?** (What kind of dish would you like? Traditional or international?) U: አለም አቀፍ (International) [Café]		hot_res: <?>

Figure 4. 8 Example Conversation With A Spoken Dialogue System Illustrating The Benefit of Maintaining Multiple Dialogue State Hypotheses

Prior to start of the dialogue, there is an equal distribution of belief. On the first turn the system asks if the user want a hotel, restaurant or café. The user responded as “hotel” and is recognized as “hotel” by the recognizer. Therefore the belief of hotel increased and based on that the dialogue manager goes to the next question requesting the type of food. On the second turn a recognition error is made. The user chose international but recognized as café. This will cause handcrafted methods to accept a piece of bad information because the system requested the next question, whereas the POMDP belief state is more robust. As shown in Figure 3.3 the belief it had before decreased because of the new evidence. In this example no account is taken of which user actions are more or less likely, or of confidence score. This method also enables the user to change goal.

4.2.1. Confidence Score

POMDPs provide a principled approach to confidence scoring by incorporating the magnitude of the confidence score by scaling the belief state update correspondingly. Weak evidences in more than one turn can be accumulated to increase the belief. Two competing evidences can be disambiguated on subsequent turns by shifting belief mass.

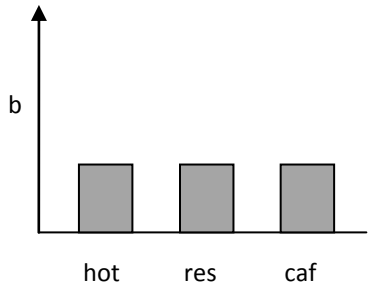
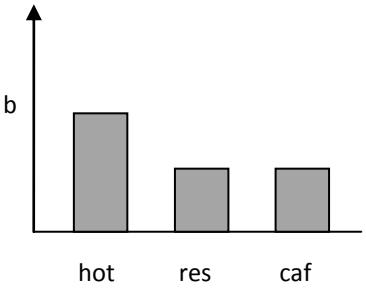
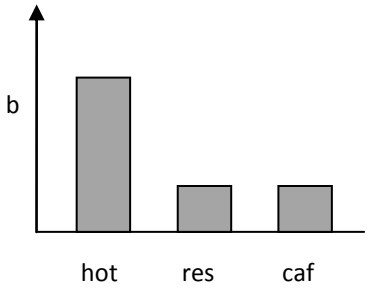
System /User /ASR	POMDP belief state	Handcrafted
Prior to start dialogue		hot_res: <empty>
S: የፈለጉት ሆቴል፣ ሬስቶራንት ወይስ ካፌ ነው? (Are you looking for a hotel, restaurant or a café?) U: ሆቴል (Hotel) [Hotel] ~ 0.38		hot_res: HOTEL
S: ይቅርታ በደንብ አልተሰማኝም፣ የፈለጉት ሆቴል፣ ሬስቶራንት ወይስ ካፌ ነው? (Sorry, are you looking for a hotel, restaurant or a café?) U: ሆቴል (Hotel) [Hotel] ~ 0.39		hot_res: <?>

Figure 4. 9 Example Conversation

The above example shows the users response is recognized with lowest confidence score. This could be due to low performance of the ASR. Because of the low belief the dialogue manager asks the question again. The score is still low for even for the second time. But the belief increases because this is additional evidence.

4.2.2. With Partition Distribution

The Figure 4.10 illustrates how the algorithm "Incremental Partition Recombination" works. In this example the system is requesting sub-city and neighborhood. In the first turn when sub-city is requested by the system, the user replied as "Yeka" with a high confidence score and the dialogue manager continues with the next question i.e. asking for the neighborhood. Then it will be divided into two partitions (not yeka, everything) and (yeka, everything). But in the second turn there is a recognition error and the recognized neighborhood ("kolfe") has low score. With this new evidence the system further partitions to (not Yeka, Kolfe) and (Yeka, Kolfe). Then the belief is updated and those with lowest belief are recombined. The partition with (not Yeka, Kolfe) is the lowest and it is recombined. The probability of Kolfe being in Yeka is also low. Since there is no partition above the threshold, the system asks the question again. This time "Kotebe" is recognized with higher confidence score. With this new evidence it splits, partitions with lower beliefs recombined. The partition (Yeka, Kotebe) is accepted supported by the higher probability of Kotebe being in Yeka. This example shows how evidences can aggregate in POMDP-based dialogue systems.

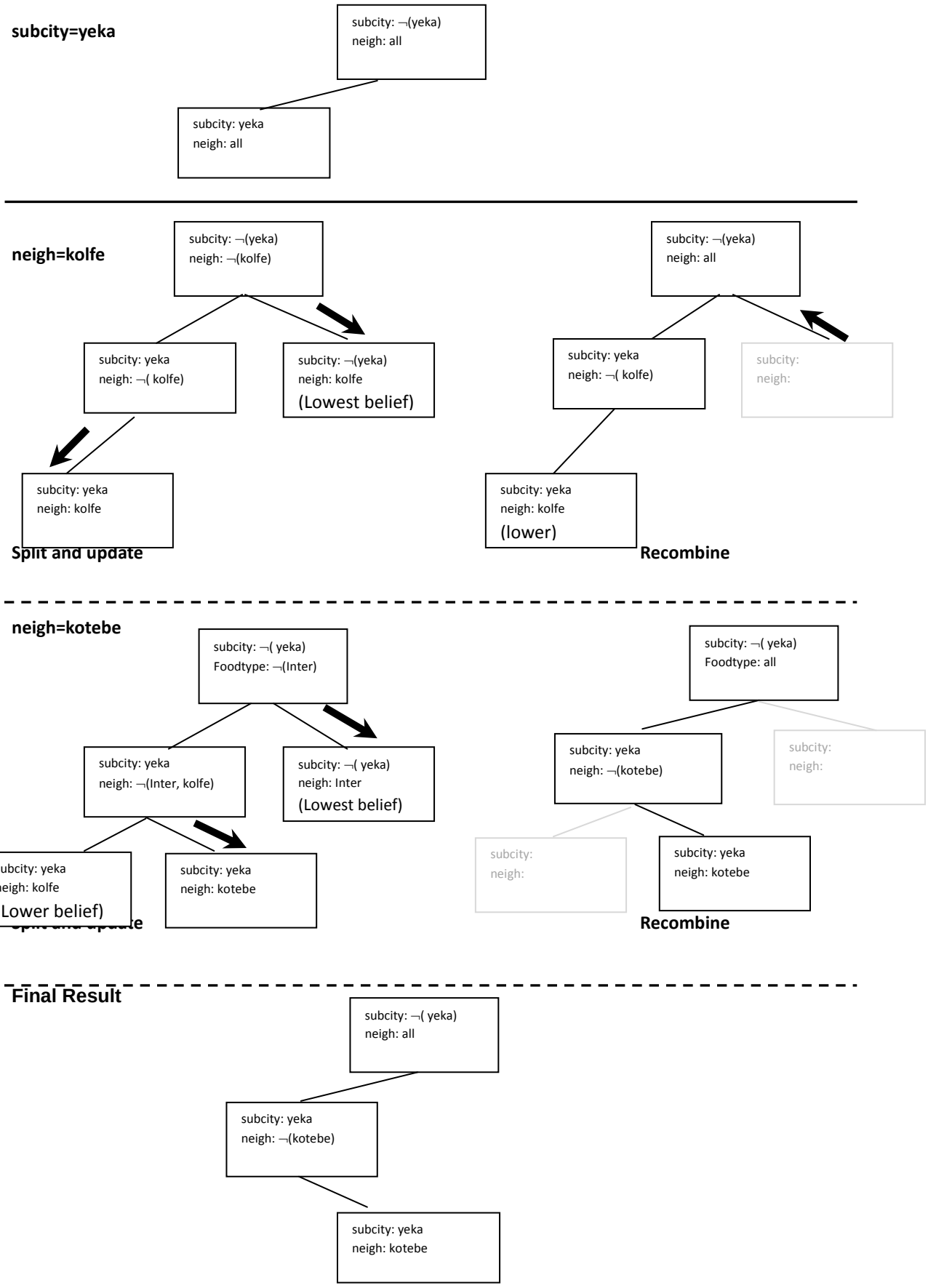


Figure 4. 10 Belief Monitoring Illustration

4.3. Experiments

4.3.1. Evaluation Methods

ASDT provided two methods to evaluate correctness of a dialogue.

The first method which is provided in ASDT scans through the belief state to find the highest belief partition in which all fields have been instantiated, and compares this to the user's true goal. In a simulated environment, the user is believed to know what he/she is doing. This method measures the performance of the dialog manager within the limits of the ASR N-Best list: in other words, to be scored correctly by this method, each field in the user goal must appear in an N-Best list.

The other method provided in ASDT scans through the belief state to find the highest belief partition with count=1 i.e., with exactly 1 match in the database. This method allows the dialog manager to ask the database for fields which it never observed in any N-Best list.

4.3.2. Experiment one: N-best list and Maximum Partition (simulated environment)

The following experiment was performed in a simulated environment of 1000 dialogues. Each dialogue contain a minimum of 4 turns from the user side.This experiment help to evaluate whether there is a useful information in n-best list that can be accommodated in the number of partitions and hence improve the correctness score.

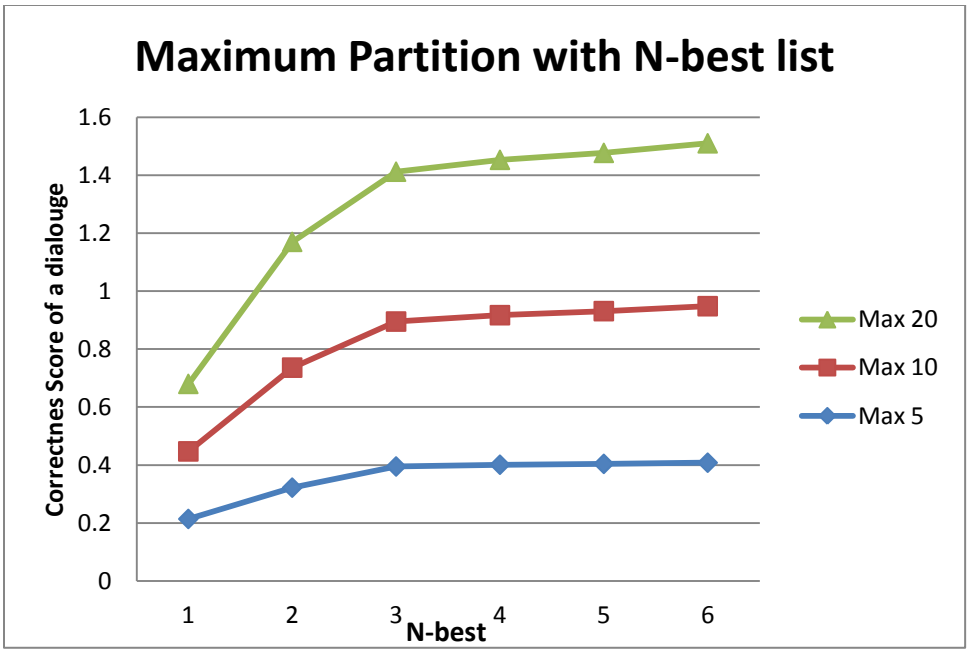


Figure 4. 11 Maximum Partition with N-Best List

As can be shown in Figure 4.11, correctness score increased when number of n-best list increased. Similarly, when the amounts of maximum partition increased the correctness score increased. An increase in both n-best list and number of maximum partition yielded better correctness score. For example with 6 n-best list, the correctness score had a growth rate of 54.67% when the maximum partition changed from 5 to 20 and keeping the partition at 20 it showed a growth rate of 15.86%.

4.3.3. Experiment two: Error rate with N-best list (simulated environment)

The purpose of this experiment was to evaluate the effect of error rate with the number of N-best list. The minimum number of N-best list 1, which means only one speech recognition hypothesis with confidence score was provided. As discussed in Section 4.2., handcrafted systems allow the first top ASR hypothesis as input. However, combined with prior probabilities and the history of the dialogue N-best list items other than the top one may contain important information about the state where the dialogue might be in.

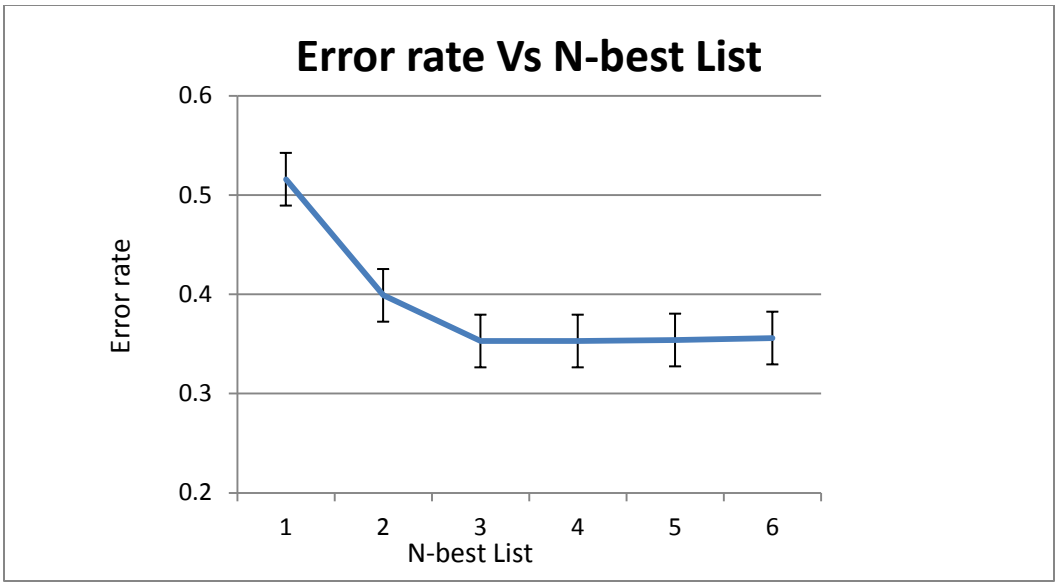


Figure 4. 12 Error Rate Vs N-Best List

Figure 4.12 shows that the error rate decreased with increase of n-best list. This shows that important information is contained in the N-best list. More evidences that result in an increase of the belief of a particular state can be extracted from the N-best list. The worst error rate was with only one n-best list. On average, the error rate reduced by 5.78 %.

After the 3 n-best list the error seemed to be flat. This might be explained due to the size of the first two fields (out of four) of the database with only three instances under each field. For instance for the field food type only international, traditional or both were available. Out of the six n-best list three of them could be out of grammar which will not decrease the error rate. For this domain N-best list above 3 did not decrease the error rate.

4.3.4. Experiment three: Speed Vs accuracy (simulated environment)

The purpose of this experiment was to test to what extent should the number of n-best list and maximum number of partition can be increased without degrading the speed and accuracy of the dialogue manager.

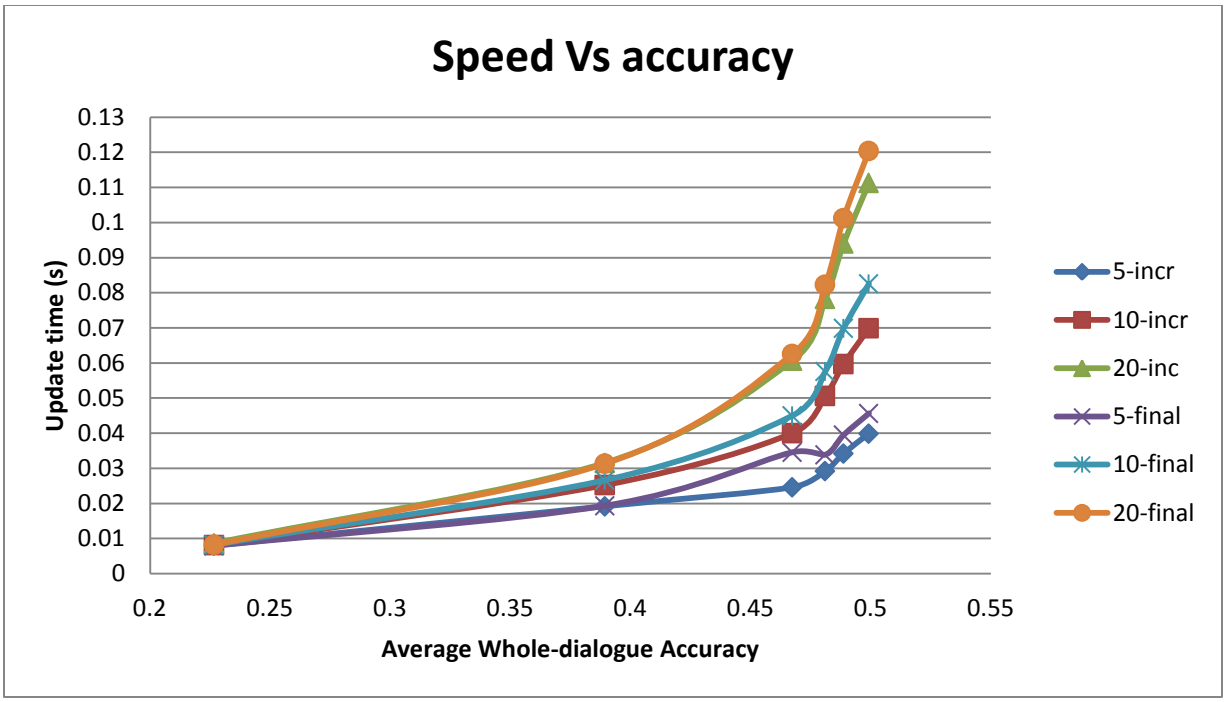


Figure 4. 13 Speed Vs Accuracy

The results, as seen in Figure 4.13, indicates that accuracy increased as number of n-best list and maximum partition increased. The dots represent number of n-best list and the line with different shaped dots represents number of partitions. Figure 4.13 shows the effect in update time when number of maximum partition increased in both incremental and final stages of partition and recombination. As can be shown in the graph with increase of accuracy the update time was well below one second which is acceptable in turn taking. With maximum of 20 partitions and 6 n-best list, the belief was updated in 0.12 seconds. The best accuracy and speed was achieved at the incremental stage of partition with 6 n-best list.

4.3.5. Experiment four: N-Best list with average WER in task completion (Real users)

This experiment was designed to evaluate robustness of the dialogue manager. It was done with four volunteer participants from computational linguistics program of Addis Ababa University.

The participants were told four goals each with five dialogues to control the goal. To control the speech condition a recorded response of the participants for subsequent dialogues. A word based request was asked with four slots. The Word Error Rate was calculated comparing the transcription of user's utterance and recognition output at a word level. Figure 4.14 shows the result of the experiment. The result indicates when the Word Error Rate increased the system completed the dialogue with an increased number of turns which could show robustness of the system. With an average word error rate of 50%, the dialogue manager can complete a task with average of 8.75 turns. This shows it completed a task at the cost of longer dialogue turns.

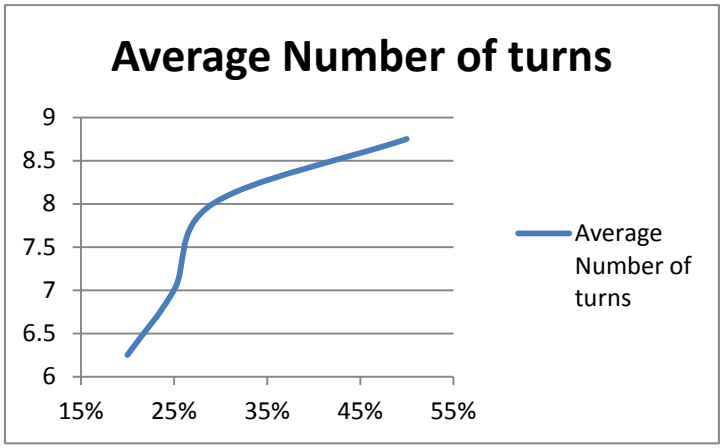


Figure 4. 14 Average Number of Turns Vs Word Error Rate

4.4. Discussion

The original motivation behind POMDP-based dialogue management for resourceful languages was 1) ASR can never be perfect and get worse in noisy environments and 2) speech is naturally ambiguous. POMDP-based dialogue performed better in a noisy environments and resolved ambiguity better. In this research, based on the underlining commonality between an ASR working in a noisy environment and an ASR with low performance due to insufficient language resource, it is hypothesized that POMDP-based dialogue management could perform better.

This research proposed a POMDP-based dialogue management which can be used to build spoken dialogue systems for under-resourced languages which can complete a task with low performing ASR with a benefit of collecting Data, and improving. In the course of the study, it was also demonstrated how to build an end-to-end SDS prototype which may be used as a data collection framework while serving to provide infomation for domains specific tasks. It is possible to build an initial task oriented SDS which can be improved and extended to other.

The experiments demonstrated that maintaining multiple hypotheses in dialogue management increases accuracy and robustness which is measured by task completion rate. Particular attention was paid to the information contained in the n-best list and the partitions. Due to ASR and NLU errors the state of the dialogue cannot be known directly. Maintaining multiple hypotheses, evidences aggregate to infer the state of the dialogue. One of the methods used was to utilize most of the N-best list the ASR outputs and partition with each of these to use as evidence. This will help to check even lowest score in the N-best list may have higher beliefs when other evidences add up. Increasing the number of n-best list by 6 reduced the error rate by 5.78% and with 6 n-best list, the correctness score had a growth rate of 54.67% when the maximum partition changed from 5 to 20. The belief updated below 0.12 seconds with 20 partitions and 6-nbest list. And the dialogue manager able to complete a task with an average 8.75 turns with 50% word error rate.

Data discrepancy, lack of other experiments on POMDP-based SDS for under resourced languages and lack of established framework to integrate SDS components were the major challenges for this research. The collected data used as domain knowledge had problems such as duplication, miss-categorization and missing of information. The absence of base-line system made it difficult to make comparison and refer prior work. In addition, out of the numerous

available language technologies for spoken dialogue system components, it is a time consuming task to choose the ones appropriate for this work. However, availability of POMDP based dialogue toolkit was limited.

To our knowledge, this is the first study to use POMDPs capacity model uncertainty for under-resourced languages like Amharic.

CHAPTER FIVE

CONCULUTION AND RECOMMENDATIONS

5.1 Conclusion

This research was aimed at proposing POMDP-based methods, approaches and algorithms to design a dialogue manager that is robust to ASR and NLU errors for the purpose of developing Amharic spoken dialogue system. Lack of language resource and off-the-shelf components are the major problems develop such systems. In this research it is argued that limited language resource results in low performing SDS components and therefore low task completion frustrating users. If a robust dialogue manager is designed, task completion will increase at the cost of longer dialogue turns. By employing the "deploy, collect and improve" approach, the data collected during the interaction can be used to retrain the ASR, reconsider NLU and better understand user behavior. Hence the dialogue manger will improve with higher task completion rate. With this resource and knowledge, extending the system to other domains is also possible.

Data collection was done using Wizard of Oz technique, address search websites and using human and simulated users. Data was required as a domain specific knowledge, to train the ASR, to identify dialogue acts and for experimentation.

A POMDP-based dialogue manager was used for this research to handle uncertainty. Uncertainty arises due to errors from ASR, NLU and users. This research employed an approximation algorithm called Partition Recombination to tackle the issue of tractability. The dialogue manager evaluated for task completion under varying hypothesis and ASR Word Error Rate.

The findings suggest that in general, POMPDP frame work explicitly models uncertainty and hence it is possible to build a Spoken Dialogue System for under resourced languages.

This paper offers empirical evidence that POMDP-based dialog modeling and maintaining multiple hypotheses, increases both accuracy and robustness of the dialogue manager. Increasing the number of n-best list by 6 reduced the error rate by 5.78% and with 6 n-best list, the correctness score had a growth rate of 54.67% when the maximum partition changed from 5 to 20. The belief updated below 0.12 seconds with 20 partitions and 6-nbest list. And the dialogue manager was able to complete a task with an average 8.75 turns with 50% word error rate.

The key strength of this study is that it approached SDS components for under-resourced languages as a source of uncertainty to solve it with the Artificial Inelegance concept of "planning under uncertainty". POMDP models "planning under uncertainty" and SDS-POMDP allows maintaining multiple hypotheses that can add up the evidences available to infer a dialogue state.

However, even though online speech data collection is demonstrated to show how "Deploy, Collect and Improve" can be implemented, the ASR and NLU units are not modified. In addition, this research is limited to the uncertainty part of POMDP. The planning part, which includes policy learning and optimization are not investigated.

5.2 Recommendations

Further study of the components of SDS is still required with the specially designed for the purpose of spoken dialogue systems. This research employed an ASR with low performance a simplified NLU unit which accommodates a sigle slot only (implemented with string matching) to avoid sematic error and NLG unit with single system act. For instance a better perfoming

Amharic ASR can be utilized. Expert knowledge and state-of-the-art approaches to SDS components can be incorporated.

It is recommended to utilize the "Deploy, collect and improve" to collect data an improve domain specific SDS which were not performed due to time limitations. A POMDP-based dialogue manager enables users to complete task at the cost of longer dialogue turns. At the end, they provide the information needed. While serving this, the same framework can be used to collect data and improve the system much better.

It is also recommended to implement a more complicated domain and use planning algorithms to automatically create policy. For this research a simple domain is selected to deal only the uncertainty that arise mainly from ASR. It is interesting to see the reinforcement learning capacity to create policy automatically.

POMDP, in principle can accomadate more evidence. The research showed as more evidence presented to a POMDP-based dialogue manager, task completion, accuracy and robustness increase. It is also vital to explore other ways of adding evidences to enhance the dialogue manager. Additional evidences might be added from language understanding unit and other features of the ASR.

REFERENCES

Ahmed, Manzoor; Riyaz, Romana; Afzal, Saduf (2013). A Comparative Study of Various Approaches for Dialogue Management. *COMPUSOFT, An international Journal of Advanced Computer Technology*, II.

Andersson, Sebastian (2014). *Context Dependent Speech Recognition*. Masters Thesis. University of Edinburgh, Edinburgh. School of Philosophy, Psychology and Language Sciences.

Austin, J. L. (1962). *How to do things with words.* Amen house, London: Oxford University Press.

Bell, Linda (2014). *Linguistic Adaptations in Spoken Human–Computer Dialogues: Empirical Studies of User Behavior*. PhD Thesis. Royal Institute of Technology (KTH), Stockholm.

Bhagavan, M. R. (1990): Technological Leapfrogging by Developing Countries, *Globalization of Technology*.

Benotti, Luciana and Blackburn, Patrick. (2011), *Classical planning and causal implicatures*. Universidad Nacional de Cordoba, Argentina

Blythe, Jim (1999). An Overview of Planning Under Uncertainty. Information Sciences Institute University of Southern California, pp. 37–54.

Bohus, Dan; Horvitz, Eric (Eds.) (2009). Learning to Predict Engagement with a Spoken Dialog System in Open-World Settings. *Proceedings of SIGDIAL 2009. London: Queen Mary University of London*.

Bühler, Dirk; Minker, Wolfgang (2011). *Domain-Level Reasoning for Spoken Dialogue Systems.* Springer Science+Business Media, LLC, New York.

Chen, Yun-Nung; Wang, William Yang; Rudnicky, Alexander I. (2013). *Unsupervised Induction and Filling of Semantic Slots for Spoken Dialogue Systems Using Frame-Semantic Parsing.* School of Computer Science, Carnegie Mellon University

Chinaei, Hamidreza (2013). *Learning Dialogue POMDP Model Components from Expert Dialogues*. Phd. Laval University, Quebec.

Crummey, Donald (2014). Literacy in an oral society: The case of Ethiopian land records. Available online at http://www.jstor.org/stable/25473353.

Cuayahuitl, Heriberto (2005). Spoken Dialogue Management Using Hierarchical Reinforcement Learning and Dialogue Simulation.

De Mori, Renato; Béchet, Frederic; Hakkani-Tür, Dilek; McTear, Michael; Riccardi, Giuseppe; Tur, Gokhan (2008). Spoken Language Understanding.

Ege, Svein; Aspen, Harald; Teferra, Birhanu; Bekele, Shiferaw (Eds.) (2009). Amharic Speech Recognition: Past, Present and Future. *Proceedings of the 16th International Conference of Ethiopian Studies*. Trondheim.

Feldmann, Valerie (2003): Mobile Overtakes Fixed: Implications for Policy and Regulations. In *International Telecommunication Union (ITU)*.

Fong, Michelle W. L. (2009). Technology Leapfrogging for Developing Countries. *IGI Global*,

Gabsdil, Malte; Lemon, Oliver (2004). Combining Acoustic and Pragmatic Features to Predict Recognition Performance in Spoken Dialogue Systems.

Gould, John D.; Lewis, Clayton (1985): Designing for Usability: Key Principles and What Designers Think. 28 volumes: Communications of the ACM.

Grover, Aditi Sharma; Plauché, Madelaine; Barnard, Etienne; Kuun, Christiaan (2008): HIV Health Information Access using Spoken Dialogue Systems: Touchtone vs. Speech.

Henderson, Matthew; Thomson, Blaise; Young, Steve (2014): Word-Based Dialog State Tracking with Recurrent Neural Networks. In Engineering Department, Cambridge University, CB2 1PZ, UK.

Jokinen, Kristiina (2000). Learning Dialogue Systems, Centre for Evolutionary Language Engineering.

Jurafsky, Daniel; Martin, James H. (2006): Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition.

Kothari, C. R. (2004). Research Methodology : Methods and Techniques, checked on 26/12/2014.

Lamel, Lori (1998). Spoken Language Dialog System Development and Evaluation at LIMSI, checked on 21/12/2014.

Landay, James A. (2000). SUEDE: A Wizard of Oz Prototyping Tool, checked on 21/12/2014.

Lee, Cheongjae; Jung, Sangkeun; Kim, Kyungduk; Lee, Donghyeon; Lee, Gary Geunbae (2010): Recent Approaches to Dialog Management for Spoken Dialog Systems.

Lison, Pierre (2013): Structured Probabilistic Modeling for Dialogue Management.

Li, William; Glass, Jim; Roy, Nicholas; Teller, Seth (2013): Probabilistic Dialogue Modeling for Speech-Enabled Assistive Technology.

Malaka, Rainer; Zipf, Alexander (2000). DEEP MAP Challenging IT research in the framework of a tourist information system.

Marilyn A. Walker; Diane J. Litman; Candace A. Kamm; Alicia Abella (1997): Evaluating Interactive Dialogue Systems: Extending Component Evaluation to Integrated System Evaluation.

Minker, Wolfgang; Bennacef, Samir (2004): Speech and Human-Machine Dialog (The Springer International Series in Engineering and Computer Science).

Morbini, Fabrizio; Audhkhasi, Kartik; Sagae, Kenji; Artstein, Ron; Can, Dogan; Georgiou, Panayiotis et al. (2013): Which ASR should I choose for my dialogue system?.

Moreira, Catarina; Mendes, Ana Cristina; Coheur, Lusa; Martins, Bruno (2011): Towards the Rapid Development of a Natural Language Understanding Module. In Instituto Superior Tecnico, INESC-ID Volume 6895, pp. 309–315. Available online at http://link.springer.com/chapter/10.1007%2F978-3-642-23974-8_33, checked on 27/12/2014.

Pérez-Marín, Diana; Pascual-Nieto, Ismael (2011): Future Trends for Conversational Agents, pp. 395–400.

Petrik, Stefan (2004): Wizard of Oz Experiments on Speech Dialogue Systems. Design and Realisation with a New Integrated Simulation Environment. Masters. Graz University of Technology, Graz. Institute of Signal Processing and Speech Communication.

Pietquin, Olivier (2009): Machine Learning Methods for Spoken Dialogue Simulation and Optimization.

Plauché, Madelaine; Çetin, Özgür; Nallasamy, Udhaykumar (2006): How to Build a Spoken Dialog System with Limited (or no) Language Resources.

Qiao, Fang; Sherwani, Jahanzeb; Rosenfeld, Roni (2010): Small-Vocabulary Speech Recognition for Resource-Scarce Languages.

Sagae, Kenji; Christian, Gwen; DeVault, David; Traum, David R. (2009): Towards Natural Language Understanding of Partial Speech Recognition Results in Dialogue Systems.

Salvi, Giampiero and Vanhainen, Niklas (2014) KTH, School of Computer Science and Communication, Stockholm, Sweden

Solomon Teferra; Binyam Ephrem; Enchalew Yifru; Kassa Tilahun; Lemlem Hagos; Mohammed-hussen Abebeker; Taye Girma (2012): Human Language Technologies for Ethiopian Languages: Challenges and Future Directions. In LIG, Université Joseph Fourier (UJF) ITPhD Program; Addis Ababa University.

Taguchi, Genichi (1993). Taguchi on Robust Technology Development: Bringing Quality Engineering Upstream. New York, NY: ASME Press.

Thomson, Blaise (2009): *Statisitcal Methods for spoken dialogue manager Blaise Thomson*. PhD Thesis. University of Cambridge. Department of Engineering.

Thomson, Blaise; Young, Steve (2009): Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. In University of Cambridge, Engineering Department, Cambridge CB2 1TP, United Kingdom.

W. Eckert, H. Niemann(1994): *Semantic Analysis in a Robust Spoken Dialog System*, Friedrich-Alexander University, Erlangen, Germany

Wilks, Yorick; Catizone, Roberta; Turunen, Markku (2006): Dialogue Management.

Williams, Jason D. (2010): Incremental Partition Recombination for Efficient Tracking of Multiple Dialog States.

Williams, Jason D.; Young, Steve (2005): Partially Observable Markov Decision Processes for Spoken Dialog Systems.

Yimam, Baye (1997): The Pragmatics of Greeting, Felicitation and Condolence Expressions in Four Ethiopian Languages. In *African Languages and Cultures*, pp. 103–128. Available online at http://dx.doi.org/10.1080/09544169708717816, checked on 21/12/2014.

Young, Steve (Ed.) (2014): Statistical Approaches to Open-domain Spoken Dialogue Systems. *15th Annual SIGdial Meeting on Discourse and Dialogue*. Philadelphia, Pennsylvania, USA, June 15-20. SIGdial, checked on 30/12/2014.

Young, Steve; Breslin, Catherine; Gasic, Milica; Henderson, Matthew; Kim, Dongho; Szummer, Martin et al. (2014): Evaluation of Statistical POMDP-based Dialogue Systems in Noisy Environments.

Zhang, Pengfei (2013): Using POMDP-based Reinforcement Learning for Online Optimization of Teaching Strategies in an Intelligent Tutoring System.

APPENDIX A

Example Output

Maximum Partition 5 and n-best list 1

```
max partition 5 dialog1 n = 1
------ Prior to start of dialog ------
PartitionDistribution:
( id,pid) belief  logBel  [logPri ] description
(  ,  -) 1.00000   0.000 [  0.000] hotres x();foodtype x();subcity x();neigh x();count=281
         1.00000   0.000          -

------ Turn 1 ------
System Action: [ask] request(hotres) {hotres:3} "እስቲ በ hotres ይንገሩኝ?"
User Action: [ig] hotres=RESTORANT
** ASR Result: **
[ig] hotres=HOTEL (0.213855)
[ig] hotres=RESTORANT (0.348027)
[ig] hotres=KAFE (0.128917)
[rest] (0.309201)
** PartitionDistribution: **
( id,pid) belief  logBel  [logPri ] description
(  ,  -) 0.39640  -0.925 [ -0.885] hotres x(HOTEL);foodtype x();subcity x();neigh x();count=116
         0.39640  -0.925          -
(  1,  0) 0.60360  -0.505 [ -0.532] hotres=HOTEL;foodtype x();subcity x();neigh x();count=165
         0.60360  -0.505          -
Update time: 0.016000

------ Turn 2 ------
System Action: [ask] request(foodtype) {foodtype:3} "እስቲ በ foodtype ይንገሩኝ?"
User Action: [ig] foodtype=AALEMAAQEF
** ASR Result: **
[ig] foodtype=BAHLAWI (0.313080)
[ig] foodtype=AALEMAAQEF (0.036942)
[ig] foodtype=AINTERNAXNAL (0.013684)
[rest] (0.636294)
** PartitionDistribution: **
( id,pid) belief  logBel  [logPri ] description
(  3,  1) 0.02939  -3.527 [ -4.029] hotres=HOTEL;foodtype=BAHLAWI;subcity x();neigh x();count=5
         0.02939  -3.527          -
(  2,  0) 0.04941  -3.008 [ -3.441] hotres x(HOTEL);foodtype=BAHLAWI;subcity x();neigh x();count=9
         0.04941  -3.008          -
(  ,  -) 0.35421  -1.038 [ -0.966] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh x();count=107
         0.35421  -1.038          -
(  1,  0) 0.56699  -0.567 [ -0.563] hotres=HOTEL;foodtype x(BAHLAWI);subcity x();neigh x();count=160
         0.56699  -0.567          -
Update time: 0.000000

------ Turn 3 ------
System Action: [ask] request(subcity) {subcity:13} "እስቲ በ subcity ይንገሩኝ?"
User Action: [silent]
** ASR Result: **
```

[silent] (1.000000)
[rest] (0.000000)
** PartitionDistribution: **
(id,pid) belief logBel [logPri] description
(3, 1) 0.02939 -3.527 [-4.029] hotres=HOTEL;foodtype=BAHLAWI;subcity x();neigh x();count=5
        0.02939 -3.527   -
(2, 0) 0.04941 -3.008 [-3.441] hotres x(HOTEL);foodtype=BAHLAWI;subcity x();neigh x();count=9
        0.04941 -3.008   -
(, -) 0.35421 -1.038 [-0.966] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh x();count=107
        0.35421 -1.038   -
(1, 0) 0.56699 -0.567 [-0.563] hotres=HOTEL;foodtype x(BAHLAWI);subcity x();neigh x();count=160
        0.56699 -0.567   -
Update time: 0.000000

------ Turn 4 ------
System Action: [ask] request(subcity) {subcity:13} "ይቅርታ, አስቲ በ subcity ይንገሩኝ?"
User Action: [ig] subcity=QIRQOSE
** ASR Result: **
[ig] subcity=QIRQOSE (0.786819)
[ig] subcity=NONE (0.005374)
[ig] subcity=BOLE (0.003493)
[ig] subcity=AAQAQIQALITI (0.002912)
[ig] subcity=YEKA (0.002529) + 8 more
[rest] (0.187812)
** PartitionDistribution: **
(id,pid) belief logBel [logPri] description
(2, 0) 0.00664 -5.015 [-3.441] hotres x(HOTEL);foodtype=BAHLAWI;subcity x();neigh x();count=9
        0.00664 -5.015   -
(1, 0) 0.06616 -2.716 [-0.704] hotres=HOTEL;foodtype x(BAHLAWI);subcity x(QIRQOSE);neigh
x();count=139
        0.06616 -2.716   -
(3, 1) 0.06794 -2.689 [-4.029] hotres=HOTEL;foodtype=BAHLAWI;subcity x();neigh x();count=5
        0.06794 -2.689   -
(6, 1) 0.41517 -0.879 [-2.594] hotres=HOTEL;foodtype x(BAHLAWI);subcity=QIRQOSE;neigh x();count=21
        0.41517 -0.879   -
(, -) 0.44409 -0.812 [-0.966] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh x();count=107
        0.44409 -0.812   -
Update time: 0.016000

------ Turn 5 ------
System Action: [ask] request(neigh) {neigh:109} " neighw ምንድን ነው?"
User Action: [ig] neigh=THEMENJA YAZH
** ASR Result: **
[ig] neigh=THEMENJA YAZH (0.730887)
[ig] neigh=THEMAMAFOQ (0.001159)
[ig] neigh=SUMALETERA (0.000786)
[ig] neigh=DESALEGNHOTEL (0.000686)
[ig] neigh=TEWODROSAADEBABAY (0.000626) + 95 more
[rest] (0.242899)
** PartitionDistribution: **
(id,pid) belief logBel [logPri] description
(2, 0) 0.00334 -5.703 [-3.441] hotres x(HOTEL);foodtype=BAHLAWI;subcity x();neigh x();count=9
        0.00334 -5.703   -
(3, 1) 0.03417 -3.377 [-4.029] hotres=HOTEL;foodtype=BAHLAWI;subcity x();neigh x();count=5
        0.03417 -3.377   -

(, -) 0.22123 -1.509 [-0.975] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh x(THEMENJA YAZH);count=106
0.22123 -1.509 -
(1, 0) 0.24204 -1.419 [-0.563] hotres=HOTEL;foodtype x(BAHLAWI);subcity x();neigh x();count=160
0.24204 -1.419 -
(7, 0) 0.49923 -0.695 [-5.638] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh=THEMENJA YAZH;count=1
0.49923 -0.695 -
Update time: 0.016000

------ Turn 6 ------
System Action: [ask] request(subcity) {subcity:13} "ይቅርታ, እስቲ በ subcity ይንገሩኝ?"
User Action: [ig] subcity=QIRQOSE
** ASR Result: **
[ig] subcity=NONE (0.097508)
[ig] subcity=AADISKETEMA (0.022530)
[ig] subcity=KOLFEQERANIYO (0.014642)
[ig] subcity=NEFASE SELK LAFTO (0.012207)
[ig] subcity=QIRQOSE (0.010603) + 8 more
[rest] (0.796140)
** PartitionDistribution: **
(id,pid) belief logBel [logPri] description
(2, 0) 0.00334 -5.703 [-3.441] hotres x(HOTEL);foodtype=BAHLAWI;subcity x();neigh x();count=9
0.00334 -5.703 -
(3, 1) 0.03417 -3.377 [-4.029] hotres=HOTEL;foodtype=BAHLAWI;subcity x();neigh x();count=5
0.03417 -3.377 -
(, -) 0.22123 -1.509 [-0.975] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh x(THEMENJA YAZH);count=106
0.22123 -1.509 -
(1, 0) 0.24204 -1.419 [-0.563] hotres=HOTEL;foodtype x(BAHLAWI);subcity x();neigh x();count=160
0.24204 -1.419 -
(7, 0) 0.49923 -0.695 [-5.638] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh=THEMENJA YAZH;count=1
0.49923 -0.695 -
Update time: 0.016000

------ Turn 7 ------
System Action: [ask] request(subcity) {subcity:13} "ይቅርታ, እስቲ በ subcity ይንገሩኝ?"
User Action: [oog]
** ASR Result: **
[silent] (1.000000)
[rest] (0.000000)
** PartitionDistribution: **
(id,pid) belief logBel [logPri] description
(2, 0) 0.00334 -5.703 [-3.441] hotres x(HOTEL);foodtype=BAHLAWI;subcity x();neigh x();count=9
0.00334 -5.703 -
(3, 1) 0.03417 -3.377 [-4.029] hotres=HOTEL;foodtype=BAHLAWI;subcity x();neigh x();count=5
0.03417 -3.377 -
(, -) 0.22123 -1.509 [-0.975] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh x(THEMENJA YAZH);count=106
0.22123 -1.509 -
(1, 0) 0.24204 -1.419 [-0.563] hotres=HOTEL;foodtype x(BAHLAWI);subcity x();neigh x();count=160
0.24204 -1.419 -
(7, 0) 0.49923 -0.695 [-5.638] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh=THEMENJA YAZH;count=1
0.49923 -0.695 -

Update time: 0.016000

------ Turn 8 ------
System Action: [ask] request(subcity) {subcity:13} "ይቅርታ, አስቲ በ subcity ይንገሩኝ?"
User Action: [silent]
** ASR Result: **
[ig] subcity=NEFASE SELK LAFTO (0.512569)
[ig] subcity=QIRQOSE (0.030907)
[ig] subcity=AADIS KETEMA (0.020087)
[ig] subcity=BOLE (0.016746)
[ig] subcity=YEKA (0.014545) + 8 more
[rest] (0.341535)
** PartitionDistribution: **
(id,pid) belief logBel [logPri] description
(2, 0) 0.00326 -5.725 [-3.441] hotres x(HOTEL);foodtype=BAHLAWI;subcity x();neigh x();count=9
        0.00326 -5.725   -
(3, 1) 0.03340 -3.399 [-4.029] hotres=HOTEL;foodtype=BAHLAWI;subcity x();neigh x();count=5
        0.03340 -3.399   -
(1, 0) 0.23662 -1.441 [-0.563] hotres=HOTEL;foodtype x(BAHLAWI);subcity x();neigh x();count=160
        0.23662 -1.441   -
(, -) 0.23867 -1.433 [-0.975] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh x(THEMENJA
YAZH);count=106
        0.23867 -1.433   -
(7, 0) 0.48804 -0.717 [-5.638] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh=THEMENJA
YAZH;count=1
        0.48804 -0.717   -
Update time: 0.015000

------ Turn 9 ------
System Action: [ask] request(subcity) {subcity:13} "ይቅርታ, አስቲ በ subcity ይንገሩኝ?"
User Action: [silent]
** ASR Result: **
[silent] (1.000000)
[rest] (0.000000)
** PartitionDistribution: **
(id,pid) belief logBel [logPri] description
(2, 0) 0.00326 -5.725 [-3.441] hotres x(HOTEL);foodtype=BAHLAWI;subcity x();neigh x();count=9
        0.00326 -5.725   -
(3, 1) 0.03340 -3.399 [-4.029] hotres=HOTEL;foodtype=BAHLAWI;subcity x();neigh x();count=5
        0.03340 -3.399   -
(1, 0) 0.23662 -1.441 [-0.563] hotres=HOTEL;foodtype x(BAHLAWI);subcity x();neigh x();count=160
        0.23662 -1.441   -
(, -) 0.23867 -1.433 [-0.975] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh x(THEMENJA
YAZH);count=106
        0.23867 -1.433   -
(7, 0) 0.48804 -0.717 [-5.638] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh=THEMENJA
YAZH;count=1
        0.48804 -0.717   -
Update time: 0.000000

------ Turn 10 ------
System Action: [ask] request(subcity) {subcity:13} "ይቅርታ, አስቲ በ subcity ይንገሩኝ?"
User Action: [ig] subcity=QIRQOSE
** ASR Result: **
[ig] subcity=NEFASESELKLAFTO (0.323016)
[ig] subcity=NEFASE SELK LAFTO (0.099007)

[ig] subcity=KOLFEQERANIYO (0.064344)
[ig] subcity=AARADA (0.053642)
[ig] subcity=QIRQOSE (0.046593) + 8 more
[rest] (0.209632)
** PartitionDistribution: **
(id,pid) belief logBel [logPri] description
(2, 0) 0.00309 -5.781 [-3.441] hotres x(HOTEL);foodtype=BAHLAWI;subcity x();neigh x();count=9
        0.00309 -5.781   -
(3, 1) 0.06030 -2.808 [-4.029] hotres=HOTEL;foodtype=BAHLAWI;subcity x();neigh x();count=5
        0.06030 -2.808   -
(, -) 0.22575 -1.488 [-0.975] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh x(THEMENJA
YAZH);count=106
        0.22575 -1.488   -
(1, 0) 0.24924 -1.389 [-0.563] hotres=HOTEL;foodtype x(BAHLAWI);subcity x();neigh x();count=160
        0.24924 -1.389   -
(7, 0) 0.46162 -0.773 [-5.638] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh=THEMENJA
YAZH;count=1
        0.46162 -0.773   -
Update time: 0.016000

------ Turn 11 ------
System Action: [ask] request(subcity) {subcity:13} "ይቅርታ, እስቲ በ subcity ይንገሩኝ?"
User Action: [ig] subcity=QIRQOSE
** ASR Result: **
[ig] subcity=KOLFEQERANIYO (0.104254)
[ig] subcity=AADIS KETEMA (0.003843)
[ig] subcity=NEFASESELKLAFTO (0.002498)
[ig] subcity=QIRQOSE (0.002082)
[ig] subcity=AAQAQIQALITI (0.001809) + 8 more
[rest] (0.877604)
** PartitionDistribution: **
(id,pid) belief logBel [logPri] description
(2, 0) 0.00308 -5.782 [-3.441] hotres x(HOTEL);foodtype=BAHLAWI;subcity x();neigh x();count=9
        0.00308 -5.782   -
(3, 1) 0.06021 -2.810 [-4.029] hotres=HOTEL;foodtype=BAHLAWI;subcity x();neigh x();count=5
        0.06021 -2.810   -
(, -) 0.22540 -1.490 [-0.975] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh x(THEMENJA
YAZH);count=106
        0.22540 -1.490   -
(1, 0) 0.25041 -1.385 [-0.563] hotres=HOTEL;foodtype x(BAHLAWI);subcity x();neigh x();count=160
        0.25041 -1.385   -
(7, 0) 0.46090 -0.775 [-5.638] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh=THEMENJA
YAZH;count=1
        0.46090 -0.775   -
Update time: 0.015000

------ Turn 12 ------
System Action: [ask] request(subcity) {subcity:13} "ይቅርታ, እስቲ በ subcity ይንገሩኝ?"
User Action: [ig] subcity=QIRQOSE
** ASR Result: **
[ig] subcity=QIRQOSE (0.380142)
[ig] subcity=GULELE (0.022195)
[ig] subcity=AADISKETEMA (0.014425)
[ig] subcity=NEFASE SELK LAFTO (0.012025)
[ig] subcity=AADIS KETEMA (0.010445) + 8 more
[rest] (0.515087)

RESTORANT AALEMAAQEF QIRQOSE THEMENJA YAZH
በእምነት ባርና ሬስቶራንት
የካፌ አገልግሎት፣ የምግብ እና የመጠጥ አገልግሎት
ከተማ: አዲስ አበባ ከተማ ክፍለ ከተማ: ቂርቆስ ሰፈር: ጠመንጃ ያዥ
ቀበሌ: 01/19
ምልክት: ኢትዮጵያ ንግድ ባንክ ጠምንጃ ያዥ ቅርንጫፍ ፊት ለፊት
ስልክ: +251-11-4162204 +251-91-1643280
None
None
None
None
None
None
None
** PartitionDistribution: **
(id,pid) belief logBel [logPri] description
(7, 0) 0.00000 - [-] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x(QIRQOSE);neigh=THEMENJA YAZH;count=0
  0.00000 - -
(2, 0) 0.00070 -7.267 [-3.441] hotres x(HOTEL);foodtype=BAHLAWI;subcity x();neigh x();count=9
  0.00070 -7.267 -
(, -) 0.11247 -2.185 [-0.975] hotres x(HOTEL);foodtype x(BAHLAWI);subcity x();neigh x(THEMENJA YAZH);count=106
  0.11247 -2.185 -
(1, 0) 0.14868 -1.906 [-0.532] hotres=HOTEL;foodtype x();subcity x();neigh x();count=165
  0.14868 -1.906 -
(15, 7) 0.73815 -0.304 [-5.638] hotres x(HOTEL);foodtype x(BAHLAWI);subcity=QIRQOSE;neigh=THEMENJA YAZH;count=1
  0.73815 -0.304 -
Update time: 0.016000

------ Turn 13 ------
System Action: [transfer] destination(subcity=QIRQOSE,neigh=THEMENJA YAZH,foodtype=AALEMAAQEF,hotres=RESTORANT) "Transferring to RESTORANT QIRQOSE in THEMENJA YAZH, AALEMAAQEF"

Example Output (Maximum Partition 5 and n-best list 2)

max partition 5 dialog3 n = 2
------ Prior to start of dialog ------
PartitionDistribution:
(id,pid) belief logBel [logPri] description
(, -) 1.00000 0.000 [0.000] hotres x();foodtype x();subcity x();neigh x();count=281
  1.00000 0.000 -

------ Turn 1 ------
System Action: [ask] request(hotres) {hotres:3} "እስቲ በ hotres ይንገሩኝ?"
User Action: [ig] hotres=KAFE
** ASR Result: **
[ig] hotres=RESTORANT (0.703861)
[ig] hotres=HOTEL (0.008714)
[ig] hotres=KAFE (0.003228)
[rest] (0.284198)
** PartitionDistribution: **

(id,pid) belief logBel [logPri] description
(2, 0) 0.09881 -2.315 [-0.532] hotres=HOTEL;foodtype x();subcity x();neigh x();count=165
        0.09881 -2.315   -
(, -) 0.12734 -2.061 [-1.578] hotres x(HOTEL,RESTORANT);foodtype x();subcity x();neigh x();count=58
        0.12734 -2.061   -
(1, 0) 0.77386 -0.256 [-1.578] hotres=RESTORANT;foodtype x();subcity x();neigh x();count=58
        0.77386 -0.256   -
Update time: 0.016000

------ Turn 2 ------
System Action: [ask] request(foodtype) {foodtype:3} "እስቲ በ foodtype ይንገሩኝ?"
User Action: [ig] foodtype=AALEMAAQEF
** ASR Result: **
[ig] foodtype=AALEMAAQEF (0.457610)
[ig] foodtype=BAHLAWI (0.082811)
[ig] foodtype=AINTERNAXNAL (0.030675)
[rest] (0.428903)
** PartitionDistribution: **
(id,pid) belief logBel [logPri] description
(6, 2) 0.00104 -6.869 [-4.029] hotres=HOTEL;foodtype=BAHLAWI;subcity x();neigh x();count=5
        0.00104 -6.869   -
(7, 0) 0.00305 -5.793 [-4.252] hotres x(HOTEL,RESTORANT);foodtype=BAHLAWI;subcity x();neigh
x();count=4
        0.00305 -5.793   -
(8, 1) 0.02317 -3.765 [-4.029] hotres=RESTORANT;foodtype=BAHLAWI;subcity x();neigh x();count=5
        0.02317 -3.765   -
(2, 0) 0.02598 -3.651 [-1.269] hotres=HOTEL;foodtype x(AALEMAAQEF,BAHLAWI);subcity x();neigh
x();count=79
        0.02598 -3.651   -
(, -) 0.03255 -3.425 [-2.343] hotres x(HOTEL,RESTORANT);foodtype x(AALEMAAQEF,BAHLAWI);subcity
x();neigh x();count=27
        0.03255 -3.425   -
(3, 2) 0.06897 -2.674 [-1.244] hotres=HOTEL;foodtype=AALEMAAQEF;subcity x();neigh x();count=81
        0.06897 -2.674   -
(4, 0) 0.08429 -2.473 [-2.343] hotres x(HOTEL,RESTORANT);foodtype=AALEMAAQEF;subcity x();neigh
x();count=27
        0.08429 -2.473   -
(1, 0) 0.15385 -1.872 [-2.594] hotres=RESTORANT;foodtype x(AALEMAAQEF,BAHLAWI);subcity x();neigh
x();count=21
        0.15385 -1.872   -
(5, 1) 0.60711 -0.499 [-2.173] hotres=RESTORANT;foodtype=AALEMAAQEF;subcity x();neigh x();count=32
        0.60711 -0.499   -
Update time: 0.047000

------ Turn 3 ------
System Action: [ask] request(subcity) {subcity:13} "እስቲ በ subcity ይንገሩኝ?"
User Action: [ig] subcity=QIRQOSE
** ASR Result: **
[ig] subcity=QIRQOSE (0.563899)
[ig] subcity=GULELE (0.003198)
[ig] subcity=LDETA (0.002078)
[ig] subcity=BOLE (0.001733)
[ig] subcity=NEFASESELKLAFTO (0.001505) + 8 more
[rest] (0.421007)
** PartitionDistribution: **
(id,pid) belief logBel [logPri] description

(8, 1) 0.00513 -5.272 [-4.029] hotres=RESTORANT;foodtype=BAHLAWI;subcity x();neigh x();count=5
0.00513 -5.272 -
(2, 0) 0.00723 -4.929 [-1.208] hotres=HOTEL;foodtype x(AALEMAAQEF);subcity x();neigh x();count=84
0.00723 -4.929 -
(, -) 0.00789 -4.843 [-2.204] hotres x(HOTEL,RESTORANT);foodtype x(AALEMAAQEF);subcity x();neigh x();count=31
0.00789 -4.843 -
(3, 2) 0.01037 -4.569 [-1.544] hotres=HOTEL;foodtype=AALEMAAQEF;subcity x(QIRQOSE);neigh x();count=60
0.01037 -4.569 -
(4, 0) 0.01126 -4.487 [-2.805] hotres x(HOTEL,RESTORANT);foodtype=AALEMAAQEF;subcity x(QIRQOSE);neigh x();count=17
0.01126 -4.487 -
(1, 0) 0.03409 -3.379 [-2.594] hotres=RESTORANT;foodtype x(AALEMAAQEF,BAHLAWI);subcity x();neigh x();count=21
0.03409 -3.379 -
(10, 3) 0.05447 -2.910 [-2.594] hotres=HOTEL;foodtype=AALEMAAQEF;subcity=QIRQOSE;neigh x();count=21
0.05447 -2.910 -
(5, 1) 0.08103 -2.513 [-2.643] hotres=RESTORANT;foodtype=AALEMAAQEF;subcity x(QIRQOSE);neigh x();count=20
0.08103 -2.513 -
(11, 4) 0.09509 -2.353 [-3.336] hotres x(HOTEL,RESTORANT);foodtype=AALEMAAQEF;subcity=QIRQOSE;neigh x();count=10
0.09509 -2.353 -
(12, 5) 0.69345 -0.366 [-3.153] hotres=RESTORANT;foodtype=AALEMAAQEF;subcity=QIRQOSE;neigh x();count=12
0.69345 -0.366 -
Update time: 0.062000

------ Turn 4 ------
System Action: [ask] request(neigh) {neigh:109} " neighw ምንድን ነው?"
User Action: [ig] neigh=BEQLO BET
** ASR Result: **
[ig] neigh=BEQLO BET (0.643322)
[ig] neigh=BOLE RUWANDA (0.004995)
[ig] neigh=QEBENA (0.003387)
[ig] neigh=SOSTEGNAPOLISTHABIYA (0.002954)
[ig] neigh=PASTER (0.002699) + 95 more
[rest] (0.243736)
** PartitionDistribution: **
(id,pid) belief logBel [logPri] description
(2, 0) 0.00288 -5.850 [-1.208] hotres=HOTEL;foodtype x(AALEMAAQEF);subcity x();neigh x();count=84
0.00288 -5.850 -
(, -) 0.00314 -5.763 [-2.204] hotres x(HOTEL,RESTORANT);foodtype x(AALEMAAQEF);subcity x();neigh x();count=31
0.00314 -5.763 -
(3, 2) 0.00413 -5.490 [-1.544] hotres=HOTEL;foodtype=AALEMAAQEF;subcity x(QIRQOSE);neigh x();count=60
0.00413 -5.490 -
(4, 0) 0.00460 -5.382 [-2.805] hotres x(HOTEL,RESTORANT);foodtype=AALEMAAQEF;subcity x(QIRQOSE);neigh x();count=17
0.00460 -5.382 -
(1, 0) 0.01562 -4.159 [-2.380] hotres=RESTORANT;foodtype x(AALEMAAQEF);subcity x();neigh x();count=26
0.01562 -4.159 -

(10, 3) 0.02169 -3.831 [-2.594] hotres=HOTEL;foodtype=AALEMAAQEF;subcity=QIRQOSE;neigh
x();count=21
        0.02169 -3.831   -
(5, 1) 0.03298 -3.412 [-2.643] hotres=RESTORANT;foodtype=AALEMAAQEF;subcity x(QIRQOSE);neigh
x();count=20
        0.03298 -3.412   -
(11, 4) 0.03408 -3.379 [-3.441] hotres
x(HOTEL,RESTORANT);foodtype=AALEMAAQEF;subcity=QIRQOSE;neigh x(BEQLO BET);count=9
        0.03408 -3.379   -
(12, 5) 0.27614 -1.287 [-3.153] hotres=RESTORANT;foodtype=AALEMAAQEF;subcity=QIRQOSE;neigh
x();count=12
        0.27614 -1.287   -
(16, 11) 0.60475 -0.503 [-5.638] hotres
x(HOTEL,RESTORANT);foodtype=AALEMAAQEF;subcity=QIRQOSE;neigh=BEQLO BET;count=1
        0.60475 -0.503   -
Update time: 0.079000

------ Turn 5 ------
System Action: [ask] request(hotres) {hotres:3} "ይቅርታ, እስቲ በ hotres ይንገሩኝ?"
User Action: [ig] hotres=KAFE
** ASR Result: **
[ig] hotres=KAFE (0.675215)
[ig] hotres=HOTEL (0.089948)
[ig] hotres=RESTORANT (0.033319)
[rest] (0.201519)
KAFE AALEMAAQEF QIRQOSE BEQLO BET
** PartitionDistribution: **
(id,pid) belief logBel [logPri] description
(11, 4) 0.00000   - [-] hotres x([3 entries]);foodtype=AALEMAAQEF;subcity=QIRQOSE;neigh x(BEQLO
BET);count=0
        0.00000   -   -
(16, 11) 0.00000   - [-] hotres x([3 entries]);foodtype=AALEMAAQEF;subcity=QIRQOSE;neigh=BEQLO
BET;count=0
        0.00000   -   -
(, -) 0.00120 -6.723 [-2.204] hotres x(HOTEL,RESTORANT);foodtype x(AALEMAAQEF);subcity x();neigh
x();count=31
        0.00120 -6.723   -
(1, 0) 0.00316 -5.756 [-2.380] hotres=RESTORANT;foodtype x(AALEMAAQEF);subcity x();neigh
x();count=26
        0.00316 -5.756   -
(2, 0) 0.00651 -5.035 [-0.532] hotres=HOTEL;foodtype x();subcity x();neigh x();count=165
        0.00651 -5.035   -
(5, 1) 0.00668 -5.008 [-2.643] hotres=RESTORANT;foodtype=AALEMAAQEF;subcity x(QIRQOSE);neigh
x();count=20
        0.00668 -5.008   -
(4, 0) 0.00736 -4.911 [-2.805] hotres x(HOTEL,RESTORANT);foodtype=AALEMAAQEF;subcity
x(QIRQOSE);neigh x();count=17
        0.00736 -4.911   -
(21, 11) 0.04903 -3.015 [-3.441] hotres=KAFE;foodtype=AALEMAAQEF;subcity=QIRQOSE;neigh x(BEQLO
BET);count=9
        0.04903 -3.015   -
(12, 5) 0.05595 -2.883 [-3.153] hotres=RESTORANT;foodtype=AALEMAAQEF;subcity=QIRQOSE;neigh
x();count=12
        0.05595 -2.883   -
(22, 16) 0.87010 -0.139 [-5.638] hotres=KAFE;foodtype=AALEMAAQEF;subcity=QIRQOSE;neigh=BEQLO
BET;count=1

0.87010 -0.139 -
Update time: 0.047000

------ Turn 6 ------
System Action: [transfer] destination(subcity=QIRQOSE,neigh=BEQLO BET,foodtype=AALEMAAQEF,hotres=KAFE) "Transferring to KAFE QIRQOSE in BEQLO BET, AALEMAAQEF"