Spoken Dialog System: Direction Guide for Lahore City

Aneef Izhar ul Haq[1], Aneek Anwar[1], Aitzaz Ahmad[1], Tania Habib[1],
Sarmad Hussain[1], Shafiq-ur-Rahman[2]

[1]*Centre for Language Engineering,*
Al-Khwarizmi Institute of Computer Science,
UET, Lahore, Pakistan
aneef.izhar@kics.edu.pk,
firstname.lastname@kics.edu.pk

[2]*FAST NUCES, Lahore, Pakistan*
shafiq.rahman@nu.edu.pk

Abstract

A Direction Guide Spoken Dialog System is a system that asks an Urdu language user about his current and the destination location over a telephone and then gives direction guidance to the user from the user's present location to the destination location. A prototype end-to-end system has been developed that is distributed and Hub & Spoke message based system using the open-source GALAXY Communicator. The end-to-end system uses a Telephony framework and a software based infrastructure that involves the modules of Automatic Speech Recognizer, Text-to-Speech Synthesizer, RavenClaw Dialog Manager, a Backend database and an Interaction Manager. Currently the end-to-end system works for a single session and future work includes the multiple session handling as well.

Keywords— GALAXY; RavenClaw; Spoken Dialog System; Urdu Path Finder for Lahore City

1. Introduction

A Spoken Dialog System is a system that aims to provide services such as Weather and Direction guidance in which the interaction between the system and the user is made using dialogs. This research paper focuses on a Spoken Dialog System framework that was made to be used as a path finder for the city of Lahore. The system includes 49 places of Lahore and provides paths from user's present location to the destination location. Generally, a dialog system includes a telephony framework, a software based infrastructure and a dialog manager.

Outline-- This research paper comprises of 8 sections in which section 2 discusses a brief literature review of different Spoken Dialog end-to-end systems and their comparisons, section 3 discusses the architecture of the prototype system's design while section 4 discusses the RavenClaw Dialog Manager used in the current system, respectively. The details about the telephony framework has been reviewed in the Section 5 and an example dialog has been presented and explained in the Section 6 of this paper. At the end of the paper, the challenges, limitations and future scope of this system has been discussed in section 7 and 8.

2. Literature Review

GALAXY is an open-source architecture for developing new spoken dialog systems. It has a centralized architecture where a Hub communicates with individual modules or servers. The Hub and Spoke GALAXY architecture -- HUB at the center while spokes are the connections made to each of the servers

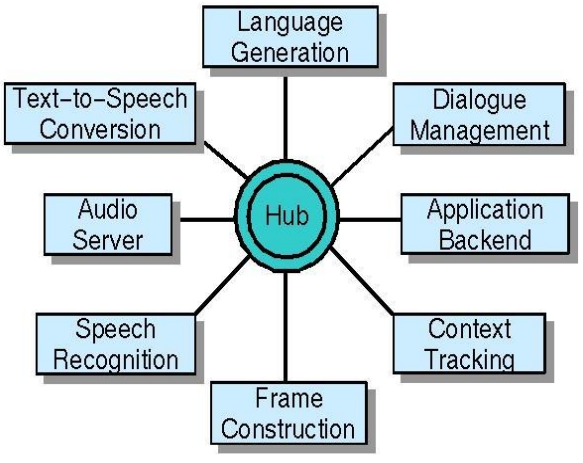

Figure 1: Galaxy Communicator [2]

from the HUB -- has been shown in Figure 1. GALAXY II is an evolutionary form of MIT GALAXY System [1]. It retains the distributed client-server architecture of GALAXY and became the reference architecture for the *DARPA Communicator Program* [2] [3]. All the subsequent systems developed under the program use the GALAXY II architecture to build their respective dialog systems.

2.1. Jupiter

Jupiter is a conversational system made using GALAXY architecture, which was developed by MIT SLS (Spoken Language Systems) group, became online in 1997. It was a weather information retrieval system using a telephone. Later it was redesigned using Web GALAXY architecture [4] to provide support for graphical interface and was restructured again in 1999 using GALAXY II programmable Hub architecture [2]. The system has a vocabulary of over 2000 words and can provide weather information for nearly 500 cities around the world [4]. Jupiter can respond to the queries about general weather forecast as well as specific queries like temperature, humidity, wind speed, sunrise and sunset time etc. It can also inform the user about the cities in a particular region whose weather information it can provide [5].

2.2. Mercury

Mercury is a telephone-based conversational flight-reservation system designed to allow the end user to plan their air-travel between 226 cities around the world [6]. Based on GALAXY II, it uses the same architecture as Jupiter except for the dialog manager whose management and design are given in [6]. The accuracy of the system, in the context of recognition by *ASR*, is discussed in [7]. This system is now offline and in its place a web-based version is available by the name of *Flight Browser*.

2.3. CMU Communicator

The *CMU Communicator* (CMUnicator) is a travel-planning system developed by Carnegie Mellon Sphinx Group under *DARPA Communicator* program [7]. It uses a central architecture by using a *GALAXY Hub* for inter-module communication. It comprises of 10 different modules. Moreover, it provides both the telephony and the desktop interface. A Process Monitor, which starts the modules and the Hub, monitors the whole end-to-end process as well as the individual modules and restarts them in case they crash.

The complete end-to-end system was developed on Windows 2000 originally and no forward compatibility is guaranteed. The complete system was not tested on UNIX but each individual module is known to work on UNIX so the system can be ported to UNIX. Its distribution is available free under the BSD license (permission to modify and distribute). Later dialog systems by CMU like *RoomLine*, *Olympus* and *Conquest* were built using the same architecture and only certain modules have been replaced or modified.

2.4. GALATEA

GALATEA is a human-like visual agent which can communicate via spoken language. It has been developed by a team of Japanese researchers from various universities and currently maintained by people at University of Tokyo. Unlike other spoken dialog systems, it has an additional module for face-image synthesis which mimics the human face while speaking like synchronizing the lips movement with the speech, nodding the head while answering in affirmation etc. It is an open-source system developed on Ubuntu, Linux and is available for use and redistribution under a BSD license [8] [9]. Despite of being compact and customizable, the system has been developed as a Desktop application strictly and does not support an interface for a telephone connection. Also, most of the related literature is available in Japanese instead of English.

2.5. OLYMPUS

The Olympus architecture is an improved version of the CMU Communicator project. The motivation behind the development of the Olympus was to come up with a framework which is open, transparent, flexible, modular and scalable. The system provides detailed accounts of the internal state of each module to aid the analysis of the running system. The *dialog management* module in the Olympus uses the RavenClaw dialog management framework [10] which comes as a two-tier architecture that separates the domain-specific tasks from the domain independent dialog engine.

A key feature shared by all Olympus components is the decoupling of the domain-specific resources and the domain-independent programs which promotes reusability and also allows the users to modify a particular module as per their needs with lesser effort and keeping the rest of the system intact. The Olympus framework comes with a BSD license that allows the use of the source code for research and commercial use [10].

2.6. Comparison of Different Spoken Dialog Systems

The comparison between different spoken dialog systems is shown in Table 1.

Table 1: Comparison of different Spoken Dialog Systems

Dialog System	Platform	License/ Source Code	Architecture	Speech Recognition (SR)	Speech Synthesis (SSM)	User Interface
JUPITER	Linux	Not Available	GALAXY	SUMMIT	ENVOICE	Telephone / Web-based
MERCURY	Linux	Not Available	GALAXY	SUMMIT	ENVOICE	Telephone / Web-based
CMUnicator	Windows	BSD	GALAXY	SPHINX	Festival	Telephone / Desktop
GALATEA	Linux	BSD	Centralized	JULIAN (Japanese)	HTS	Desktop
Olympus	Windows	BSD	GALAXY	SPHINX-II/SPHINX-III	Kalliope	Telephone/ Desktop
ConQuest	Windows	Not Available	GALAXY	SPHINX-II	Cepstral	Telephone/ Desktop
SUNDIAL	Not Known	Not Available	Pipeline/ Sequential	HMM based	HMM based	Telephone

Jupiter, Mercury, CMUnicator, Olympus, Conquest all are based on GALAXY architecture. Jupiter, Mercury and Galatea are based on Linux platform while CMUnicator, Olympus and Conquest are Windows based. Olympus Spoken Dialog System and the GALAXY communicator Hub & Spoke framework that contained only dummy servers/modules were chosen in building this current prototype system design. GALAXY Communicator Software Infrastructure was used for building the current prototype design along with the Dialog Manager module from the Olympus Spoken Dialog System. Olympus was not chosen as an end-to-end system in our current prototype design as it had a lot of additional functionalities that were not required in this prototype system, hence instead of scaling down the Olympus to remove the additional functional modules, the RavenClaw Dialog Manager from Olympus was used as a stand-alone application in the prototype system along with the GALAXY Communicator.

3. System Architecture

The Spoken Dialog System has been made using two open source entities; the GALAXY framework, based on a networking solution, and RavenClaw [11]. Each module in the GALAXY Communicator communicates with the hub only [12]. In the present system, the communication protocols that are used are the same as those in the GALAXY architecture but every module has been developed and implemented from scratch. The hub uses a program file, written using a high-level scripting language called the *Hub Program File Syntax*, which has a set of rules defined for routing of the information received from various modules and to control the flow of communication. Each module runs as a separate server which proves to be beneficial in scenarios where certain modules need to run on separate server machine. Each server machine's IP and port numbers are listed in the Hub's program file. To start up a communication, first all the servers start up on their respective ports and then Hub starts by loading the routing rules, the Hub Programs and the list of servers along with their corresponding addresses and ports and consequently connects with the servers. Once all the servers and Hub are up, a new dialog session can initiate. A new session is created whenever a user calls on the given number using his telephone or mobile phone.

Figure 2 shows the high level architecture of the prototype's framework design. The GALAXY framework constitutes of the dialog manager (RavenClaw), and the components on the Fedora machine; the Telephony framework constitutes the Asterisk server on the CentOS machine and the physical Hardware. Each block connected with the GALAXY Hub is a server process, called a GALAXY server. The Asterisk server on the other hand, is not a GALAXY server, unlike the rest of the servers, and thus cannot communicate with the GALAXY Hub directly. Therefore, it communicates with the servers on the Fedora machine using socket connections for passing the call initiation information, the user inputs and for receiving the system's response and signaling information for synchronization. The speech recognizer used in this prototype system is an Urdu Automatic Speech Recognizer (ASR) that has been trained on the Sphinx Recognizer. The ASR vocabulary currently uses 49 places inside Lahore and these places were chosen on the basis that they had large training data which resulted in good accuracy. The list of these places is provided in Appendix I. The speech synthesizer uses the Festival Text-To-Speech Synthesizer (TTS). The TTS outputs the system response in English as the Urdu TTS is currently in the developing phase. A Backend server was used as an Application Database, which contains

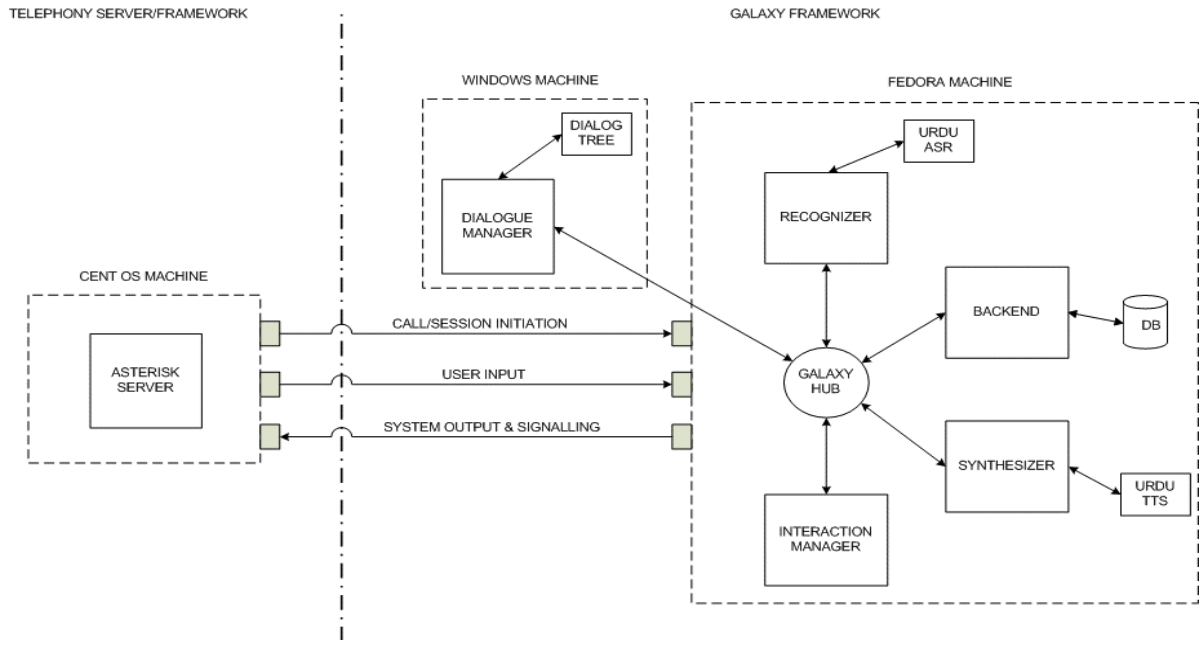

Figure 2: High level System architectural diagram

the direction information of the different locations used in this Location based prototype system. The path descriptions for various locations inside Lahore are retrieved from The Google Directions API [13]. An Interaction Manager has been used that acts as a mediator between the Dialog Manager and the GALAXY Framework. The pre-processing required after determining the nature of the Dialog Manager's output is done by the interaction manager prior to routing the frame to the appropriate destination (Backend or Synthesizer modules).

4. Dialog Manager -- RavenClaw

RavenClaw is the dialog manager used in the Olympus spoken dialog system. A product of the CMU Speech Group, Olympus has been designed for Microsoft Windows based dialog applications built on top of the foundations laid by GALAXY framework. RavenClaw is implemented with the provision of being reused; the dialog management logic and responsibilities are defined separately, along with generic error handling and error recovery techniques. The dialog itself is defined separately in a file as a dialog tree. For each new dialog system, the system developer needs to write a new dialog tree. The dialog manager RavenClaw had to be extracted from the Olympus Spoken Dialog System so as to enable it to use it as a standalone application/module with the GALAXY framework. But the usage of the dialog manager as an independent entity was not possible as it had dependencies on the Parser, Confidence Annotator and Natural Language Processing (NLG) modules that were present in the Olympus Spoken Dialog System but were not required in the current prototype system. So these dependencies had to be accounted for in RavenClaw.

In OLYMPUS, the decoded inputs from the Recognizer enter as an input to the DM via the Parser Module, the Confidence Annotator (that assigns a confidence score to the parsed decoded results) and the Interaction Manager [14]. The output DM frame had to pass through an NLG Module that performed a lookup in generating a complete information message that was then passed to the Synthesizer. In the current framework, the dependency from the Confidence Annotator module of the OLYMPUS system has been catered by assigning a confidence score of either 0 or 1 to the decoded output result of the Recognizer. A correctly decoded result gets the confidence score of 1 while wrong decoding result means a confidence score of 0. The dependency of the NLG module of OLYMPUS on the RavenClaw DM was managed by writing a small routine that was capable of performing a lookup based on the DM's output frame's *prompt_key* key. The *prompt_key* key can be a *welcome* type or a *going_to* type or a *going_from* type, etc. The *prompt_key* tells which text message is to be synthesized and the *prompt_type* tells the nature of the system response (a question which shall lead to an input, a simple informative message, or the requested route to be retrieved from the database).

5. Telephony Framework

A telephony framework/server is the most important part of the Mobile Based Spoken Dialog System as it is the telephony framework that enables the user to input his speech to the Spoken Dialog System or to listen to

the system generated responses and prompts. The telephony framework in the prototype system consists of the following core components:

- Telephone line (Hardware)
- Linksys-spa-400 VOIP box (Hardware)
- x86-based server (Hardware)
- CentOS (Software)
- TrixBox IP PBX (Software)

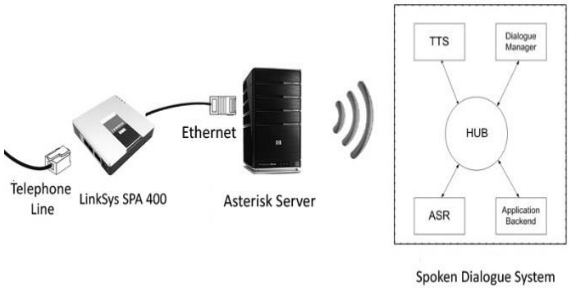

Figure 3: Telephony framework

Figure 3 shows the current design of the telephony framework. The current design of the telephony framework includes a Telephone line that is provided by PTCL(the service provider) and terminated at an FXO port present in the LinkSys SPA 400 VOIP Gateway device. An Ethernet cable is used to connect the Gateway device to the dedicated Asterisk Server that runs on the Operating System of CentOS and the communication software that is used between the Gateway device and the dedicated Asterisk Server is TrixBox. The asterisk server is made to communicate with the Spoken Dialog System – GALAXY Communicator – using a wired or wireless connection. Currently, the system uses a single telephone line but this system can be well-expanded and scaled up if multiple analog trunk lines are used or a single E1 PRI (Primary Rate Interface) line is used that supports a data rate of up to 2.048 Mbps and up to 32 simultaneous channels.

6. Process Flow

Figure 4 shows the process flow diagram of the GALAXY framework. The Asterisk server on the Telephony framework side receives the phone call and connects the caller with the GALAXY framework. Other than that, the Asterisk server plays all the system messages to the user (received from the Synthesizer) and records the user input which it then sends to the Recognizer. The communication protocol of the GALAXY framework is a custom standard. The messages exchanged between various servers are in the form of GALAXY frames. The Hub receives the output frame from each server and routes it to the intended server. This routing information comes from the Hub program file which has the list of rules defined for routing the output frames from each server. The

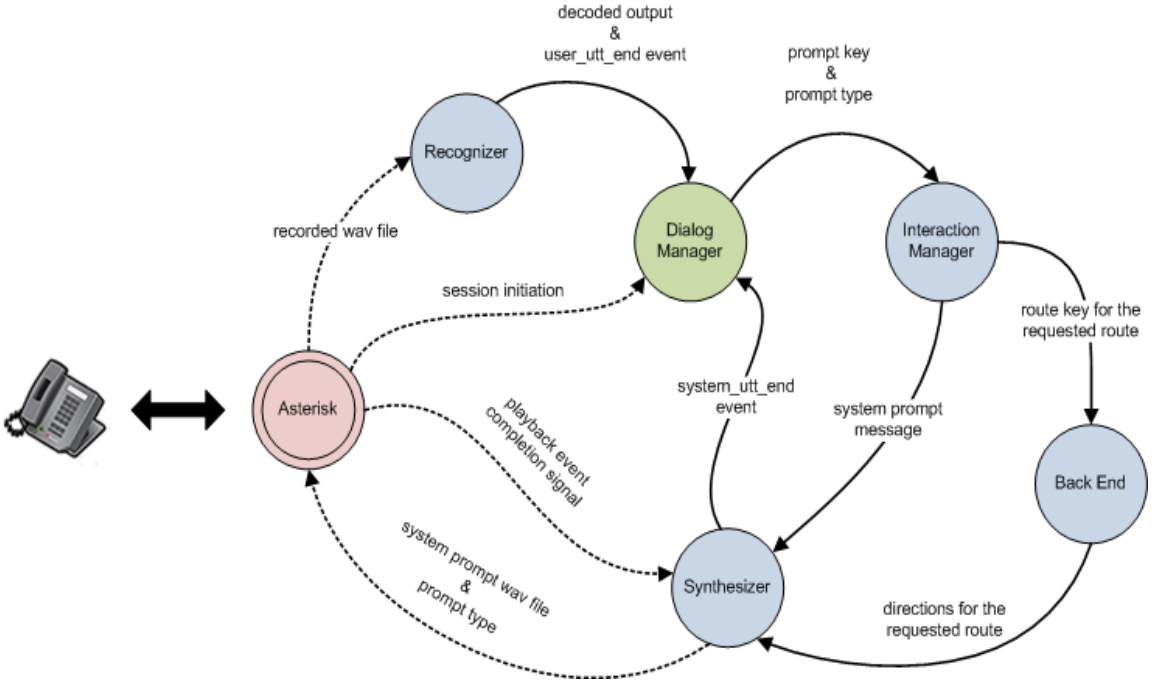

Figure 4: Process flow diagram

recognizer receives the recorded speech file from Asterisk and performs recognition operation before sending the decoded output to the Dialog Manager. The Synthesizer receives the message text of the system's response, performs the text-to-speech synthesis, and sends the synthesized speech file to Asterisk. Furthermore, it also signals to proceed with the dialog after the system response has been played to the user on the Asterisk end. The Dialog Manager controls and executes the flow of the dialog. The Interaction Manager receives the output of the Dialog Manager, performs some pre-processing operations on the basis of the type of the system response to be generated, and sends the processed information to either the Backend (in case of a Backend query) or to the Synthesizer otherwise.
The process flow diagram is explained below.

6.1. Flow diagram

For each system response, the Dialog Manager sends a *prompt key* and *prompt type* to the Interaction Manager as an output. Depending on the *prompt type*, the Interaction Manager chooses to route the key to the Backend (for the retrieving the requested route) or uses the key to construct the text message to be sent to the Synthesizer. The Synthesizer gets the system response synthesized, and sends the speech file to Asterisk along with the prompt type. The Asterisk server plays the system response to the user and then sends the *playback event completion signal* to the Synthesizer. On receiving this signal, the Synthesizer sends the *system_utt_end* event to the Dialog Manager so that it may proceed with the execution of the dialog.

If the Asterisk server receives a *request* type prompt, after sending the *playback event completion signal* to the Synthesizer it plays a beep and records the caller's locations that follow the beep. The recorded speech input is sent to the Recognizer which decodes the speech input and sends the decoded result to the Dialog Manager along with the *user_utt_end* event. On receiving this event, the Dialog Manager processes the inputs and performs the necessary validation operations before commencing the dialog. If there is an error in decoding the input from the user, the error handling is implemented in such a way that the user is then asked to input the location again (up to a maximum of 3 iterations). If still the system is unable to decode correctly, the system informs the user with an error message that his inputs were not decoded successfully and the call is then terminated.

6.2. Example Dialog

An example dialog conversation has been shown below:

System: *Welcome to the CLE Direction Guide.*
System: *What is your current location?*
User: انارکلی
System: *Where are you headed to?*
User: گوالمنڈی
System: *Just a minute let me check that for you.*
System: *Total distance is 2.3 km. Head southwest towards Katchery Rd. After 77m take the 1st left towards Katchery Rd. After 0.2 km at the roundabout, take the 1st exit onto Mayo Hospital Rd. After 0.5 km, turn left onto Napier Rd. After 0.2 km, turn left onto Hospital Rd. After 66 m, continue onto Mayo Hospital Rd. After 0.4 km, turn right onto Railway Rd. After 0.7 km, turn right at Barafkhana chowk onto Flemming Rd. After 0.1 km, Gawal Mandi, Lahore.*

System: *Thank you for using the CLE Direction Guide. Goodbye!*

7. Challenges

The currently made spoken dialog system faces a few challenges due to the Automatic Speech Recognizer. These challenges can be categorized as misrecognition and non-recognition. In the first scenario, the recognizer can misrecognize a location with some other location in the vocabulary. In such a situation, no error would be generated as the word exists in the vocabulary and the corresponding system response would be played-back to the user. An example in the case of misrecognition is shown below:

System: *What is your current location?*
User: انارکلی
(Recognizer decoded output: انارکلی)
System: *Where are you headed to?*
User: گوالمنڈی
(Recognizer decoded output: چونامنڈی)
System: *Total Distance is 3.0 km, Head southwest toward Katchery Rd, After 77 m, Take the 1st left onto Katchery Rd, After 0.2 km, Take the 2nd left onto New Anarkali Road, After 0.6 km, Slight left at Jalandhar Moti Choor House onto Ganpat Rd, After 0.2 km, At the roundabout, take the 2nd exit onto Circular Rd , After 0.4 km, Keep left to stay on Circular Rd , After 0.2 km, Slight left onto Shahalami Rd, After 81 m, Turn left onto Shah Alami Road, After 0.6 km, At the roundabout, take the 2nd exit onto Dabbi Bazar, After 0.3 km, Turn left onto Gali Kabli Mal, After 0.3 km, Turn right onto Jamadaran Gali, After 24 m, Turn left*

toward Choona Mandi Bazar, After 31 m, Turn left onto Choona Mandi Bazar, After 0.1 km, Choona Mandi, Lahore.

In this example, the destination location *Gawal Mandi* was misrecognized as *Choona Mandi* and system returned the path from *Anarkali* to *Choona Mandi.*

The other type scenario could be when the Automatic Speech Recognizer does not recognize anything. The un-recognition is mainly due to limited training data and vocabulary or it could also be due to the background noise present in the user's environment. In the case when no output is given by the recognizer, then the system asks the user again and again for a maximum count of 3 unless it finds some correct location in the vocabulary. An example of non-recognition is given below:

System: *What is your current location?*

User: انارکلی

(Recognizer decoded output: *Null*)

System: *What is your current location?*

User: انارکلی

(Recognizer decoded output: انارکلی)

System: *Where are you headed to?*

User: گوالمنڈی

(Recognizer decoded output: *Null*)

System: *Where are you headed to?*

User: گوالمنڈی

(Recognizer decoded output: *Null*)

System: *Where are you headed to?*

User: گوالمنڈی

(Recognizer decoded output: گوالمنڈی)

System: *Just a minute let me check that for you.*

System: *Total distance is 2.3 km. Head southwest towards Katchery Rd…*

In the above given example, if there is an instance that no useful output is obtained from a recognizer for 3 consecutive iterations, the user is informed with an error message saying "*An error occurred during recognition. Sorry for the inconvenience",* and the call is terminated. Hence these two challenges i.e. misrecognition and non-recognition, need to be accounted for in the future.

8. Future Work

Currently the prototype has been made to work in a single session, but this can be up-scaled by giving it the ability to establish multiple sessions at the telephony framework end using an E1 line. At the software end, the Dialog Manager module used in the current framework design does not allow concurrent sessions. On the other hand, GALAXY Communicator framework is capable of handling multiple and concurrent sessions. Hence for a workaround, an intermediate entity or module (basically a Conversation Manager) must be made that spawns a new Dialog Manager when a new parallel call is received from a new user. The Conversation Manager will act as an interface between the Dialog Manager(s) and the rest of the framework. The rest of the framework shall be enhanced accordingly. Work is currently being done in developing an Urdu TTS, hence the future scope of this paper also includes the integration of the prototype Urdu TTS in the current system. We also aim to expand the ASR vocabulary to all the important places in Lahore city (currently 49 locations are in the ASR's vocabulary).

Acknowledgments

This work has been conducted through the project, Enabling Information Access for Mobile based Urdu Dialogue Systems and Screen Readers supported through a research grant from ICT RnD Fund, Pakistan.

References

[1] P. R. Cohen, A. Cheyer, M. Wang and S. C. Baeg, "An Open Agent Architecture," in *Proc. AAAI Spring Symposium*, Stanford, California, March 1994.

[2] S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid and V. Zue, "GALAXY-II: A Reference Architecture for Conversational System Development," in *Proc. ICSLP'98*, Sydney, Australia, 30th November - 4th December, 1998, pp. 931-934.

[3] J. Polifroni and S.Seneff, "GALAXY-II as an Architecture for Spoken Dialogue Evaluation," in *Proc. Second International Conference on Language Resources and Evaluation (LREC)*, Athens, Greece, May 31- June 2,2000.

[4] R. Lau, G. Flammia, C. Pao and V. Zue, "WebGalaxy—Integrating spoken language and hypertext navigation," in *Proc. of Eurospeech*, Rhodes, Greece, September, 1997.

[5] V. Zue, S. Seneff, J. R. Glass, J. Polifroni, C. Pao, T. J. Hazen and L. Hetherington, "Jupiter: A Telephone-Based Conversational Interface for Weather Information," *IEEE Trans. Speech and Audio Processing*, vol. 8, no. 1, January, 2000.

[6] J. Polifroni and S. Seneff, "Dialogue Management in the Mercury Flight Reservation System," in *Satellite Dialogue Workshop of the ANLP-NAACL Meeting*, Seattle, Washington, April 2000.

[7] [Online]. Available: http://www.speech.cs.cmu.edu/Communicator/Communicator/. [Accessed 18 November 2013].

[8] [Online]. Available:http://sourceforge.jp/projects/galatea/releases/ [Accessed 30 Novermber 2013].

[9] [Online]. Available: http://hil.t.u-tokyo.ac.jp/~galatea/. [Accessed 30 October 2013].

[10] January 2007. [Online]. Available: http://wiki.speech.cs.cmu.edu/olympus/index.php/Download. [Accessed 7 January 2014].

[11] D. Bohus and A. I. Rudnicky, "The RavenClaw dialog management framework: Architecture and systems," *Computer Speech and Language*, vol. 23, January, 2009.

[12] "GALAXY Communicator Documentation," [Online]. Available: http://communicator.sourceforge.net/sites/MITRE/distributions/GalaxyCommunicator/docs/manual/. [Accessed 13 January 2014].

[13] Google. [Online]. Available: https://developers.google.com/maps/documentation/directions/. [Accessed 13 November 2013].

[14] D. Bohus, A. Raux, T. K. Harris, M. Eskenazi and A. I. Rudnicky, "Olympus: an open-source framework for conversational spoken language interface research," in *Proc. HLT-NAACL 2007*, Rochester, New York, USA, April, 2007.

Appendix – I

ماڈل ٹاؤن	گارڈن ٹاؤن
دھرم پورہ	مسلم ٹاؤن
راوی روڈ	شالیمار باغ
بادامی باغ	مال روڈ
شاہدرہ	بھاٹی چوک
ملتان روڈ چونگی	بتی چوک
لاری اڈہ	بابو صابو

لکشمی چوک	سمن آباد
والٹن	یتیم خانہ چوک
چونا منڈی	علامہ اقبال روڈ
کوٹ لکھپت	گوال منڈی
فردوس مارکیٹ	بند روڈ
نشتر روڈ	چاندنی چوک
فیصل ٹاؤن	باغ جناح
مغل پورہ	داتا دربار
لوئر مال	فیروزپور روڈ
بھٹا چوک	نیلا گنبد
فاطمہ جناح میڈیکل کالج	واہگہ بارڈر
گنگا رام	شیخ زید ہسپتال
شاہ عالم مارکیٹ	شالیمار ہسپتال
شاہی محلہ	شالیمار چوک
تاج پورہ	شملہ پہاڑی
جوڑہ پل	انار کلی
	حال روڈ