DEVELOPMENT OF MULTIPLE AUTOMATIC SPEECH RECOGNITION SYSTEMS IN THE GALAXY FRAMEWORK

Muhammad Qasim, Aneek Anwar, Tania Habib, Sarmad Hussain

Center for Language Engineering
KICS, UET
Lahore, Pakistan
firstname.lastname@kics.edu.pk

ABSTRACT

This paper discusses a spoken dialog system, Bus Reservation System, which makes use of multiple automatic speech recognition systems. A single Speech recognition system for a large vocabulary results in high error rate. Using separate speech recognition system for each dimension of user input reduces the vocabulary size which may lead to better performance. The system was built using open source Galaxy framework and Olympus RavenClaw dialog manager. The system supports multiple sessions and each session's state is maintained by using session ids. The voice response to the user is generated by concatenating the pre-recorded audio files. We learned through testing that performance results for speech recognition are good enough both in laboratory and field. Use of multiple speech recognition systems leads to better performance in systems with various dimensions of user input.

Index Terms— spoken dialog system, multiple automatic speech recognition, galaxy, ravenclaw, HMM, bus reservation system

1. INTRODUCTION

Spoken dialog systems provide speech interface to user in order to access information. Most spoken dialog systems use a single Automatic Speech Recognizer (ASR) to understand the user's response. This paper presents a system that uses multiple ASRs depending on the dimension of the user's response. Currently, the state-of-the-art in speech recognition is far from being perfect which results in high error rate in case of large vocabulary. Therefore, it makes sense to use separate ASRs for each dimension of user input as it reduces the vocabulary size for each ASR which in turn can lead to better performance. Also, it allows to divide the dialog in such a way that system asks the user about each field individually using a directed question which helps in limiting the user response. Such a system assists user to decide what its response should be.

The above mentioned method is implemented on a bus reservation system where the user's response can consist of a city name, time value or a number and a separate ASR is used for each of these fields. The bus reservation system was built to be used for travel reservation from Lahore city to 44 other cities of Pakistan. It uses a form-filling approach and is based on finite state machine model where each state is associated with a question or prompt and user's response is used to determine the next state of the system.

The rest of the paper is structured as follows: Section 2 presents the literature review, section 3 describes the system architecture and major modules. Process flow of the system is covered in section 4 and section 5 presents the results of speech recognizer. Finally, Section 6 draws some conclusions and outlines the future work.

2. LITERATURE REVIEW

An open-source framework, Galaxy, is used to develop spoken dialog systems [1]. Galaxy consists of a centralized structure where the Hub is a central module through which the modules communicate. Several systems have been developed using the Galaxy architecture. Jupiter, a system developed by MIT Spoken Language Systems group based on Galaxy, was a weather information system for telephone users [2]. Jupiter can provide weather forecast information of nearly 500 cities. Mercury is another Galaxy based flight reservation system which facilitates a telephone caller in planning air-travel to and from 226 cities across the world [3].

Carnegie Mellon Sphinx Group developed CMU Communicator under Defense Advanced Research Projects Agency (DARPA) [4]. It comprises of 10 modules and uses Galaxy Hub for inter-module communication. Olympus, an improved version of CMU Communicator, was designed to be an open, modular, flexible and scalable framework [5]. Olympus uses RavenClaw framework for dialog management. RavenClaw architecture separates dialog tasks from the dialog engine which executes the dialog tasks, this allows easy modification of individual modules [6].

Galaxy Hub and Spoke framework and Olympus dialog system were selected to build the bus reservation system. Olympus offers lot of extra features that bus reservation system did not required, hence the RavenClaw dialog management module was used as stand-alone application together with Galaxy communicator.

3. SYSTEM ARCHITECTURE

Bus Reservation System is developed by utilizing two open source elements; Galaxy framework and RavenClaw. The communication protocols of Galaxy architecture are used but all the individual modules have been developed from scratch. The Hub controls the flow of communication using a Hub program file, written in Hub specific scripting Language [7]. The modules transfer messages among each other using Galaxy frames.

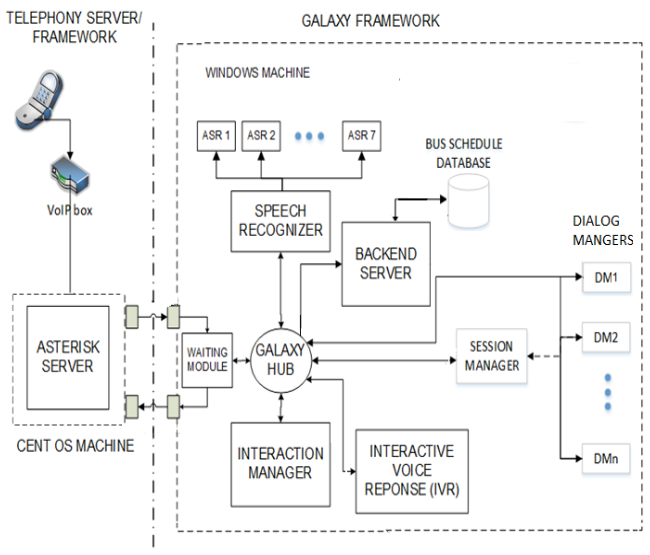

Figure 1 System Architecture Diagram

Figure 1 shows the high level architecture of bus reservation system. Telephony framework enables interfacing of human speech with the Spoken Dialog System. The telephony framework in bus reservation system consists of a telephone line, Linksys spa-3102 VOIP box and an asterisk server running Trixbox Operating System. An Ethernet cable connects VOIP box to the Asterisk Server. The Asterisk server communicates with the Galaxy modules using socket connection. The working of each individual module is presented in the following sub sections.

3.1. Dialog Manager

RavenClaw dialog manager is used in the Olympus dialog system developed for Microsoft Windows and based on Galaxy framework. Details of the RavenClaw Dialog Manager are discussed in Aneef et al, 2014 [8]. The entire dialog is written as a tree in the dialog task tree file of RavenClaw using RavenClaw Task Specification Language (RCTSL) [9]. Figure 2 shows the dialog task tree for bus reservation system. The dialog task tree is traversed from left to right starting from the left most leaf node.

The bus reservation system engages the user in a question-answer conversation to acquire the desired traveling details that include destination city, departure day, departure time and number of seats. It starts with the *Welcome* node which greets the user and after its execution, the dialog moves to next node. The *Get User's Knowledge* node asks the user whether the user knows how to use this system or not. If the user response is in affirmation, the system skips the instructions and moves to next node. Otherwise, the system plays the instructions to user and then moves to next node. The *Get Required Info node* asks the user about the desired destination, departure day and time and number of seats. The *Perform Query* node checks the bus schedule database for any available bus according to the specifications of user. If it finds such a result the reservation is made and skips the *Not An Exact Match* node. In case the system is unable to find any available bus according to the exact specifications of the user, it asks the user about any nearby available bus and makes the reservation for that bus. If even no nearby bus is available, the system informs the user that no bus is available and terminates the call.

3.2. Session Manager

Multiple Galaxy Sessions can run concurrently which demands that a separate Dialog Manager be used for each Galaxy Session. Session Manager handles multiple Dialog Managers and maintains their state. Whenever a new Galaxy session starts, Session Manager locks a particular Dialog Manager with that session. Each message to be sent to Dialog Manager is sent to Session Manager which then forwards the message to a particular Dialog Manager based on the Galaxy session ID. On termination of a Galaxy session, Session Manager closes the corresponding Dialog Manager. The Session Manager can run 10 Dialog Managers concurrently in Bus Reservation System.

3.3. Speech Recognizer

The Speech Recognizer module decodes the utterance spoken by the user and sends the result back to Dialog Manager. Speech recognition system works for isolated words and is

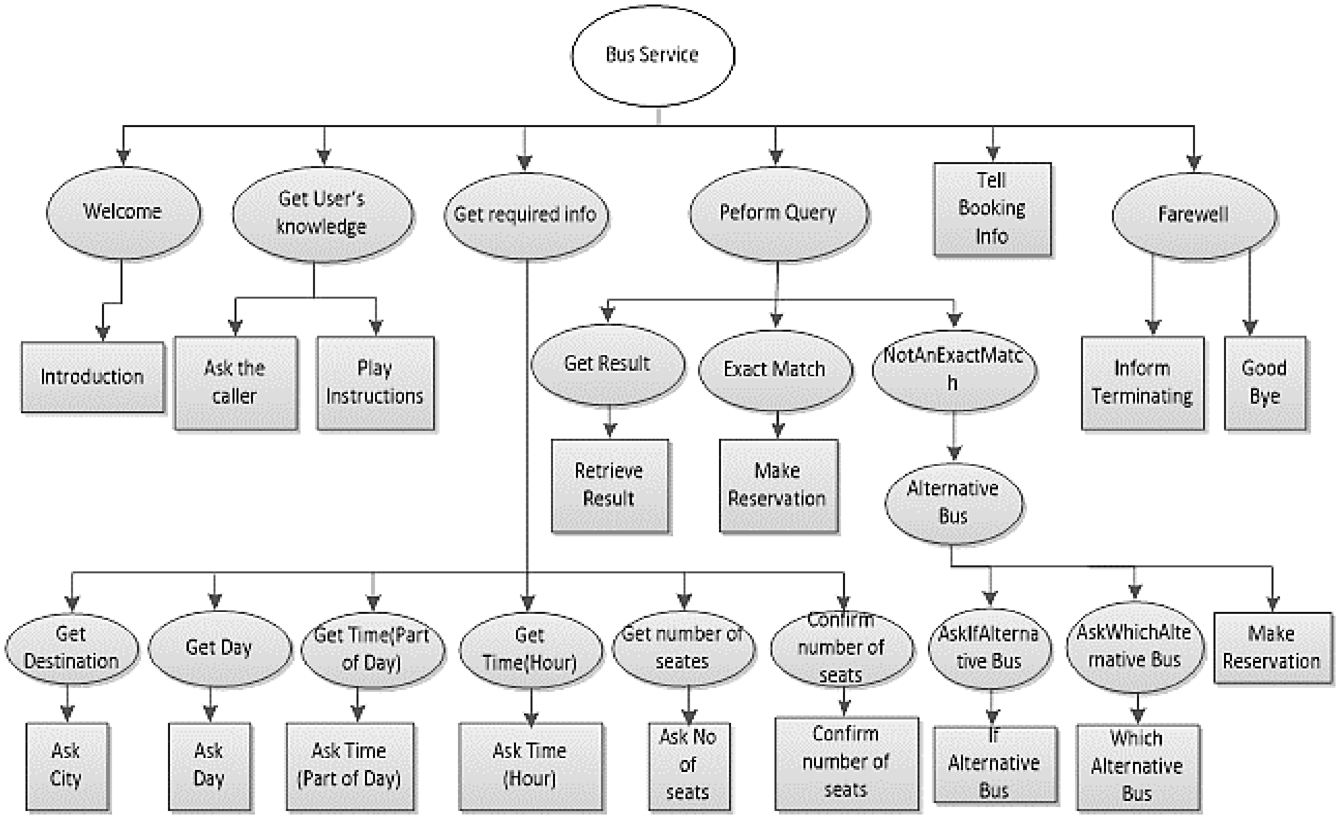

Figure 2 Dialog Task Tree of Bus Reservation System

independent of the speaker. Several different Automatic Speech Recognizers (ASRs) are developed; each ASR decodes a user input in a specific domain, e.g. a separate ASR for city names and a separate ASR for day of departure. There are following seven different ASRs for bus reservation system:

1) ASR for destination city name
2) ASR for departure day
3) ASR for departure time (part of day)
4) ASR for departure time (hour of day)
5) ASR for number of seats
6) ASR for affirmation
7) ASR for determining choice of bus

The above mentioned ASRs are trained on data recorded from the speakers of Punjab province. Table 1 shows the amount of training data and gender ratio.

TABLE 1: Training Data for All ASRs

Recorded data duration	**Number of Speakers**	
	Male	**Female**
18 hours	418	300

The data was recorded in office environment over the telephone channel as the application was to be used through telephones. The collected data was then cleaned to be used for training of ASRs. ASRs are developed using an open source HMM based toolkit Sphinx III [10].Using separate ASRs for all dimensions of user input, the accuracy of speech recognizer improves a lot. Asterisk server records the user's input and sends it to Galaxy server which forwards it to the Speech Recognizer module. Speech Recognizer calls a specific ASR for decoding user's response depending on the question asked from the user. After decoding, the Speech Recognizer sends the decoded output to the Hub which then forwards it to other modules for further processing.

3.4. Backend Module

The backend module populates the Database on its startup and then looks up the query whenever it is called. Bus schedule information is saved in a text file in the following format:

Destination	*Departure Day*	*Departure Time*	*Number of Seats Available*

The system has reservation service for 44 destination cities, 7 days of week and 96 (24 hours x 4) time slots for each day. After the information from the user is collected, a query is sent to the backend. If backed finds an entry in the database according to the desired information of the user and the attribute *Number of Seats available* is greater than or equal to the requested number of seats for reservation, the backend updates the database and returns a 4-digit booking number. In case there is no entry in the database according to the requested information or the *Number of Seats available* is less than the requested number of seats, the backend server returns two entries from the database; one bus available before the user's requested departure time and one after the user's requested departure where *Destination* and *Departure Day* values are according to the user's requirement.

3.5. Interactive Voice Response

This module is called to generate the voice response which is to be played to the user. What needs to be played is determined through the *prompt_value* sent by the Dialog Manager. It performs two functions –find appropriate fixed prompts and synthesize variable prompts. Fixed prompts are just in the form of pre-defined sentences or questions and are played to the user as it is when requested by the Dialog Manager. Variable prompts depend on the information provided by the user or the database through the Backend server and these are generated by concatenating the appropriate audio files. Galaxy session ID is appended at the end of the filename of the generated audio file to make distinction between different files during multiple sessions. Interactive Voice Response (IVR) sends the generated audio file to Asterisk to play it to user and waits for the confirmation from the Asterisk server that voice response to the user has been played.

3.6. Interaction Manager

Dialog Manager is based on the Olympus framework while the rest of the modules are based on the Galaxy Communicator framework, therefore an intermediate module is required to pass messages between Dialog Manager and other modules. The Interaction Manager (IM) serves like a bridge between RavenClaw Dialog Manager and other modules. The frame sent from Dialog Manager is parsed by the Interaction Manager and a new frame is sent to other modules including all the information required to be passed from Dialog Manager to them. The logic is simple, the *prompt_type* and *prompt_value* are extracted from the frame received from Dialog Manager and based on their values, a frame is sent to Backend or IVR. The only functionality of Interaction Manager is to re-route the message of Dialog Manager to either Backend or IVR module based on the prompt type.

4. PROCESS FLOW

When a user calls, the asterisk server answers the call and sends a session initiation signal to Galaxy Server which in response starts a Galaxy session and sends a signal to Session Manager to start a Dialog Manager session. The Session Manager runs an instance of Dialog Manager. The Dialog Manager starts the execution of its dialog task tree. First, it sends a frame to the Interaction Manager containing the message "*inform welcome*". The Interaction Manager parses this message and sends a frame to the IVR module. The IVR module sends the welcome audio file to the asterisk server to be played and waits for the confirmation from the Asterisk server. The asterisk server plays the received audio file and sends a confirmation message to the Galaxy server which is forwarded to Dialog Manager through Session Manager. After the confirmation is received, the Dialog Manager moves to the next node to ask the user whether the user knows how to use this system or not by sending a frame to the interaction Manager with a message of *"request ask_user_knowledge"*. The Interaction Manager parses this message and sends a frame to IVR. IVR sends the appropriate audio file to the Asterisk server and waits for confirmation. But in this case, as the prompt type was *request*, after playing the audio file the Asterisk server records the user response and sends it to the Galaxy framework. Upon receiving the recorded file, the Speech Recognizer module is called to decode the user input. The Speech Recognizer module decides to call the ASR for Confirmation on the basis of the prompt key i.e., *ask_user_knowledge*. The decoded result is sent to Dialog Manager which then executes rest of the nodes of dialog task tree.

In the similar way, the system acquires all the information from the user required for reservation. Afterwards, the system performs a query on the bus schedule database. If there is a bus available, the backend module makes the reservation, generates a random 4-digit number and sends it to the IVR module. IVR informs the user that reservation is done and plays the 4-digit number to the user as reservation number. In case the bus is not available according to the user's requirements, backend returns two alternative buses and user is asked to choose one. Then reservation is made in the user's chosen bus and user is informed about the reservation. Afterwards, a good bye message is played and the call is terminated. A sample dialog between a caller and the system can be viewed online [11].

5. RESULTS AND DISCUSSION

The ASRs developed for bus reservation system were tested using the prerecorded audio files. Table 2 shows the accuracy results of each of the ASR.

TABLE 2: Results of all Automatic Speech Recognizers

Type of ASR	**Vocabulary size**	**Training Utterances**	**Testing Utterances**	**Correct Decoded**	**Accuracy (%age)**
Destinations ASR	44	1543	584	563	96.40
Reservation Day ASR	23	805	307	291	94.78
Reservation Time ASR (Part of Day)	5	170	31	29	93.54
Reservation Time ASR (Hour)	19	659	219	204	93.15
Number of Seats ASR	10	385	150	146	97.33
ASR for choice of Bus	2	70	20	20	100
Confirmation ASR	2	70	26	26	100
Overall	86	3043	1118	1075	96.15

The accuracy results show that all ASRs are performing well. After integrating the ASRs in the Dialog System, field testing of the system was conducted to evaluate its performance in the scenarios and places where the system is intended to be used. Table 3 shows the results of field testing of bus reservation system.

TABLE 3: Field Testing Results of ASRs

Testing Utterances	**Correct Decoded**	**Incorrect Decoded**	**Accuracy (%age)**
222	201	21	90.54054

The overall accuracy of 90.5% of the ASRs is satisfactory enough. One of the challenges to spoken dialog system is the misrecognition of ASRs. This misrecognition is handled by using a keyword "غلط" (GALAT); this keyword makes the dialog manager to go back to the previous question asked. For instance, the user responds with MULTAN when asked for destination and system misrecognizes it as MARDAN. The system then plays the decoded response along with the next question asked from the user; upon hearing the user can correct the misrecognized input by saying the keyword "غلط" (GALAT). The system will now again ask for the desired destination city. Some other challenges include the user's inappropriate response such as the user speaks nothing or speaks multiple words or speaks something out of the vocabulary. In all such cases the user is asked to repeat the response. The system prompts the user to repeat for at most three times and if still the response is not recognized correctly, the call is terminated.

6. CONCLUSION AND FUTURE WORK

The system performs reasonably well in low noise. The use of multiple ASRs has certainly improved the recognition of user input and it can help to improve performance in systems where user input consists of various dimensions. The error handling capabilities of the system make it very user friendly. But the user response is very restricted and selective which makes it slightly hard to be used by a novice user. Interaction between user and system can be modified to be more flexible. The overall time of call for a successful reservation is more than two minutes, it needs to be reduced by decreasing the number of fields. Work is being done in this regard by merging reservation time (part of day) and reservation time (hour) fields. The system can be further improved by using a keyword spotting technique where user can state the desired reservation details in a continuous speech sentence and then keywords regarding destination, day, time and seats can be spotted.

ACKNOWLEDGMENT

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

REFERENCES

[1] S. Senef, E. Hurley, R. Lau, C. Pao, P. Schmid and V. Zue, "GALAXY-II:A Reference Architecture For Conversational System Development," in *ISCA*, Sydney, Australia, 1998.

[2] V. Zue, S. Senef, J. R. Glass, J. Polifroni, C. Pao, T. J. Hazen and L. Hetherington, "JUPITER: A Telephone-Based Conversational Interface," *IEEE Trans. on Speech and Audio Processing,* vol. 8, no. 1, pp. 85-96, 2000.

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

[4] A. Rudnicky and W. Xu, "An agenda-based dialog management architecture for spoken language systems," in *IEEE Automatic Speech Recognition and Understanding Workshop*, 1999.

[5] D. Bohus, A. R. Carne, T. K. Harris, M. Eskenazi and A. I. Rudnicky, "Olympus: an open-source framework for conversational spoken language interface," in *NAACL-HLT-Dialog '07 Proceedings of the Workshop on Bridging the*

Gap: Academic and Industrial Research in Dialog Technologies, 2007.

[6] A. I. Rudnicky, C. Bennett, A. W. Black, A. Chotomongcol, K. Lenzo, A. Oh and R. Singh, "Task and Domain Specific Modelling in the Carnegie Mellon Communicator System," in *Interantional Conference on Spoken Language Processing (ICLSP)*, 2000.

[7] "Galaxy Communicator Documentation: Communicator Hub Programs," [Online]. Available: http://communicator.sourceforge.net/sites/MITRE/distributions/GalaxyCommunicator/docs/manual/reference/pgm.html. [Accessed 10 April 2015].

[8] A. Izhar ul Haq, A. Anwar, A. Ahmad, T. Habib, S. Hussain and Shafiq-ur-Rahman, "Spoken Dialog System: Direction Guide for Lahore City," in *Conference on Language and Technology*, Karachi, Pakistan, 2014.

[9] "RavenClaw Task Specification Language," [Online]. Available: http://wiki.speech.cs.cmu.edu/olympus/index.php/Tutorial_1#The_RavenClaw_Task_Specification_Language. [Accessed 8 April 2015].

[10] "Robust Group's CMU Sphinx Tutorial," [Online]. Available: http://www.speech.cs.cmu.edu/sphinx/tutorial.html. [Accessed 13 March 2015].

[11] "Bus Reservation System's Vocabulary List," Center for Language Engineering, [Online]. Available: http://cle.org.pk/dialog/bus_reservation_system_vocab.pdf.