Natural Language Processing with Text-to-Speech on Android (May 2011)

Sonal Bhatt, *Graduate Student, Arizona State University, Division of Computing Studies.*
Email: sbbhatt1@asu.edu

Abstract— **As the use of mobile devices is expanding and affecting various aspects of human life, the number and smartphone users is dramatically increasing. Consequently, the robustness of interaction between smartphone and human is essential for better system performance. This paper presents the detail implementation approach for interactive natural language system with 'Cab Reservation' application on android smartphone. By using the speech synthesizer technology for the android, the application presents the modality of text-to-speech responses on android device.**

Index Terms— **natural language processing, speech recognition, text-to-speech synthesizer, artificial intelligence, semantic structure, conversational dialog, ad-hoc-quarries.**

I. Introduction

Speech application can be defined as interaction between the user and the computer in more natural way. As people find speaking naturally is easy, it is the most advantageous to incorporate speech into any natural language processing software. Conversational dialog is a verbal action which takes place turn by turn between human and computer with feedback and acknowledgement to indicate understanding. The field of *Artificial Intelligence (AI)* and the idea of a machine dialog with humans are as old as the field of Computer Science. Fifty-three years ago, the British Mathematician Allan Turing proposed the Turing Test in his paper "Computing Machinery and Intelligence" [1]. In the Turing Test, a user A is placed at a terminal with a keyboard and at the other end another user B is placed at a different terminal. In addition at the other end there is a computer program designed to maintain humanlike conversations. The user A cannot see who or what is at the other end. The user A is then engaged in conversations with user B and with the computer program. If the user A cannot tell the difference between the user B and the computer program, then we say that the computer program has passed the "Turing Test". Since Alan Turing's paper was published, for many years the Turing Test has been the ultimate goal of AI and conversational systems.

Speech application should be based on an understanding of the different ways that people use language to communicate [2]. Nowadays people use texting and IVR (Interactive voice response) to communicate with the computers via cell phone. IVR system can be used by telephone's keypad or with the speech recognition. To order or book something with this kind of application, it follows the exact conversational dialog. IVR is prerecorded audio to direct user how to proceed. With the use of speech recognizer and speech synthesizer, the applications based on IVR can be deployed to automobile systems for hands-free operation.[3] Most of the time IVR based application can be used for transactional dialog where grammar is predefined, and user is bound to say or type restricted quarries.

Despite of the advanced AI tools available, the question always remained for how to translate a semantic structure into computer queries or commands that can re-use existing commercial applications and databases that are proprietary to a specific business. Furthermore, such AI suited languages are difficult to use and comprehend. In the latter years software developers have been forced to abandon these languages that are better suited for natural language and opt to develop specific dialog flows from scratch using Java, C++ and now VoiceXML. The dialog is then designed for the specific application, but it tends to limit the user to specify commands, due to the task-oriented nature of these languages. Although these languages have Object Oriented capabilities they are still very much task oriented.

A. Problem Statement

This project implements the process of developing a *conversational dialog* for booking a cab. It allows the complex natural language requests with text and speech. In this application, I have designed and implemented language dialogue for Android smartphone that allows a process of booking a cab. It also allows the user to ask open ended question related to cab, are more conversational. I have accounted all possible outcomes of the user's utterances and have built a reply for every possible situation. The application takes user input as text, entered by the keypad of Android smartphone or it can be the text from speech-to-text facility provided with speech recognizer option for Android smartphone. In this project I have developed the dictionaries of words used for the domain of making a reservation for cab. The words are categorized with English linguistic knowledge. The dictionaries contain nouns, verbs, adjectives, pronouns and numbers. 'Cab Reservation' application can be interrupted by exact transactional dialog to simple question-answer knowledge base conversational dialog. The exact dialog allows the user to book a cab from one destination to another. The dialog is designed to take inputs of departure, destination, date, time and the type of vehicle user want to reserve. It also provides the ability for user to ask open ended

question not related to reservation. This type of question can be called as ad-hoc query that allows the user to get information about cab, i.e. rates, car seats and payment methods. This dialog has to be paired with complex business logic at the application level, in order to support all such possible outcomes. Currently there are VoiceXML based systems are available but those system do not handle spontaneous user requests and interruptions in the dialog. Those applications only support static dialog flow.

The implementation uses the speech synthesizer for the android device. The text-to-speech (TTS) library allows the user to hear the response from the system.

B. *Motivation*

Since speech is such a natural medium for communication, users' expectations of a speech application tend to be extremely high. There are some particular situations when people want to use speech application - for example, when the user's hands and eyes are busy – while driving a car, accessing some location or ordering something over the phone. Sometimes people just want to access their electronic mail while driving. In this kind of situation people tend to use speech application expecting a successful dialog with accurate translation of speech-to-text.

People use airline information system, banking, ordering, reservation system used at a hotel, ATM and many more as the transactional exact requests. Any online system may require transaction that are just information retrieval open ended question and some systems use exact request confirmation dialog. By moving forward developing this application, the idea of ease of access for 'Cab Reservation' can be achieved. The integration of speech-to-text and text-to-speech allows the application to be used in certain environment where people cannot text. Multimodality of texting and speech of this application allows the user to book a cab while they are in the situation where they cannot speak loudly. Furthermore, the features of 'Cab Reservation' application allow easy usability of the application for the people with disabilities.

C. *Applications*

The first thing comes into mind when we talk about text-to-speech application is aiding the handicapped people. Blind people widely benefit from text-to-speech systems, when coupled with Optical Recognition Systems (OCR), [4] which give them access to written information. The market for speech synthesis for blind users of personal computers will soon be invaded by mass-market synthesizers bundled with sound cards. DECtalk (TM) is already available with the latest SoundBlaster cards now. When the computer Aided Learning System combines with a Text-to-speech synthesizer (TTS), it will provide more helpful language education tool [4].

More natural communication can be done between human and machine with the text-to-speech (TTS) and also with the help of good voice recognizer. In this category only having good voice recognizer does not help having a successful communication in more natural way. There has to be a precise natural language understanding. For the domain specific communication, natural language parser plays an important role for high quality multimedia communications [5].

D. *Expected End Results/Goals*

By implementation the 'Cab Reservation' application, following accomplishments are expected to be achieved.

· Allowing the users to be spontaneous, at any point in time of the transactional dialog of reservation and informational retrieval about the cab.

. Allowing the users to speak naturally and ask questions in different ways. At each step providing feedback or acknowledgement prompts.

· Understand the user request and categorize it as exact transaction or open ended fuzzy logic with simply question-answer conversation.

. Building dialogs for interacting with the users in more natural way and engage them in more natural conversations.

. Recognize and understand over 90% of the speaker's requests even with long and complicated sentences.

· Interacting with backend process for information retrieval, updating, inserting or deleting data.

. Allowing the users to get the response in either medium of speech or text.

Once all of the above items are completed, the end result will be the 'Cab Reservation' application with Artificial Intelligent Agent ("AI Agent") which will be capable of processing user's natural speech and text for reserving a cab and engaging the user in conversations with follow up prompts, interacting with a backend application and in turn perform true automated customer self-service providing multi modality response with use of text-to-speech synthesizer on android. Interactive voice response systems are relied on telephone keypad input. Therefore, in this 'Cab Reservation' speech application user can speak any phrase such as "I would like to reserve a cab from Phoenix zoo to Arizona State University" instead of pressing different buttons or numbers for various options on the mobile device.

II. Related Work

Speech technology can use composition, transcription, transaction and collaboration dialog based on particular domain [4]. Natural Language Understanding (NLU) and speech recognition are two independent technologies. When these two technologies can be combined, it provides the powerful human-computer interaction (HCI). Natural language understanding has been an active area research for decades. Since then, the field of Artificial Intelligence (AI) has evolved researchers are borrowing ideas from the fields of mathematics, linguistics, psychology and philosophy. From the research of long decades it can be derived that the conventional computer programs and procedural paradigms were not suited for the challenge at hand. By procedural paradigms I am referring to task oriented programming, such as a program written in a 3rd generation language like COBOL, Fortran, or C [5]. Completely different languages and tools had to be created to help the development

of Conversational Systems, such as Lisp (based on Lamda Calculus), Prolog (based on predicate calculus), SmallTalk (based on objects), semantic nets, frames, etc.

There are many systems in the world to demonstrates the various linguistic issues such as ELISA, Winograd's SHRDLU simulated a robot that control blocks on a tabletop, LUNAR, LIFER [6].

ELIZA is a very early example of natural language processing. When I research about the natural language processing, first comes into mind, ELIZA written at MIT by Josef Weizenbaum between 1964 to 1966 [6]. ELIZA worked by parsing and substitution of key words into phrases [ref]. ELIZA computer program using almost no information about the human thought or emotion, it provided human-like interaction.

Secondly, a chatterbot is designed to simulate an intelligent conversation with humans via speech or text. This chatterbot is based on the theory of Turning Test described in the introduction of this paper. The technology, used by the chatterbot, to generate response is simply finding a keyword from the input and get the reply from database with matching keywords or wording patterns [6].

In the field of linguistic, Chomsky proposed the X-bar theory in 1970 and was further developed by Jackendoff in 1977. X-bar theory identifies the syntactic presumably for human languages [7]. The letter X is used for part of speech. So in the process of parsing a speech or utterances, all the lexicon or ontological categories are assigned to each of the word in speech. Therefore N assigned to noun, A assigned for adjective, V for Verb and P assigned to preposition. The main proposal of X-bar theory is all phrases are defined by rules. According to the X-bar theory, every rule has conceptual structural schema [8].

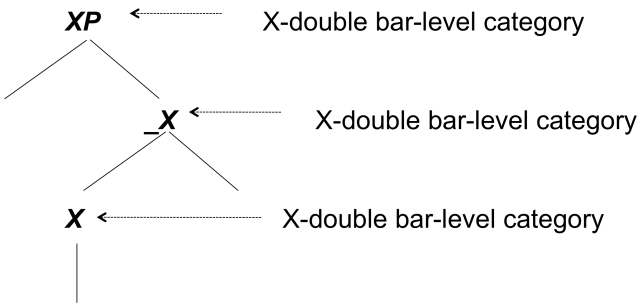

Figure 1: General Understanding of X-bar theory

According to this theory the simple sentence "Mike likes Maria" is parsed as following syntactic category as shown in Figure 2. Where a zero-level word (category) "Maria" combines with some other element, "like" and an x-bar level category formed called V-bar. When X-bar level category combines with some further element, called "Mike" is formed the XP level category called VP [9].

Text-to-Speech synthesizer converts the written text to sounds. Nowadays in the world there are number of speech synthesizers available which have low performance ratio yet satisfactory audio output in different languages such as English, Japanese and Swedish. The techniques used in speech synthesizer includes concatenation of digital recordings, synthesis by rule, where the information is provided for the words to make intonation, and tone.

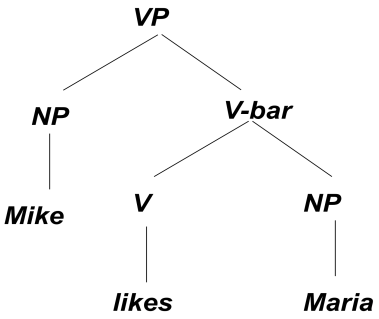

Figure 2: Parsing with X-bar theory

Figure 3 shows the functional diagram of a very general TTS synthesizer. A simple text is process by Natural language processing software with linguistic knowledge and some logical inferences. Then the text goes to make some phonetic transcription with desired intonation and rhymes [5]. Then it passes through the Digital Signal processing to transform that symbolic information into speech with the help of mathematical models, algorithms and computations.

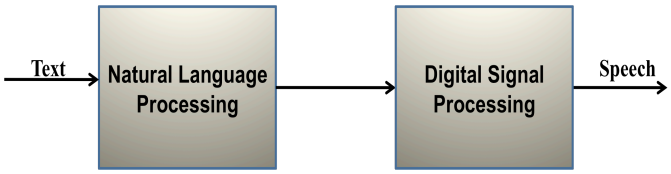

Figure 3: Text-to-Speech Synthesizer

Most of the time text-to-speech synthesizer costs the user to say some specific and restricted text to pronounce. Sometimes the quality of the "emotional dynamics" also comes into the play as the outputs are not as comparable to human speech performances. Although giving less satisfactory output, these synthesizers solve the problem in real time with limited memory requirements. The example of this technology is Emily(ref), that acts as reading coach for children. It provides reading passages and makes correction. Other examples of speech processing consist DECtalk, DragonDictate, and Phonetic Engine [6].

III. Implementation

A. Architecture

Figure 4: Software Architecture

To develop the "Cab Reservation" application with Natural Language Processing and Text-to-Speech, I have used the Client-Server Architecture approach. As the goal of the application is to provide text-to-speech functionality on mobile device, it uses Android operating system device as client. It can be any model of the phone using Android 2.2 or above version. On the server side, The Natural Language Parser is developed in Java. The grammars and dictionaries used for the 'Cab Reservation application are in the .txt format packaged in 'Language' directory of the application package. User can connect to the server from the android phone by giving the host name and the port number of the server. User can speak or type the question to the device and then it connects to the server to parse the speech and process it using natural language processing. The server communicates to the android client and gives the response in text format. Android client is using Text-to-speech library to process the text into speech. Finally user can get the response of the question asked in speech format on android device.

B. *Technologies and Libraries Used*

The "Cab Reservation" application with Natural Language Processing (NLP) and text-to-speech (TTS) functionality is developed using the Java language version 1.6 on the Windows operating system. In addition it requires the Android operating system tool to use as client. In this application the Text-to-Speech library used for the Android device is open-sourced. It can be downloaded to any Android device from android market. This TTS library is also packaged in the software running on Windows machine.

I have developed two different types of grammar for this application. First, to facilitate transactional dialog for booking a cab, I have created cabexactgrammar.txt file and second, for ad-hoc, open-ended questions for cab information retrieval I have created cabfuzzygrammar.txt. To generate these grammars I have used the NuGram IDE [10] plug-in for Eclipse SDK. NuGram IDE is open sourced Eclipse plug-in that offers to generate speech recognition grammar in Augmented Bakus Naur Form (ABNF) format. This format is a plain-text, non-XML, representation of a traditional Bakus Naur Form(BNF) grammar. The body of a grammar consists of a set of *rule definitions*. Each rule definition associates a rule name with a rule expansion. The purpose of the rule definition is to associate a legal rule expansion with a rule name. Most grammars identify a set of possible words that a user might text or say, the top-level rule expansion in a grammar rule is usually a set of alternatives. For example,

$Vehicle = sedan | coup | minivan;

This rule is named 'Vehicle', and the rule expansion is the set of alternatives. This grammar is matched if the user says or texts "sedan", "coup", or "minivan". Here the terminology used for these actual words, user might say, is tokens. I have generated ABNF format grammars and convert them into simple text format for the Natural Language Parser to be understood and processed easily.

C. *Design*

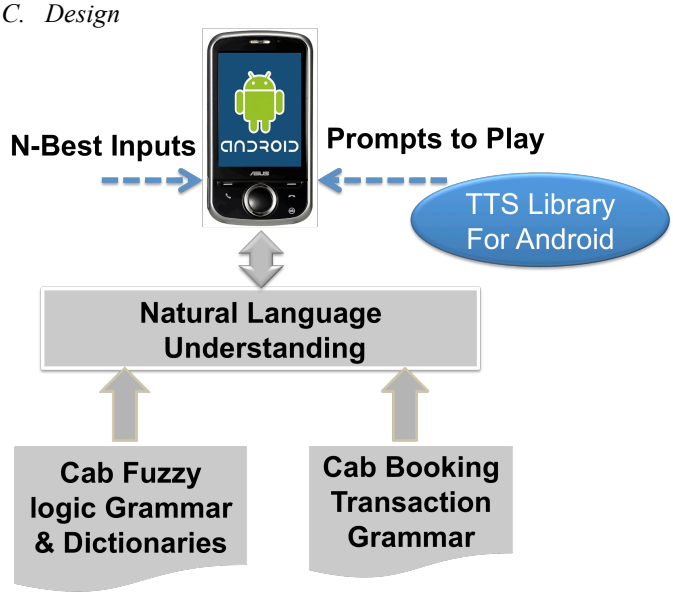

Figure 5: Application Flow

Figure 5 describes the basic communication structure of android client to 'Cab Reservation' application on the server. The android smarphone takes the input as speech or as text and is processed though the server using natural language parser. The natural language parser understands the input and parses the speech in parts and categorizes the request as question-answer fuzzy logic or cab booking transactional dialog based on the inputs of two grammar files. The server gets the appropriate response from the knowledge base text database. This response is processed by Text-to-Speech (TTS) library on android device and plays and displays the prompt on android screen by android client.

An utterance is converted into multiple recognition (n-best) results or captured from text box of the client API and sent to the server API. The server API takes the (n-best) phrase and processes it through the Natural Language Understanding Parser, the Parser first checks the *Cab Booking exact grammar*. If the parsing is not successful, then it checks the *cab fuzzy logic Grammar*. The phrase is then broken apart syntactically by the Parser using the tokens and rules defined in ABNF grammars. As the phrase is being broken apart syntactically, the *Conceptual Structure* is being assembled through the process of parsing. From this point on, the system no longer deals with phrases, but rather with conceptual structures. Dictionaries of nouns, verbs, preposition, adjectives and numbers are used to form the semantic conceptual structure from the parts of speech. This process is done defining the tokens in the grammar file. As the parser goes through each of the rule for correct match, it puts the proper ontological category defined as token in grammar file. For the booking grammar, the ontological category defined as the name of the states of the states file which are the pieces of the information need to fulfill the cab booking process.

For example, if the user asks the open ended question "Can I bring my pet?", Natural Language Parser breaks the sentence

into parts (words). For each word its finding the right category based on the dictionaries in the system. Here the word 'pet' falls into the 'noun' category, so the parser puts that word into the ontological category named DOBJECT. For the verb 'bring' the ontological category is EVENT. So below is the conceptual structure parsed made for the above example question. It changes the first person of 'I' from the question to second person 'you' when it parts the request.

```
(THING(SUBJ(VALUE you))(EVENT bring(DOBJECT my(
VALUE pet))))
```

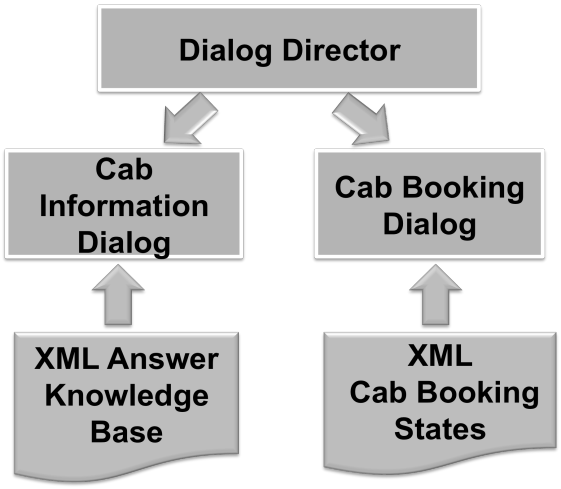

Figure 6: Dialog Director

As shown in Figure 6, The *Dialog Director* is responsible for keeping track of the conversation and building the dialog flow on the fly. If the parsing with the cab booking transactional grammar is successful, then the Dialog Director determines that this is an "exact request" (booking type). If the parsing with the *cab fuzzy logic Grammar* is successful, then it determines that this is a "fuzzy request" (ad-hoc query). If the request is an ad-hoc query or question, then the Dialog Director determines whether the request is interrupting a sub-dialog and it keeps the current dialog context. If the request interrupts the dialog for any other reasons, then the Dialog Director determines whether to interrupt the current sub dialog with the new request. The Dialog director also keeps track of where the dialog is at any given point in time, whether this is a correction, or whether the dialog is at a confirmation or user wants to repeat the prompt.

Once the Dialog Director determines how the request must be processed, a conceptual structure is sent to either the *cab information dialog* or to the *cab booking dialog* as show in Figure 6. If the request is open ended information retirieval based then it contacts the *XML Knowledge Base* named cabanswer.xml file to retrieves a correct answer (in the form of a conceptual structure). These answers are also converted into conceptual networks and are compared heuristically with the conceptual structure. A successful comparison yields a correct answer. If the request is categorized by the director as transactional booking dialog then director contacts the cabstates.xml file to fire the next prompt for the user.

Coding Details

The cab booking application with natural language processing configures the Artificial Intelligence environment. The application engine builds the dialog on the fly, depending on a specific situation in the conversation, where the number of situations could potentially be exponential. That is why the system gives a more natural flow to the dialog that would be difficult to simulate with another programming paradigm.

The nature of the conversations is driven by both grammars described above along with the XML declarative definitions as cabstates.xml for booking dialog and cabanswers.xml to retrieve knowledge base answer. Changes in grammars and the XML definitions make up the nature and content of the conversations. The implementation contains the following components:

· The Java based API for natural language parser that takes user input and parse the speech into meaningful and system understandable semantic, conceptual form.

· A directory consists of text files defined as dictionaries. These dictionaries are categorized based on the English language grammar.

· A cab booking grammar for exact queries/booking transactions in ABNF format.

· A cab fuzzy logic grammar for handling ad-hoc open ended information retrieval request.

These grammars are more powerful as they allow the tokens of the comprehensive dictionaries included within the system. The semantic information refers to ontological categories or other special categories freely chosen by the user. As the booking request transactional dialog uses cab booking grammar and cabstates.xml file to build the conceptual structure, ad-hoc query type questions uses the fuzzy logic grammar to parse the speech and build a semantic structure. This process uses nouns, verbs, preposition, adjectives, and numbers dictionary files to map the words to semantic ontological category. Ontological dictionary in the system helps the word to define in category called place, quantity, time, thing and manner.

This type of category is mapped to the word when the question starts with 'how', 'what',' why', 'where', 'when', 'how many', and 'how much'. Below example shows the sentence parsed to different ontological semantic ontological categories. For example, below Figure 7 shows the different ontological category for sentence, "Can I bring my pet in cab?"

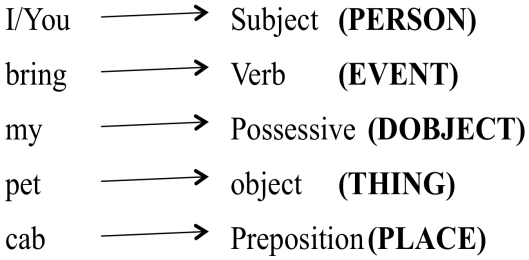

Figure 7: Ontological Category

Once the conceptual structure is made for the user request, the actual answer search begins. The answer has to fulfill the missing ontological category. Therefore when there

are two answers in the answer set with same subjects and events then the knowledge base xml system tries to get the best answer with the right conceptual structure which fulfills the missing ontology from the question to answer the request. Following scenario explains when there is more than one answer in knowledge base system, how the semantic transformation and answer search processed.

The cab driver may have picked the order from office on Sunday

The cab driver picked the order from office

Who picked the order on Sunday

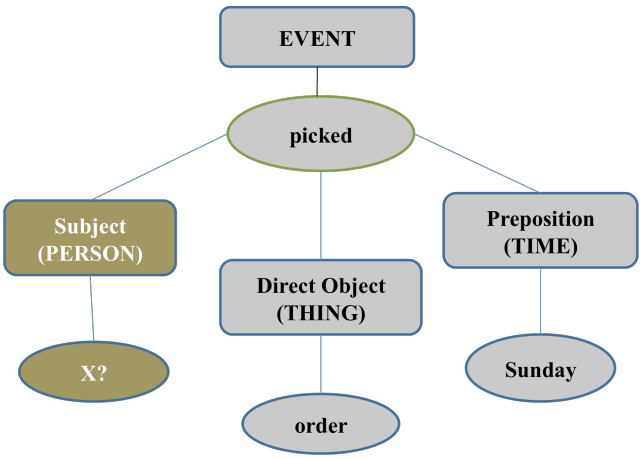

Figure 8: Semantic Transformation and Answer Search

Figure 8 demonstrates the scenario when user asks 'Who picked the order on Sunday?' After recognizing the conceptual structure, system understands that 'subject' category is missing when asking 'who' type of question. From the answer set, it finds two similar answers. By making the conceptual structure of those two answers, application knows that the first answer's conceptual structure is more relevant to the conceptual structure of the question. Therefore it returns the first answer for question asked.

Based on the dialog design on a natural dialog study ensures that the input grammar will match the phrasing actually used by people when speaking in the domain of the application [2]. A natural dialog study also assures that prompts and feedback follow conversational conventions that users expect in a successful interaction. Natural Language Processing based speech applications should adopt language conventions that help the end user know what they should say next and that avoid conversational patterns that violate standards successful and cooperative behavior. This process is done in 'cab booking' application by providing user the follow up prompts that let the user know what to ask or answer next. "Cab Booking" application contains two types of grammars, cab fuzzy logic grammar and cab exact transaction grammar described above.

```
$FirstRule = $ReserveRequest;

$ReserveRequest = (REQUEST reserve) $ReserveAction;
$ReserveAction =(ACTION) $ReserveRules;
$ReserveRules =  $ReserveVerb $ReservePermutations | $ReserveVerb  |
$ReservePermutations;

$ReserveVerb = $Reserve $What;
$Reserve  = reserve | book;
$What  = a taxi | a cab;
$ReservePermutations =  $Time | $Date | $Vehicle | $Departure | $Destination ;

$Departure = (DEPARTURE) $DeparturePre;
$DeparturePreList = i am leaving from |  leaving from | from ;
$DeparturePre = $DeparturePreList $DepartureTerminal | $DepartureTerminal;
$DepartureTerminal = $DepartureDet $DepartureTerminalWords |
$DepartureTerminalWords;
$DepartureTerminalWords = (VALUE) $DepartureValue ;
$DepartureDet  = a | an | the;
$DepartureValue = phoenix zoo;

$Vehicle = (VEHICLE) $VehiclePre ;
$VehiclePreList = a;
$VehiclePostList = vehicle;
$VehiclePre = $VehiclePreList $VehiclePost | $VehiclePost;
$VehiclePost = $VehicleTerminal $VehiclePostList | $VehicleTerminal;
$VehicleTerminal = (VALUE) $VehicleValue ;
$VehicleValue = minivan | sedan | coup;
```

Figure 9: Cab Grammar for 'Booking' request

Figure 9 shows the template of ABNF grammar for transaction sub-dialog. The cab application uses the 'booking' request for the transactional dialog to book a cab. It contains the grammar rules to parse the speech in different semantic transformation. The words in brackets define the ontological category for different words. In above figure (DEPARTURE) and (VEHICLE) are the ontological categories. Grammar rules always start with '$Firstrule'. Each rule is separated by '|" symbol. This grammar file also contains the states of destination, date, and time the user want to reserve a cab for. It contains the token as described above for each state with words in figure as 'minivan', 'sedan', 'coup' for 'Vehicle' state and 'phoenix zoo' for 'Departure' state. These tokens are listed based on user likely say or text for booking a cab. The functionality of ABNF grammar helps the user to say different words for same meaning. For example, in above figure, the grammar rule for '$Reserve' contains the words 'book' and 'reserve'. That grammar rule I combined with '$What' rule which has two tokens of 'taxi' and 'cab'. So now user can triggers the this grammar by asking the same thing in four different ways as 'book a cab', 'reserve a cab', 'book a taxi', and 'reserve a taxi'. For the time and date category when user says different number, the number dictionary file can be referenced as token in this grammar file to take all kind of different number combinations user likely to ask.

Figure 10 describes the cabstates.xml file. When user wants to do the exact request for booking a cab, this state file is read by the system. The request is triggered by the 'reserve' command when user says 'I want to reserve a cab'. The pieces of information needed to complete the booking process, are destination, departure, date, time and what type of vehicle user

```
<REQUEST name="reserve" type="action">
    <STATE type="VEHICLE">
        <PROMPT> What type of vehicle would you like? (minivan, sedan or
            coup)</PROMPT>
    </STATE>
    <STATE type="TIME">
        <PROMPT>What time do you want to be picked up? (Include A.M. or
            P.M.)</PROMPT>
    </STATE>
    <STATE type="DATE">
        <PROMPT> What date do you want to be picked up? </PROMPT>
    </STATE>
    <STATE type="DESTINATION">
        <PROMPT>  What is your destination?</PROMPT>
    </STATE>
    <STATE type="DEPARTURE">
        <PROMPT> Where are you departing from? </PROMPT>
    </STATE>
    <CONFIRMATION>Ok so you want to reserve a VEHICLE being picked up at DEPARTURE
        and being dropped off at DESTINATION, DATE at TIME.
    </CONFIRMATION>
    <CORRECTION>
        <ITEM> What would you like to do then? </ITEM>
    </CORRECTION>
</REQUEST>
```

REQUEST is activated when certain information IS provided

STATE is activated when a piece of information IS NOT provided.

CONFIRMATION is activated when information for all states is provided.

CORRECTION is activated when user says 'no' CONFIRMATION .

Figure 10: Layout of Exact Transaction Dialogue in XML

want to reserve. These pieces of information are stored in <STATE> tag of the XML file. The name of the state is used by the system to make the conceptual structure with the ontological categories. The <STATE> tag is activated when the user first triggers the request by asking the 'reserve' action. Once it is in state file, it prompts the user with text in <PROMPT> tag within <STATE> tag. Once the <STATE> value has been fulfilled it jumps to the next state to capture the value. If the user fulfils all the states in one request then, the system captured all the values and puts in the proper ontological category to form the conceptual structure. Once all the values are captured within xml file, it goes to <CONFIRMTATION> tag. This tag allows the user to review the request. At this stage, user can change his mind and change the values he entered before. Once the user confirms the reservation, the application returns the assertive response of reservation by 'Ok, your reservation has been confirmed'. If the user says 'no' at the confirmation step, <CORRECTION> tag will trigger, and prompts the user for next question. These tags allows the end users know what they should say next and avoid conversational patterns that violate standards successful and cooperative behavior for the reservation process.

Figure 10 describes the parsing of exact request from the cabstates.xml file and cab exact grammar. Action command triggers when user asks for reservation. All the blue boxes indicate the states have to be captured to go to the confirmation dialog. Until system gets all the states value, it asks the user with prompt to fulfill the value. These values can be given to the system at once or one by one. System parses the speech and fills the ontological category with values parsed. Missing states are gathered at end to give the follow up response. This implementation gives the ability user ask for the transaction in many ways naturally. Below figure parses the sentence of "reserve a sedan for today at 3 p.m."

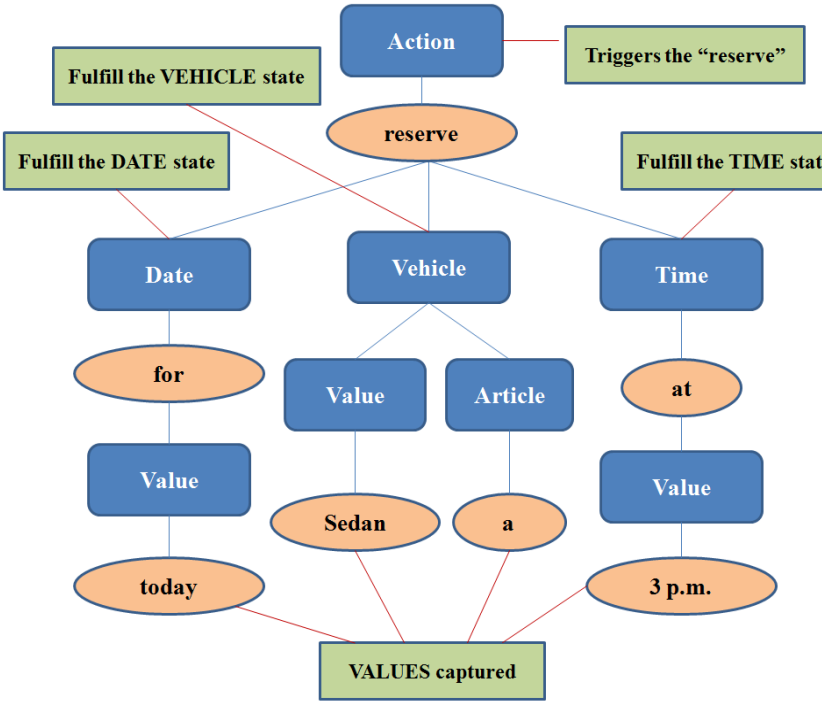

Figure 10: Conceptual Structure of Exact Request

The process of building 'cab booking' application started with building answers sets. As described above this application contains two types of answer sets, knowledge base and exact booking transaction dialog set. From the user requirement, I have determined what queries needed to be answered and provided actual answer in the cabanswer.xml file. For the reservation process, I have designed the transactional dialog for booking process. I have built the states depends on pieces of information needed to fulfill the reservation of cab. Based

on that requirement I have built the cab exact grammar that supports the states.xml file. This grammar is designed by considering all options user might ask or say to complete the state values. The robustness of ABNF grammar allows the developer to add more tokens easily for each state to provide more options to the grammar. For example, the destination state for the cab reservation process consists limited number of destination. In future the value of the destination can be easily changed from 'Phoenix zoo' to 'Aquarium'.

Figure 11 describes the cabanswer.xml file. This file is an xml knowledge base answer set. It is used when the user request is categorized as ad-hoc request. The parser makes the conceptual structure based on cab fuzzy logic grammar. The conceptual structure is mapped from the fuzzy grammar. The system understands which answer to return from the context and meaning of the user's request.

The "Cab Booking" application with Natural Language Processing has the following features which allow the user to communicate to the android phone in more natural way. I have developed these functionalities by adding more dictionaries to the application, and designing the dialog director to allow the interruption between exact and ad-hoc type request. This feature facilitates the user to complete the reservation process at any time in the conversational dialog. User can ask the information retrieval request and come back to complete the reservation. For example, if the user is in middle of the reservation process and application prompts the user of "What type of vehicle would you like to reserve?" , user can refuse to give the answer of this prompt and interrupt the system by asking another open ended question i.e. "What kind of vehicle this cab company offer?" . The Dialog director saves the reservation state, and retrieves the matching answer from knowledge base answer set. The system gives the response of answer and also gives the follow up prompt of reservation process.

Sentence pre-fixes:

Sometimes the users just say some useless words to get the information. The words like 'umm', 'oh', 'please', can be ignored by the system to successfully parse the speech. Sometimes the long phrases like 'I want to', 'Can you please tell me', 'I would like to' used by the user, also does not need to be parsed when retrieving the correct prompt or dialog from the system. I have achieved this functionality by creating the text file "Filters.txt". This file contains all these useless words and phrases user likely to say. The Parser loads the file, and before it starts to parse the request, it checks weather the string is starting with these useless words. If the request contains this word, it will be spitted and the parser will use filtered request to make conceptual structure.

Corrections:

In my project, I have developed the functionality of the correction at any point in the dialog. In real world, when people make reservation, they should have facility to change mind. I have accomplished this functionality by providing explicitly the <CORRECTION> tab in the cabstates.xml file. Example: <CORRECTION>What would you like to do then?</CORRECTION>. Therefore in the reservation process, at confirmation stage, when application plays a confirmation prompt, user has facility to change the values he has entered previously and parser will process the request and will replace the values in the same conceptual structure. <CORRECTION> tag is triggered when confirmation is denied by the user.

For example:

System: You want to reserve a sedan. Is that correct?
User: No.
System: What would you like to do then?
User: Actually I want to reserve a minivan.

At any point in the dialog, when user wants to change the mind, the Filter.txt file has "ChangeMind' words listed, which will be filtered out and parser will get new values. For example,

User: I want to book a sedan.
System: To what date do you want to be picked up?
User: Actually I want to book a minivan.

Navigation Commands:

"Cab Booking" application allows the user to flow the dialog in any direction. It is not like the IVR system, where user is bound to say specific response or type particular number for navigation. I have developed this functionality by creating navigation_command.txt file. It helps to flow the

The system determines which answer to return depending on the context and meaning of the user's question

```
<ANSWER_TEMPLATE>

 <ANSWER>Green Cab accepts cash and all major credit cards. </ANSWER>

<ANSWER>Green Cab is wheelchair accessible. We offer customized vans that accommodate
 wheelchairs and scooters up to 31 inch wide at the regular fare price. </ANSWER>

<ANSWER>Green Cab operates with less than a 15 minute pickup time for all of our calls. </ANSWER>

<ANSWER>Green Cab follows the allowances of four passengers per taxi cab. </ANSWER>

<ANSWER>Green Cab offers a 10% discount for passengers 60+ and the City of Houston $6.00 in
 the city rate that is valid in the downtown district. </ANSWER>

<ANSWER> Green Cab and for-hire vehicles are exempt for the laws regarding car seats.
 Passengers are encouraged to bring their own car seats if travelling with a small child. </ANSWER>

<ANSWER>Green Cab is operated by private contractors who will make individual decisions
 regarding pet policies. All ADA or service animals will be transported.</ANSWER>

</ANSWER_TEMPLATE>
```

Figure 11. XML Knowledge base Answer set for "Cab Booking" application.

dialog in certain directions depends on user's response with yes, no, repeat and cancellation. Some of the mappings with repeat and cancel commands are as follows. REPEAT maps to a variety of phrases: What? What did you say? Can you repeat? Say that again please? Pardon? Pardon me? Please repeat. CANCEL maps to a variety of phrases: Please cancel this transaction. Stop please. I don't want to reserve. YES maps to yes, of course, sure, yeah, yup. NO maps to no, nah, no thanks. This facilitates the user to speak or type more naturally without binding to enter certain input. The application understands all kind of responses listed in the text file.

Synonyms:

I have created 'synonyms.txt' file to map similar context words into one word used in dictionary. For example, user can use bird, animal, cat, dog instead of using word 'pet'. I have mapped these words to word 'pet' which is in 'noun' dictionary. Therefore, user can request with any word, but the parser will refined the request by replacing other word to 'pet' and will continue to parse the sentence.

D. Adopted Design

The "Cab Reservation" with Natural Language Processing is based on the X-bar theory [7] explained above. I have designed the Natural Language parser and Dialog director for the application. I have adopted the execution interface technology for retrieval of the answer, based upon the correct match of the conceptual structure, I developed from the parser. This interface uses "tree" data structure. Using mathematical equations and algorithms, the execution interface calculates the weight and position of the word in the knowledge based answer set. It finds the best matching paragraph. The searching is done based on the "verb" in the sentence. The parser puts the verb in ontological category named "EVENT". Therefore, the execution interface starts the search from leaf node of the tree which is the verb in the sentence.

IV. VALIDATION

A. Validation

The "Cab Reservation" with Natural Language Processing (NLP) using Text-to-speech (TTS) is developed considering validation at every stage of the application development cycle. Using the requirements as the baseline, all the functionalities of the application described above were tested. Application is tested for ad-hoc queries and transactional dialog system. For the android client the application is installed on Motorola device provided by Dr. Richard Whitehouse, Lecturer CTI Department of Engineering at Arizona State University. The server is installed on one of the virtual server of CTI department, Arizona State University. The application is also tested with the speech recognizer for speech input.

B. Results

After testing "Cab Reservation" application on actual android device, it can be derived that answers are more accurate and precise when all words the user likely to use in formulating the question, are found in dictionaries. Answer performs better if the question is fully parsed through either from the cab exact transactional grammar or cab fuzzy logic grammar rather then not finding the matching rule for building proper conceptual structure. 95% of the question maps the proper ontological category and found the words from synonyms files.

Although most of the questions worked, I have found out that if the phrase is not in active voice it is difficult for the application to parse the speech in right semantic structure. Also when the answer for particular question is found in multiple places in knowledge base answer set then system does not give the accurate response. That time the application is found the answers to be ambiguous. Although this 'Cab Reservation' application is not an expert in the subject area, the undesirable effect is that it gives the wrong answer for some information retrieval area rather than no answer.

V. CONCLUSION AND FUTURE WORK

The Natural Language Processing tool with text-to-speech looks through dictionaries and grammars to gather relevant response of user request. I believe that the 'Cab Reservation' application is developed using the concepts and design of natural language processing described in this paper, when deployed, helps the end user to speak more naturally for the reservation process. This project uses primarily the android device, with text-to-speech library supported for android operating system. Second, this application gives the appropriate responses based on the natural language parser and understanding tool to flow the user communication with android smartphone more natural way. Further the solution implemented during this project is scalable, portable, can be deployed to any android device.

Some areas for future work would be to add more intelligence to cab grammar to achieve more natural and robust dialog responses by using collaborative user communication. Hence an important extension to the current work would be to consider mobile agent security mechanisms such as authentication, authorization, encryption and others as in [12] to ensure that secure transaction can be done throughout the booking process. In the future, a more graphical user interface can be developed for ease of the end user and more functions like recording the transaction dialog can be added on as the mobile application. At this stage, the system uses speech recognition of android device itself, it can be developed separately to detect the user's speech input time and made the conversation more natural without touching the device. This application contains the dictionaries and grammar for booking a cab in English language. In future the functionality can be added to work with multiple language dictionaries and grammars.

Furthermore, when the system is unable to find the answer from the answer set or text database, it would be able to search thru intranet or internet to give the related answer of user request. For more accurate communication, the functionality of pushing a related page with the appropriate answer can be added in future. For example, when the user does not know the exact location of the departure, the system will be able to detect the current location of android device with the help of GPS. In addition, the application can be enhanced to automate grammar generator process from the states files. This feature will reduce the amount of human error in linguistic field and

syntax errors. Consequently, it would save the man hours to build the robust conversational dialog. For testing purpose the tester can be developed which will take all the questions user likely ask and will feed into the application, will generate the related answer and dump into the text file to analyze the results to save the time of testing the application with different types of questions.

ACKNOWLEDGMENT

I would like to thank my committee chair Dr Timothy Lindquist for providing his valuable guidance and advice throughout this work. I also extend my thanks to my committee members Professor Richard Whitehouse and Dr. John Femiani for their feedback. Professor Richard's android class inspired me to work on this project idea and helped to build the client-server architecture with the android device.

REFERENCES

[1] A. Turing, "Computing Machinery and Intelligence", *Mind 49, 1950, pp.* 433-460.

[2] "Designing Effective Speech Application", Java™ Speech API Programmer's Guide, Sun Microsystems, Inc.

[3] C.Bajorek, "The state of IVR navigation technology", Computer Telephony Magazine, New York, NY, Volume 8, September 2000.

[4] J. A. Jacko, A. Sears, "The human –computer interaction handbook: fundamentals, evolving technologies, and emerging" New Jersey: Lawrence Erlbaum Associates, 2003, pp. 712-750.

[5] T.Dutoit, "*An Introduction to Text-to-Speech Synthesis."* TTS Research team , TCTS Lab, pp. 2-6.

[6] B.Manaris. "Natural Language Processing :A human-Computer Interaction Perspective", University of Southwestern Lousiana, Louisiana.

[7] Chomsky, N.. "Remarks on Nominalization." In R.Jacobs & P. Rosembaum, eds., *Readings in English, 1970*.

[8] Jackendoff, R. ' *Foundations of Language*.' Oxford University Press, New York, NY. 2002.

[9] Jackendoff, R. ' *Semantics and Cognition*.' The MITPress, Cambridge, MA. 1983.

[10] NuGram Platform, "http://nugram.nuecho.com/product_app/welcome", nu Echo Inc., 2003-2011.

[11] M.D. Riley. "Tree-based modeling for speech synthesis", In G. Bailly, C. Benoit, and T.R. Sawallis, editors, *Talking Machines: Theories, Models, and Designs*, pages 265–273.

[12] Yang Kun, Guo Xin, Liu Dayou, "Security in mobile agent system: problems and approaches", ACM *SIGOPS Operating Systems Review*, Volume 34 Issue 1, January 2000.