A Chat-Bot answering queries of UT-Dallas Students

Himanshu Kandwal
The University of Texas at Dallas
Richardson, TX 75080 USA
hxk154230@utdallas.edu

Abstract

A verbal communication is the most natural way of interaction and the most convenient way of sharing the information. This interaction can be between two people or between a human and a bot, or a chat bot. This project presents a chat-bot system that aims to generate relevant and well reasoned answers by understanding natural language elements of the user query in a restricted semantic and knowledge Domain. It also aims to analyze and publish the sentiments contained in the user query tokens. The knowledge used to train is the information about the UT Dallas University and University library. The bot itself is an android application which supports voice communication by making use of android speech API. This project also introduces a well-structured knowledge representation that examines the user query patterns and provides a veritable response.

Keywords: chat-bot, speech recognition, AIML, natural language processing

1 Introduction

Human-machine conversation is a technology integrating different areas to facilitate the communication between users and the computers using natural language. A related term to machine conversation is the *'chat-bot'*, a conversational agent that interacts with the users turn by turn using natural language. Different terms have been used for a chat-bot such as: machine conversation system, virtual agent, dialogue system, and chatterbot. The purpose of a chat-bot is to simulate a human conversation. The chat-bot architecture integrates a language model and computational algorithms to emulate informal chat communication between a human user and a computer using natural language.

Chatting with computer programs has come a long way since the pioneering artificial intelligence demonstrations like ELIZA and Parry. Early chatter bots were designed as testing platforms for ideas in NLP. Current chatter bots are more sophisticated and have found application in interactive games, as website navigation tools, and for simulating personal help desk assistants. However no program has achieved the level of a human so far (Floridi et al. 2009).

1.1 Domain-specific knowledge system

Domain specific knowledge system refers to the information base which is relevant to a particular domain, for example: a university, country etc.

All chat-bots rely on a knowledge base to deduce the response for the user query. We need a knowledge system in order to train a chat-bot for a domain. Using this learned knowledge the chat-bot can predict or provide response to the user queries. The most formalized way of creating a comprehensive and scalable knowledge base is by using AIML (Artificial Intelligence Mark-up Language).

1.2 Sentiment Analysis

Sentiment analysis is the process of determining whether a piece of input is positive, negative or neutral. It's also known as opinion mining, deriving the opinion or attitude of a speaker.

In our project, we are also analyzing the sentiments present in the user query to evaluate the attitude of the user. This will act as a reflection and hence, help user in understanding the tone of the conversation.

2 Related Work

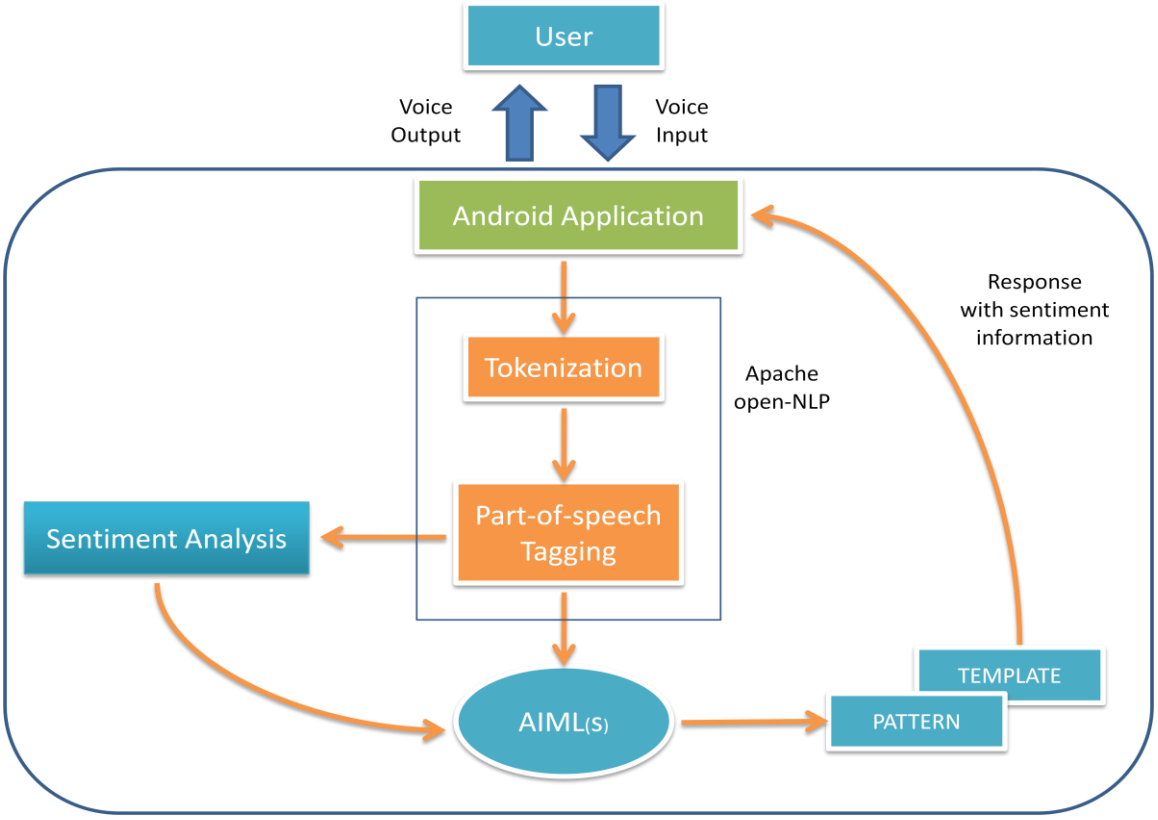
Conversational agents are restricted to the knowledge that is manually *'hand-coded'* in their knowledge files and to the natural language. These restrictions are very costly while scaling the knowledge domain or extending the support for another language. There are research advancements going on to remove these restrictions. One of such research is 'Using corpora in machine-learning chatbot systems' (Abu Shawar B. and Atwell E. 2005), which makes use of Dialogue Diversity Corpus (DDC) to extract the dialog patterns using machine learning.

Another research work which inspired this project is the semantic approach in the 1960s, epitomized by the first implementation of the chatterbot Eliza (Weizenbaum 1966),

3 Project Architecture

In this project, there are many software components and technologies involved and the complete architecture can be depicted as below.

However, this architecture is very rudimentary and rigid. For example, we must often update the knowledge base to include knowledge about the domain.



Architecture of chat-bot

3.1 Android Application

We have made use of a simple android application, providing a nice and rich interactive interface to the user. We have made use of android speech API in order to have the best and precise speech recognition library.

3.2 Apache OpenNLP

Apache OpenNLP is a Java machine learning toolkit for performing numerous natural language processing (NLP) tasks such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and co-reference resolution.

In our project, we are considering only the tokenization and Maxent-POS Tagging functionality of OpenNLP. The training model that we are using is the default model which has been provided by Apache OpenNLP.

3.3 AIML

It is an XML/JSON dialect for creating natural language software agents. AIML is the language most often used by conversational agents, including educational conversational agents such TutorBot and TQ-Bot. Although its simplicity and the relatively good performance of the conversational agents using it makes it attractive, AIML is however a very limited language that can be summarized in a simple pattern matching. Patterns of inputs (users' sentences) and outputs (responses of the conversational agent) is defined largely by expansion and a priori. For example, the well-known Alice bot, one of the best non-task-oriented conversational agents, winner in 2001, 2002 and 2004 of the Loebner contest and second in 2010, is in fact a simple list composed of tens of thousands of predefined questions/answers.

AIML contains several elements or AIML data objects. These elements are made up of units called topics and categories. The topic is an optional top-level element, has a name attribute and a set of categories related to that topic. Categories are the fundamental unit of knowledge in AIML. Each category is a rule for matching an input and converting to an output, and consists of a pattern, which matches against the user input, and a template, which is used in generating the chat-bot answer.

```
<category>
        <pattern>WHAT IS YOUR NAME</pattern>
        <template>My name is John.</template>
</category>
```

XML based representation

```
"category" :{
        "pattern" : "WHAT IS YOUR NAME",
        "template": "My name is John."
}
```

JSON based representation

There are two types of categories: a. ***Atomic categories****:* are those with patterns that do not have wildcard symbols, _ and *.

b. ***Default categories:*** These are those with patterns having wildcard symbols * or _. The wildcard symbols match any input but they differ in their alphabetical order.

3.4 Sentiment Analysis

For sentiment analysis, we are using a manually made, weighted corpus containing the commonly used adjectives, adverbs and the other part-of-speech elements with their sentiment weight.

4 Approach

Our approach is modular as we are applying various concepts and technique while retrieving the response data.

4.1 Preparing domain specific knowledge system

We prepare the domain specific knowledge system by creating the relevant AIMLs which are populated with the domain information.

These AIMLs will act as brain in the whole process. A custom AIML parser is then used to de-serialize the prepared AIML file into AIML Java objects.

We have restricted our domain to the UT Dallas university campus and UT Dallas University library. Hence, the chat-bot will only be able to correctly respond to the user queries if they are related to that domain.

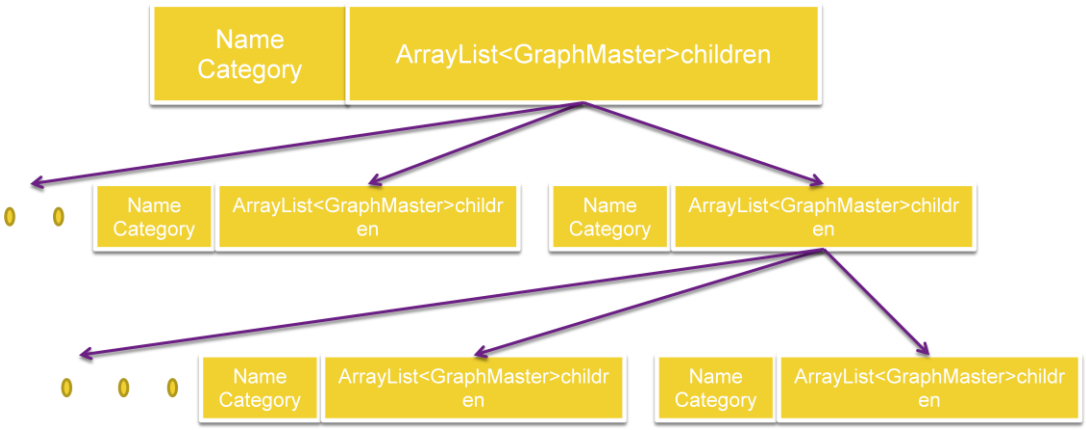
4.2 Creating custom search tree structure

We then devised a special search tree data structure 'KnowledgeGraphMaster', to stores the stimulus-response (pattern-template) categories in this tree.

This data structure merges all the categories within one structure while keeping the value of category in its leaf node. The non-leaf nodes within the search tree carry the name and the future associated children elements.

When a bot client inputs text as a stimulus, the KnowledgeGraphMaster searches the categories for a matching input pattern, along with any associated context, and then outputs the associated template as a response.

The search is done recursively on one pattern token at a time, from left to right. These categories can be structured to produce more complex humanlike responses with the use of a very few markup tags.



Pictorial depiction of Data Structure KnowledgeGraphMaster

4.3 Pattern Matching Algorithm

Before the matching process starts, a normalization process is applied for each input, to remove all punctuations. Using the NLP features from apache openNLP, the user's sentence is analyzed lexically (tokenization) and broken into words. This set of tokens is then taken to perform a grammatical labeling (part-of-speech tagging), whose objective is to classify words according to their grammatical function (nouns, pronouns, verbs, etc...). This classification is based on a dictionary and on the context in which the word appears.

After the normalisation, the AIML interpreter tries to match word by word to obtain the longest pattern match, as we expect this normally to be the best one. This behaviour can be described in terms of the KnowledgeGraphMaster set of files and directories, which has a set of nodes called node-

mappers and branches representing the first words of all patterns and wildcard symbols.

KnowledgeGraphMaster matching is a special case of backtracking, depth-first search. In most cases however there is very little backtracking, so the matching often amounts to a linear traversal of the graph from the root to a terminal node.

Let w1,…,wk be the input we want to match. The KnowledgeGraphMaster matching function may be defined recursively. Initially, the program calls Matcher(r, 1), where r is the root and the index 1 indicates the first word of the input.

The heart of the algorithm consists of three cases:

1. Does the node contain the key"_"? If so, search the subgraph rooted at the **children** node linked by"_." Try all remaining suffixes of the input to see if one matches. If no match was found, proceed ahead with the other possibilities.

2. Does the node contain the key w_h, the jth word in the input sentence? If so, search the subgraph linked by w_h, using the tail of the input $w_{h+1},\ldots,w_k$

. If no match was found, proceed ahead with the other possibilities.

3. Does the node contain the key"*"? If so, search the sub graph rooted at the child node linked by"*". Try all remaining suffixes of the input to see if one matches. If no match was found, return false.

We can define the matching process formally as:

```
Match(n, h) :-
{
 if h > k
  return true;
else if $m = G(n,_) and $ j in [h+1..k+1]
| Match(m, j),
  return true;
else if $m = G(n, w_j) and Match(m, h+1)
  return true;
else if Exists m = G(n, *) and $j in
[h+1..k+1] | Match(m, j),
  return true;
else
  return false;
}
```

The first case defines the boundary condition: 0. If there are no more words in the input, the match was successful.

Points to be noted:

a) At every node, the "_" wildcard has highest priority, an atomic word second priority, and the"*" wildcard has the lowest priority.

b) The patterns need not be ordered alphabetically. They are partially ordered so that"_" comes before any word, and"*" comes after any word.

c) The matching is word-by-word, not category-by-category.

d) The matching algorithm is a highly restricted form of depth-first search, also known as **backtracking**.

4.4 Sentiment Analysis

For sentiment analysis, we manually build the weighted sentiment training model. Word weighting is a technique that provides weights to words in the model to enhance the performance of opinion mining.

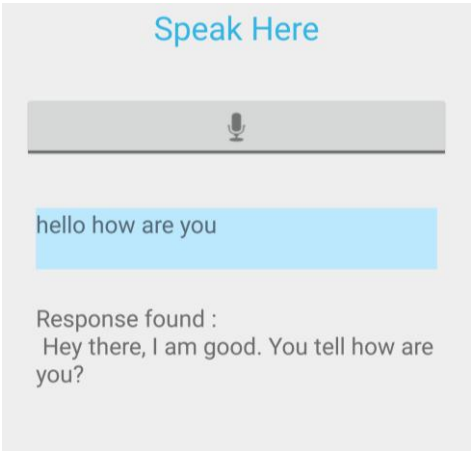
The incoming tokenized and POS tagged user query when received by the analyzer, it scans the words and tries to compute the overall sentiment value. This is done by iterating over the tokens and summing up token weights. If the sum comes out be 0, then it means neutral else if the sum is more than 1, then it means its positive else it is negative.

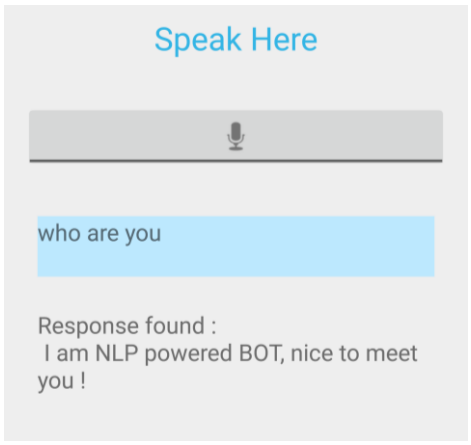
This model is decent enough to provide veritable and reliable answers if the word weights are thoughtfully and statistically distributed.

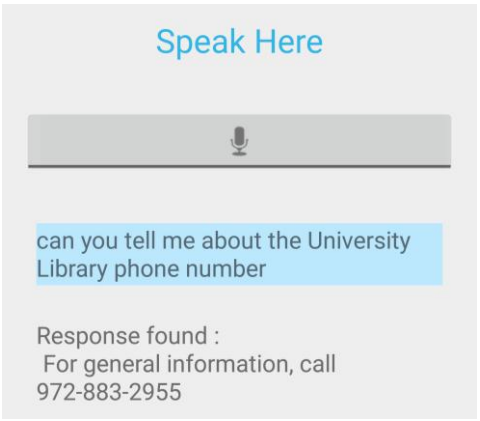
5 Experiment and Results

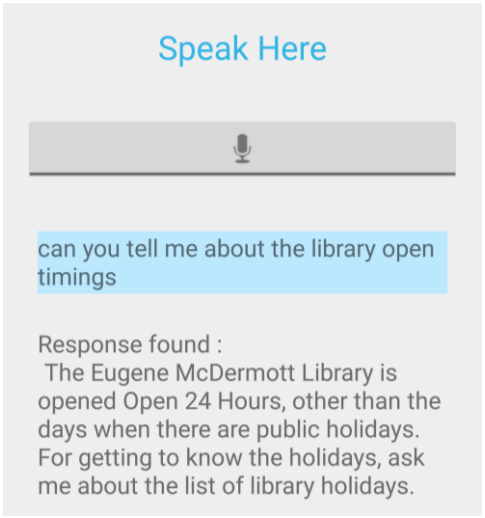
We show some sample conversations that our chat-bot are able to perform. These are actual screenshot of conversations that our chat-bot can perform when fed with simulated customer inputs. The conversations are related to knowledge of self-awareness, university campus and university library, which chat-bot tries to respond in the best well-reasoned manner.

Case 1: Neutral Sentiments

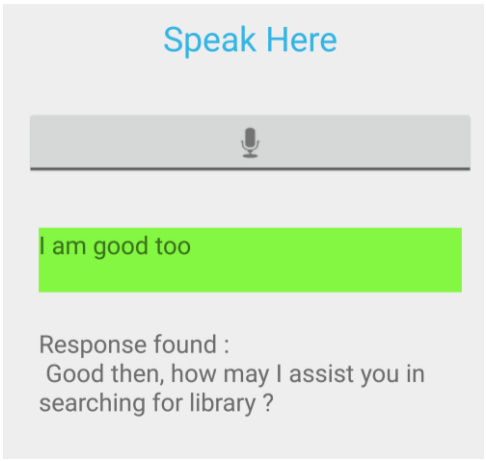


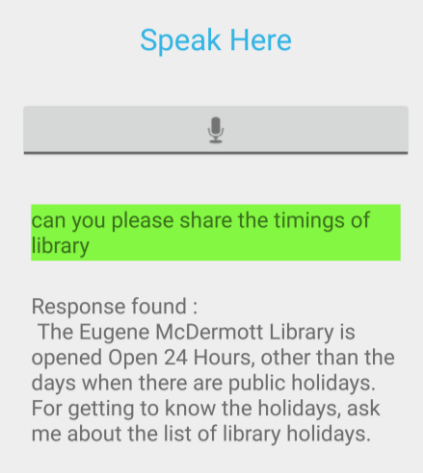


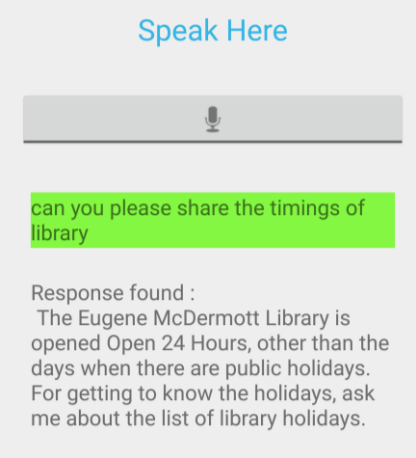




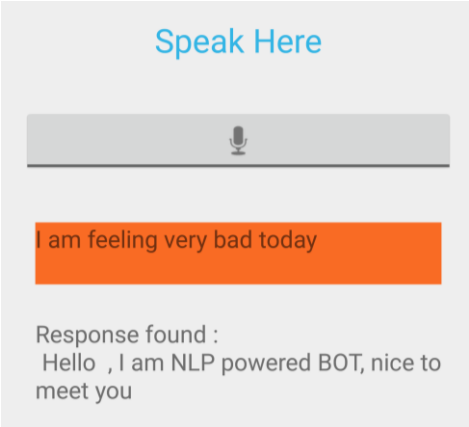
Case 2: Positive Sentiments

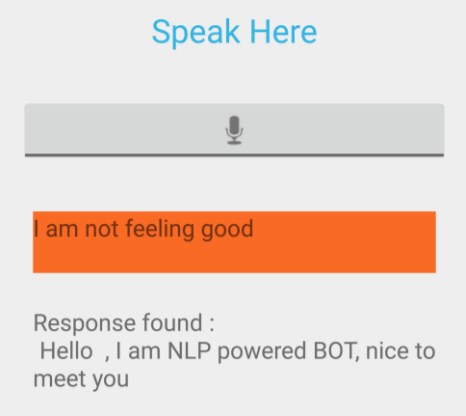






Case 3: Negative Sentiments





6 Conclusion

A chat-bot is one of the easiest ways to fetch information from a system without having to think for proper keywords to look up in a search engine or browse several web pages to collect information. Users can easily speak their question and retrieve the related information. In this project we looked into how a simple chat-bot can be created and tailored to be used as a domain specific chat-bot.

In addition, my work on conversational agents had many issues with android such as integration issues of apache openNLP, limited primary memory (RAM) allocation to application (16 MB) by android os causing issues like OutOfMemoryException, while loading training models like maxent for pos tagging etc. As we know that NLP activities are resource intensive hence, in order not to limit ourselves with the hardware, I would not to create a ***standalone*** android application, if I was given a chance to do this project again. I would rather consider a ***client server architecture***, wherein all the resource centric processing happens at server

side and the client (android app) remains as light weight as possible.

7 References

- Abu Shawar, B. and Atwell, E. (2003a). *"Using dialogue corpora to retrain a chatbot system."* In Archer, D., Rayson, P., Wilson, A., and McEnery, T., editors, Proceedings of the Corpus Linguistics 2003 conference (CL2003). Lancaster University, UK, pages 681–690.

- Abu Shawar, B. and Atwell, E. (2002). *"A comparison between alice and elizabeth chatbot systems"*. Research Report 2002.19, University of Leeds – School of Computing, Leeds.

- Franck Dernoncourt (2012). *"Designing an intelligent dialogue system for serious games"*. Rencontres Jeunes Chercheurs en EIAH, RJC-EIAH'2012

- ALICE. (2002). A.L.I.C.E AI Foundation, http://www.Alicebot.org/

- http://www.alicebot.org/anatomy.html

- Floridi, L., Taddeo, M., and Turilli, M. 2009, *"Turing's Imitation Game: Still a Challenge for Any Machine and Some Judges"*, Minds and Machines, 19(1), 145-150.

- Abu Shawar B. and Atwell E. 2005. *"Using corpora in machine-learning chatbot systems"*. International Journal of Corpus Linguistics

- Weizenbaum, Joseph (January 1966), "*ELIZA—A Computer Program For the Study of Natural Language Communication Between Man And Machine*", Communications of the ACM 9 (1): 36–45

- Fernando, A., Burguillo, C., Rodríguez A., Rodríguez, E. and Llamas, M. 2008. "*T-Bot and Q-Bot: A couple of AIML-based bots for tutoring courses and evaluating students."* Frontiers in Education Conference. FIE. 38th Annual.

- Bayan Abu Shawar, Eric Atwell(2007). *"Chatbots: Are they Really Useful?"*

- Apache opennlp- https://opennlp.apache.org

- AIML : http://www.alicebot.org/aiml.html