**Individual Report on Project (13):** CHATBOT, Restaurant Recommendation System

**Name:** Louis Arokiaraj Gilbert

**Email address:** [louisarg@usc.edu](mailto:your_email_address@email.provider)

**Rest of my group:** Arun Soundararaj, Bharath Sivaraman, Vrushali Peshave

**URL of GitHub repository:** <https://github.com/louisarokiaraj/BSLVChatbot>

**Project Overview**

This project aims to build a dialogue system a.k.a Chatbot that could recommend or suggest restaurants to users based on their preferences. The primary motive and enthusiasm that triggered the development of this project is the fast paced research and technological outbreak happening in the world of cognitive and digital dialogue systems. Some of the key names are IBM Watson, Apple’s Siri, Amazon’s Alexa, Google Home and many more.

The Chatbot interacts through the terminal and speech inputs. The Chatbot focuses on three key factors namely user’s preferred CUISINE TYPE, LOCATION and PRICE based on the user’s conversation. Having the given values, it finds the top 3 restaurants, ordered by the ratings, and recommends the same to the user with the help of YELP API. In addition, the Chatbot does review analysis of these 3 restaurants and presents the same to the user’s suggestion. The Chatbot’s architecture is based on the following four components:

* Speech Synthesis
* Natural Language Understanding
* Dialogue Management / Natural Language Generation
* Sentimental Analysis

The project was evaluated on various metrics as follows:

* Word Error Rate for Speech Synthesis
* Accuracy of Prediction for Natural Language Understanding
* Survey collected from various user’s interaction with ChatBot
* Performance Comparison of multiple Classification Algorithm.

The team observed consistent promising performance of the chatbot in all of the above-mentioned evaluation.

**My Primary Responsibility**

I took up the primary responsibility of the design and implementation of Dialogue Management and Natural Language Generation modules of the project. The goal of dialogue management (DM) is to make decisions on how the system should respond to the current user input and in addition to that, the decision must also be grounded on the semantic and pragmatic contexts of the conversation as defined by the recent dialogue history. In short they are the decision-making engines of the dialogue system (ChatBot).

I have followed the methodology of ***Rule Based Dialogue Management*** where in the user’s input is mapped to one of the predefined rules, which is already understood by the system.

I have defined two types of rule following the directions and examples of the research paper [1]. These are the rules, which are fed into the system with the help of key values pairs of patters and its corresponding actions to be taken.

1. **Request Rules** – In this rule, the System takes the initiative and the rules are intended to prompt the user for specific information.
2. **Collect Rules**. – These rules allow the system to either collects the requested information from he user or to handle cases in which the user takes the initiative to answer and respond to the same. The collect rules are the output of the action to be performed from the Dialogue manager to the Natural Language Generation Unit.

The following examples illustrate example of the collect rules and request rules in the system:

[PATTERN]: price\_given

[ACTION]: ask\_for\_location | ask\_for\_cuisine | ask\_for\_location\_and\_cuisine

[PATTERN]: price\_given && location\_given && cuisine\_given

[ACTION]: factors\_collected && hit\_yelp

[PATTERN]: not\_satisfied

[ACTION]: request\_which\_factor\_not\_satisfied

The dialogue manager, having understood the given user’s pattern and fetching the corresponding ACTION to be made, passes the action to the NLG unit of the system to retrieve the corresponding output / response back to the user. The dialogue manager uses these set of patterns and rules in regard to drive the user conversation and functionality of the chatbot. The system is well intelligent enough that if the user is not satisfied with the factors, before hitting the yelp API, he is asked for confirmation from the chatbot once. At this time the user can say “NO” to the system and change / update his preferences as well. The DM does not flush all the patterns or rules that it had collected before, instead it just resets that corresponding rule, and triggers its corresponding action to be made in order to receive the updated preference from the user. With this the chatbot is able to perform better to the user rather annoying him/her by making the conversation right from the beginning. The DM says the system to fire the Yelp API to fetch the live results based on the collected preferences only upon the confirmation of the user “YES” at any point of the conversation with the chatbot.

***The Natural Language Generation*** unit is implemented on the basis of ***Dictionary based retrieval***. In this there is a dictionary of sample response templates stored in the system. Based upon the action given by the DM, the NLG unit chooses one randomly from the pool of responses mapped to that corresponding action and output to the user.

Example:

STARTUP\_FILTER [‘ASK\_FOR\_LOCATION\_CUISINE] = ["Great! May I know what location would you like?","Great!, May I know what type of cuisine would you like ?","Location ?", "Cuisine?","Awesome!, can you tell me location and Cuisine ?", "Any preferences of location and cuisine ?","Location please", "May I know your choice of Cuisine please?", "Cuisine and Location please"]

STARTUP\_FILTER [‘REQUEST\_FACTOR\_UNSAT] = ["May I know what factor you are uncomfortable with?","Can you tell me which one you are uncomfortable with? (Price / Location/ Cuisine)", "Which preference you want to update?"]

***Process Explained:***

I have written the complete “**BSLVChatBot.py**” which acts the main executable file of the project integrating all the various modules into one complete system. The Dialogue Manager acts the central system to co-ordinate the roles and sequence of actions between the various modules of the Chatbot.

Once the user inputs his statement into the system, the Speech recognizer listens to his input, parses it and sends as a string to the DM. The dialogue manager passes on this information to the CRF Tagger, which tries to tag the given sentence based on the trained model as one among (CUISINE, PRICE and LOCATION). The CRF unit sends this tagged data to the DM. The DM manager being provided with this tagged data, find the corresponding pattern with the current user’s conversation. Let us assume that the user said

“Suggest some good Italian restaurants near Marina Del Rey”.

The output from the CRF unit would be

“Suggest – O, some – O, …Italian – B-CUISINE, … Marina – B-LOCATION, del – I-LOCATION, Rey – I-LOCATION”

The system understands that the current user has given two information (CUSINE and LOCATION). Thus now the dialogue manager, will try to find the corresponding pattern for the current status of the system. Having found that pattern that it has to ask for the PRICE, it triggers the corresponding action of “ASK\_FOR\_PRICE” to the NLG unit. The NLG unit receives this action parameter, and chooses one response at random to that corresponding action mapping. Thus the DM keeps the conversation happening until the user has provided all the required factors, so that it can tell that system to contact the Yelp API in order to retrieve the top recommended results based on the user’s provided preferences. At any point of time during the conversation, if the user is not comfortable with any specific preference / all of the preferences, the DM understands it, and correspondingly resets that particular Rule in the Rules mapping. It does not reset everything and thus making the conversation to be re-trigged from the beginning.

**Other Project Work**

My other project works involves the following

* Building the training dataset and manually annotating the same

I took up majority of the task of creating the sample user responses and annotating the same in regard to train the CRF Suite. I had drafted and collected around 500 sample user responses. I annotated them with the following standard

B-LOCATION – Beginning of a location

I-LOCATON – Inside the location

O – No tag

Link to the sample dataset: <https://docs.google.com/document/d/1tLt7JPdmU2g7TERcy8LbK6XS190Mdu7jMXOjbIaZHzk/edit>

* Collection survey from various users based on the interaction with the chatbot

I created a Google survey form and circulated among few students in campus to submit their feedback upon the interaction with the chatbot. The results and analysis of the survey is explained earlier in the report.

* Recording the DEMO video

Link: <https://www.youtube.com/watch?v=DWipYowZmaU&t=18s>

**Online Resources**

[1]. Natural Language Toolkit: <http://www.nltk.org/_modules/nltk/tag/crf.html>

[2]. An Interactive guide to writing bots in Python: <https://apps.worldwritable.com/tutorials/chatbot/>

**References**

[1]. Seokhwan Kim & Rafael E. Banchs, 2014, R-cube: A dialogue agent for restaurant recommendation and reservation. Asia-Pacific Signal and Information Processing Association, 2014 Annual Summit and Conference (APSIPA), IEEE.

[2]. Yushi Xu & Stephanie Seneff, 2010, Dialogue Management Based on Entities and Constraints, MIT Computer Science and Artificial Intelligence conference