Assignment 1 Report submitted in fulfilments of the requirements of the degree of

MSc in Computer Science

By

Deepak kumar Singh 20493345

Under the guidance of Joel Fischer and Jeremie Clos
Department of Computer Science
University of Nottingham
2022-2023

University of Nottingham
UK | CHINA | MALAYSIA

1. Introduction

I have made a chatbot which can recommend restaurant and movies to the user based on their preference and past watched movie respectively. These two different recommender systems are connected by the help of intent classifier. The classifier takes the query of the user and classify weather the query can be solved by movie recommender chatbot or restaurant recommender chatbot. The restaurant recommender chatbot tries to understand the type of restaurant and in which location the user wants recommendation from the chatbot. You can get the results sorted on rating of the restaurant and lower rated restaurants can be filtered out as well. The movie recommender chatbot tries to which is the exact movie the user has already seen and then use the genre of that movie to recommend new movies based on genre similarity. Filtering the lower rated movies out of all the recommendations is possible for this chatbot as well. Apart from these 2 there are 3 more intents. These are Welcome Intent, Appreciation intent and Goodbye Intent. Welcome intent starts the conversation, appreciation intent can be used when a good result is obtained from the chatbot and Goodbye intent end the conversation.

The chatbot could help the user to explore new restaurants according to their choices of cuisine and location, and end his search of good movies according to rating and genre very easily. It could be used in entrainment and hospitality industries. There are a lot of movies and restaurant, and the number is increasing every day. A lot of these movies and restaurants are good but cannot be identified by the user which brings loss to the movie makers and the restaurant owners. So, this chatbot can offer its help to both parties. On top of that I have tried to implement a light weight but powerful model which can be used easily.

2. Background

I took the idea from lectures of Human-AI Interaction to implement a similarity-based recommender chatbot. I have used TF-IDF to assign weights to words in documents and then with the use of these cosine similarity was calculated to find the closest documents. This approach is used to make the core of 2 recommender systems and then output of chatbot is produced. The same concept is also used to calculate the intent of the query. The users query was passed in a TF-IDF model and then cosine similarity was pattern documents of the tags and then the result of cosine similarity gave us the correct intent.

I have tried to manage the situations where the max cosine similarity is very less which is the case where ambiguity in the statement of the user arises. In these cases the chatbot will ask a clarification question. Which will help the chatbot to classify the intents correctly.

In restaurant chatbot I have implemented a disambiguation intent and a clarification intent. The clarification intent is also trying to carry the context in the conversation. When user wants to search a particular type of restaurant but forgot to say anything about the location where it needs to be found then disambiguation intent goes of and try to get the location. Notice in below conversation chatbot askes “Tell me the city you want the restaurant to be in”. It happened when user query did not have any the name of city.

```
Start your conversation with the chat bot
User: Hi
ChatBot: Hello
User: Find me a vegan restaurant with a rating 5
ChatBot: Tell me the city you want the restaurant to be in
User: London
ChatBot: Found 191 restaurants.
1   R & H cafe gallery
2   Liman Restaurant
3   Holy Smoke
4   The Clink Restaurant
5   Bar 61 Restaurant
ChatBot: Do you want to see all?
User: no
```

Clarification intent works when the user wants chatbot to find restaurant in a city but don't want any specification about what kind of restaurant. Like Indian, Chinese, Vegan options. It gives result of best restaurant in the same city and then ask the user if he wants to have restaurant names based on some specifications.

```
User: Hi
ChatBot: Hi there
User: Find me a restaurant in london
ChatBot: Found 13596 restaurants.
1   R & H cafe gallery
2   Osteria Romana
3   The Oystermen Seafood Bar & Kitchen
4   Liman Restaurant
5   Holy Smoke
ChatBot: Do you want to see all?
User: no
ChatBot: What kind of food would you like to have? Write No if you have
already found your restaurant.
User: Vegan, dutch
ChatBot: Found 1 restaurants.
1   My Old Dutch Pancake House
```

Notice in the above conversation bot askes "What kind of food would you like to have?" This is because the user query didn't have any specification.

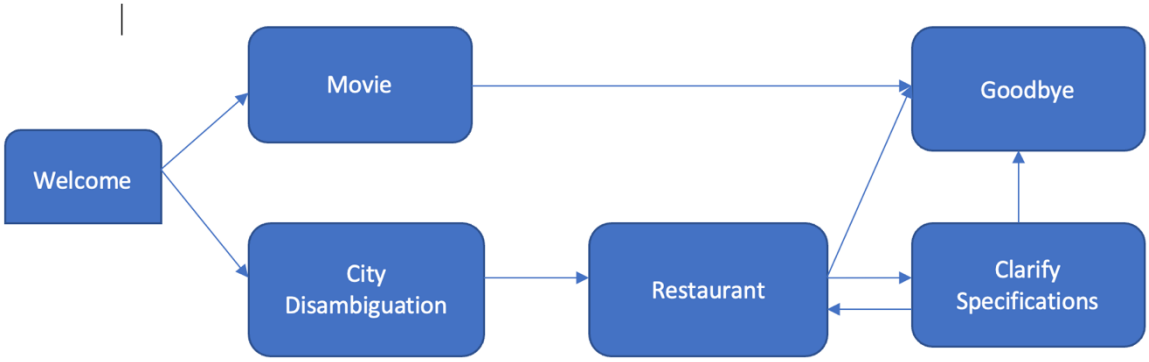
The use of disambiguation and clarification intents makes it different as it is trying to carry context of the conversation and dealing with ambiguous situations.

3. Proposed Systems

3.1. Intent hierarch

The below diagram explains how the different intents of the chatbots hierarch ally are placed. Welcome intent being at the start of conversations and then we see layers of intents. These

layering in intents are guiding the flow of conversations. Technically restaurant and movie intents should be at the same level but city disambiguation intents help to figure out in which city the restaurants need to be searched. In the same way after “Clarify specifications” tags if there is no specification present in the query the conversation could end move towards “Goodbye” tag or user can add specification tags and again search the restaurant



3.2. Restaurant Chatbot

This chatbot make use of the dataset which has name or restaurant, rating, tags on specification of restaurants, location of the city(city name).

```
In [16]: def find_similar_movies(user_query):
             k = TfidfVec.transform([user_query])
             vals = cosine_similarity(k, tfidf)
             index = 0
             val_with_index = []
             for val in vals[0]:
                  tuple1 = (val, index)
                  val_with_index.append(tuple1)
                  index = index + 1
             val_with_index.sort(key=sortFirst, reverse=True)
             return val_with_index
```

It takes the user query and try to find all the tags present in the query then try to calculate cosine similarity with the restaurants tags + city_name. It gives you the restaurants with similar specification and same location. Then we sort the similarity array and finds the restaurants which has all the tags identified in the query and is in the same city.

How do we check if all the tags in the query is present in the recommended restaurant? So, we have extracted all the tags from the dataset and now we find out how many of those are present in user query and then those are present in recommendations or not. In case its not present we remove those from the recommendation list. This is how we ensure the correction of output.

3.3. Movie Chatbot

This chatbot makes use of the dataset which has movie_id, movie_name, rating_of_multiple users and the expected recommended movie. Expected recommended movie will be used to calculate the accuracy of the model later. We need the user to say which movie is already seen and based on the genre of that movie we will try to find movie from out dataset. So, we first identify the movie in the user query extract its genre and try to calculate similarity with the genre of other movies. Then we gives the best 5 movies to the user.

How do we find out which movie is there in the user query?

```
In [8]: def search_movie_genre(user_query):
            k = TfidfVec_search_movie_genre.transform([user_query.lower().translate(str.maketrans(string.punctuation, ' '*len(st
            vals = cosine_similarity(k, tfidf_search_movie_genre)
            idx = vals.argsort()[0][-1]
            flat = vals.flatten()
            flat.sort()
            req_tfidf = flat[-1]
            if req_tfidf == 0:
                return "", 0
            else:
                return movie_genres[int(movie_name_token[idx].split(" ")[0])], int(movie_name_token[idx].split(" ")[0])
```

We find the similarity of the query with the names of the movie. One with the highest similarity is our movie user is talking about.

4. Evaluation

There are 3 different parts of this system which needs testing. The restaurant recommendation chatbot, Movie recommendation chatbot and intent classification of the chatbot. Movie recommendation dataset has column (Expected Recommendation) which gives which is the most recommended movie if we watch this but this way we cannot test the rating filters of the chatbot. We use the data of the column to evaluate the accuracy of the Movie chatbot. The other 2 parts of the system is tested manually.

Manual testing of the restaurant bot has test cases: -

1. Find restaurants with a tag does not present in the dataset
2. Find restaurants with specification and rating which gives zero recommendations
3. Find restaurants with recommendable restaurants more than 5
4. Find restaurants with recommendable restaurants less than 5
5. Find restaurants in a city for which data is not present

Testing Movie chatbot uses the expected recommendation column. I ran the bot for all the movies present in the dataset and compared the first recommendation of chatbot with the expected recommendation of the movie. The results shows an accuracy of 0.87.

Manual testing of the intents classification has test cases: -

1. User query should be classifiable to all the tags.
2. Only movie chat tag is selected for movie queries
3. Only restaurant tags are selected for restaurant query
4. Goodbye tag should end the conversation
5. Anything gibberish should ask the user to try again
6. Anything other than goodbye tag should not end the conversation

5.Discussion

5.1. Evaluation Inference

Manual testing of restaurant has given a lot of inferences. Wrong recommendation is given by it a few times. But what went wrong? Actually, the user query finds similarity on concatenation of tags, restaurant name and rating. Sometimes the tags are present in the names of the restaurant.
Eg1: - Name of the Restaurant: - "Vegan Express". Tags: - Vegan, Indian, Dutch. Now If I am searching for vegan options Vegan express has higher higher accuracy because it has 2 Vegan words in it.

Eg2: - Name of restaurant: - "5 Star Restaurant". Tags:- Fast Food, Indian. Rating :- 3
Now if I want to search 5 rated movies this will also give high similarity because it has 5 in the name though it has less rating which is wrong.

Testing for Movie gives us an idea that there are movies which has sequels like "Toy Story" and "Toy Story 2". In These cases the chatbot is confused and gives wrong results. The same rating problem discussed in Restaurant chatbot is arising here.

Intent classification is giving us fair results. Very few times it doesn't classify the query because you have written something very difficult to understand other its working fine.

5.2. Potential Impact

The chatbot could help the user to explore new restaurants according to their choices of cuisine and location, and end his search of good movies according to rating and genre very easily. It could be used in entrainment and hospitality industries. There are a lot of movies and restaurant, and the number is increasing every day. A lot of these movies and restaurants are good but cannot be identified by the user which brings loss to the movie makers and the restaurant owners. So, this chatbot can offer its help to both parties. On top of that I have tried to implement a light weight but powerful model which can be used easily.

5.3. Bias

The chatbot is calculating similarity so I think the bias can be embedded in the data only. If we consider dataset of restaurant it is biased to the liking of city because restaurant which are liked by the people of the city will only be present. So, someone coming from any other city could face disliking of the recommended restaurant.

6.Conclusion

The chatbot used TF-IDF and cosine similarity for the purpose of recommendation of movies and restaurant. These two chatbots were combined by intent classification. Some level of context matching was also tried to achieve.

A few problem with the working is presence of tags in name of the movie or restaurant which can be fixed by replacing the tags with maps of special character.

It solves problem of both the parties on each end of the system. One being the user and other is the bearer of the chatbot.