Zhongwen Lian

LING 575 - Spoken Dialogue System

March 19, 2021

Pizza-ordering system Write-up

In this project, my major contribution is to perfect the pizza-ordering system I designed on Hw2. To be specific, the task is divided into five parts. First, I incorporated the NLU system I implemented in homework4 into the pizza ordering system. Second, I included the sentiment analyzer component into the system so that the dialogue management system can respond to negative or positive feedback accordingly. Third, I implemented the recommendation system so that the dialogue management system can respond to user's questions related to recommendation. Fourth, I incorporated the ASR (audio speech recognition) into the system. Last but not least, I incorporated the T2S (text to speech) into the system. Now, I will illustrate each component individually.

1. **Rule-based and Statistical-based NLU component**

In homework 4, I was able to dive deep into both rule-based and statistical-based NLU systems. Their overall f1-score and accuracy are close to each other and for a specific task such as pizza-ordering, their performances vary on different intents. I found it useful to integrate both rule-based and statistical-based NLU to generate a potential better performance. For intents such as order_pizza, the rule-based NLU can easily capture this intent by having keyword detection on the pizza menu. On the other hand, for intents such as ask_for_recommendation, the statistical-based system is good at classifying whether an utterance is a question or not. The statistical-based system first predicts whether an utterance should be labeled as a question. If so, the NLU will then use rules to check if the question is asking specifically for recommendation. I

used a gradient boosting classifier from sklearn package for training the model. The corpus is loaded from nltk corpus “nps_chat”.

2. **Sentiment Analyzer**

I incorporated a sentiment analyzer to the NLU component and it can generate a sentiment score between -1 and 1 for each utterance. -1 represents the worst sentiment from the user and vice versa for 1. I adopted the SentimentIntensityAnalyzer available in nltk package. Mainly, SentimentIntensityAnalyzer utilizes a large, pre-labeled corpus for scoring sentiment. Lexicons with positive and negative sentiment scores are available and the SentimentIntensityAnalyzer relies heavily on these lexicons. I set a threshold for judging sentiment in a sentence. If the sentiment score is above 0.2, the system will show appreciation to the user. If the sentiment score is between -0.1 and 0.5, the system should apologize for inconvenience it brought to the user. Lastly, if the score is below -0.5, the system will prompt for potential cancellation of the current order. One of the weaknesses of the sentiment analyzer is the hardcoded standard for sentiment score judgmentation. More tests should be performed for better performances. Lastly, the system will focus on the sentiment score of the response from users after the system gives back the answer to a recommendation question. Mainly, this feature is used to determine how useful the system recommendation is to users.

3. **Recommendation system**

I implemented a recommendation system where the user can ask for recommendations for pizza, topping, crust and size in any time and in any sequences. As I specified in part 1, the system first checks if the utterance is a question. Then it will look for keywords such as “recommend” to validate the recommendation request. If the user does not specify what kinds of foods they are asking for, the system will recommend specialty pizza on default.

4. **ASR system**

I have looked through many different API in python that are designed specifically for ASR purposes. Eventually, I chose the speech_recognition package as my implementation since it incorporates google, ibm, bing and CMU sphinx engines. Among all available API, only CMU sphinx engines support offline ASR which is realized by HMM and n-gram statistical models. Other API need stable internet connection and are constantly training and perfecting their models based on their ever-lasting data stream. As a result, as trained on a fixed corpus, CMU sphinx engine does not outperform google api in most of the cases as I tested. The system will choose either google API or CMU sphinx API depending on the internet connection.

5. **Text-to-Speech system**

In terms of text-to-speech system, I chose the gTTS package which stands for Google Text-to-Speech. I saved the audio file generated after calling the API the system will remove the generated audio file no matter if the audio is successfully played or not.

Challenges encountered

Overall, the greatest challenge for my project is centered on the NLU component. For example, for a rule-based system, it is important to locate the correct keyword for classifying an intent. The system used to treat any utterance that has the word “think” as an intent for requesting “thin” crust. On the other hand, the system also suffers from misspelling. If the keyword such as “recommend” is misspelled, the feature of the system will fail completely. I solved these problems for my system by including potential confused words into rules. For instance, the system will double-check whether the user is asking for a thin crust or just merely saying “think”. These solutions are far from perfect but work well on this specific task.

Conclusion

In all, I have tried to add many features to the current pizza-ordering system and all of them are capable of doing a demo presentation at this point. For future works, I expect to integrate a misspelling detection component and an auto grammar correction component to the system which can serve a more formal resolution compared to the solution I had at this point. Also, it would be a lot better if I can train my statistical model (sentiment analysis, question detection) on corpus that is specifically designed for a pizza-ordering task. Overall, I am glad that I got the opportunity to use a lot of cutting-edge techniques in this project and it had been a pleasure to finish this project.

Bibliography

NLTK: https://www.nltk.org/index.html

gTTS: https://gtts.readthedocs.io/en/latest/#

Speech_recognition: https://github.com/Uberi/speech_recognition#readme

Sklearn: https://scikit-learn.org/