NLP Project

Ahmed ElZamarany 40-11881
Malak Rassem 40-3571

In the early days of the project, we firstly tried translating the data using the regular Google Translate API. The results were horrifying. Then, courtesy of Omar Nael, he sent his data that was translated by Google Developer API to everyone via Dr. Mervat.
The data still needed a lot of cleaning. Many lines were only half full and overlapping with other lines. This caused a weird shift in data. After fixing this by code, we started by first tokenizing the utterances. Then getting the feature count. This obviously led to weird features. We successfully were able to remove numerical values and gibberish data.

We then attempted performing tokenization on the translated utterances. Which was a much harder task. There was still English data in the set which we needed to get rid of as well as arabic punctuation that was not registered; however, tokenizing Arabic was considered a success. We then attempted to perform stemming on the tokenized Arabic utterances. We used multiple stemmers which were supposed to give good results for Arabic. But this was not the case. We performed classification of sentiments over 4 classes. The accuracy of many classifiers we used to perform sentiment analysis, did not cross the 28% mark. Leading us to believe we should take another path since the results are almost random.

The other path was simply to translate the input from the user into English, perform all tokenizations, lemmatization, stop word removal and classifications in English and then return the result translated into Arabic.

This approach was deemed a lot more successful. With an accuracy over 60% for sentiment analysis. Sentiment analysis was performed using a Logistic Regression classifier.

We ensured that the chatbot identified the sentiment first and stayed within it. Meaning that when the chatbot detects that the user is giving them sad input, the utterance the bot will reply with will always be sad.

The reply was handled by performing TF IDF over the entire data and then performing cosine similarity between the input and both utterances and prompts with weights for each of them. We picked the top 3 replies with the highest cosine similarities. Then in order to choose between them we then made sure that the reply that was chosen was still relevant by making sure it does not have extra irrelevant information to the input by checking if it has very common words that do not exist in the input.

After achieving good testing results with the chatbot we implemented the greeting and farewell replies. Which allows the user to quit the conversation. This, along with the response generation was in a loop while receiving input from the user.

We then applied our own Google Developer API translation of the input from Arabic to English. And then translated the acquired result back to English. There was an unusual shift in the data that disabled us from using the pretranslated data that Omar sent. However this means that the file would not translate remotely unless the evaluator places their own credentials.

Below is a flowchart of our pipeline.