Hamza Alfarrash | 261017161 | **Ahmed Mossa | 261001739 |**

Preliminary Results

MAIS 202 - Project Deliverable 2

Problem Statement:

Our initial idea was to implement an AI chatbot and use embedded speech to take speech input and output speech by the bot (converted to text temporarily). After getting feedback from the TPM, we realized this project would be too heavy for the given time frame. Consequently, we decided to proceed with a “text to text” chatbot instead and focus on having a pleasant front-end UI.

Data Preprocessing:

Since we slightly altered the project, we had to pick another dataset. The dataset is collected from 147M conversation-like exchanges extracted from Reddit comment chains over a period spanning from 2005 through 2017. We did not need to make any changes to the set as it was already extracted for us. There are also three versions of the dataset, small, medium, and large. We are currently utilizing the medium-sized dataset, but we could potentially try the large one depending on the performance.

Machine Learning Model:

In the first deliverable, we indicated that we will be using a Transformer as our model, but with a slight adjustment to our project, we will be using a pre-trained model called DialoGPT, an extension of the *Hugging Face PyTorch Transformer.* DialoGPT is formulated as an autoregressive (AR) language model and uses a multi-layer transformer as model architecture (Microsoft, 2019). We are also using the transformers and torch libraries. As for testing the model, we have offered a thorough discussion of the model testing methodology in the preliminary results section. Up to this point, we have not encountered any problems regarding testing the model. However, we are yet to decide on the validation methods, so we are discussing this with our TPM.

Preliminary Results:

Our initial implementation was to use the greedy search algorithm to generate responses based on the highest probability of choosing each time step. You see the model repeats a lot of responses, as these are the highest probability, and it is choosing it every time. By default, `model.generate()` uses greedy search algorithm when no other parameters are set. Thus it was not the most robust solution. We then looked into an algorithm called beam search, which allows us to reduce the risk of missing high-probability sequences by keeping the most likely `num_beams` of hypotheses at each time step and then taking the sequences that have the overall highest probability. This gave us better results but we realized that both *greedy search and beam search* were not ideal for an open-ended generation as in chatbots.

Therefore, we shifted our focus to “sampling” because we can incorporate randomness into our generation. We clearly saw some improvements but we still need to improve it by fine-tuning our hyperparameters; temperature and top-k sampling.

Next steps:

We are optimistic about the direction we’re taking our model in terms of performance. We’ll continue to fine-tune our model and utilize sampling until we reach a good performance that satisfies us. We also decided that we will use Heroku and possibly flash for the web deployment of our application.