SI 650 Information Retrieval

Homework 3

Name: Yashaswini Kaggle Name: Yashaswini Joshi Unique Name: yjoshi

Part 2: Implement an IR-based Chatbot

2a:
As I had to implement a scoring function that can be used with the interactive web framework, I used the BM25 Okapi from rank_bm25 package. The rank_bm25 package has different BM25 algorithms such as BM25 Okapi, BM25L, BM25+ etc. These algorithms were taken from this paper, which gives a nice overview of each method, also benchmarks them against each other. A nice inclusion is that they compare different kinds of preprocessing like stemming vs no-stemming, stop-word removal or not, etc. For the implementation of get_bot_response() method I have used the BM25 Okapi algorithm, as it gave me better results. BM25Plus : 0.00899 when I used test set for ranking with get_top_n(n=10). BM25 Okapi gave me 0.2666.

2b:
For this as well I have used BM25 Okapi algorithm as it gave better results. So, first I iterated those messages, and try to find the responses with it in the test set. If there are more than 10 responses in the test data with the current message, then use bm25 to rank and get the top-10 scoring responses among those "test-set responses". Here what I found was some of the message_id did not have 10 responses within test set. However, for some messages, there are less than 10 responses with them in the test set. Then what I did was to first rank the responses with that message (let's assume there are only 6 responses with it in the test set), so that I get rank1-6, for the remaining 4 slots, I then use this message as the query again, and then get the most relevant message in the entire database (which is the original tsv file) using bm25 again, and extract the top 4 responses of this "most relevant message" using bm25 again (the second time using bm25) as the eventual rank 7-10, so finally I can make sure I have top-10 responses for each message every time. ‘dtncx90’ and ‘dtncx79’ message_ids which did not have 10 responses in test set. I ranked the responses as mentioned. All this while I had not considered preprocessing of the data. Hence, I was not able to beat the random sample in the Kaggle system’s NDCG@10.
Realizing, the importance and need of preprocessing, I preprocessed the columns message and response. The preprocessing involved converting message and response text to lowercase and then removal of numbers and punctuations from the lowered text and finally removing the stop words. I also did lemmatization, but I lowered my score, hence removed the lemmatization of the text.