ECS763P - Natural Language Processing: Assignment - 2

TANISHQ VERMA
210642458

Queen Mary University of London — January 17, 2022

In this assignment, we need to create a vector representation of the document given to us which contains the lines spoken by the characters and be able to improve the representation such that it distinguishes from the other character present in the document. In this coursework, we are allowed to make training data comprising a maximum of 360 lines spoken by each character and 40 lines for each characters for test data. To improve the vector representation, I did pre-process, feature extraction and transformation techniques as instructed in the questions and then tries to reduce the mean score and accuracy to 1 from 5.12 and 0.3125 respectively which was provided in the notebook.

1 Q1 Improve pre-processing

As stated in this question, to get the mean rank close to 1, we need to do pre-processing techniques which we have to learn in the module and change the $pre_process$ function accordingly. We need to use the $computer_IR_evaluation_score$ function to see the improvement in the code which calculates the mean rank and the accuracy. I did some standard pre-processing techniques like lowercase, removing punctuation, removing end of line. I also decontracted the corpus changing "won't" to "will not". Further, I did contractions, tokenizing the result and removing the stopwords. Finally, I did lemmatization using pos-tag. I considerably reduce the mean rank and accuracy to 2.75 and 0.5625 respectively.

2 Q2. Improve linguistic feature extraction

In this question, we need to use feature extraction techniques to improve the $to_feature_vector_dictionary$ and improve the accuracy and mean rank. As given some suggestions in the question, I tried n-gram of different lengths by creating the get_ngrams function. I tried it for unigram, bigram, trigram etc and so got the best score in bigram. For some reason, the function that I defined which is get_ngrams gives a better result than the bigram function present in the nltk. Further, I did POS-tagging as word and tag and concatenated it with '@'. Finally, I did sentiment analysis which further helped in improving my mean rank.The result of $to_feature_vector_dictionary$ is a dictionary with keys as words and values as counts. The final mean rank is 2.125 and an accuracy of 0.6875 respectively.

3 Q3. Add dialogue context data and features

For this question, we need to make appropriate changes to the $create_character_document_from_dataframe$ and to other places in the notebook so that the data takes in the content of the lines spoken by the character and other characters in a chronology of all the characters present in the same scene however we cannot use the name and the gender directly. Since the surrounding and environment both contribute and depends on the semantic of the word, I added information for each line. I further added features and then I added context to the corpus. To make this work, I added the episode and the scene in what specific line it is used followed by the scene information to each line. This made the mean rank of 2 and accuracy of 0.625.

4 Q4 Improve the vectorization method

In this question, we need to implement the tf-idf as specified. We need to change the $create_document_matrix-from_corpus$ as specified. From the above question, we already got the data set and $vector_representation$ I implemented the tf-idf in it. Further, I also did select k-best using sklearn.feature_selection.SelectKBest where we can select best feature which further help in reducing the mean rank. After doing both the techniques, the mean rank and the accuracy are 1.5625 and 0.6875 respectively.

5 Q5. Select and test the best vector representation method

Finally, after doing all the optimisation and making tf-idf representation, I check mean rank and accuracy for both the training as well as testing dataset. We created test data using the given test csv file. For the test data I got mean rank of 1.4375 and accuracy of 0.625.