Assignment 2
Vector Space Semantics for Similarity between Eastenders Characters

Erblin Marku, ID:210762815, Msc Computer Science
NLP, ECS763P

Q1: Improve pre-processing (10 marks)

I have started pre-processing with simple *lowercasing* the words and then *expanding the constructions*. I build a dictionary with construction words and their expanded version and apply them to the token if they match the keys in the dictionary. I check the output of a single doc to see if the pre-processing works. This first 2 steps had a slight increase of mean rank but improves accuracy to 0.375 with 4 correct out of 16.
I commented out the lowercasing and we see that just the expanding constructions improves the mean rank to 4.125. Then, I remove punctuation which improved the mean rank to 3.9375, and then I removed the stop words. Removing stop words gets us the lowest mean rank till now with 3.0 and a precision of 10 out of 16.
I tried 2 more pre-processing techniques with *lemmatization* and *replacing words that express numbers with actual digits*. Both these resulted in a slight increase of the mean rank from 3.0 to 3.0625.
To conclude we get the lowest mean rank with expanding constructions, removing punctuation and stop words.

Q2: Improve linguistic feature extraction (15 marks)

For the linguistics features I will start with the *nltk pos-tagger*. Which returns in a tuple the token and the tag for them. I will convert it to string and add as a feature, so the function incorporates it into the dictionary. After running it, we get a mean rank of 2.8125 from 3.0 so it is slightly doing better, so we are keeping it in the code.
Next, I am adding *bigrams* and *trigrams* from *nltk.util library*. Bigrams seem to perform well decreasing our mean rank to 2.5625 and precision to 10 correct out of 16. Trigrams result in a slight increase to our mean rank from 2.56 to 2.62 so I will comment it out in the code to get other features to improve my mean rank.
I will add sentiment as a feature. I am using *Vader Sentiment* library, which returns a dictionary of scores for *Negative, Positive, Neutral and Compound* for a sentence. For this I need each token to be a single sentence so I will divide the token list in chunks of 1 and perform the sentiment analysis for each word. First, I tried adding the compound score to each word as a feature (since it is the score for the overall sentiment in a sentence) but if resulted in a large increase in mean rank so I did not consider it.
Instead, I am using the definition of “Neutral”,” Positive” and “Negative” which are given and concatenated to the token if their compound score is in the defined ranges. The ranges are defined in the Vader Sentiment documentations.
After writing the function I use it on the token list and update the output dictionary of the features function.
The sentiment analysis seems to be not a great change to our training and validation sets but the increase the mean rank in the test set is too much, so it is not worth using it for this case.
The mean rank in the test set from the best features till now is 4.3125 with an accuracy of 9 out of 16 correct.

Q3. Add dialogue context data and features (15 marks)

For this question I will try 2 approaches. The first one is just adding the whole sentences from the same scenes, previous and next. This makes the character document very large and to d it I will create 4 new columns in the panda data frame. 2 columns with true or false values comparing the scene names with a shift 1 and -1. This enables us to get previous lines if the scenes are the same and the same for the next lines.
At the end we will drop these columns so the df does nit change. The other 2 columns will be the lines shifted 1 up and 1 down, while deleting the element that is removed. This helps us get the previous and next lines in

the same row so we can add them to the characters document. I will add also "_PRE_" and "_NEX_" at the end of previous and next sentences to make it clear where the characters sentence ends and context starts. This results in a mean rank of 2.0625 which is good but including all the words from the next and previous sentence does overload the document and it does not seem like much context, because some words will increase their presence while not having much actual weight in the document. For this I will use *Rake-nltk* which is domain independent and determines key phrases in a doc or sentence. So I will get only the key phrases from the next and previous sentences and add them to the character document maintaining the same logic of the previous try. At first this results in a mean rank of 1.625 but guessing that the sentiment analysis would make sense when the context is added I am adding it to the features again and this resulted in a mean rank of 1.5625 and an accuracy of 12 correct out of 16 in the train and held-out sets, while the test corpus gives a mean rank of 1.75 with the same accuracy. This is the best I could get with the steps till now.

Q4. Improve the vectorization method (10 marks)

In this part I will initialize the *Tf-Idf* vectorizer method from *sklearn*. This will weigh the words in term of frequency in a document and overall document frequency. While in the first try with the defaults, meaning the *idf , smoothing*(adding 1 to avoid 0 division), and no variant of *tf* being used. In this case we get a mean rank of 1.312 which is good. I am looking at the sentences and some words happen very often in one scene because they are related to the scene, but not much in other documents. Still by using straight *tf-idf* multiplication the word gets a higher weight that it should have. I will try to normalize the weights and turn to variants of term frequency and will use the *sublinear tf*, so I will set it to true.
This try results in a mean rank of 1.25.
Looking at the way character documents in detail I can see that adding 1(meaning setting smoothing to true) can make some words that increase their *tf* and I don't see 0s the way the documents are created. So, turning smoothing to False resulted in the same mean rank of 1.25. It did not change much in mean rank.
The last try will be with no smoothing and straight term frequency, which increases the mean rank to 1.3125 but the mean cosine similarity is lower so we will keep the smoothing to false for now.

Q5. Select and test the best vector representation method (10 marks)

I have been selecting the best techniques from each step so at this ponit I don't have much to do but try all the best in the test corpus. In *Pre-processing* I will leave in the function the *expanding of constructions and the removing of punctuation and stop words*. These gave me the best mean rank we will comment out the others. In the *features* we will keep the *count, pos tags, bigrams, and Vader sentiment analysis* since the last one was found to work better when we add context from previous and next lines from the same scenes. In the *vectorizer method* we will keep the *smoothing to False* and will use the *sublinear tf* to normalize the weighing for word occurring too much in a document but fewer times in others.
After keeping all these setting and running the code in the last cell by using all the 400 lines and 40 in the test file we get the mean rank 1 and accuracy 16 correct out of 16 which means the Information Retrieval classification method can select documents from test data as belonging to the correct character document.

Each steps mean rank, mean cosine similarity and accuracy are saved in different data frames at the end of the notebooks and clear table with these statistics are saved in an Excel document for each question.