NLP Assignment 2 Distributional Semantics **Yash Desai**

Q1) For the first question we aim to improve the mean rank of our feature matrix by improving the pre-processing techniques in the previous approach that was demonstrated on the notebook. Previously the pre-process function that we used performed a simple tokenization operation. This generated a mean rank of 4.5625 which needs to be improved by applying different pre-processing, feature extraction and transforming techniques throughout this assignment. There is a significant amount of change in the mean rank after using the new pre-processing techniques.

The best combination of these techniques (in order) which helped us achieve such a good mean rank involved: -

1)Tokenization 2) Removing all the punctuations using the .isaplha() function which returns an alphanumeric value for any input 3) Removing the stop words using the spacy library

4) Lemmatization (takes the root or the lemma of the word)

In **tokenization** we split every word from a sentence and make a list of those words. Now to remove the punctuations from these words first I tried to remove each **punctuation** manually by stating all the punctuations and using string methods to remove them. I observed a better result when I removed the punctuations by just using the .isalpha() method which removes everything except letters and numbers (also the _EOL_ part). Further I used the spacy library in python to remove the **stop words** from our data. Then finally I performed **lemmatization** on the remaining data using the NLTK library in python which basically returns the root of any word. The mentioned approach helped us to achieve a mean rank of **1.25** which is a significant improvement from **4.5625** that was achieved using just tokenization.

Q2) Our goal for question 2 is to **improve the feature extraction** by adding meaningful features to our feature vector and observe a change in the mean rank. To achieve the same, we have added five novel features to our feature dictionary i.e., the to_feature_vector_dictionary_q2() function.

1) ***POS*** *(Part-of-Speech) Tags ("word"+ "pos")*

Parts of speech tagging simply refers to assigning parts of speech to individual words in a sentence. Here I have performed the POS tagging on the sentence level which means that I have provided each sentence as context to extract the part-of-speech tags. The library used to perform the same is "spacy" which we earlier used to remove the stop words in question 1.

2) Using ***bigrams****,* ***tri-grams*** *and* ***four-grams:***

(3 different features)

The n grams are simply a sequence of words. 2 for bigrams, 3 for tri-grams and 4 for four-grams. Adding these features with their counts will provide a great word level context and may improve our mean rank. The n grams are calculated using the 'nltk' library in python.

This new feature dictionary will contain the bi-grams, tri-grams, four-grams and POS tags as the keys and their respective counts as the values. *For example:- a tri-gram feature looks like this* ***{'tone + got + pathetic':1}***

3) ***Sentiment Analysis***

Here we have performed sentiment analysis for all the sentences in our corpus and added them as features. We will get the number of positive, negative, and neutral sentences for each character in our dataset. Moreover, in the sentiment analysis' features the keys will be the type of sentiment and values will be the number of sentences. For example:-

{'positive_sentences': 15, 'negative_sentences': 7, 'neutral_sentences': 18}

The mean rank after adding these five new features and computing its counts has increased by a considerable margin i.e., **from 1.25 it became 2.9375** . The features, POS tags, bigrams, tri-grams, four grams and sentiment analysis were added to improve the context and in turn improve the mean rank but it was observed that it degraded the performance. Therefore, in the next question we have adopted a new approach where we work with context on the sentence level instead of word level.

Q3) In the 3rd question we need to **add context of the line spoken by the characters**. This task is performed at the sentence level. To achieve this, I have added the previous and the next lines along with the current lines into the data frame as context in the form of a dictionary. This dictionary may have four unique key values i.e. "**PRE**", "**CURRENT**", "**POST**" and "**None**. The previous lines will have the have the key "PRE", the next line will have the key "POST" and the current line will have the key "CURRENT". If the previous or the post lines are from the same speaker then we use the key for that line as "None". So before extracting features we add this context to our data frame

first and then later use this context as a feature after pre-processing. To process this context data, we have created another pre-process function which will tokenize, lemmatize, and remove the stop words from this context data according to the PRE", "CURRENT" and "POST" labels. Along with these labels for words we have added the **POS** tags of corresponding words to improve the word level context. A pre-processed word will look like this: - "POST_nowadays|ADV". A post word 'nowadays' with its POS tag 'ADV' i.e., adverb. A new list with the training corpus containing words with these labels is then transformed using the Dict Vectorizer. The above approach yields a decently **improved mean rank of 1.1875**. This implies that considering the sentence level context is a really good way of improving the mean rank and in turn improving the accuracy. Hence, we will continue this approach in our next question where we change the method of transformation of features.

Q4) In this question we have used the same techniques used in question 3 (sentence level context) and further aimed to improve the mean rank by using a better transformation method for the training feature matrix which is by using the **TFIDF** (term frequency–inverse document frequency) transformer. TFIDF basically evaluates how relevant a word is to a document in a collection of documents. Some words may occur frequently across the entire training corpus, but they may not be relevant for a specific document. The TFIDF transformation was performed using a function TfidfTransformer() from the 'sklearn' library in python on the training data. By using this approach, we got a **mean rank of 1.0625** which is probably the best we can achieve. The methodology that we implemented throughout the questions has been able to achieve a near perfect mean rank which means we have created one of the best possible feature sets for the provided dataset to train.

Q5) The main aim of the fifth question is to choose the **best method out of the questions 1-3** which provides the best mean rank. The same must be performed on the entire training and the testing data hence first we need to pre-process both the datasets to be compatible with our established approach in the previous questions. Hence first I have added the "training" labels (as performed in the notebook) to the train data and "holdout" labels to the entire test data. After trying and testing the methodologies (function from our code) from the previous questions the approach which provided the best accuracy was using the functions (context of lines from question 3) to perform sentence level context extraction and adding POS tags to the tokenized words using the pre-process function used in question 3. I was able to achieve a mean rank of **1.0** and a perfect accuracy (1.0) with the entire training and testing data which is the best that we can achieve. To better visualise our approach, I compared the mean rank and accuracy obtained after every question into a well-arranged data frame. (Q5 has the best)

	Operation	Mean Rank	Accuracy
Default	No Operation	4.5625	0.1875
Q1	Improved Pre-Processing	1.2500	0.9375
Q2	Improved feature extraction(Adding POS, n-grams and sentiment analysis)	2.9375	0.4375
Q3	Adding context to our data (considering previous and next lines)	1.1875	0.8750
Q4	Changing the transformation (transforming the matrix using the TFIDF Transformer)	1.0625	0.9375
Q5	Combination of best methods from previous questions	1.0000	1.0000

Conclusion:-

A model is said to perform better when it has meaningful features and excellent data pre-processing. By adding the context of lines spoken by the characters in terms of lines spoken by other characters and further pre-processing the data and adding POS tags to tokenized words, I have achieved the best mean rank possible. From starting to work on the word level pre-processing (mean rank **1.25**), to improving the quality of features (POS + n-grams + sentimental analysis) (mean rank **2.9375**), to working on sentence level context (mean rank **1.1875**), to improving the transformation (TFIDF) **mean rank 1.0625**) we finally have achieved **a perfect mean rank of 1.0** from a poor mean rank of **4.5625**. Moreover, I have also improved the accuracy from 18.75% to a 100%. Hence, I can conclude that this is one of the best methodologies of pre-processing, feature extraction and feature transformation that can be formulated for the provided "Eastenders script data". Finally, I conclude that such methods when used in real world NLP tasks, should produce accurate and reliable models.