**NLP Assignment 2: Distributional Semantics**

For each question, mean rank was calculated both on validation set and test set. The techniques/features/context that gave best results on the test set were finalised for that question and were carried over to the next question.

**Question 1 : Improve pre-processing**

Using the pre-processing techniques you have learned in the module, improve the pre_process function above, which currently just tokenizes text based on white space.

When developing, use the 90% train and 10% validation data split from the training file, using the first 360 lines from the training split and first 40 lines from the validation split, as per above. To check the improvements by using the different techniques, use the compute_IR_evaluation_scores function as above. The **mean rank** is the main metric you need to focus on improving throughout this assignment, where the target/best possible performance is **1** (i.e. all test/validation data character documents are closest to their corresponding training data character documents) and the worst is **16**. Initially the code in this template achieves a mean rank of **5.12** and accuracy of **0.3125** on the test set- you should be looking to improve those, particularly getting the mean rank as close to 1 as possible.

Answer 1

Pre-processing techniques such as lowercase, removal of numbers, stop words and punctuation, tokenization, stemming, lemmatization and POS tags were experimented with. The biggest boost on mean score for validation set came from the combination of lower casing, removing punctuation, and word tokenization – the score dropped to 1.25. However, on the test set this combination did not prove to be effective; meaning, the training set is more similar to the validation set compared to the similarity between the entire training set and test set. Adding the removal of stop word technique to this combination proved to be effective with the test set – the score dropped to 2.06. This would mean that there were more stop words in the training data character documents that set it farther from the test data character documents. Although, lemmatization helped reduce the mean value on the validation set, it made no difference on the test set. Stemming, on the other hand, increased the mean value.

The training set (360 lines) and the validation set are closest with a combination of – lowercase, removal of punctuation and word tokenization. Considering the best results achieved on the test dataset, techniques - lowercase, removal of punctuation and stop words, word tokenization and lemmatization – are finalized in this step.

**Question 2: Improve linguistic feature extraction**

Use the feature extraction techniques you have learned to improve the to_feature_vector_dictionary function above. Examples of extra features could include extracting n-grams of different lengths and including POS-tags. You could also use sentiment analysis and gender classification (using the same data) as additional features.

You could use some feature selection/reduction with techniques like minimum document frequency and/or feature selection like k-best selection using different criteria https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html. Again, develop on 90% training and 10% validation split and note the effect/improvement in mean rank with the techniques you use.

Answer 2

POS tags of the tokenised words, ngrams of length 2 and 3 were experimented with. POS tags were concatenated with their words to form unique strings to be fed into the feature list. Ngrams were

created by fetching n sequential words and concatenating them with a space in between. Each feature was clearly differentiable with their string designs. The counts dictionary, which initially had only the words and its frequency, now has each of the above feature along with the frequency of their occurrence in each document.
While ngrams with length 3 had the detrimental effects on the mean value, bigram proved to be effective. As the string gets longer, the frequency drops due to which ngrams of higher lengths seem to perform poorly. With the combination of pre-processing techniques finalised in the previous step, the mean rank was at 2.81. By adding only POS tags into the feature dictionary, the mean rank reduced to 2.4. However, the mean rank on the test_data was 2 after the previous step, now increased to 2.25. A few combinations of features were experimented with - combination of POS tags, bigram and trigram reduced the mean rank on validation set to 2.31 while it increased on test data from 2 (in the previous step) to 2.19. By using only bigram as an extra feature, the mean rank was maintained at 2 on the test data.
Observing the result pattern on validation and test data, it was clear that validation set had more similarities with the training dataset, probably because they came from the same document.

**Question 3: Add dialogue context data and features**
Adjust create_character_document_from_dataframe and the other functions appropriately so the data incorporates the context of the line spoken by the characters in terms of the lines spoken by other characters in the same scene (immediately before and after). You can also use **scene information** from the other columns **(but NOT the gender and character names directly)**.

Answer 3
To add the context, episode_scene column from the datasets was used to ensure additional context added was relevant to the document. To avoid repetition, only those lines that were not spoken by the same person in sequence were considered. To avoid repetition between next line of current iterations and previous line of the next iteration, sequential conversations between only 2 people were ignored.
Tags _PREV_ and _NEXT_ were used to differentiate the sentences. With the entire context – previous line, next line, and line tokens (_PREV_, _NEXT_, _EOL_), the mean score on validation set reduced from 2.69 to 2.25 and increased on the test data from 2 to 2.25. Cleary, just by increasing the size of the vocabulary, the mean rank could not be reduced. By providing only the previous line and prev token, the mean rank reduced to 1.81 on validation set, but increased on test data to 2.43. Next line provided good context for the test data. By removing the token _PREV_, the mean rank reduced on both validation set (1.69) as well as test set (1.8). Upon doing this, each document had lines that belonged to different people, thereby increasing their similarity. Few more experiments were conducted with different features (extra features) and subsets of contexts. Previous line turned out to be most significant in terms of reducing the mean rank. Hence only the previous line, current line and end of line token were finalised in this step.

**Question 4: Improve the vectorization method**
Use a matrix transformation technique like TF-IDF (https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html) to improve the create_document_matrix_from_corpus function, which currently only uses a dictionary vectorizor (DictVectorizer) which straight-forwardly maps from the feature dictionaries produced for each character document to a sparse matrix.
As the create_document_matrix_from_corpus is designed to be used both in training/fitting (with fitting set to True) and in transformation alone on test/validation data (with fitting set to False), make sure you initialize any transformers you want to try in the same place as corpusVectorizer = DictVectorizer() before you call create_document_matrix_from_corpus. Again, develop on 90%

training 10% validation split and note the effect/improvement in mean rank with each technique you try.

Answer 4
The model worked upon used frequency of the vocabulary. By upgrading it to provide weightage for the word (i.e. comparing number of times a word occurred in a doc with the number of documents the word occurs in), the performance improves significantly.
With features finalised in the previous step, just by changing the vectorization process, mean rank dropped to 1.5 and 1.25 on validation and test sets respectively. By further tweaking the parameters of this vectorizer, further experiments can be conducted.

**Question 5: Select and test the best vector representation method**
Finish the optimization of your vector representations by selecting the best combination of the techniques you tried in Q1-3 and test using the code below to train on all of the training data (using the first 400 lines per character maximum) and do the final testing on the test file (using the first 40 lines per character maximum).
Make any neccessary adjustments such that it runs in the same way as the training/testing regime you developed above- e.g. making sure any transformer objects are initialized before create_document_matrix_from_corpus is called. Make sure your best system is left in the notebook and it is clear what the mean rank, accuracy of document selection are on the test data.

Answer 5
Given the entire exercise was performed on both validation set and test set hand-in-hand, at this stage, experiments were restricted. A few experiments were conducted to check if providing the full context – previous line, prev token, current line, next line, next token, and end of line token would improve the mean value. But, again, only the previous line, current line and end of line token proved to be most significant.
Few additions were made to pre-processing techniques – removing numbers and fixing contractions – they seemed to increase the mean rank on both validation and test sets.
Adding POS tag as an extra feature helped reduce the mean to 1.5 on validation set and 1.25 on test data set.
Few more experimentations performed are available in the Appendix section of the notebook.
Further enhancements can be made by trying to include sentiment analysis as a feature.