Q1

For question 1, I had to improve the pre-processing function from just a simple tokenization, in order to improve the mean rank and accuracy. So I downloaded **punkt**, and imported **word_tokenize** and **PorterStemmer** from **nltk**. The Porter Stemmer is a process of removing the commoner morphological and inflexional endings from words (A stemming method). For example, “liked”, “likely” and “liking” would just be “like”. So I tokenized the character text with the **word_tokenize** and created a list of sentences that appends all the stemmed words from the character text. While I was going through this process I noticed that when I printed out the sentences there were some symbols that could affect the mean rank and accuracy negatively. Some symbols were: “--”, “…”, “_eol_”. After observing these symbols I tried removing everything so I created a list with the symbols and words that I wanted to remove. Apparently if I remove every symbol the mean rank increases and accuracy decreases. Although, after removing only “_eol_” I got better results. After computing the scores I observed that the mean rank decreased to 2.31, and accuracy increased to 0.38.

Q2

For question 2, I had to improve the linguistic feature extraction. I approached this task by adding POS-tags by importing **spacy** and by importing **ngrams** from **nltk**. Then I tagged all the content from **character_doc** and created a list called dictionary that appends a for loop which adds the text, special character and tag together. After that I created the final dictionary and made a function that adds bigrams to the dictionary. Then I used the same procedure for the rest of the cells to compute the scores and the mean rank decreased to 1.88, and accuracy increased to 0.50

Q3

Question 3 requires adjustments to some functions so that 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.

Q4

The task for question 4 was to improve the vectorization method. So I imported the **TfidfTransformer** from **sklearn.feature_extraction.text** and transformed the matrix outputted by the **DictVectorizer.** I used the default parameters, because whenever I tried to change the parameters the results were getting worse. Then I followed the process to compute the scores. Mean rank decreased to 1.31 and accuracy increased to 0.81.

Q5

For question 5 I had to optimize my vector representations by selecting the best combination of the techniques and functions that I have used throughout question 1, up to question 4. I started off with importing all the required libraries:

```
import spacy
from nltk import ngrams
from sklearn.feature_extraction.text import TfidfTransformer
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
nltk.download('punkt')
```

Then run the pre-process function that was used in question 1, to pre-process all the training data by stemming (using the **PorterStemmer**), and tokenizing (using the **word_tokenize**). After the pre-processing, I trained all the training data and proceeded by adding the **"to_feature_vector_dictionary"** function that I used for question 2. This function as mentioned earlier tags the content by using **"en_core_web_sm"** from **spacy** and adds bigrams to the final dictionary by using the **ngrams** from **nltk**. The next step I used is the **TfidfTransformer** to transform the training feature matrix.

Finally, I tested the model by importing the test data. The final results for the test data is:

Mean rank: 2.06
Mean cosine similarity: 0.71
Accuracy: 0.63