NLP Assignment 2

Salva karimisahari 240753803

Question 1: (implemented in NLP_Assignment_2_distributional_semantics.ipynb)

The methods that were tested one by one include:

- Lowercasing
- Removing specific punctuation (.,!?-) to simplify the text while keeping the semantic structure as the punctuations will probably not be aiding in the identification of a character.
- Removing tokens with digits as some of the tokens were observed to have them.
-Expanding contractions because the text includes terms like “don’t” and “you’re” that reduce semantic clarity
- Using NLTK’s tokenizer achieves better results than splitting by whitespaces with the structure of resulting tokens.
-Removing redundant and meaningless filler words like “um”, “haha” reduces noise.
- Conversational text has exaggerated words like ‘nooo’. To reduce noises while preserving the meaning, the limitation for repeated characters is set to 2.
- Removing stop words by itself increased the mean rank but after ding the other preprocessing step and stipping the words to their stems, it reduced the mean rank to 2.5
- Adding more complicated preprocessing steps like tagging named entities or lemmatizing increased mean rank and decreased accuracy which is predictable given the nature of our task which is recognizing characters and excessive preprocessing might remove important linguistic or stylistic cues that distinguish characters.

	Tokenize (initial)	Lowercasing	Remove punctuation	Removing numbers	Expand contractions and tokenizer	Removing filler words	Removing exaggerated words	A combination of all methods
Mean Rank (val)	3.6	3.6	3.18	3.6	3.125	3.62	3.62	1.68
Accuracy(val)	0.31	0.31	0.43	0.31	0.372	0.31	0.31	0.75

After running multiple tests on multiple combinations, the best result (mean_rank = 1.68, accuracy = 0.75) was achieved with converting to lower text, removing punctuations, expanding contractions, removing repeated characters, tokenizing, removing words with digits, removing filler words and removing stop words in the mentioned order.

Question 2: (implemented in NLP_Assignment_2_distributional_semantics)

Given the enhancements in the preprocessing step, after applying a combination of all preprocessing methods, our validation mean rank = 1.68 and accuracy = 0.75 with using count feature vectors. To enhance to_feature_vectore_dictionary(), different methods have been added to the count feature. One of the methods is to add ngrams to capture contextual features. To see the effect bigrams and trigrams were added to the features. The frequency of part-of-speech tags such as POS_DET, POS_NOUN,... was added to identify linguistic patterns but it appears to be not suitable for recognizing the speaker (characters might have similar distributions of nouns, verbs, and adjectives). Sentiment analysis (positive, negative, neutral probabilities) can capture emotional tones from a character. Adding emotion probabilities (joy, sadness, anger, and …) can distinguish characters (using Hugging Face’s transformers library). Adding linguistic complexity metrics such as average sentence length and type-token-ratio can help identify characters as certain characters might use longer sentences or different textual patterns.

	Count feature vector	bigrams	Bigrams and trigrams	Pos tagging	Sentiment analysis	Emotion analysis	Linguistic metrics	Combination of bigrams, sentiment, emotion, linguistic
Mean Rank (val)	1.68	1.62	1.68	4.56	1.68	1.68	1.68	1.62
Accuracy (val)	0.75	0.6875	0.6875	0.18	0.75	0.75	0.75	0.68

For crying the document matrix, one way is to use the feature vectorizer, an addition can be using tf-idf scores to highlight unique words in document and lessen the effect of frequent words (TF-IDF also removes documents with 0 weight so it also handles filtering based on document frequency). Another addition can be selecting more important features to reduce the size of the final matrix (the initial matrix is of shape (16, 21040))

	Feature vectorizer	TF-IDF	Selecion (k=500)	Selecion (k=1000)	Selecion (k=2000)	Selecion (k=3000)	Selecion (k=5000)	Selecion (k=10000)	Selecion (k=20000)	Feature vectorizer, TF-IDF and selection (k=2000)	Feature vectorizer, TF-IDF and selection (k=10^5)
MeanRank (val)	1.62	1.87	1.75	1.68	1.62	1.62	1.62	1.56	1.62	1.68	1.62
Accuracy (val)	0.68	0.75	0.56	0.63	0.68	0.68	0.68	0.62	0.62	0.875	0.875

Using TF-IDF increased the mean rank probably because common/less informative words have an effect on identifying characters but using feature selection impacted the mean rank by reducing it to 1.56 as its best result but affected the accuracy negatively. A combination of feature vectorizer, tf-idf and selection (k=10000) seems like the optimal solution to reduce mean rank while increasing accuracy.

Question 3: (Implemented in NLP_Assignment_2_Q3.ipynb)

In order to add dialogue context, the data frame was grouped by episodes and scenes and to distinguish lines spoken by a character and by the other people without using the character name directly, lines are tagged by _TARGET_SPEAKER_ and _OTHER_SPEAKER_ in the end and beginning of lines. For each character, a limit of 300 sentences is kept so the train_character_docs will include 300 lines spoken by the character or by the oher characters shared in the same scene. (example (SHIRLEY): [`_TARGET_SPEAKER_ Look at ya, not a mark on ya. And you think you're an unlucky man. _END_TARGET_ _OTHER_SPEAKER_ Shirl... _END_OTHER_ _OTHER_SPEAKER_ Where you going? _END_OTHER_ _TARGET_SPEAKER_ I'm gonna get help. Oh where's my phone? Oh Kevin. Kevin you smashed it, didn't ya? Kevin, Kevin, where's your phone? _END_TARGET_`]). With this approach using the function `create_character_document_from_dataframe` we have less of the context of the target speaker, especially given that we use a BOW model, but using bigrams will help the model learn which tokens follow the target speaker and which follow the other speaker. However the 300 lines of target text limitations appears to make the focus shift towards other characters as the accuracy drops to 0.43 and the mean rand increases to 2.68. Another implemented approach (`create_character_document_from_dataframe_windowed`) is to use a fixed window size for adding other character's lines to the 300 target speaker lines from a scene so that no more than 2 (window size) lines before and after the target speaker's lines would be added from another speaker. These changes reduced the mean rank to 1.9 and increased the accuracy to 0.87 and a window size of 1 resulted in a 1.81 mean rank and 0.87 accuracy. In general, comparing the results with context features to previous result it can be seen that the model performs best without the added confusion from scene context (even window=0 didn't improve performance because of the added tags)

Question 4: (Implemented in NLP_Assignment_2_Q4.ipynb)
A total grid search has been implemented on different methods of preprocessing being peasant or not (example: `{"remove_punct":True,"lower":True,"expand_contractions":True,"normalize_space":False,"remove_repeat":True,"remove_word_numbers":True,"remove_filter":True,"stop_words":True,"pos_filtering":False,"lemmatize":False}`) On different additional feature (example: `["bigram", "sentiment", "emotion", "linguistic"]`), and on different matrix types (example: `["tfidf", "selection"]`). The result on all permutations is available in **grid_search_results.csv.** The first 22 rows show similar results with minimal mean rank. One of the optional results (which is similar to the initially suggested solution that was obtained through a limited grid search) is:

preprocess	feature_methods	matrix_types	mean_rank	mean_cosine_similarity	accuracy
{'lower': True, 'remove_punct': True, 'expand_contractions': True, 'normalize_space': False, 'remove_repeat': True, 'remove_word_numbers': True, 'remove_filter': True, 'stop_words': True, 'pos_filtering': False, 'lemmatize': False}	['bigram', 'sentiment', 'emotion', 'linguistic']	['tfidf',"selection"]	1.6875	0.9466374156755650	0.75

Which will be used for Q5 and Q6 (using the same methods implemented in question 1 and 2 as the preprocess methods, feature methods and matrix type appear to be one of the optimal choices). Adding scene context showed to reduce performance in general (and specifically with the chosen attributes so it was not used)

Question 5: (implemented in NLP_Assignment_2_distributional_semantics.ipynb)
According to the heat map we want to identify which held out character is most similar (other than the character himself and which character is least similar and does a character's held out vector best match with themself or is another character's vector even closer?

Character	Christian	Clare	heather	Ian	Jack	Jane	Max	Minty	Other	Phil	Ronnie	Roxy	Sean	Stacey	Tanya
Closest match	Jane(0.322)	Clare (0.234)	Heather (0.40.9)	Ian (0.189)	Jack (0.308)	Jane (0.704)	Max(0.506)	Minty (0.356)	Other (0.117)	Phil (0.139)	Ronnie (0.232)	Roxy (0.211)	Sean (0.23)	Stacey (0.186)	Tanya (0.221)
Furthest match	Phil(0.028)	Phil(0.039)	Sean (0.043)	Ronnie (0.017)	Phil(0.077)	Ronnie (0.011)	Ronnie (0.055)	Clare (0.057)	Minty (0.048)	Clare (0.051)	Stacey (0.028)	Phil (0.04)	Ronnie (0.03)	Ronnie (0.017)	Ronnie (0.017
Self	0.175	0.234	0.409	0.189	0.30	0.704	0.506	0.356)	0.117	0.139	0.232	0.211	0.23	0.186	0.221

All character apart from Christian have their own character as the closest vector but the cosine similarity values are generally close together which is understandable because in a series, they talk about the same subjects and they probably have similar words vectors with a bag of word approach and short document (300 lines) don't help capturing the difference (Christian and Jane probably use similar slangs and words or talk about the same context and that is the reason why Christian got a higher similarity with Jane compared to himself). In general, characters who speak about the same events or have similar personalities (lexically) appear closer in the vector space. Characters with unique jargon or repeated phrases remain easily distinguishable. In general, the system successfully matched characters to themselves (highest similarity being between Jane's vector with herself indicating she uses unique words and is more easily distinguishable with our approach.

Question 6: ((implemented in NLP_Assignment_2_distributional_semantics.ipynb)
Test result: `total words 8939, mean rank 1.3125, mean cosine similarity 0.30309059756497103, 13 correct out of 16 / accuracy: 0.8125`