ECS763U/ECS763P - NATURAL LANGUAGE PROCESSING - 2023/24

COURSEWORK 2

Vickshan Vicknakumaran (student)

Contents

Question 1

1. **Lower Case:** Converted all characters in the text to lowercase. This helps during normalisation of the text and ensures that the same words in different cases are treated identical.
2. **Tokenisation:** Splits the text into sperate words or tokens. Using the NLTK word tokenize, it will handle punctuation and complex word structures better. It also handles whitespaces as well.
3. **Stopword Removal:** Filters out common words which does not have semantic value in the text. This step will reduce noise and focus analysis on more meaningful words.
4. **Whitespace Removal:** Removes any extra spaces so that tokens are clean and uniform.
5. **Stemming and Lemmatisation with Part-Of-Speech Tagging:** This will reduce words to their base or root form. Stemming is the process of removing/trimming words which end with common patterns, while lemmatisation reduces the words to their dictionary form. This will help reduce the complexity of the text and focusing on the essence of words.

Question 2

To improve linguistic features, I have used the following techniques:

1. **N-gram Features:** Including the n-grams enables to capture more contextual information than single words alone. It is powerful in particularly understanding phrase structures and can provide more contextual information. The CountVectorizer with ngram_range=(1, 2) captures both unigrams and bigrams, adding depth to the feature set.
2. **TF-IDF Features:** This is a numerical statistic which help understand the importance of a word in a document in a corpus. This approach helps distinguish the significance of a word across the whole data. By using **TF-IDF**, more weight is given to the terms that are frequent in a specific document. The TfidfVectorizer is used here to transform the text data into a TF-IDF-weighted feature matrix.
3. **POS Tag Features:** Part-of-Speech tagging assigns tags to each word in the text for example noun, verb, adjective. This incorporates these features so it helps the model understand the structure of the sentences grammatically. This is an advantage in certain types of text analysis.
4. **Sentiment Analysis:** The function analyses the overall sentiment of the document. . It uses NLTK's SentimentIntensityAnalyzer from the vader_lexicon, which is particularly good at handling social media text and informal language. The function calculates sentiment scores which includes positive, negative, neutral, and compound (overall) scores. Its impact adds an emotional dimension to the feature set.
5. **create_document_matrix_from_corpus function:** Seamlessly transforms unstructured text into a structured matrix representation, enabling machine learning algorithms to process and extract meaningful insights from textual data. By incorporating a TF-IDF transformation, it moves beyond mere frequency counts, weighting terms based on their importance across the entire corpus.

Question 3

The improvements made in the **'create_character_document_from_dataframe'** are the following:

1. **Collecting Scene-Specific Lines:** This function iterates through the DataFrame and writes line for each character. More importantly, it also stores the lines spoken in each scene and

scene, and stores this information in the **'scene_lines'** dictionary, which is done by the scene and scene numbers.

2. **Dialogue Context:** After the collection of the lines. The function is looped through the 'scene_lines' dictonary. It adds lines for each character from other characters that are in the same episode and scene. This ensures that the only relevat contextual dialogues are included.
3. **Limiting Line Count**: 'max_line_count' ensures that only a specific maximum number of lines are added for each character. Helps prevent single characters in documents from becoming disproportionately large.

Question 4

Grid search function is used to find the best combination of pre-processing parameters. When testing through the grid search for the preprocessing it showed that the best mean rank was 3.5 and the best accuracy was 0.3 with **'Top Parameter Combinations: Parameters: (('lowercase', True), ('stemming_and_lemmmatizer', False), ('stopwords', False), ('tokenisation', False), ('whitespace', True))'** . This was a good gradual improvement from the original code was given.

Question 5

The similarity matrix reveals that some characters have more linguistically similar dialogue than others. Chandler Bing and Joey Tribbiani are the most similar (0.924), suggesting they share language patterns or topics. Ross Geller's dialogue is least like the aggregated #ALL# category (0.863), indicating distinct language usage. Characters with intertwined narratives or similar roles tend to have higher similarity scores, as seen with Monica Geller and Other_Female (0.903), while those with unique language styles or diverse topics are further apart in the matrix.

Question 6

Analysing the results from your information retrieval system, two key metrics stand out: the mean rank and the accuracy.

Mean Rank: The mean rank achieved is 1.7. This metric is particularly significant as it represents the average position in which the correct document appears across the queries. A mean rank of 1.7 indicates that on average, the correct document is usually found very close to the top of the ranked list.

Accuracy: The program achieved an accuracy of 60% (6 correct out of 10). This means that for 60% of the queries, the top-ranked document was indeed the correct one. An accuracy of 0.6 reflects a decent level of effectiveness in retrieving the most relevant document as the first result. However, there is still room for improvement, as 40% of the top choices were not the most relevant documents.