QUESTION 1:

The provided code implements a pre-processing function for character text in Natural Language Processing (NLP) using the Natural Language Toolkit (NLTK) library in Python. The primary objective of this code is to enhance the quality of text data by applying various pre-processing steps.

The contractions' function replaces common contractions in the input text with their expanded forms, ensuring that contractions like "won't" are transformed to "will not." The pre_process function, the core of the code, takes character text as input and performs several crucial pre-processing steps. It begins by lowercasing the text to standardize case, offering a consistent basis for analysis.

Optionally, contractions in the text are expanded, providing a more explicit representation of words. Tokenization, the process of breaking down text into individual words or tokens, is then applied using a specified tokenizer. Stopwords, common words with little semantic meaning, are removed, while preserving negations like "no" and "not." Lemmatization follows, converting words to their base forms (e.g., "running" to "run"). Finally, only nouns, adjectives, and verbs are retained based on their part-of-speech tags, focusing on essential linguistic elements.

This comprehensive pre-processing pipeline aims to clean and structure raw character text, rendering it suitable for subsequent NLP tasks such as sentiment analysis or classification. The steps undertaken serve to reduce noise in the text and extract relevant information for more accurate and meaningful analyses.

QUESTION 2:

The provided code conducts a comprehensive grid search for feature extraction configurations in the context of Natural Language Processing (NLP) using the feature extraction functions defined earlier. The main objective is to optimize the performance of the feature extraction process by systematically exploring a range of parameter values.

The `evaluate_feature_extraction` function takes a set of feature extraction configurations, processes the training corpus with each configuration, and evaluates the performance using an Information Retrieval (IR) framework on a validation set. The configurations include parameters such as n-grams range, top n-grams to select, document frequency thresholds, normalization techniques, and the number of features after dimensionality reduction.

The grid search explores various combinations of these parameters, applying them to the feature extraction process, and evaluating the resulting feature matrices. The evaluation considers metrics such as mean rank, mean cosine similarity, and accuracy in classifying characters. The objective is to identify the configuration that yields the best performance, enhancing the effectiveness of feature extraction for downstream tasks such as text classification.

In summary, the function aims to systematically optimize feature extraction parameters, providing insights into the most effective configuration for character-based text analysis in the given NLP context. The grid search approach ensures a thorough exploration of parameter space, facilitating the selection of an optimal configuration for subsequent model training and evaluation.

QUESTION 3:

The provided code focuses on the evaluation of configurations for creating character-level documents from a dataframe, which is crucial for subsequent tasks in Natural Language Processing (NLP). The primary objective is to systematically explore different configurations and identify the most effective parameters for assembling character documents. These documents incorporate contextual information, including lines from the same scene, enabling a richer representation of character interactions and dialogue.

The `create_character_document_from_dataframe` function takes a dataframe containing information about characters, lines, episodes, and scenes. It organizes the data, considering contextual lines within a specified scene, and compiles character documents by combining the target line with its surrounding context. The resulting character documents are then pre-processed and used for subsequent feature extraction.

The `evaluate_create_character_document_from_dataframe` function conducts a grid search over different configurations, such as context size, to find the optimal settings for creating character documents. It evaluates each configuration by processing the training and validation datasets and subsequently leveraging the best pre-processing and feature extraction configurations determined in earlier steps (Q1 and Q2).

The evaluation includes metrics like mean rank, mean cosine similarity, and accuracy, providing insights into the effectiveness of different configurations. The code aims to identify the configuration that yields the best performance in character document creation, contributing to improved downstream tasks such as information retrieval and character-based analysis in NLP. Additionally, the visualization of cosine similarity through a heat map enhances the understanding of document relationships in the dataset.

QUESTION 4:

The presented code segment is part of the pipeline for preparing and processing data for Natural Language Processing (NLP) tasks. It specifically focuses on the training and validation phases of the model. The primary objective is to create a feature matrix for training data and a corresponding feature matrix for validation data, adhering to the optimized configurations determined in the earlier steps (Q1 and Q2).

1. Training Character Documents Creation:

   - Utilizing the optimal configuration for creating character documents (best_config_character_doc), the `create_character_document_from_dataframe` function assembles character documents from the training dataset (train_data).

   - The context size parameter from the optimal configuration influences how much contextual information is included in each character document, enhancing the richness of the document representation.

2. Training Corpus Pre-processing (Q1):

   - The training character documents are pre-processed using the best configuration determined in Q1 (best_config_preprocess).

   - The pre-processing involves tokenization and expanding contractions, ensuring the text is normalized and ready for feature extraction.

3. Training Feature Matrix Creation (Q2):

   - The pre-processed training corpus is used to create a feature matrix using the best configuration for feature extraction (best_config_feature_extraction).

   - This involves transforming the textual data into a numerical representation, considering n-grams, top features, and other parameters optimized for the specific NLP task.

4. Validation Data Pre-processing and Feature Matrix Transformation:

   - Similar pre-processing steps are applied to the validation character documents using the same configurations determined for the training set.

   - The validation feature matrix is then transformed (not fitted) based on the pre-processed validation corpus using the optimal feature extraction configuration.

By following this process, the code ensures consistency in pre-processing and feature extraction between the training and validation datasets, facilitating the model's generalization and performance evaluation on unseen data.

QUESTION 5:

By plotting the cosine similarity in the output heat map, the function provides a visual summary of the relationships between characters based on the features extracted from their documents.

QUESTION 6:

The function evaluates a model trained on TV show character dialogues. It processes training data, creates a feature matrix based on optimized configurations, and pre-processes test data. The model is then evaluated on IR tasks, calculating mean rank, cosine similarity, and accuracy. The output includes line counts per character and performance metrics, providing insights into the model's ability to rank character lines and capture semantic similarities. This evaluation ensures the model's effectiveness on unseen data, confirming its generalization in natural language processing tasks.