Report of IR Assignment 3

Roll Number: 2021404
Name: Om Mehroliya

Answer 1

Approach

- **Loading Metadata Products**: The code loads product IDs from metadata based on search keywords. It reads a gzipped JSON file, extracts title and description from each record, and checks for keyword matches.
- **Preprocessing Review Text**: The code includes a placeholder function for preprocessing review text. However, it's not implemented in the provided snippet.
- **Loading Reviews for Products**: Another function loads and preprocesses reviews for selected products. It iterates through reviews, checks product IDs, and preprocesses review text.
- **Main Function**: Orchestrates the process by calling functions to get relevant product IDs, load reviews, and perform additional processing.

Methodologies

- The code uses Python's `gzip` module to handle gzipped files and `json` module to parse JSON records.
- It leverages Pandas for data manipulation and analysis, especially for loading data into DataFrames and performing preprocessing tasks.
- The approach employs simple string matching techniques to identify relevant products based on search keywords.

Assumptions

- The metadata and review data are assumed to be in a specific format (JSON) and compressed using gzip.
- Preprocessing steps are left as placeholders, assuming that more complex preprocessing will be implemented later.

Results

- The code generates a DataFrame containing preprocessed reviews for products matching the search keywords.
- It saves the head of the processed DataFrame to a text file (`ans1.txt`) for inspection and the processed data to a CSV file (`processed_data.csv`). However, specific preprocessing steps and resulting data are not shown in the provided code snippet.

Answer 2-4

- **Loading Data**: The code imports a custom module named "1.py" using `importlib.import_module` and defines functions to load data from a CSV file.
- **Data Preprocessing**: Functions are provided to preprocess the loaded data by removing duplicates and handling missing values.
- **Descriptive Statistics**: Another function calculates and prints descriptive statistics of the dataset, including total rows, number of reviews, average rating, unique products, and counts of good and bad ratings.
- **Rating Categorization**: There's a function to add a rating category ('Good' or 'Bad') based on the overall rating.
- **Word Cloud Visualization**: The code generates a word cloud visualization for good ratings using the processed review text and displays it using Matplotlib. Additionally, it saves the generated word cloud image to a file named `word_cloud_image.png`.
- **Main Analysis Function**: The main function orchestrates the entire analysis process. It loads the data, performs preprocessing, calculates descriptive statistics, adds rating categories, generates the word cloud, and saves the image.

Methodologies

- The code utilizes Pandas for data manipulation and analysis, Matplotlib for visualization, and the WordCloud library for generating word clouds.
- It employs numpy for numerical computations and exception handling to gracefully manage errors during data loading.

Assumptions

- The CSV file specified in `processed_file_path` contains the required data with necessary columns like 'overall' and 'processed_reviewText'.

- The preprocessing steps (removing duplicates and handling missing values) are sufficient for the analysis, and no further cleaning is required.
- Ratings are categorized as 'Good' if they are greater than or equal to 3, and 'Bad' otherwise.

Results

- The code successfully loads, preprocesses, and analyzes the dataset, providing descriptive statistics and generating a word cloud visualization for good ratings.
- The word cloud image is saved to the specified file path (`word_cloud_image.png`).
- Any errors encountered during data loading or processing are gracefully handled and reported to the user.

Answer 5

- The code utilizes Pandas for data manipulation, Matplotlib for visualization, and BeautifulSoup for HTML parsing.
- NLTK (Natural Language Toolkit) is used for text preprocessing tasks such as removing HTML tags, accented characters, and stopwords, as well as lemmatizing words.

Assumptions

- The CSV file specified in `file_path` contains the necessary data with columns like `processed_reviewText` and `overall`.
- The NLTK resources (e.g., stopwords, WordNet) need to be downloaded for text preprocessing, assuming internet connectivity.

Results

- The code successfully loads the dataset, preprocesses it, and adds a new column for rating categorization.
- Descriptive statistics are calculated and printed, providing insights into the dataset.
- Word cloud visualizations are generated to represent the most frequent words in good ratings.

Answer 6

- **Module Import**: The code imports a module named "5.py" using `importlib.import_module`.
- **NLTK Setup**: Necessary NLTK resources such as tokenizers, stopwords, and WordNet are downloaded using `nltk.download()`.
- **Text Preprocessing**: The function `preprocess_text()` from module 5 is encapsulated and applied to the DataFrame to clean and lemmatize the text data.
- **Data Loading and Preprocessing**: Data is loaded from a CSV file and preprocessed using the encapsulated preprocessing function.
- **Saving Data**: The preprocessed DataFrame is saved to a new CSV file using `to_csv()` method.
- **Word Cloud Generation**: A word cloud is generated from the preprocessed text and saved as an image file.

Methodologies

- The code leverages Pandas for data manipulation, Matplotlib for visualization, and NLTK for text preprocessing.
- BeautifulSoup is used for HTML parsing, and WordCloud library is utilized for generating word clouds.

Assumptions

- The CSV file specified in `file_path` contains the necessary data with a column named `processed_reviewText`.
- NLTK resources are downloaded and available for text preprocessing.

Results

Answer 7-10

- The code successfully loads and preprocesses the data, saves it to a CSV file, and generates a word cloud image representing the processed text.
- The first few rows of the preprocessed DataFrame are written to a text file for inspection.
- **Module Import**: The code imports a module named "5.py" using `importlib.import_module`.

- **Environment Setup**: The function `setup_environment()` sets up the environment by printing a message.
- **Dataset Loading and Preprocessing**: The function `load_and_preprocess_dataset()` loads the dataset from a CSV file, preprocesses it, and adds a new column for rating class based on the 'overall' column.
- **Dataset Splitting**: The function `split_dataset()` splits the dataset into training and testing sets.
- **Model Definition**: The function `define_models()` defines machine learning models using pipelines with TF-IDF vectorization and different classifiers.
- **Model Evaluation and Reporting**: The function `evaluate_and_write_reports()` evaluates the models, generates classification reports, and writes them to a file.
- **Main Execution**: The `main()` function orchestrates the entire process by calling the setup, loading, preprocessing, splitting, model definition, evaluation, and reporting functions in sequence.

Methodologies

- The code leverages scikit-learn for machine learning tasks such as vectorization, model training, evaluation, and reporting.
- It uses pipelines to combine vectorization and classification steps for each model, ensuring a streamlined workflow.
- Classification reports are generated using scikit-learn's `classification_report()` function, providing detailed metrics for model evaluation.

Assumptions

- The CSV file specified in `file_path` contains the necessary data with a column named `processed_reviewText` and an 'overall' column representing ratings.
- The dataset is assumed to have already undergone some preprocessing, such as HTML tag removal and text normalization.

Results

- The code successfully sets up the environment, loads and preprocesses the dataset, splits it, defines machine learning models, evaluates them, and writes classification reports to a file.
- The reports provide insights into the performance of each model, helping in model selection and further analysis.