IR Assignment-3

Loading the Electronics Dataset

- While working with this dataset, I realised that there are some encodings that are not convertible to utf-8 format, so I changed my encoding format while parsing into latin1 encoding.
- Since this CSV file is over 3.0 GB, I used pandas as it is highly optimized code and read the data chunks; I defined a suitable chunk size of 10,000 rows per chunk and started reading the file.
- While reading the csv, there were numerous occasions where I encountered parsing Errors; since that row was ill-managed or defective, I decided to skip the rows with parsing errors in my code, as they won't play such role in my dataset.
- After processing all the valid chunks, I then concatenated them all - and formed a bigger data frame called "main_df", with ignoring the indexes of valid chunks and maintaining our own indexing.
- Since CSV was not openable in MS Excel or other fields, I decided to print the top 50 rows, analyse each row, cell, and column, and understand the dataset.
- This was a highly crucial stage as it helped me understand those rows which have the most NAN values or will play little role in my dataset.
- <class 'pandas.core.frame.DataFrame'>
-
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13090043 entries, 0 to 13090042
Data columns (total 12 columns):
 #   Column          Dtype
---  ------          -----
 0   overall         object
 1   vote            object
 2   verified        object
 3   reviewTime      object
 4   reviewerID      object
 5   asin            object
 6   style           object
 7   reviewerName    object
 8   reviewText      object
 9   summary         object
 10  unixReviewTime  object
 11  image           object
dtypes: object(12)
memory usage: 1.2+ GB
None
```

- I also created a subset of this dataset , top 10,000 rows and stored it in csv format to view and study the data and since I am going to eliminate the some useless columns, which won't be much required in my assignment. The file was **"sample_data_main_df_10000_rows_wo_preprocessing.csv."**
- `Number of rows in the Main DataFrame: 13090043`

Loading of the meat_electronics.json file

- Similar to the first dataset, the JSON file size was 1.2GB +; loading such a massive dataset was impossible for my backend specs, so I defined chunks; somehow, a chunk size of 1000 rows worked out for me.
- After getting valid chunks, I concatenated them into a data frame, “meta_data_df” and printed its shape.
- Since this JSON was not openable in the code editor or other fields, I decided to print the top 50 rows, analyse each row, cell, and column, and understand the dataset.
- Like the first one, I also created a subset of this dataset, the top 10,000 rows and stored it in csv format to view and study the data since I will eliminate some useless columns, which won’t be much required in my assignment. The file eas - **"sample_data_meta_data_df_10000_rows_wo_preprocessing.csv"**
- **Shape of metadata DataFrame: (786445, 19)**
- `Number of rows in the Metadata DataFrame: 786445`
-
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 786445 entries, 0 to 786444
Data columns (total 19 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   category        786445 non-null  object
 1   tech1           786445 non-null  object
 2   description     786445 non-null  object
 3   fit             786445 non-null  object
 4   title           786445 non-null  object
 5   also_buy        786445 non-null  object
 6   tech2           786445 non-null  object
 7   brand           786445 non-null  object
 8   feature         786445 non-null  object
 9   rank            786445 non-null  object
 10  also_view       786445 non-null  object
 11  main_cat        786445 non-null  object
 12  similar_item    786445 non-null  object
 13  date            785741 non-null  object
 14  price           786445 non-null  object
 15  asin            786445 non-null  object
 16  imageURL        786445 non-null  object
 17  imageURLHighRes 786445 non-null  object
 18  details         785607 non-null  object
dtypes: object(19)
memory usage: 114.0+ MB
None
Shape of metadata DataFrame: (786445, 19)
```

Choosing a Product and Defining Keywords and Synonyms

- I chose my product as headphones and then chose the keywords as stated: ['headphone', 'headphones','headphones(in-ear)','noise-canceling headphones']
- I clearly distinguished it from the headsets and earphones, and earbuds, and we are interested in headphones, which may or may not have a microphones.
- I have gone through various synonyms and acronyms and came up with this set of keywords for my product.

Isolating ASIN Product Numbers Related to Headphones in meta_data_df:

- Examined product titles, descriptions, and categories in the dataset to identify ASIN numbers associated with headphones.
- Focused primarily on product titles for keyword searches and also considered descriptions and categories.

- Detected and stored ASIN numbers related to headphones in pickle files to avoid false positives.

```
Number of rows in the DataFrame after droping missing value rows and duplicate rows: 26060
<class 'pandas.core.frame.DataFrame'>
Index: 26060 entries, 11480 to 3829070
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   overall     26060 non-null  object
 1   verified    26060 non-null  object
 2   reviewTime  26060 non-null  object
 3   reviewerID  26060 non-null  object
 4   asin        26060 non-null  object
 5   reviewText  26060 non-null  object
 6   summary     26060 non-null  object
dtypes: object(7)
memory usage: 1.6+ MB
None
```

-

Preprocessing the main_df According to the Need:

- Selected essential columns ('overall', 'verified', 'reviewTime', 'reviewerID', 'asin', 'reviewText', 'summary') for analysis, as they provide valuable insights for further investigation.
- Removed unnecessary columns containing many NaN values that did not contribute significantly to the analysis.
- Examined and explored the filtered_main_df using info() and head() methods to understand its structure and content.
- Saved the inspection results as a raw CSV file named "RAW_FILE_CSV" for reference.
- Removed rows with missing values and duplicates from the DataFrame to ensure data quality.
- Converted columns to appropriate data types ('overall' to float, 'reviewTime' to datetime, 'verified' to boolean, 'reviewerID' and 'asin' to string) for consistency and ease of analysis.
- Saved the cleaned and processed DataFrame as "final_file.csv" for further analysis and modeling.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26057 entries, 0 to 26056
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   overall     26057 non-null  float64
 1   verified    26057 non-null  bool   
 2   reviewTime  26057 non-null  object 
 3   reviewerID  26057 non-null  object 
 4   asin        26057 non-null  object 
 5   reviewText  26057 non-null  object 
 6   summary     26057 non-null  object 
dtypes: bool(1), float64(1), object(5)
memory usage: 1.2+ MB
None
```

-

Performing the Tasks:

- Printed descriptive statistics of the product reviews to understand the distribution of ratings and other key metrics.

- Processed review text to extract relevant information and stored it in a new column named 'processed_reviewText' for sentiment analysis and keyword extraction.
- Conducted a left join between main_df and meta_data_df using ASIN as the key to create merged_df, which was then deduplicated to ensure unique product representation.
- Identified the top and least 20 brands based on review counts in merged_df to understand brand popularity and market share.
- Determined the most positively reviewed product and explored its details using metadata from meta_data_df.
- Generated word clouds using processed summary and review text to visualize key themes and sentiments expressed in the reviews.
- Analyzed review counts over the last 5 consecutive years to identify trends and patterns in customer feedback.
- Visualized the distribution of ratings versus the number of reviews using pie charts to gain insights into customer satisfaction levels.

Train the Five ML Models and input tf-idf vector as the input-feature, computing, precision, recall, support, f1score:

- We are using TfidfVectorizer from sklearn.feature_extraction.text to convert text data (X) into TF-IDF features (X_tfidf).
- TF-IDF is a numerical representation technique that weighs terms based on their importance in a document relative to a corpus of documents. It helps in capturing the significance of terms in a document.
- We are using train_test_split from sklearn.model_selection to split the TF-IDF features (X_tfidf) and corresponding labels (y) into training and testing sets (X_train, X_test, y_train, y_test).
- The data is split with a ratio of 75% for training (X_train, y_train) and 25% for testing (X_test, y_test).
- random_state=42 ensures reproducibility of the split.
- We are importing various classifiers (SVC, RandomForestClassifier, LogisticRegression, MultinomialNB, KNeighborsClassifier) from sklearn to compare their performance.
- Each classifier is instantiated and stored in a dictionary (classifiers) with its respective name as the key.
- For each classifier in the classifiers dictionary:
    a. We train the classifier (clf.fit(X_train, y_train)) using the training data (X_train, y_train).
    b. We make predictions (y_pred) on the test data (X_test) using the trained classifier (clf.predict(X_test)).
    c. We generate a classification report (classification_report) to evaluate the classifier's performance on the test data.
    d. The classification report includes metrics such as precision, recall, F1-score, and support for each class label ('Good', 'Average', 'Bad').

    e. The results are printed for each classifier, showing how well each one performs in classifying the test data.

- The output for each classifier includes a detailed classification report highlighting its performance metrics.
- By comparing the classifiers' performance metrics (precision, recall, F1-score), we can assess and select the most suitable classifier for our text classification task based on the provided dataset (X and y).

Collaborative Filtering:

- We also calculate the user_item_matrix by making the df[‘reviewerID’] as rows and df[‘asin’] as columns and filling values ratings and with 0 where the rating is not s not specified.

  user_user_collaborative_filtering Function:
- Purpose: Implements user-user collaborative filtering to predict ratings based on user similarity.
- Input Parameters:
    - user_item_matrix: User-item rating matrix represented as a numpy array.
    - k_values: List values representing the number of nearest neighbors (K) to consider.
    - n_neighbors: Number of nearest neighbors to use for similarity computation.
    - k_folds: Number of folds for cross-validation (default is 5).
- Steps:
    - Splits the user-item matrix into training and validation sets using k-fold cross-validation.
    - Computes cosine similarity between users based on the training set.
    - Identifies top n_neighbors similar users for each user.
    - Predicts ratings for the validation set by averaging ratings from similar users.
    - Calculates mean absolute error (MAE) between predicted and actual ratings for evaluation.
    - Returns a list of MAE scores corresponding to each value of k_values.

  item_item_collaborative_filtering Function:
- Purpose: Implements item-item collaborative filtering to predict ratings based on similarity between items.
- Input Parameters:
    - user_item_matrix: User-item rating matrix represented as a numpy array (transposed to item-user matrix).
    - k_values: List of values representing the number of nearest neighbors (K) to consider.
    - n_neighbors: Number of nearest neighbors to use for similarity computation.
    - k_folds: Number of folds for cross-validation (default is 5).
- Steps:
    - Transposes the user-item matrix to create an item-user matrix.

    - Splits the item-user matrix into training and validation sets using k-fold cross-validation.
    - Computes cosine similarity between items based on the training set.
    - Identifies top n_neighbors similar items for each item.
    - Predicts ratings for the validation set by averaging ratings from similar items.
    - Calculates mean absolute error (MAE) between predicted and actual ratings for evaluation.
    - Returns a list of MAE scores corresponding to each value of k_values.

Key Points:

- Both functions use cosine similarity to measure the similarity between users/items based on their rating patterns.
- They utilize k-fold cross-validation to train and evaluate the collaborative filtering models.
- MAE is used as a metric to assess the performance of the models in predicting ratings.
- The functions are parameterized to experiment with different values of n_neighbors (number of neighbors) and k_values (number of nearest neighbors) for evaluation and comparison.

Generating Top 10 Products by Sum Ratings and Displaying Product Information:

- Calculating Sum of Ratings for Each Product:
    - Computes the sum of ratings for each product (item) across all users (rows) in the user_item_matrix using axis=0 to sum along columns.
- Sorting Products by Sum Ratings:
    - Sorts the products by their sum of ratings in descending order using nlargest(10) to get the top 10 products with the highest sum of ratings.
- Retrieving and Displaying Product Information:
    - For each of the top 10 products:
        - Retrieves additional information (title and description) from the meta_data_df DataFrame based on the ASIN (Amazon Standard Identification Number) of the product.
        - Assumes that the meta_data_df DataFrame contains columns like 'asin', 'title', and 'description' that correspond to product information.
        - Prints the rank, ASIN number, sum of ratings, title, and description of each product in a formatted manner for display.
        - Uses a loop (enumerate) to iterate over the top 10 products, displaying their information sequentially.

  Key Points:

    - Utilizes the sum() method on the user_item_matrix to calculate the sum of ratings for each product.
    - Uses nlargest(10) to identify and retrieve the top 10 products with the highest sum of ratings.
    - Retrieves corresponding product information (title and description) from the meta_data_df DataFrame based on the ASIN number of each product.

- Prints the top 10 products along with their ratings sum, title, and description in a structured format for easy readability and analysis.

Created an interactive input taking User-user recommender and item-item recommender

- It takes a reviewerID for user-user and an asin number for item-item as input. Additionally, it accepts the number of rankings as input. Using cosine similarity, it shows the most similar product and the most similar items, respectively.