Assignment II
Recommendation System

Anirudh Potlapally

Introduction

The objective of this assignment is to create an advanced recommendation system utilizing the Amazon user reviews dataset. The specific dataset used for this project is focused on the Automotive data. Each individual data sample within the dataset represents a user-item interaction record that contains the following fields:

- User ID: denoted as "reviewerID" in the dataset.
- Product ID: denoted as "asin" in the dataset.
- Rating: a 1-5 integer star rating that the user assigned to the product, denoted as "overall" in the dataset.
- Review: a textual review of the product that the user provided, denoted as "reviewText" in the dataset.
- Title: the title of the review, denoted as "summary" in the dataset.
- Timestamp: the time that the user made the rating and review.
- Helpfulness: contains two numbers, i.e., [#users that think this review is not helpful, #users that think this review is helpful].
- Image: for each product, the dataset provides the image of the product in the form of a 4096-dimensional vector learned by a CNN deep neural network. These vectors are provided in an independent dataset called "Visual Features," which is also available on the website.
- Metadata: some metadata information for each product, including product title, price, image URL, brand, category, etc. It is also provided as an independent dataset called "Metadata," which is also available on the website.

Throughout this project, I used Python's surprise package for data preparation and model training and evaluation.

Github-link

Data Selection

The entire dataset contains 7,990,166 ratings, 3,873,247 unique reviewers, and 925,387 unique products. To streamline the project and manage computational resources effectively, a smaller subset of 100,000 samples was selected. Within this subset, there were 96,805 unique reviewers and 61,933 unique products. The detailed steps involved in the data selection process include:

1. **Read Data from GZ File:** In this step, we read the data from a GZ file line by line and store each line in a list.
2. **Convert List to DataFrame:** After reading the data, we convert the list into a dataframe. A dataframe is a tabular data structure commonly used in data analysis and manipulation.
3. **Column Selection:** Once we have the dataframe, we identify and drop the unwanted columns that are not required for further analysis.
4. **Choose Relevant Data Fields:** From the remaining columns, we select three specific data fields: 'ReviewerID', 'asin', and 'overall'. These fields provide information about the reviewer's ID, the product's ASIN (Amazon Standard Identification Number), and the overall rating given by the reviewer.
5. **Drop Null Fields:** We identify and drop any null fields in the dataframe. Null fields are missing or undefined values that could affect the accuracy of our analysis.
6. **Reduce Dataset Size:** Since the dataset is extremely big, for ease of calculation, a subset of 100,000 data samples was chosen the dataset for further analysis.

Data Processing

Following the data selection phase, the dataset was further preprocessed to prepare it for an in-depth analysis. This involves splitting the dataset into distinct training and testing subsets, ensuring that there are no overlapping samples between the two sets. The step-by-step process for this stage is outlined below:

1. **Group Data by Review IDs**: In this step, we group the data by review IDs. This allows us to analyze and process the data at the reviewer level.
2. **Split Data into Train and Test Sets**: Once the data is grouped by review IDs, we split the total dataset into train and test subsets. The train set will be used to train our model, while the test set will be used to evaluate its performance.
3. **Create Train and Test Sets using surprise's Reader() Class**: To prepare the data for further analysis, we utilize surprise's Reader() class. This class helps parse through the dataframes and create train and test sets that can be used with various collaborative filtering algorithms.

Algorithm selection

Four algorithms were considered while building recommendation systems in this experiment. The models were trained and evaluated using Surprise, which is a popular library for building and analyzing recommender systems. It provides various collaborative filtering algorithms and evaluation metrics to make it easier to develop recommendation systems. The following are the four algorithms:

1. **BaselineOnly:** BaselineOnly is used to predict ratings based on baseline estimates. It uses a basic model that considers the average rating of items, the average rating given by each user, and the deviations of individual items and users from these averages.

2. **SVD (Singular Value Decomposition):** SVD is a matrix factorization-based algorithm used in recommendation systems. Its underlying principle is that it decomposes the user-item rating matrix into lower-dimensional matrices to capture latent factors contributing to user preferences. Subsequently, SVD attempts to reconstruct the original matrix by estimating missing ratings and can provide personalized recommendations based on these estimates.

3. **NMF (Non-Negative Matrix Factorization):** NMF is a matrix factorization-based algorithm. It decomposes the user-item rating matrix into non-negative matrices and approximates the original matrix by multiplying them.

4. **CoClustering:** CoClustering is a clustering-based algorithm that simultaneously clusters users and items based on the users’ rating patterns. It divides the user-item matrix into sub-matrices and finds clusters with similar rating behaviors.

Model training and evaluation

After preparing the datasets for training and testing, I proceeded to evaluate the performance of four recommendation algorithms: SVD, BaselineOnly, NMF, and CoClustering. The implementation of these algorithms was done using the Python package 'Surprise'.

The training and evaluation process involved the following steps:

1. **Initialization**: I initialized an instance of each algorithm using the Surprise package. This step sets up the algorithm with default or specified parameters.
2. **Training and Testing**: The initialized algorithms were trained using the training set, which contains user-item interactions. The models were then tested using the testing set, which consists of unseen user-item pairs. The algorithms utilized the training data to learn patterns and make predictions for the testing data.

3. **Performance Metrics**: To assess the accuracy of the models, I calculated the Root Mean Squared Error (RMSE) and Mean Squared Error (MSE) scores on the predictions obtained from the test set. These metrics provide insights into the quality of the recommendations made by the models.
4. **Top 10 Recommendations**: After obtaining the predictions, I selected the top 10 recommendations for each user from each algorithm's output. These recommendations represent the most highly recommended items for each user based on the trained models.
5. **Recommendation Evaluation**: Additionally, I evaluated the quality of the recommendation predictions for each algorithm. Specifically, I calculated precision, recall, F-measure, and conversion rate for the top 10 recommendations generated by each algorithm for each item in the test set. The code for precision and recall calculations was adapted from the Surprise package's documentation, which can be found at [this source].

Results

The results obtained from the test set are summarized in the table below

Algorithm	RMSE	MSE	Precision	Recall	F-Measure	Conversion Rate
Baseline	0.9740	0.9487	0.945833	1.0	0.97216	0.9458333
SVD	0.9669	0.9348	0.945833	1.0	0.97216	0.9458333
Co-Clustering	1.2828	1.6456	0.958333	0.895833	0.92602	0.9583333
NMF	1.2740	1.6231	0.895833	0.69166	0.78062	0.8958

It was observed that the SVD and the Baseline methods had comparable performance, and both outperformed the Co-Clustering method and the NMF method. Although the RMSE and MSE scores of the NMF method were slightly lower than the Co-Clustering method, the former's precision, recall, and F-measure scores were considerably lower than the latter.

The recommendations for each of the algorithms are stored in a text file in the results folder in the GitHub repository in the directory named Recommendation Systems.

Link to GitHub: https://github.com/AnirudhPotlapally/CSE-272.git