Liubov Ivashov

Amazon Fine Food recommender systems using R and Python

1.Introduction

Recommender systems is the essential tool for cross-sale for many online platforms such as Amazon.
There are two main types of recommendation systems:

- Content-based recommendation system when we use similar features of the product to predict next purchased item. For example, we can recommend the book of the same author or books in the same category based on our previous purchased and highly rated ones.
- Collaborative recommendation system which uses opinion of the users that bought similar products.

Collaborative recommendation

In this project, we will use a basic collaborative recommendation system only, or more specifically:

- User-based collaborative filtering (UBCF model): it is based on finding similar users and find the items those users have liked but we haven't tried yet.
- Item-based collaborative filtering (IBCF model): in this case, we will find similar products to the one the user has bought, and we will recommend those products that are like those which has ratted as best.

For modeling this project has been used R package "Recommender Lab" and Python's Natural Language Toolkit (NLTK). The main goal of the recommendation system – recommend top 3 products from Amazon that a person can potentially buy next.

2.Data

- The CSV data file was collected from the Kaggle website.

- This dataset contains reviews of fine foods from Amazon.
- The data spans over a period of more than 10 years, including all ~500,000 reviews up to October 2012.
- Reviews include product and user information, ratings, and a plain text review. It also includes reviews from all other Amazon categories.
- Link to the Dataset: https://www.kaggle.com/snap/amazon-fine-food-reviews

3.Hypotheses

H0: We can predict next purchases of the customers based on their previous purchase experience and based on the purchase experience of people who bought similar products and rated them 4 starts and more.
Ha: Rating of the previous purchases of the similar customer cannot help us predict future purchases.

4.Methodology and Approach

4.1 Recommender Lab package

For this modeling we use ProductID, UserId and Score columns only. We drop text columns, time stamp and other columns that are not relevant for the purpose of recommendation system. To work with Recommender Lab, we need to clean and prepare data to the certain format: required pre-process data and convert it into the matrix format. Next step is to normalize the score. We have unbalanced data – most reviews are 4 or 5 stars. We can assume that at that time people mostly gave positive reviews. We need to normalize the data (method = ”Z-score”) before starting modeling.

In our data set we have items that have been reviewed just a few times, to avoid the bias we will use users who have rated at least 30 foods, and foods that have been reviewed at least 100 times.

After we transformed cleaned data to the matrix format, we can start modeling. We split data 80/20 – training/test data. Number of recommended items = 3.

IBCF (Item-based):
1. For each two items, measure how similar they are in terms of having received similar ratings by similar users.
2. For each item, identify the k-most similar items.
3. For each user, identify the items that are most like the user’s purchases.
4. IBCF recommends items based on the similarity matrix.

UBCF(User-based):
1. Measure how similar each user is to the new one. Like IBCF, popular similarity measures are correlation and cosine.
2. Identify the most similar users. The options are: Take account of the top k users (k-nearest_neighbors). Take account of the users whose similarity is above a defined threshold.
3. Rate the items purchased by the most similar users and the approaches are: Average rating. Weighted average rating, using the similarities as weights.

4. Pick the top-rated items.

We evaluate **eight different models (commonly used to measure similarity)** to get ourselves the best working recommender system:

- Item Based Collaborative Filtering with Cosine (IBCF_cosine)
- Item Based Collaborative Filtering Pearson (IBCF_Pearson)
- Item Based Collaborative Filtering Pearson Euclidean Distance (IBCF_Euclidean) for similarity score
- Single Value Decomposition (SVD)
- Alternating Least Squares (ALS)
- Popular
- Random

4.2 Recommender system using Python's Natural Language Toolkit (NLTK)

To predict a user's response to a product, it is necessary to understand the tastes of the user and the properties of the product (Review Summary). The taste of the user can be learned from user's review of various products. The properties of the product can be obtained by mining the reviews given by multiple users. Attainment of the tastes of the user and properties of the product will allow us to estimate whether a user will have positive or negative reactions to the products. We use KNN classifier to find similar products. During the modeling we will follow three main steps.

Step 1: attempts to find similar products based on qualitative reviews (Summary)
Step 2: compares quantitative review (Score) and qualitative review (Summary) to create a connection between them
Step 3: analyze each user's qualitative review and recommend the products using Step 1 and filter out the low-ranking products by incorporating Step 2

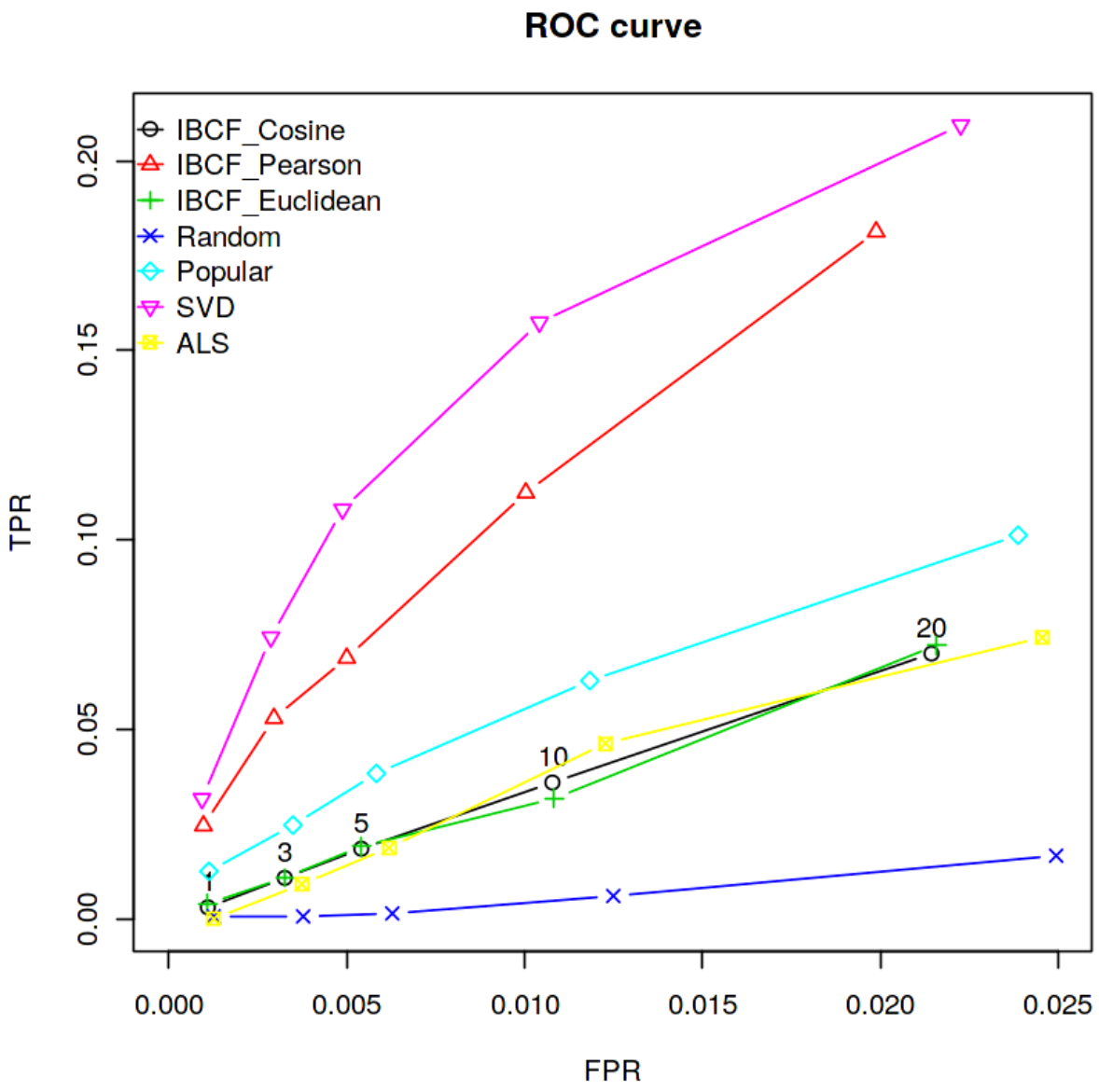
Following these steps, we will achieve strong recommendation system that using also a text data from the data set in regards to improve precision metric from the previous model. We create same collaborative filtering models: user-based and item-based.

5.Performance Measure

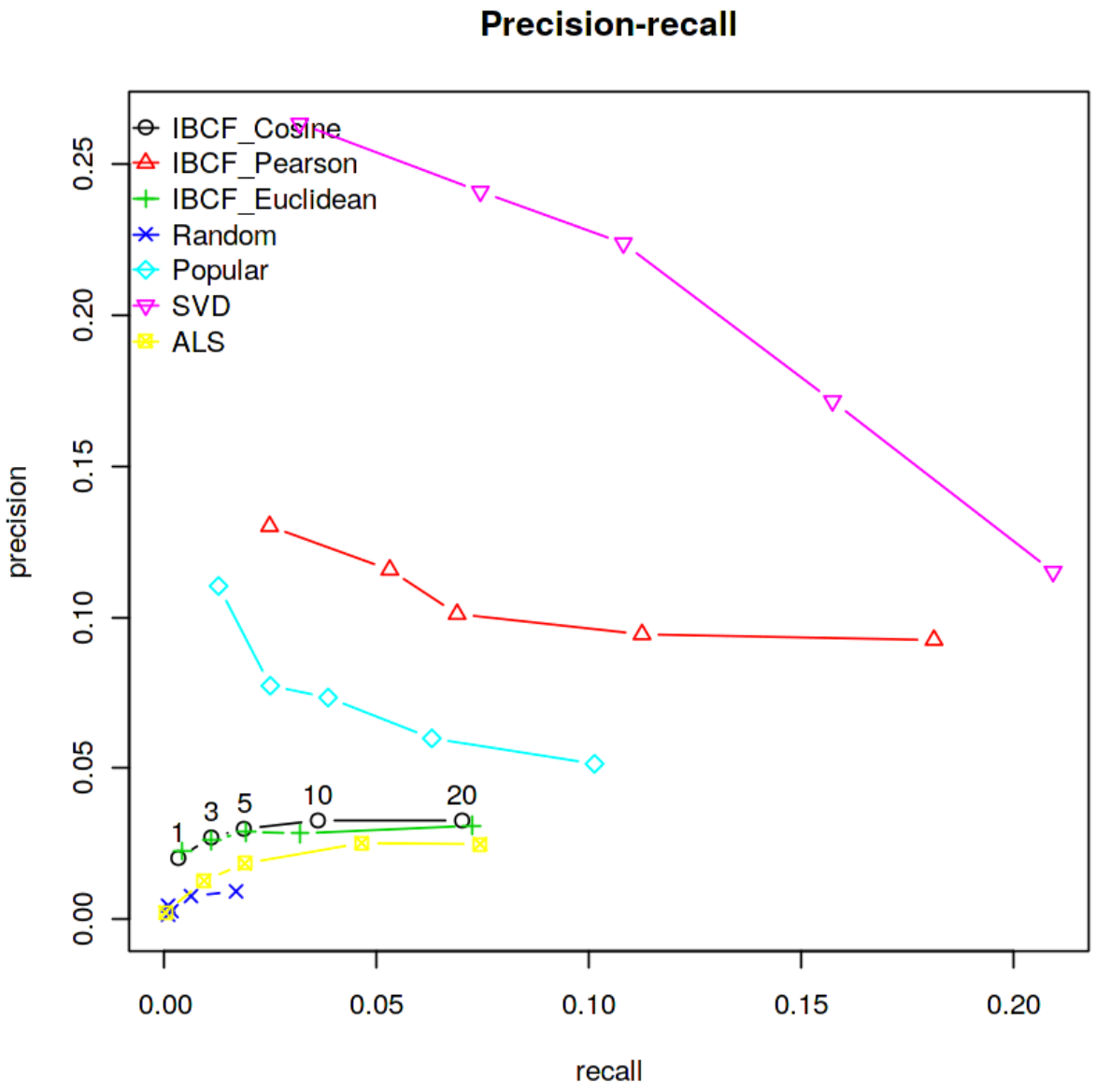
The most popular way to evaluate the recommender system is to use precision metric. Precision is the number of selected items that are relevant.

5.1 Evaluation by Recommender Lab

For evaluation of these models, we use R function "evaluationScheme" which allow us to find one the most successful model from the list.



SVG has the highest precision for three items suggestions of all the models. This is the model we should likely use when we choose with recommender system.



5.2 Evaluation by k-nearest neighbor (Python)

For item-based filtering model – the precision is 73%.

	precision	recall	f1-score	support
3	0.50	0.80	0.62	10
4	0.90	0.70	0.79	27
accuracy			0.73	37
macro avg	0.70	0.75	0.70	37
weighted avg	0.80	0.73	0.74	37

0.7297297297297297

For user-based filtering model, the precision is 100%.

```
Predicting review score for testset user are :  [3 4 4 4]
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         0
           4       1.00      0.75      0.86         4

    accuracy                           0.75         4
   macro avg       0.50      0.38      0.43         4
weighted avg       1.00      0.75      0.86         4
```

For recommendation system we used only data with review score 4 and 5, so the precision is perfect for this model.

However, as we can see Summary did not provide us any significant improvements in precision. All models provided in this project are basic models and even though they provide good results, they could not be used for new users, only for established users who actively reviewed items and left recommendation.

6.Word Cloud for different Score categories

In this section, we can explore word clouds for 5 categories of reviews starting from 1 to 5. WordCloud is a visualization technique for text data wherein each word is pictured with its importance in the context or frequency.

Score 1

Review Score One

Most frequently occurred word in the reviews with score one (1): disappointed, taste, horrible, bad, terrible. We can even assume what kind of product received such unpleasant reviews: coffee, tea, dog food.

Score 2

Liubov Ivashov

Most frequently occurred word in the reviews with score Two(2): taste, overprices, good, flavor, disappointing, great, bland. Not helpful. May be required additional list of stop words to prune to avoid confusion like this.

Score 3

The most frequently occurred words in Score Three (3) category: ok, bad, small, flavor, taste, disappointing, average.

Score 4

The most frequently occurred words in Score Four (4) category: good, tasty, great, delicious, great.

Score 5

The most frequently occurred words in Score Five (5) category: delicious, love, best, great product, awesome, tasty. These words mostly overlapped with Score four categories, that is why it's been a good idea to split reviews on two categories – positive (Score 4 and 5) and negative (Score 1 to 3).

7.Implications and Future Work

In this project we used simple recommender systems that don't use the whole potential of the text data that we have - mostly rely on reviews' score or reviews' summary. Even though these models showed pretty good results they are far from perfect recommendation system: work good with people who is actively reviewing and scoring products. These models predict good for no more than 5 items (used for prediction of top 3 items). Surprisingly, Amazon still uses this

collaborative filtering recommendation model, even though they test deep learning models constantly.

As a next step we need to implement **BERT Embedding recommender system:** feature extraction techniques and hybrid deep learning methods for sentiment analysis exploiting the advantages of BERT, to incorporate sentiments into recommendation methods as additional feedback and thus improve the performance and the reliability of recommender systems. However, it will require update data and use not only ProductId, but the names of the items and short description of each item.

Reference:

Michele Usuelli, Suresh K. Gorakala "Building a Recommendation System with R"