Amazon Fine Food Recommendation System

Final Report

Dongxue Shao
dongxue.shao@essec.edu

Xiaoyu Tian
xiaoyu.tian@essec.edu

Shenghua Du
shenghua.du@esesc.edu

Tailai Zhang
tailai.zhang@essec.edu

ABSTRACT

In this report we implemented the following models to build a recommendation system based on data from Amazon Fine Food Reviews [1]: Matrix Factorization, SVD, Random Forest and Times Series.

Our main assumption is that for a certain user, the higher review score is, the more likely the article should be recommended. We tried to compare different methods to establish a more reasonable and reliable system. Performances of the models are evaluated using MSE for the score and Confusion Matrix for recommendation or not. We creatively considered the time factor as a threshold function, say, if the score based on time model is below three, we will reject our recommendation. This will prevent us from recommending good but outdated products to our customers. This model is also helpful for users who do not any records in the database: we can only recommend them the products in trend.

After evaluating each model, we decided to use Matrix Factorization and considered Time Series Model as a Threshold.

KEYWORDS

Recommendation system, Machine Learning, Matrix Factorization

1 INTRODUCTION

Almost every e-commerce websites allow users to rate the products or services which they received when shopping. These feedbacks serve as suggestions to other users and are influential to people’s decisions on whether to buy the product. Therefore, exploring and understanding the ratings has become an important way to understand the need of users. Moreover, applying the data to build a better recommendation system is an integral part of the success of a company. Recommendation systems of Amazon brings more than 30% of revenues, and Netflix, where 75% of what people watch is from some sort of recommendation.

Based on the Amazon Data, we built a recommendation system for Amazon users. We implemented Matrix Factorization, SVD, Deep Learning, Random Forest and Times Series. We compared different methods and made a combination of some methods to provide a better recommendation.

This project is particularly important for the following reasons:

1) Business aspect: Boost revenue: discovering potential demand; Increase client retention: companies are more likely to obtain more repeating clients and build stable relationships with them; Cost Reduction: discovering potential demand may also help to improve supply chain efficiency.

2) Learning aspect: no existing kernel available from Kaggle has tried Matrix Decomposition, SVD and Deep Learning. We benefit from implementing our algorithms to develop a better understanding of the trade- off between different methods.

Potential application: This project can be easily applied to different business areas, for instance, movie, music, restaurant, book, hotel or even dating application.

2 PROBLEM DEFINITION

The problem we are going to solve is how to help users select products which they may like and to make recommendation to stimulate sales and increase profits.
Firstly, we decided to choose the Amazon Fine Food Reviews dataset which consists of 568,454 food reviews Amazon users left up to October 2012 as our dataset.
Secondly, our recommendation system is based on users rating prediction. We assume that users tend to like the products that have a score of greater than 4 and we will consider the highest 5 scores product as our recommendation candidates.
Thirdly, we implemented several algorithms to predict the scores of each product for each user.

2.2 Distance Based Model

Here we use the cosine-distance to give the similarity between vectors. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The similarity ranges from -1 to 1 where -1 means exactly opposite, 1 means exactly the same and in-between values indicating intermediate similarity or dissimilarity.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|_2\|\mathbf{B}\|_2} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

Since the MSE of traditional Collaborative Filtering model is quite high, we decided to use quasi-SVD approach instead. In this method, the output matrix is the dot product of three matrix. Here U is user similarity matrix and P is product similarity.

$$\Sigma_{m*n} = U_{m*m} \cdot \Sigma_{m*n} \cdot P_{n*n}$$

2.2 Matrix Factorization

Matrix Factorization is another way to decompose the user-product matrix. MF models map both users and products to a joint latent factor space of dimensionality ***f*** (***f*** refers to the number of the latent factors), such that user-product interactions are modeled as inner products in that space. In this report, we create two new matrices "user-latent_factor" (size ***q*f***, ***q*** refers to number of users) and "product-latent_factor" (size ***p*f***, ***p*** refers to the number of products). Element in target matrix can be calculated as $\hat{r}_{ui} = q_i^T p_u$.
We choose Least-square method to minimize the loss. Constant ***λ*** controls the extent of regularization.

$$\min_{q^*, p^*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

2.3 Random Forest

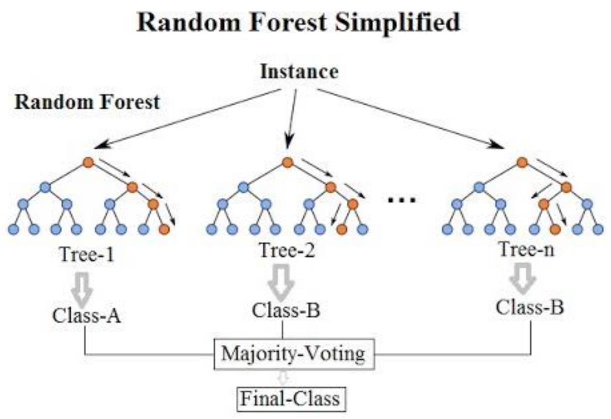
Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.
This process is sometimes called "feature bagging". These are models built from a training set for new points x' by looking at the "neighborhood" of the point, formalized by a weight function ***W***:

$$\hat{y} = \sum_{i=1}^{n} W(x_i, x')\, y_i.$$

Since a forest averages the predictions of a set of m trees with individual weight functions W_j, its predictions are

$$\hat{y} = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{n} W_j(x_i, x')\, y_i = \sum_{i=1}^{n}\left(\frac{1}{m}\sum_{j=1}^{m} W_j(x_i, x')\right) y_i.$$

One of the advantages of random forests method is that it can avoid overfitting because of the law of large numbers.



As it is currently the state-of-art method for classification problem, we decided to implement it as a reference.

Here, we chose "*ProductId*", "*UserId*", "*Time*", "*#Products*" and "*#Users*" as features and "*Score*" as the classification labels and use API form Scikit-learn to fit a random forest models.

2.4 Probabilistic Matrix Factorization

In PMF, we assume that the appearance of user and product are compiled to Gaussian distribution as following.

$$p(U|\sigma_U^2) = \prod_{i=1}^{N} \mathcal{N}(U_i|0, \sigma_U^2\mathbf{I}), \qquad p(V|\sigma_V^2) = \prod_{j=1}^{M} \mathcal{N}(V_j|0, \sigma_V^2\mathbf{I}).$$

The distribution of Scores is the joint distribution of the former two variables.

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|U_i^T V_j, \sigma^2)\right]^{I_{ij}}$$

According to Bayesian formula, we can get the final objective function as following:

$$\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - U_i^T V_j)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} \| U_i \|^2 \quad + \frac{\lambda_V}{2}\sum_{j=1}^{M} \| V_j \|^2$$

2.5 SVD

We apply Singular value decomposition (SVD) algorithms to mapping the user-product matrix into low-dimensional space.

$$A_{m*n} = U_{m*m}\Sigma_{m*n}V_{n*n}^T \approx U_{m*k}\Sigma_{k*k}V_{k*n}^T$$

Then we can get the predicted scores based on the similarity between unscored items and the others.
As for evaluation, we choose MSE (including train set and test set MSE) and Confusion Matrix (score >4 as recommend, score <4 as not recommend) to evaluate all the models above.
Additionally, we introduce a time parameter with vote to the final model. Which means if the score predicted by the time parameter is below 3, we will not recommend it in the end.

3 RELATED WORK

Our project relied on this paper "Matrix Factorization Techniques for Recommender Systems" [5] as the theoretical framework. We found out that matrix factorization models are superior to classic nearest-neighbor techniques for producing product recommendations and this is also proved by our research. Generally speaking, recommender systems are based on one of the two strategies: Content filtering and Collaborative filtering.

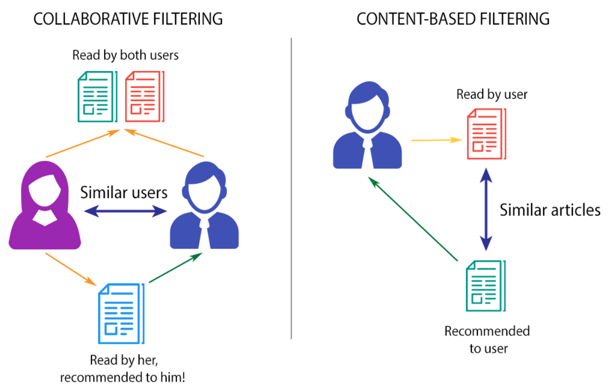


Fig 1 Content vs Collaborative Filtering

3.1 Content filtering

This approach creates a profile for each user or product to characterize its nature. For example, a movie profile could include attributes regarding its genre, the participating actors, its box office popularity, etc. The profiles allow programs to associate users to products. The paper mentioned that a known successful realization of content filtering is the Music Genome Project, which is used for the internet radio service Pandora.com. A trained music analyst scores each song in the Music Genome Project based on hundreds of distinct musical characteristics. These attributes will contribute to recommend listeners' songs. However, based on the Amazon Fine Food Database, we don't have the profile information on products nor on users. Thus, we will focus on the second method.

3.2 Collaborative Filtering

The other algorithm relies on past user behavior – for example, previous transactions or product ratings – without requiring the creation of explicit profiles. This is more suitable for our project. Collaborative filtering analyzes the relationships between users and interdependencies among products to identify new user-item associations.

The two primary areas of collaborative filtering are the neighborhood methods and latent factor models. Neighborhood methods are centered on computing and relationships between items or, alternatively, between users. This method can be categorized into two genres: user-oriented neighborhood method and product-oriented neighborhood method.

User-oriented neighborhood method: Joe likes the three movies A, B, and C. To make a prediction for him, the system finds some similar users who also like those movies, and then determines which other movies they liked.

Item-oriented collaborative filtering: this in essence just chain recommendations off a particular item. The main advantage is that it can recommend items to users who we know nothing about except for what item they may be looking at currently.

Latent factor models are an alternative approach that tries to explain the rating by characterizing both items and users on, say, 20 to 100 factors inferred from the rating patterns.

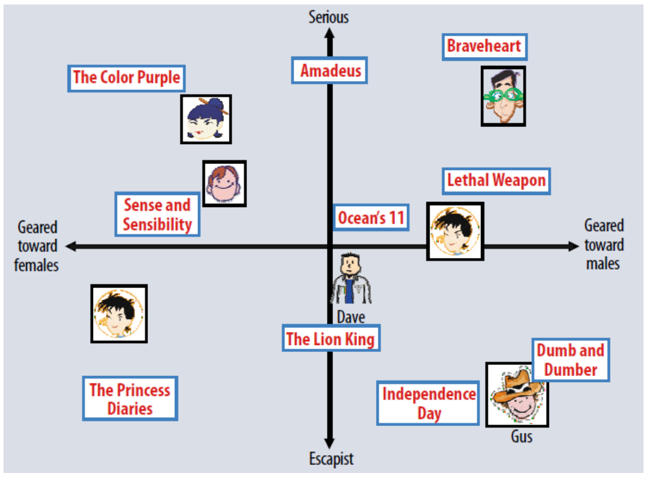


Figure 2: Latent factor approach

For our amazon fine food case, the discovered factors might measure obvious dimensions such as soft drinks, alcohol and food but also less well-defined dimensions such as how tasty is a food; or completely uninterpretable dimensions. For users, each factor measures how much the user likes foods that score high on a certain type. Figure 2 (also from the paper) illustrates the idea for a simplified example in two dimensions.

3.3 Matrix Factorization Methods

Some of the most successful realizations of latent factor models are based on matrix factorization. In its basic form, the matrix factorization characterizes both items and users by vectors of factors inferred from the rating patterns. We will discuss more about the Matrix Factorization Model in Part 4 Methodology.

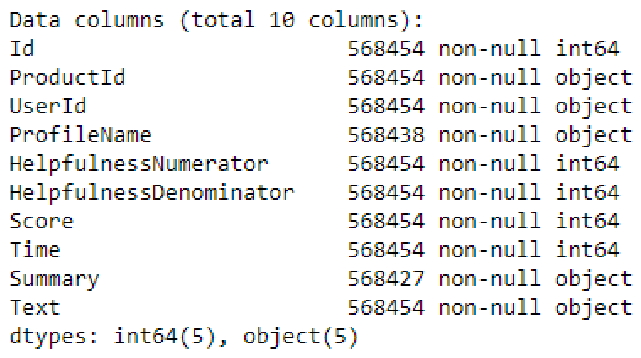
3.4 Comments and Improvement

As we mentioned in 3.1, we referred to "Matrix Factorization Techniques for Recommender Systems" [5] as our theoretical framework. We would like to verify if Collaborative Filtering based on Matrix Factorization methods is the state-of-art method for recommendation systems. Besides, we believe that it is not appropriate to omit the time factor. Whether a product is outdated is a key factor for a client to decide to purchase or not.

4 METHODOLOGY

4.1 Dataset and Pre-processing

As Fig 1 shows, our datasets have a Data frame size of 568454 * 10.



```
Data columns (total 10 columns):
Id                        568454 non-null int64
ProductId                 568454 non-null object
UserId                    568454 non-null object
ProfileName               568438 non-null object
HelpfulnessNumerator      568454 non-null int64
HelpfulnessDenominator    568454 non-null int64
Score                     568454 non-null int64
Time                      568454 non-null int64
Summary                   568427 non-null object
Text                      568454 non-null object
dtypes: int64(5), object(5)
```

Figure 3: Data Frame size

In Fig 2, we show that almost all users bought less than 60 times and almost all products have been purchased less than 200 times.

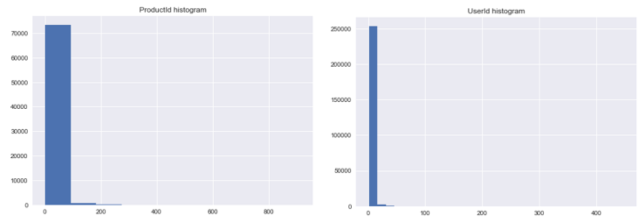


Figure 4: Histogram of *ProductId* and *UserId*

To reduce computation complexity, we filtered all items with less than 10 counts in *ProductId* and 10 counts in *UserId*.

Besides, we created two new columns *#Users*, *#Products* to capture how many times the user/product appeared in the database.

Finally, we believe that it's unlikely for us to get additional information from "text" information (it's impossible to predict one's preference only base on their text comments on different products), we decided to drop "Summary" and "Text" columns.

After feature engineering, our data consist of 64.3k rows and 5 columns (*ProductId*, *UserId*, *Score*, *#Products*, *#Users*)

4.2 Algorithms

4.2.1 Distance Based Model

In this part, we use the cosine-distance based model, and it is a little different from the traditional collaborative filtering. The traditional CF use either user similarity matrix or product similarity matrix to predict the whole user-product matrix. However, we found it was not applicable in this data set -- the MSE is way bigger than other methods. Then we tried to use a quasi-SVD approach. That is, the output matrix is the dot product of three matrix:

$$\Sigma_{m*n} = U_{m*m} \cdot \Sigma_{m*n} \cdot P_{n*n}$$

U is user similarity matrix and ***P*** is product similarity. And we use this algorithm to update the target matrix, the user-product matrix. Our train set MSE is 2.0860 and our test set MSE is 2.1480.

The advantage is that this method is quite interpretable. However, the drawbacks are also apparent: this algorithm is naive with a deficient performance. For a large data set, this algorithm need m^2+n^2 times to calculate the similarity matrix.

4.2.2 Matrix Factorization

Latent factor models are an alternative approach that tries to explain the ratings by characterizing both items and users on, say, 20 to 100 factors inferred from the ratings patterns. For products, the discovered factors might measure obvious dimensions such as candy vs drinks, or adult food vs children's; For users, each factor measures how much the user likes the product that score high on the corresponding movie factor.

Using latent factor model, we transform the way to calculate the similarity of users and products. The features become more stable and condense.

we first create two new metrics, user-latent_factor and product-latent_factor. The size are ***q*f*** and ***p*f***, ***p***, ***q*** are the total number of users and products, ***f*** is the number of latent factors. So, every element in the target matrix can be calculate as: $\hat{r}_{ui} = q_i^T p_u$.

And the target matrix is shown as $Pred = Q \cdot P^T$. Next, we need to create the objective function based on least-square method to minimize the loss:

$$\min_{q^*,p^*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

The system adjusts the model by fitting the previously observed ratings. However, the goal is to generalize those previous ratings in a way that predicts the unknown ratings. Thus, the system should avoid overfitting the observed data by regularizing the learned parameters by adding ***L2 term***. The constant ***λ*** controls the extent of regularization and is usually determined by cross-validation.

Next, we use stochastic gradient descent to optimize the objective function. The processing is:

$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$$
$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$$

Where gamma is the stochastic learning rate and ***e*** is the error term. In the iteration, when the change in loss is larger than 0, the learning rate increases by 5%; if delta-loss is smaller than 0, it means the new loss is becoming larger. The learning rate decreases by 50% so that the loss can converge.

The MF gives a very good prediction: the train set MSE is 0.03542 and the test set MSE is 0.03385. While both MSEs are equally small, thus no overfitting problem occurs.

In addition, as Fig 3 shows that the number of latent factors significantly affects the results.

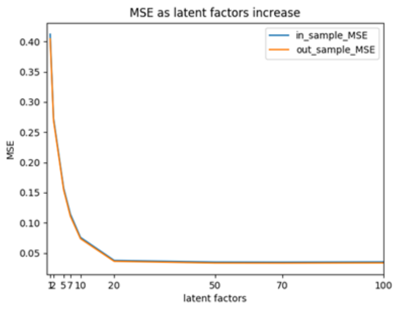


Figure 5: Latent factors and MSE

As the number of latent factors increase, the MSE decreases. And MSE converges around n=20.

4.2.2 Probabilistic Matrix Factorization

PMF is similar to MF. It is nothing more than a MF assuming the distribution of user and production are Gaussian. That is:

$$p(U|\sigma_U^2) = \prod_{i=1}^{N} \mathcal{N}(U_i|0, \sigma_U^2 \mathbf{I}), \qquad p(V|\sigma_V^2) = \prod_{j=1}^{M} \mathcal{N}(V_j|0, \sigma_V^2 \mathbf{I}).$$

And the joint distribution of user and product is the distribution of the scores.

$$p(R|U, V, \sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij}|U_i^T V_j, \sigma^2) \right]^{I_{ij}}$$

The problem comes to maximum the probability of ***U***, ***V*** based on ***R*** and variance. This transformation is based on Bayesian formula. Rewrite the function above and applying log to both sides:

$$\ln p(U, V|R, \sigma^2, \sigma_V^2, \sigma_U^2) = -\frac{1}{2\sigma^2} \sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij} (R_{ij} - U_i^T V_j)^2 - \frac{1}{2\sigma_U^2} \sum_{i=1}^{N} U_i^T U_i - \frac{1}{2\sigma_V^2} \sum_{j=1}^{M} V_j^T V_j$$
$$- \frac{1}{2} \left(\left(\sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij} \right) \ln \sigma^2 + ND \ln \sigma_U^2 + MD \ln \sigma_V^2 \right) + C$$

When optimizing this function, the standard errors are fixed, so the objective function is:

$$\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij} \left(R_{ij} - U_i^T V_j\right)^2 + \frac{\lambda_U}{2} \sum_{i=1}^{N} \parallel U_i \parallel^2 \quad + \frac{\lambda_V}{2} \sum_{j=1}^{M} \parallel V_j \parallel^2$$

Gradient descent process is the same with basic MF.

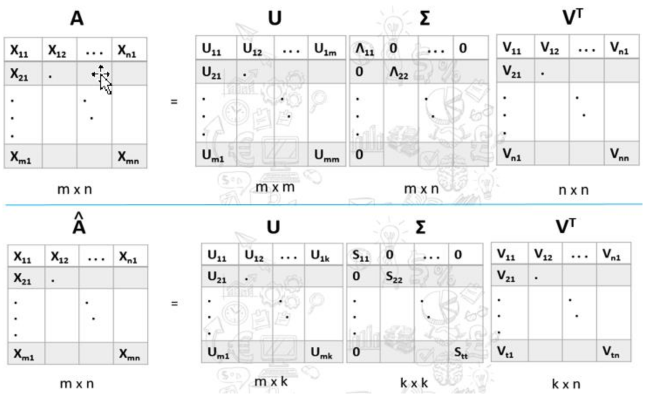
PMF does better than MF for sparse matrices. The assumption of Gaussian makes it more accurate to predict. But for our dataset, the chosen data is not sparse, so the performance of PMF is almost the same with MF. the in sample MSE is 0.036697 and the out sample MSE is 0.030156.

4.2.3 SVD

In our recommendation system, we have such a matrix which has many scores from the users to the items. We hope to predict the targeted users' score to other unevaluated items and then recommend the items with the highest five scores.

The advantage of SVD is that: users' score matrix is a sparse matrix, so we can map the original data into a Low-dimensional space and then calculate the similarity of different items. This can help us reduce calculation complexity.

$$A_{m*n} = U_{m*m} \Sigma_{m*n} V_{n*n}^T \approx U_{m*k} \Sigma_{k*k} V_{k*n}^T$$



4.2.4 Time Series

After our exploratory data analysis, we found out that the ratings are related to the time: the latter, the better.

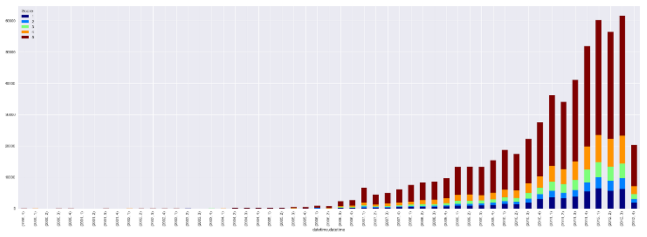


Figure 6: Score distribution based on Time

Besides, the popularity of a product is naturally linked to different time periods. iPhone 4 was a 4.8/5 product in 2010 but it will be inappropriate to recommend the same product in 2013.

We decided to convert different ProductId and *UserId* into dummy variables, and ran *RandomForestRegressor* on *ProductId*, *UserId* and *Time*. We consider time series forecasting score as a threshold: if it is below 3, we will drop the candidate.

4.2.4 Text Mining

As Fig 7 indicates, *Text* column is essential for sentimental analysis.

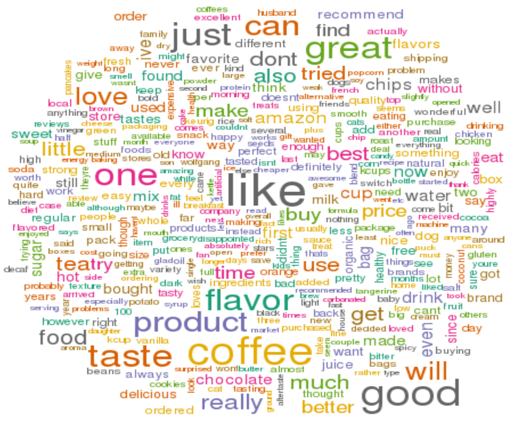


Figure 7 Word Clouds of *Text* column

To better complement our recommendation, we also use Natural Language Process methods to find 5 key words for each product.

Firstly, we group comments of each product and then tokenize them. Then, we counter the most frequent five words and drop the stopwords. (refer to 4.2.5 for a demo)

4.2.5 *Recommendation*

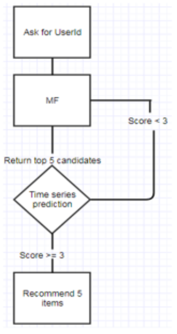


Figure 8
Recommendation Pipeline

After comparing different methods, we decided to use Matrix Factorization as our basic algorithm to predict for a certain product for a certain customer.

We select the top 5 scores as our recommendation candidates. Then we introduced the time series method to determine if the score is below 3 or not. If yes, we will drop this candidate. Otherwise, we keep the original five.

Demo: we randomly pick a *UserId* and ran our recommendation system.

Table 1: Product bought the user

ProudctId	Text
B0058AMY74	Kettle Brand Potato Chips, Unsalted, 5-Ounce Bags (Pack of 15)
B002NHYQAS	Newman's Own Organics Organic Premium Chocolate Bar, Dark 54% Cocoa, 2.25-Ounce Bars (Pack of 12)
B0012XBD7I	Kettle Brand Potato Chips, Backyard Barbeque, 8.5-Ounce Bags (Pack of 12)
B000G6MBUA	Kettle Brand Potato Chips, Sea Salt and Vinegar, 5 Ounce (Pack of 15)
B007I7Z3Z0	Lipton To Go Stix Iced Black Tea Mix, Tea and Honey, Lemon, 0.14 oz, 10Count (Pack of 12
B0012XBCZQ	Kettle Brand Potato Chips, Krinkle Cut, Buffalo Bleu, 5-Ounce Bags (Pack of 15)
B00472I5A4	Kettle Brand Jalapeno Chips, 2-ounces (Pack of24)
B004LL97EO	Kettle Krinkle Chips Zesty Ranch, 5-Ounce (Pack of 15)
B000LKXBL4	Kettle Brand Potato Chips, Sea Salt
B0054TWQMM	Nutiva Organic White Chia Seeds 12oz
B0061IUKDM	Higgins & Burke, Green Tea
B001EO5Q64	Nutiva Organic Coconut Oil, Virgin, 15 Ounce
…	…

Table 2: Product recommended by our system

ProudctId	Text
B00438XVGU	Starbucks VIA Ready Brew Coffee, Colombia, 3.3-Gram Packages,50 Count
B007JT7AEY	Clear Men 2 in 1 Shampoo, Clean and Refresh Scalp Hydrator, 12.9 Ounce
B003Z6W32E	Starbucks VIA Ready Brew Coffee, Colombia Medium, 3- Count Packets (Pack of 6)
B00451WLYI	Starbucks VIA Ready Brew Coffee, Decaf Italian Roast, Extra Bold, 3-Count Packets (Pack of 6)
B007RTR8UW	Clear Scalp and Hair Conditioner, Strong Lengths 12.7 oz

We then ran each ProductId in our time series models and conclude that all candidates meet the requirements.

Table 3: Keywords for ProductId: B00438XVGU

Keyword	coffe	cup	starbuck	via	instant	tast	brew	good	make	use
Freq	175	64	60	59	55	51	46	42	36	32

5 ELVALUATION

After using the different model, we will use the MSE and the confusion matrix to evaluate them. The performance result can be summarized as follows:

MSE	Distance-based	MF	PMF	RF	SVD
In Sample	2.0860	0.0354	0.0367	0.0727	0.3982
Out sample	2.1480	0.0339	0.0302	0.3500	0.3981

Then we compared different models on Confusion Matrix:

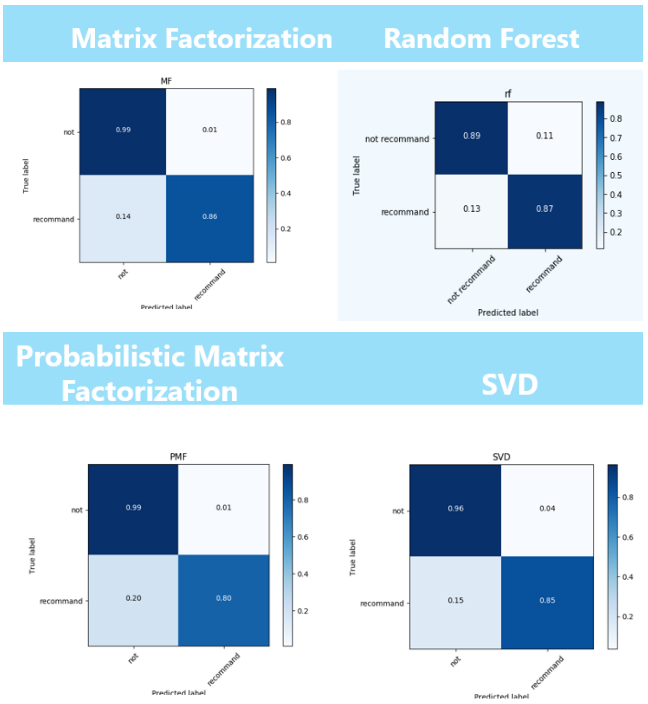


Figure 8: Confusion Matrix

Here we define articles of score > 4 as our recommendation candidates.

6 CONCLUSION

Since Matrix Factorization algorithm performed best in all the score predicting methods, we decided to use Matrix Factorization to fill all the empty cells and predict the scores for each user. We then choose the highest 5 products as our candidates. Finally, we take the time parameter into account. If the score from time prediction is below 3, we reject this candidate and choose the 6th one.

Considering the future work, we have found some limitation as follows:

1. We only extracted *UserId*, *ProductId*, *Scores* and *Time* from the original database but didn't analyze the textual information. More research on sentimental analysis for the textual data may be needed. Besides, the *HelpfulnessNumerator* and *HelpfulnessDenominator* may also help us determine who are the influencer and which information is considered as influential review by our users.

2. We assumed that if the product score is higher than 4, the user will like the product. The evaluation part can only assess the performance of rating prediction. But how to predict if users are more likely to buy these products remain uncertain. For example, two items A and B are very similar, but A is more expensive. A can't be afforded by the user even if he bought B before and rated 5. Here, high score doesn't mean more likely to be purchased.

3. To make the system perform better, we tried to include time as a factor. Although a product gets high score, we still need to consider the time feature. We believe that there are more features to be considered as well. For example, the product viewed history, the

income of the user and so on. If we take more factors into account, our recommendation system should perform better.

REFERENCES

[1] J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. WWW, 2013. The link: https://www.kaggle.com/snap/amazon-fine-food-reviews/data

[2] Salakhutdinov and Mnih, Probabilistic Matrix Factorization, NIPS 2008.The link: http://papers.nips.cc/paper/3208-probabilistic-matrix-factorization.pdf

[3] Mohammad Khoshneshin, Collaborative Filtering via Euclidean Embedding, *RecSys '10 Proceedings of the fourth ACM conference on Recommender systems*.The link: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.308.7304&rep=rep1&type=pdf

[4] Zhang et al., Collaborative User Network Embedding for Social Recommender Systems, Proceedings of the 2017 SIAM International Conference on Data Mining. The link: http://repository.kaust.edu.sa/kaust/bitstream/10754/625053/1/1.9781611974973.43.pdf

[5] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 42.8 (2009).