Recommender System

Yang Xu

ECE Department, New York University

NY, USA

yx388@nyu.edu

*Abstract*—

Through recommender system, items will be recommended to user that suit his/her personal taste. Collaborative filtering approach is widely used to predict user’s interests based on user rating history. Many techniques could be further employed to improve that approach's recommendation accuracy. In this project, we utilize Hadoop to realize some existing recommendation models. And we test the accuracy of these models through a public dataset.

Keywords—analytics, recommendation, Hadoop

# Introduction

Generally, recommender system is divided into two types: content-based approach and collaborative filtering approach. Content-based approach will create a profile for each user or product to characterize their nature, which need gathering lots of explicit external information. So it is not easy to utilize this approach in reality. In contrast, collaborative filtering approach will simply predict user’s interests by mining user rating history. It would analyze relationships between users and their related products to identify new user-item associations. The benefit of this approach is that domain knowledge is not needed to mining the relationship. The basic two methods of collaborative filtering are neighborhood method and latent factor models. But many techniques could be employed to improve those two methods' recommendation accuracy. For example, online social network relations, user interest shifting modeling, temporal pattern could be utilized to do further help. In this project, we utilize hadoop to realize some existing recommendation models. And we test the accuracy of these models through a public dataset. In the future, we will further propose our own models to do prediction.

# Motivation

Nowadays, electronic retailers and content providers offer customers a huge selection of products. Usually customers could hardly find their most interested and appropriate products by themselves when facing enormous choices. Through recommender system, appropriate items will be recommended to user that suit his/her personal taste. Thus, potentially it could enhance customer's experience and further enlarge revenue of electronic retailers.

Recommender systems are now applied in a variety of applications. The most popular ones are probably movies, music, news, books, research articles, search queries, social tags, and products in general. However, there are also recommender systems for experts, jokes, restaurants, financial services, life insurance, persons (online dating), and Twitter followers.

The accuracy of recommendation is very important to the profit of company. Thus, companies are willing to put resources on enhancing the existing system accuracy. For example, from 2006 to 2009, Netflix sponsored a competition, offering a grand prize of $1,000,000 to the team that could take an offered dataset of over 100 million movie ratings and return recommendations that were 10% more accurate than those offered by the company's existing recommender system. From academia point of view, it is not trivial to build a good recommendation system. Such analysis require accurate understanding of dataset feature and good modeling techniques. Thus, doing such research draws lots of attention from both academia and industry.

# Related Work

Among the two methods of collaborative filtering, latent factor model draws lots of attention from research society. Most of them are based on matrix factorization. Koren, etc do a survey about the latent factor models based on matrix factorization.[1] Matrix factorization models maps both users and items to a joint latent factor space of dimensionality f, so that user-item interactions are modeled as inner products in that space. Then ,the problem becomes how to find appropriate user vector and item vector based on the got user-item rating matrix. Basically, two approaches could be used to find those two vectors: stochastic gradient descent approach and alternating least squares approach. Based on the basic model , researchers could further do some extensions. One extension is to add bias factor to the model: the bias could either be the special user deviation or item deviation from the average. Or incorporating additional information sources, e.g., customer implicit feedback, user attributes to deal with cold start problem. Also, temporal dynamics, e.g., customer interest shifting, item popularity shifting could be considered into modeling. Finally, acquired data varying confidence level could also be embedded into the basic model.

Besides the above extensions, social information are also included into model to enhance recommendation accuracy. But here the simple friend relation doesn't reflect the taste similarity. A feature of "friend circles"[2] is proposed to divide social network to the domain-specific "trust circle". Social network data and rating data will be combined to do recommendation. Equal trust, expertise-based trust, trust splitting are proposed to form the social trust. This kind of idea is also used to do top-k recommendation [3]. Besides utilizing user's neighbors information in trust network in the latent feature space, author also proposes another cute idea: including missing rating information to consideration when doing recommendations.

Along with rating information, usually customers will provide one review information. The review information potentially will tell why customer will provide such rating. For example, maybe customer is satisfied with the food but not that enjoy the atmosphere too much. Thus he gives one general score 3.0 to one restaurant. By analyzing such review, the recommender system will model customer and restaurant features more accurately. Authors in [4] and [5] proposes two methodologies to utilize those information. Basically their basic idea is to combine review analysis and rating information to do recommendation. But how to combine them together efficiently to give a good result is still one open question.

# Design

Generally, our design flows consist of the following steps like figure below shows:

1) the raw JSON data set will be transferred to CSV data format. And the data will be filtered to contain only the information that our analysis needed. Then, data will be sampled into two parts: training data and test data.

2) Then, based on training data, the recommendation model will be formulated. When we formulate the model, there will be some computation work that could be paralleled. Thus, the recommendation model could be realized via Map Reduce model.

3) After recommendation model being formulated, we could predict rating for the user-item pairs in the test data using our formulated model. And we can compare the predicted rating with the real rating in our test data to assess the accuracy of the model.

In the project, we consider three popular recommendation models: matrix factorization model, item-based model and popularity model.

For matrix factorization model, as related work section stated, the rating matrix A are considered as the inner products of user-feature matrix P and item-feature matrix Q. In this project, we utilized the matrix factorization extension model that would added bias vectors. As discussed in [6], It could be solved in parallel via alternating least square (ALS):

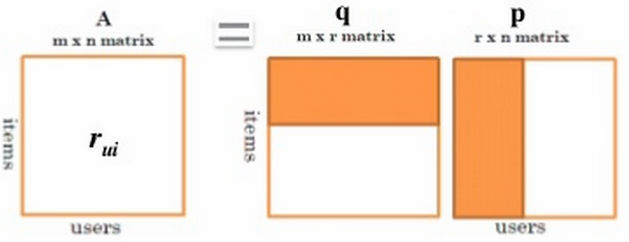
a) First initialize matrix P by assigning random number to each entry of the matrix;

b) Fix P, solve Q by minimizing the objective function;

c) Fix Q, solve P by minimizing the objective function;

d) Repeat step b) and step c) until the criterion is satisfied.

In above step b) and step c), the update of each user/item vector could be updated independently. Thus, those two steps could be realized via parallel map/reduce jobs.

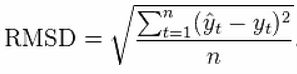


For item-based model, it belongs to the category of neighborhood method. we compute similarity between items using the rating of users who have also interacted with both items. Then, the rating of unrated item could be predicted using user's rating history and item similarity. In the experiment, we utilize Pearson Correlation coefficient [7] to measure the similarity.

For popularity model, we just calculate item's mean rating in training data to predict the rating of item in the test data. In this model, we didn't consider user's personal interest.

For item-based model and popularity model, they could also be realized in a parallel way. For example, the similarity calculation in item-based model between different items could be done independently. The mean rating calculation of different items in popularity model could also be implemented independently. In theory, those models could also be realized in Map Reduce model. But in current implementation, we didn't utilize the parallel way to realize these two models.

For the accuracy assessment, we use the classical root-mean-square deviation (RMSD) to measure the prediction accuracy. The RMSD is defined as follows:



And for this parameter, a smaller value means a better prediction result.

# Results

In our experiment, we use the dataset from yelp challenge. The yelp Challenge Dataset includes data from Phoenix,  Las Vegas, Madison, Waterloo and Edinburgh, which includes 42153 businesses, 320002 business attributes, 31617 check-in sets, 252898 users, 955999 edge social graph, 403210 tips, 1125458 reviews.

In this project, we only utilize the review rating information. We parse and filtering the raw data. We use random 90% rating information as the training data and the residual 10% rating information as the test data.

We implemented matrix factorization model using Hadoop framework. And we train our recommendation model in Amazon Elastic Map Reduce platform. For the residual two models, we implement using python. And the recommendation models are trained in our lab server. The RMSD results of our experiment are shown like below:

|  |  |
| --- | --- |
| **Approach** | **RMSD** |
| Item-based Model | 1.2389 |
| Popularity Model | 1.1927 |
| Matrix Factorization Model | 1.1557 |

Surprisingly, the simple popularity model already gives us a good result. And item-based model behaves worse than popularity model. In matrix factorization model, there are some hyper-parameters. And different hyper-parameter value combinations offers different model accuracy. We try various combinations and only the best RMSD result is listed in above forms. Result shows that matrix factorization models only behave a little better than popularity model. We think that this result could be explained via the following reason:

The dataset contains many businesses. For some businesses, people's taste doesn't vary too much. For example, people may like the most popular bar no matter what taste they are. For those kind of business, the popularity model already gives good recommendation. Thus, we should pick up the businesses that people turn to have different taste to utilize the personalized recommendation model. Through this way, we think that the recommendation accuracy could be further improved.

In our experiment, we found that actually Map Reduce platform doesn't offer us lots of time gain to build the matrix factorization recommendation model. In Amazon Elastic Map Reduce platform, it usually takes us 10 minutes to start the nodes and set up the environment. Besides, via Hadoop, the temporary result will be written to the slow-accessed hard disk. Then, the next map or reduce work will read the local file from disk to start another iteration. In the experiment, we just utilized two nodes to do Hadoop work and the dataset scale is not very big. Thus, the time of utilizing Hadoop to do such iteration modeling job is actually worse than using other parallel techniques, such as Spark or even multiprocessor programming.

# Future Work

In the analysis of experiment, we think that doing business category classification first and then doing personalized recommendation will offer us better result. We plan to try this idea in the future.

And actually in the beginning of the project, we plan to propose our recommendation model that will consider other information along with the rating information. But due to the time limit, we didn't realize it in this project. For that recommendation model, we want to combine social information, rating information and review information to generate one model that combining topic model and collaborative filtering like the following graph indicates. When doing review information, we will utilize the rating to help us better analyzing the topic of the review. We plan to implement this model in the future.

# Conclusion

In this project, we investigate some current recommender systems. Then, we implement three recommender system and test the accuracy of those models through yelp dataset. And current result shows that the matrix factorization model behaves better than item-based model and popularity model. This is a good practice for us to understand and utilize existing model. And we plan to realize and test our proposed models in the future.

##### Acknowledgment

Thanks Amazon for AWS grant support of this project. Thanks Prof. Suzanne for her kind guidance.

##### References

1. Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems.", in Computer 42.8 (2009): 30-37.
2. Yang, Xiwang, Harald Steck, and Yong Liu. "Circle-based recommendation in online social networks.", in ACM KDD, 2012.
3. Yang, Xiwang, et al. "On top-k recommendation using social networks.", in ACM Recsys, 2012.
4. McAuley, Julian, and Jure Leskovec. "Hidden factors and hidden topics: understanding rating dimensions with review text.", in ACM Recsys, 2013.
5. Diao, Qiming, et al. "Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS)." in ACM SIGKDD, 2014.
6. Zhou, Yunhong, et al. "Large-scale parallel collaborative filtering for the netflix prize." Algorithmic Aspects in Information and Management, Springer Berlin Heidelberg, 2008. 337-348.
7. Pearson Correlation Coefficient. http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio\_exports/sphilip/pear.html