Amazon Recommender System

Guicheng Wu
Xinyi Xie

March 18, 2016

Abstract

Recommender systems have played an important roles in many applications ranging from E-commerce to social media, musics, movies and news. We explore the approaches to design the recommender systems, and find out collaborative filtering(CF) related methods are the most popular ones because they have been very successful in both research and practice. In our Amazon recommender system project, we first collect and preprocess some related authoritative datasets, and build the datasets we require. Then we develop the baseline predictor, user-user based CF and item-item based CF algorithms to make prediction and recommendation. In order to compare the performance of different algorithms, we implement different similarity functions such as cosine similarity and pearson similarity for both user-user based CF and item-item based CF. Finally, we conduct experiments on our datasets using k-fold cross validation. We judge the prediction results by comparing the correct rate, mean absolute error(MAE) and root mean squared error(RMSE). Experiments shows user-user based CF based on pearson similarity function has the best performance. Also the fold size k will not significantly affect the results.

Contents

1 Introduction

Recommender systems have become very common in the past twenty years, and are applied in variety of applications. The most popular applications are probably books, movies, music, news [15] and products [16]. And recommender system has been one of most important part for many e-commerce websites such as Amazon, Netflix and Twitter, because a good recommender system can find items the specific users are interested in. It's an important way for websites to help users save time and boost users' satisfactory and fidelity. Also recommender systems can increase the number of items sold and sell more diverse items.

In general, recommender systems are software tools and techniques providing suggestions for items to be use to a user [11]. In the simplest form, recommendations are offered as ranked lists of items. In performing this ranking, recommender systems are trying to predict the most suitable items based on the users' browsing or viewing histories. In order to perform such a computational task, recommender systems collect items' preferences from their users, and the most common way is to ask users to rate the items. However, this is not a good choice for service providers because many users are not willing to rate stuffs. Therefore, we need some algorithms to automatically compute the preference of users based on the history ratings and reviewing.

There are many approaches to design the recommender systems. We can classify these systems into three broad groups.

- **Collaborative filtering.** Collaborative Filtering (CF) [12] which relies on past user behaviors such as product ratings, and does not require the explicit profiles. Notably, CF approaches require no domain knowledge and avoid the need of extensive data collection. In addition, relying directly on user behaviors allows uncovering complex and unexpected patterns that would be difficult to profile using data attributes. There are two main disciplines of CF: the neighborhood approaches and latent factor models [5].

- **Content-based filtering.** Content based filtering methods are based on a description of the item and a profile of the user's preference. In a content-based recommender system, keywords are used to describe the items and a user profile is built to indicate the type of item this user likes. For example, if a Netflix user has watched many war movies,

then recommend a movie classified in the database as having the "war" genre.

- **Hybrid recommender systems.** Hybrid approach combines collaborative filtering and content-based filtering algorithms into a recommender system. Hybrid approach could be effective when algorithms cover different aspects of the data set. For example, CF suffers from cold start(the new item has never been rated by users), but content-based methods do not.

In our project, We are trying to build an Amazon recommender system, which provides information or items that are likely to be of interest to a user, in an automated fashion. The Amazon dataset has been classified into more than twenty category: books, electronics etc. The overall dataset has more than 140 million reviews which are totally 20GB. We are going to choose several categories to perform the recommendation system since we cannot handle that much data into one matrix and it may cause overfitting problem.

Generally, our amazon recommender system is composed of two part. The first part is the prediction of rating for a specific user and a specific product which this specific user hasn't bought before. The second part is the top-K recommendation for a specific genre of products, which will find K potential products that each user mostly interest in. In this case, each user's K recommended products can be very different. Also evaluation methods will be implemented to evaluate our recommendation system.

The paper is organized as follows. The section 1 introduces the basic idea and approaches of recommender systems. The section 2 is the related work of recommender systems. Then in the section 3, we introduce the algorithms we implemented one by one: baseline predictor in the section 3.2, collaborative filtering in the section 3.3 with user-user collaborative filtering and item-item collaborative filtering. Then we implement the top K recommendation in section 3.4. In section 4, we explain the details of choosing the dataset. Also we introduce three criterion to measure the algorithm in section 5. Finally, in section 6 we use a well known dataset to validate our algorithm and in section 7 we use our algorithm on Amazon dataset and show the result.

2 Related Work

Recommendation systems are ubiquitous in applications ranging from ecommerce to social media, video, and online news platforms. Such systems help users to navigate a huge selection of items with unprecedented opportunities to meet a variety of special needs and user tastes.

For example, Carlos. A. Gomez-Uribe and Neil Hunt [1] introduced various algorithms that make up the Netflix recommendation system. The personalized video recommendation algorithm orders the subsets of videos (selected by genre) for each member in a personalized way. Different from personalized video recommendation algorithm, the goal of top-N video recommendation algorithm is to find the best few personalized recommendations in the entire catalog for each member. The Netflix has found that shorter-term trends are powerful predictors of videos that their members will watch; therefore, they have developed the trending recommendation algorithm to find the top trending videos. Moreover, the video-video similarity algorithm is used to compute a ranked list of videos for every video in their catalog.

There are many different kinds of algorithms to build recommendation systems, however, the most popular algorithm for recommendation systems is collaborative filtering. Collaborative filtering is based its predictions and recommendations on the ratings or behavior of other users in the system.

The survey conducting by Ekstrand Michael [4] aims to provide a broad overview of the current state of collaborative filtering research. In the survey, it discussed the core algorithms for collaborative filtering and traditional meaning of measuring their performance against user rating data sets. These algorithms included baseline predictor, user-user collaborative filtering, item-item collaborative filtering, dimensionality reduction, hybrid recommendation etc. It also introduced how to build reliable and accurate data sets.

George Karypis [3] presented item-based recommendation algorithms. The key steps of this algorithm are the methods used to determine the similarity between items, and also the method used to combine these similarities in order to compute the similarity between a basket of items and a candidate recommendation item. According to the paper, item-based algorithms are faster and better than traditional user-based recommendation algorithms.

Despite of its popularity, collaborative filtering based recommendation systems still have two major limits. One is the sparsity. Due to the data sparsity problem, the quality of collaborative filtering algorithm is greatly limited. Zan Huang [2] has proposed the link prediction approach for col-

laborative filtering to deal with sparsity. They adapted a wide range of link measures for making recommendations. The other limit is the scalability. For scalability problem, one possible way is to clustering the users, and then to use user clusters to derive recommendations. A.K. Jain [7] presented a taxonomy of clustering techniques, and identify crossing-cutting themes and advances.

Many adaptions for collaborative filtering have been proposed. Lee, Joonseok et al [8] assumed the rating matrix is locally low-rank which is more appropriate for real world recommendation systems. Polatidis Nikolaos [9] proposed a recommendation method that improves the accuracy of collaborative filtering and is based on multiple levels and constraints. Experiments show that their methods are both practical and effective. Based on latent factor and neighborhood models, Yehuda Koren [5] introduced some innovations to both models, and then the latent factor and neighborhood models can be smoothly merged, thereby building a more accurate combined model. Further accuracy improvements are achieved by extending the models to exploit both explicit and implicit feedback by the users.

Most recommender systems focus on analyzing patterns of interest in products to provide personalized recommendations. Another important problem is to understand relationships between products. McAuley, Julian et al [10] develop a method to infer networks of substitutable and complementary products. This method is mainly based on topic models.

For many retailers, rating data is unavailable, or unreliable because ratings are purely declarative. Personalized recommendations have to based on customers purchase history. With this consideration, Pradel Bruno [6] presented an experimental evaluation of various collaborative filtering algorithms on a read-world store dataset of purchase history from customers.

3 Implemented algorithms

3.1 Preprocess Data

Before we actually implement our algorithms, we need to preprocess our Amazon data set. Generally, there are two steps:

1. **Filter data.** For each piece of data, the original data set contains many attributes such as 'userID', 'itemID', 'rating', 'userName' and

reviewName'. In our project, We only require 'userID', 'itemID' and 'rating', so we filter the unnecessary attributes.

2. **Construct user-item matrix.** We build the user-item matrix R as $U \times I$ where the rows represent the users U and columns represent the items I. $r_{u,i}$ represent the rating value of item i rated by user u.

3.2 Baseline Prediction

To get started, we develop the baseline predictor. We denote the baseline prediction for user u and product i as $b_{u,i}$. Baseline predictor method is used as a baseline for other algorithms. Also the method is very useful for building non-personalized baselines comparing to personalized algorithms. We first implement a simplest baseline method and then optimize it.

The simplest baseline is to predict the average ratings over all ratings in the dataset: $b_{u,i} = \mu$ (Where μ is the overall average rating). And enhancement can be made by incorporating two other parameters as follows.

$$b_{u,i} = \mu + b_u + b_i \tag{1}$$

Where μ is the overall average rate of all the user and product pairs, b_u and b_i are user u and product i baseline predictors, which can be defined by using average offsets:

$$\begin{aligned} b_u &= \frac{1}{|I_u|} \sum_{i \in I_u} (r_{u,i} - \mu) \\ b_i &= \frac{1}{|U_i|} \sum_{u \in U_i} (r_{u,i} - b_u - \mu) \end{aligned} \tag{2}$$

In the equation, I_u represents the set of items which are rated by user u; U_i represents the set of users which rate item i. And $r_{u,i}$ is the rating value of item i rated by user u.

3.3 Collaborative Filtering

The main task of Collaborative Filtering (CF) are prediction and recommendation [13, 14]. It works by building a database of ratings for items by users. The Collaborative filtering process is shown in Figure 1. Nowadays CF has

been very sussessful in both research and practice, and in both information filtering applictions and E-commerce websites.

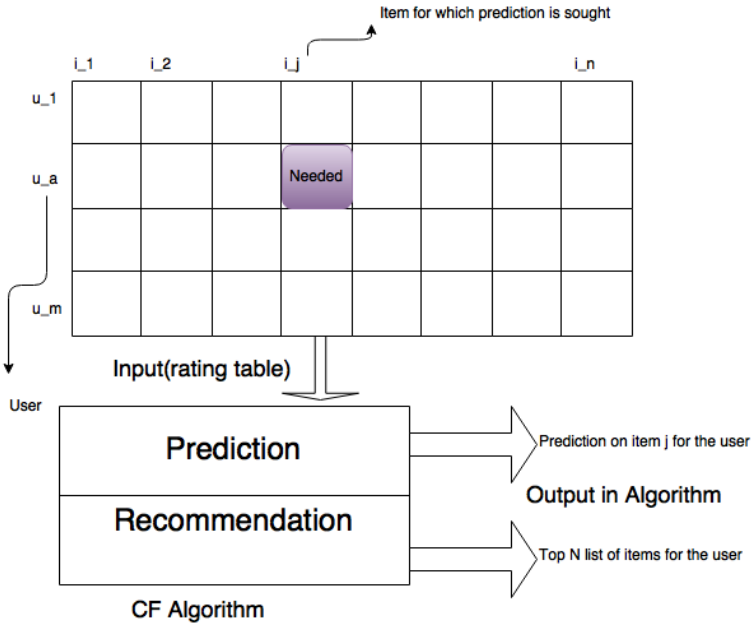


Figure 1: Collaborative filtering process

In this section, we are going to implement Collaborative Filtering (CF) which includes user-user CF, item-item CF. User-user CF is a very intuitive algorithm that based on the assumption that the past rating behavior of other users is similar to that of the current user. Thus we can use their ratings on other items to predict what the current user will like. One key element is to find a function to measure similarity. We implement cosine similarity which is measured by the cosine distance between two rating vectors and unknown ratings are considered to be 0. Also we'd like to implement Pearson correlation, constrained Pearson correlation to compare these three different similarities function. However, the deficiency of user-user CF is, while effective, suffering from scalability problems as the numbers of the users are growing. So we would like to implement item-item CF to compare.

Item-item CF also uses similarities that between the rating patterns of iterm rather than using similarities between users rating behaviors. Also, cosine similarity, Pearson correlation are implemented.

3.3.1 User-User Based Collaborative Filtering

To compute predictions or ratings for a user u, user-user CF uses similarity metric s to compute a neighborhood N of neighbors of u. This comes from a simple thought that finding the similarity rating patterns between people to group them together. For example, Cindy may loves romantic movies and hate action movies which is the same with Wendy. And what we do is to group Cindy and Wendy together to use the rest of the group member to predict Cindy's opinion to movie Die Harder.

Once we have the neighborhood information, we can compute the prediction as follows

$$p_{u,i} = \bar{r}_u + \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in N} |s(u,u')|} \tag{3}$$

where $\bar{r}_u$ is the average of ratings of user u, $s(u,u')$ is the similarity between user u and u'.

We implement three different similarity: Pearson correlation, constrained Pearson correlation, cosine similarity. First, consider Pearson correlation:

$$s(u,v) = \frac{\sum (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum (r_{v,i} - \bar{r}_v)^2}} \tag{4}$$

Cosine similarity is a vector-space approach rather than a statistical approach. In the vector-space, if two vectors v and w is the same, cosine values is 1. If two vectors are orthogonal, cosine value goes to infinity. This can be interpret as their l_2 norms in discrete space:

$$s(u,v) = \frac{r_u^T * r_v}{\|r_u\|_2 \|r_u\|_2} \tag{5}$$

User-User based CF has been very successful in the past, but our evaluation on Amazon data set has revealed some potential challenges such as

- **Sparsity.** In Amazon data set, we have 915,448 pieces of rating information, and it includes 531,891 users and 64426 items. This means each user has only rated 1.72 items at average. In this case, user based CF algorithms may be unable to predict or recommend items for a particular user, because it is possible that there are no similar users at all since the users have rated only 1-2 items. As a result, the accuracy of recommender systems may be poor.

- **Scalability.** User based CF algorithm require computation that grows with both the number of users and the number of items. With millions of users and items, the computation time of user based algorithm would be very long.

3.3.2 Item-Item Based Collaborative Filtering

Rather than matching the user to the similar users, item-item based CF matches the target items with each of the previous user's purchased and rated items. After matching, we can find the most similar items. Then Using similar items to predict the rate of the unknown item. If two items tend to have the same users like or dislike them, then these two items are considered similar and users are expected to have similar preferences for similar items. Item-item based CF generates predictions by using the user's own ratings for other items with those item's similarities to the target item, rather than other users' ratings and user similarities as in user-user based CF.

In its raw form, item-item based approach firstly need to determine the K most similar items for a given item. There are various ways such as pearson similarity we used in user-user based collaborative filtering. In our project, we use the common method cosine similarity, pearson similarity, and conditional probability similarity to find similar items.

The cosine similarity equation is shown as follows.

$$s(i,j) = \frac{r_i^T * r_j}{\|r_i\|_2 \|r_j\|_2} \tag{6}$$

In the equation, r_i and r_j are both item rating vectors. The more two items are similar, the higher the value will be.

The pearson similarity equation is shown as follows.

$$s(i,j) = \frac{\sum_{u \in U_i \bigcap U_j} (r_{u,i} - \bar{r_i})(r_{u,j} - \bar{r_j})}{\sqrt{\sum_{u \in U_i \bigcap U_j} (r_{u,i} - \bar{r_i})^2} \sqrt{\sum_{u \in U_i \bigcap U_j} (r_{u,j} - \bar{r_j})^2}} \tag{7}$$

In the equation, $\bar{r_i}, \bar{r_j}$ are the average rates of item i and item j. U_i, U_j are the user set who bought item i and the user set who bought item j.

The basic Conditional probabilities based similarity function is $s(i,j) = Pr_B(j \in B | i \in B)$ where B is the user's purchase history. With a scaling parameter α to a balance for frequently occurring items The conditional

probability equation becomes

$$s(i,j) = \frac{Freq(i \bigcap j)}{Freq(i)(Freq(j))^{\alpha}} \tag{8}$$

where $Freq(i \bigcap j)$ represents the number of users who bought item i also bought item j. $Freq(i)$ represents the numbers of users who buy item i, similar for $Freq(j)$. α serves as a damping factor to compensate for $Pr_B(j \in B|i \in B)$ being high solely due to a high margin probability $Pr(j \in B)$.

After collecting a set S of items similar to i, we can generate predictions using a weighted average. The prediction rate $p_{u,i}$ is calculated as follows.

$$p_{u,i} = \frac{\sum_{j \in S} s(i,j) * r_{u,j}}{\sum_{j \in S} |s(i,j)|} \tag{9}$$

In the equation, $s(i,j)$ represents the similarity of item i and j, and $r(u,j)$ represents the rating of item j given by user u.

3.4 Top K recommendation

In this section, we are going to implement top k-recommendation method. Top K recommendation method is to recommend K items to a particular user. The idea is for one user u and all the other item i, which u has not bought yet or rated yet, predict ratings. Then order these ratings and choose the top k item to recommend to the user u.

Since top K recommendation method is based on predicting rate for a particular user and a particular item. We can resuse our previous implemented prediction algorithms user-user based CF and item-item based CF. Different prediction algorithms may result in different recommendations.

4 Dataset

We choose two dataset. One is a benchmark dataset so that we can validate our algorithms. The other dataset is what we want to experiment on. In the beginning of this project, we use the second one and many problems occur, such as huge sparse matrix resulting no neighborhood for collaborative filtering algorithm. Therefore, we found the movie dataset which is relatively good, in other word, existing neighborhood for most of the users and items.

4.1 Movielens data set

Movielens is one famous movie data set that used to valid algorithms. It is collected for collaborative filtering research, and consists of 1 million of ratings from 6040 users and 3952 movies. Therefore, on average, each user and each movie have at least ten data for us to compare. The data set can be download from http://grouplens.org/datasets/movielens/.

4.2 Amazon data set

Our Amazon data set are collected by Professor Julian McAuley from UCSD. It contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014. Considering the running time of the whole data set is extremely large , we have to only experiment on several specific category product data sets. The Amazon data set is sourced from http://jmcauley.ucsd.edu/data/amazon/links.html.

5 Evaluation

One may wonder why we need an evaluation method. There are thousands of algorithms and we need a unified classification method to choose a 'good' one from these algorithms. Herlocker has analyzed the methods to evaluate the collabrotive fitering recommender system[17]. The evaluation of recommender systems are playing an important role in recommender systems research. You can find that the Netflix Challenge rewards are given if the team can improve RMSE of 10%.

We are using several measurement in our evaluation systems. The very straightforward method is mean absolute error (MAE), sometimes also called absolute deviation. If there are n numbers of ratings r_i and p_i are corresponding prediction ratings. The MAE is defined as

$$\frac{1}{n}\sum_i |r_i - p_i|.$$

MAE means the deviation from the mean. The deficient of the MAE is that it can't compare with different range. For example, there are two dataset which MAE 0.7: one is from 1 to 10 and the other one is from 1 to 20.

To tackle this deficiency, normalized mean absolute error (NMAE) is introduced to measure MAE by range. Denoting the range as $[r_{min}, r_{max}]$, we

have MAE/$(r_{max} - r_{min})$. The deficiency of this measurement is obvious: it can not precise interpret the original ratings scale, in other words, MAE.

Thus, we use the most popular and more widely used root mean squared error (RMSE) which is defined as follows:

$$\sqrt{\frac{1}{n}\sum(p_i - r_i)^2}.$$

Also, it can be divided by $r_{max} - r_{min}$ to normalize for the rating scale like NMAE.

We will compute MAE and RMSE to compare the same dataset. Mathematically, they are two different norm l_1 and l_2. Both of them are good measurement to errors with different emphasis. l_1 accumulates the distance from the average of every sample. And l_2 penalize more for a long distance. For example, 2 and 4 has a 4 penalization comparing with 2 and 3 for different norm.

6 Validation on movielens Dataset

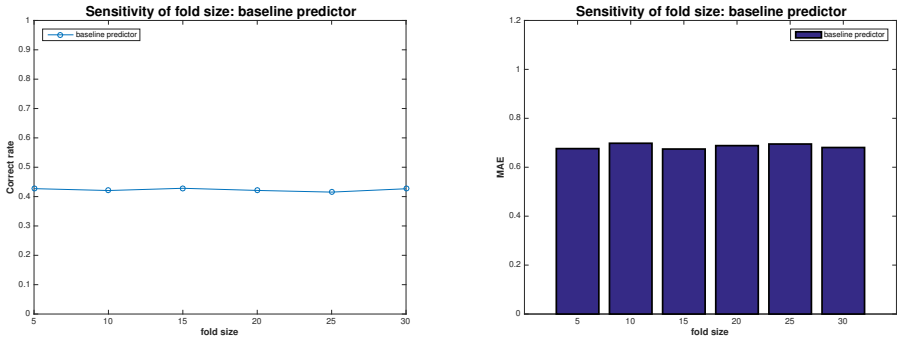
Firstly, We are going to test our algorithm on a well-formed dataset from movielens.org.

We evaluate the movielens dataset with all our implemented algorithms including baseline predictor, user-user based CF and item-item based CF. The evaluation technique we use is k-fold cross validation. The results are demonstrated in table 1.

Movielens prediction results				
Algorithm	Parameters	Correct rate(%)	MAE	RMSE
Baseline	fold=5	42.71	0.6761	0.9033
	fold=10	42.07	0.6980	0.9606
	fold=15	42.83	0.6744	0.8984
	fold=20	42.10	0.6883	0.9295
	fold=25	41.53	0.6949	0.9329
	fold=30	42.67	0.6806	0.9167
User-based CF	neighborSize=1	38.34	0.753	1.064
	neighborSize=5	46.71	0.603	0.749
	neighborSize=10	51.89	0.519	0.592
	neighborSize=15	46.77	0.615	0.781
	neighborSize=20	49.84	0.544	0.631
	neighborSize=25	54.22	0.542	0.668
	neighborSize=30	44.37	0.619	0.755
Item-based CF	neighborSize=1	27.84	1.184	2.446
	neighborSize=5	32.81	0.912	1.473
	neighborSize=10	34.48	0.891	1.442
	neighborSize=15	33.81	0.859	1.287
	neighborSize=20	30.48	0.966	1.597
	neighborSize=25	27.82	0.990	1.061
	neighborSize=30	27.43	1.061	1.878

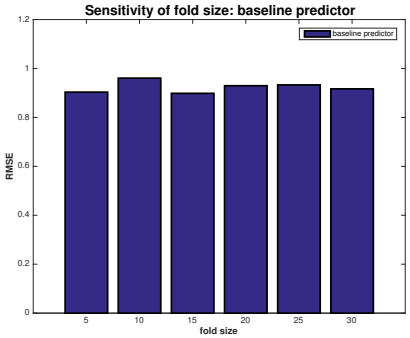
Table 1: Movielens prediction Resuts

The baseline predictor's evaluation results are shown in Figure 2. The correct prediction rate is about 42.0%, the mean average error(MAE) and root mean square error(RMSE) are about 0.68 and 0.92. We can conclude that the fold size of baseline predictor will not significantly affect the results of prediction.



(a) Correct rate

(b) MAE



(c) RMSE

Figure 2: Movielens dataset: sensitivity of fold size on baseline predictor

For the item-item based CF and user-user based CF, we valid these two algorithms based on cosine similarity function, and we find that the results change significantly as the neighborhood size changes. We compare these two algorithms' performance by graphs. In Figure 3, we compare the correct prediction rates with the change of neighborhood size. From Figure 3, we can see that user-user CF has a better performance than item-item CF. The user-user CF has the highest correct prediction rate 54.2% when the neighborhood size is 25. The item-item CF has the highest correct prediction rate 34.5% when the neighborhood size is 10. Intutively, we can see that item-item CF have larger MAE and RMSE than user-user CF from Figure 4.

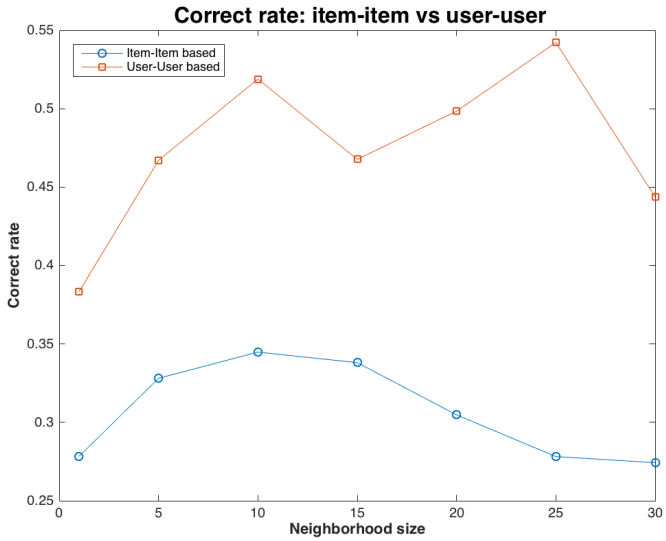
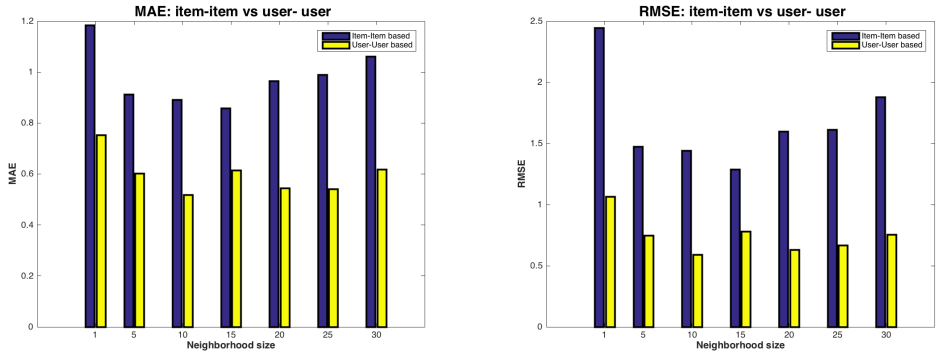


Figure 3: Movielens dataset prediction correct rate: item-item vs user-user



(a) MAE

(b) RMSE

Figure 4: Movielens dataset prediction MAE and RMSE: item-item vs user-user

7 Amazon Dataset Experiment

- **Result**

1. **Amazon Dataset Scale**
   We tested two dataset of Amazon: baby products and office products. The baby products dataset has 915,448 ratings with 531,891 users and 64,426 products. The larger one is about office products and has 1243,186 ratings with 909,314 users and 130,006 products.

2. **Parameters**
   We use cross validation to compute the correct rate, MAE and RMSE. The most common choice of validation for machine learning algorithms is tenfold cross validation. However, our dataset are too large to run a tenfold validation (it may cost days to have one outcome). Therefore, we choose to compute k fold just once and make k 'acceptable' large, which means we can compute the result in a reasonable time. Thus, we just random pick sample from the dataset as our testing data. The other remaining dataset is our training data. On the other hand, we realize that different k may influence the result. Thus we also compute different fold for baseline method to check whether k matters.

3. **Details for Each Table and Figure**
   In Table 2 and Table 3, we compress two parameters into one column fold size and neighborhood size. For CF method, we use the same number of fold and only change the neighborhood size to check the influence of neighborhood size. And for baseline method, similarity function is not available so we just leave the first column. Also, concerning about the influence of the fold size to the result, we compute different fold size in Table 3. And in Figure 5 and Figure 6, the data comes from the Table 3 to have a clear view of the change of different methods for different parameters.

- **Critical Evaluation**

1. **Does fold size matters?**
   In the Figure 7, we can see that the correct rate changes very little for different fold size. In our test, we randomly choose the sample and it is large enough for validation. In statistical, this is simple

random sampling method which has the assumption that every data is representative sample in the dataset. In our dataset, we choose one category of Amazon products which means the users have already show some similarities. Thus it is reasonable to believe each of them is representative of their group.

2. **Does neighborhood size matters?**
   Neighborhood size is the essence of the CF method because it computes how many users or items are alike. This parameter is hard to choose in practical so we emphasize the importance in our experiment. And we notice that the neighborhood size influence not only the result but also the computation time. We believe that one can learn a better neighborhood size from the result in practical situation before actually run the recommender system.

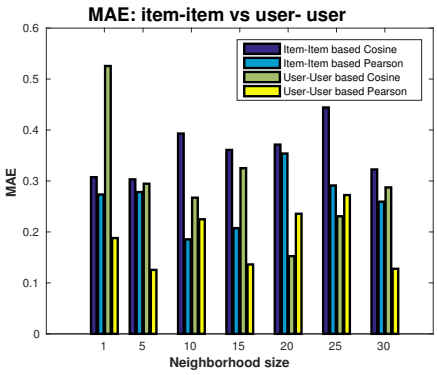
3. **Difference between cosine and Pearson Similarity**
   In Table 2, when we fix the neighborhood size and compare the similarity function. We can see that Pearson correlation outperform Cosine. While in Table 3, sometimes cosine outperform Pearson correlation and sometimes cosine outperform Pearson correlation. Thus, it hard to tell which one is better to use. In terms of similarity, cosine is better than Pearson correlation. And in terms of practical use, we believe that should depend on dataset.

Baby dataset prediction results					
Algorithm	Similarity	Fold/n	Correct rate	MAE	RMSE
Baseline	-	2000	0.743142	0.34239	0.408978
Item-based CF	Cosine	$n = 1$	0.6973	0.6078	1.5327
		$n = 5$	0.6946	0.4968	1.0689
		$n = 10$	0.6899	0.5024	1.0313
		$n = 15$	0.6357	0.5329	1.0136
		$n = 20$	0.6755	0.5694	1.2878
		$n = 25$	0.72	0.4463	0.9053
		$n = 30$	0.6491	0.6623	1.6447
	Pearson	$n = 1$	0.7048	0.4678	0.9834
		$n = 5$	0.7472	0.4154	1.1964
		$n = 10$	0.7767	0.2969	0.4727
		$n = 15$	0.6329	0.5254	0.9189
		$n = 20$	0.7681	0.2979	0.5532
		$n = 25$	0.7467	0.3467	0.5600
		$n = 30$	0.7292	0.3468	0.5273
User-based CF	Cosine	$n = 1$	0.6623	0.4870	0.8636
		$n = 5$	0.6723	0.4350	0.6836
		$n = 10$	0.6296	0.4568	0.6543
		$n = 15$	0.6976	0.3805	0.5561
		$n = 20$	0.7284	0.3642	0.5617
		$n = 25$	0.7547	0.3082	0.4465
		$n = 30$	0.7097	0.3656	0.5376
	Pearson	$n = 1$	0.8855	0.1687	0.3012
		$n = 5$	0.8176	0.2353	0.3882
		$n = 10$	0.8192	0.2203	0.3107
		$n = 15$	0.8693	0.2549	1.3791
		$n = 20$	0.8246	0.2339	0.4211
		$n = 25$	0.8314	0.2442	0.6395
		$n = 30$	0.8477	0.1827	0.2538

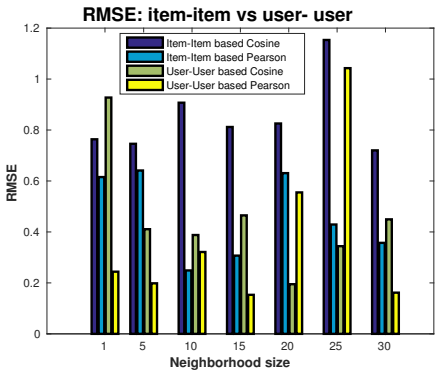
Table 2: Baby dataset prediction Resuts for baseline method, user-based CF and item-based CF. n denotes neighborhood size.

Office dataset prediction results					
Algorithm	Similarity	Fold/n	Correct rate	MAE	RMSE
Baseline	-	1000	0.835833	0.192581	0.257301
	-	1500	0.842822	0.198020	0.289604
	-	2000	0.833076	0.213292	0.315301
	-	2500	0.855670	0.171134	0.232990
	-	5000	0.868996	0.139738	0.157205
Item-based CF	Cosine	$n = 1$	0.840137	0.3075	0.7636
		$n = 5$	0.838723	0.3032	0.7459
		$n = 10$	0.7677045	0.3931	0.9072
		$n = 15$	0.7791995	0.3609	0.8119
		$n = 20$	0.834807	0.3714	0.8256
		$n = 25$	0.7667025	0.4442	1.1534
		$n = 30$	0.8141025	0.3226	0.7201
	Pearson	$n = 1$	0.850427	0.273504	0.615385
		$n = 5$	0.826613	0.278226	0.641129
		$n = 10$	0.846154	0.185520	0.248869
		$n = 15$	0.829876	0.207469	0.307054
		$n = 20$	0.776923	0.353846	0.630769
		$n = 25$	0.773946	0.291188	0.429119
		$n = 30$	0.785714	0.259398	0.357143
User-based CF	Cosine	$n = 1$	0.632479	0.5256	0.9274
		$n = 5$	0.755187	0.2946	0.4108
		$n = 10$	0.788793	0.2672	0.3879
		$n = 15$	0.736626	0.3251	0.4650
		$n = 20$	0.868644	0.1525	0.1949
		$n = 25$	0.817814	0.2308	0.3441
		$n = 30$	0.781377	0.2874	0.4494
	Pearson	$n = 1$	0.836000	0.1880	0.2440
		$n = 5$	0.902834	0.1255	0.1984
		$n = 10$	0.819277	0.2249	0.3213
		$n = 15$	0.872340	0.1362	0.1532
		$n = 20$	0.847909	0.2357	0.5551
		$n = 25$	0.836576	0.2724	1.0428
		$n = 30$	0.889362	0.1277	0.1617

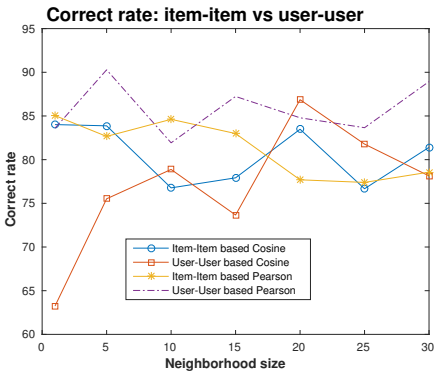
Table 3: Office product dataset prediction resuts for baseline method, user-based CF and item-based CF. For baseline method, we compare the different fold size because it does not have neighborhood size. n denotes neighborhood size .



(a) MAE using CF

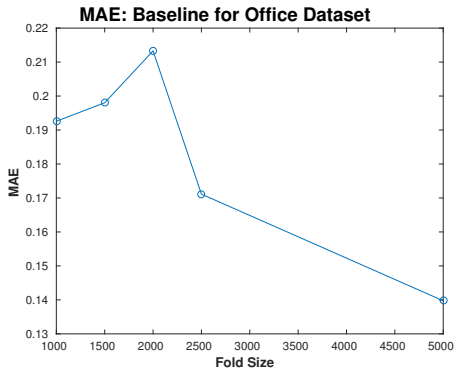


(b) RMSE using CF

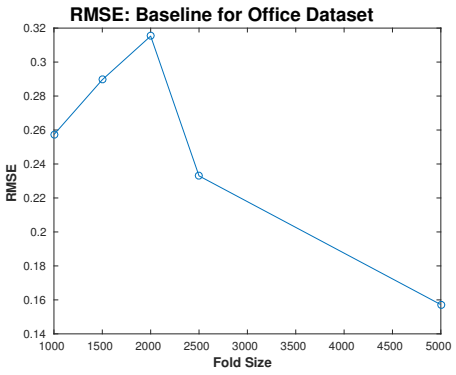


(c) Correct Rate using CF

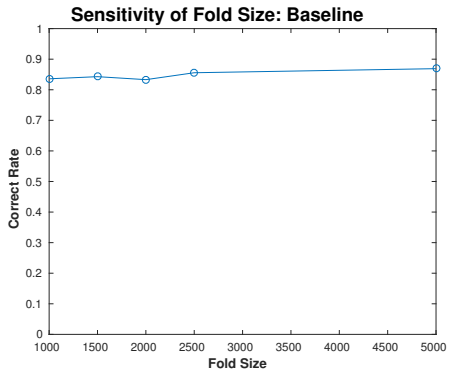
Figure 5: Office Product Dataset: x axis denotes the neighborhood size while y axis is: (a) MAE comparison between user-based and item-based using different similarity function; (b) RMSE comparison between user-based and item-based using different similarity function; (c) Correct rate comparison between user-based and item-based using different similarity function.



(a) MAE using Baseline



(b) RMSE using Baseline



(c) Correct Rate using Baseline

Figure 6: Office Product Dataset using baseline method: x axis denotes the fold size while y axis is: (a) MAE; (b) RMSE; (c) Correct rate

8 Conclusion

Amazon recommender system project is both interesting and practical for us. There have been hundreds of papers related to this area, so we have opportunities to get many novel ideas from these papers. During the process of conducting this project, we learn many interesting points from our own experiments. For example:

1. **The dataset itself significantly affects the results.**
   For the same algorithm such as user-user based CF, the prediction results can be very different in different datasets. We first conduct

the experiments on the famous movielens dataset, the average correct rate is 47.45%, the average MAE is 0.5993, and the average RMSE is 0.7486. However, for the Amazon baby-product category dataset, the average correct rate is 69.35%, the average MAE is 0.3996, and the average RMSE is 0.6148. One of the possible reasons behind that could be different datasets have different sparsity. And sparsity will strongly affect the effectiveness of similarity functions. As a result, sparsity affects the results.

2. **The similarity function really counts.**
   We develop several similarity functions for both user-user based CF and item-item based CF. We can see from our results(the detail are in section 7) that different similarity functions of the same algorithm result in very different prediction results.

3. **The neighborhood size parameter of similarity function affects both prediction results and computation time.**
   Similarity functions are the most important part of collaborative filtering algorithm, and the setting of neighborhood size in similarity function will not only affect the prediction results, but also the computation time. The larger the neighborhood size, the longer the computation time. However, it does not mean the larger the neighborhood size, the better the results will be. For the cosine similarity function in the user-user based CF, the neighborhood size of 20 has the best performance.

4. **The computation time is a big deal.**
   For recommender system, computation time is a big concern. With millions of users and items, the computation time of finding similarity users or items would be very long.

5. **Different situations should use different algorithms.**
   Different situations or different applications may use different algorithms. For example, if there are more users than items in the dataset, then item-based CF would cost less time. Otherwise, user-based CF would cost less time.

6. **The simplest method may work.**
   The baseline predictor algorithm is the simplest algorithm in recom-

mender system, but it does not mean it performs bad in all situations. Actually, the baseline predictor work well in the movielens dataset.

During the process the working on the Amazon recommender system, we do come out many ideas. Two of the ideas are listed:

1. **Recommend the substitutable and complementary items.** Instead of recommending the same category of product, we may recommend the substitutable and complementary items. For example, when a user in an online store is examining t-shirts she/he should receive recommendations for similar t-shirts, or otherwise jeans, sweatshirts, and socks, rather than (say) a movie even though she may very well be interested in it. From these relationships we can construct a product graph, where nodes represent products, and edges represent various types of product relationships.

2. **Image-based recommendations on styles and substitutes.** We are interested in the relationships between the appearances of pairs of items, and particularly in modeling the human notion of which objects complement each other. There are a range of situations in which the appearance of an object might have an impact on the desired appearance of another. For example, 'which T-shirt matches a specific pair of black shoes', maybe the white T-shirt could be a good match, but the red one is not.

References

[1] Gomez-Uribe, Carlos A., and Neil Hunt. "The Netflix Recommender System: Algorithms, Business Value, and Innovation." ACM Transactions on Management Information Systems (TMIS) 6.4 (2015): 13.

[2] Huang, Zan, Xin Li, and Hsinchun Chen. "Link prediction approach to collaborative filtering." Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries. ACM, 2005.

[3] Karypis, George. "Evaluation of item-based top-n recommendation algorithms." Proceedings of the tenth international conference on Information and knowledge management. ACM, 2001.

[4] Ekstrand, Michael D., John T. Riedl, and Joseph A. Konstan. "Collaborative filtering recommender systems." Foundations and Trends in Human-Computer Interaction 4.2 (2011): 81-173.

[5] Koren, Yehuda. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008.

[6] Pradel, Bruno, et al. "A case study in a recommender system based on purchase data." Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2011.

[7] Jain, Anil K., M. Narasimha Murty, and Patrick J. Flynn. "Data clustering: a review." ACM computing surveys (CSUR) 31.3 (1999): 264-323.

[8] Lee, Joonseok, et al. "Local collaborative ranking." Proceedings of the 23rd international conference on World wide web. ACM, 2014.

[9] Polatidis, Nikolaos, and Christos K. Georgiadis. "A multi-level collaborative filtering method that improves recommendations." Expert Systems with Applications 48 (2016): 100-110.

[10] McAuley, Julian, Rahul Pandey, and Jure Leskovec. "Inferring networks of substitutable and complementary products." Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2015.

[11] Ricci, Francesco, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. Springer US, 2011.

[12] Goldberg, David, et al. "Using collaborative filtering to weave an information tapestry." Communications of the ACM 35.12 (1992): 61-70.

[13] Breese, John S., David Heckerman, and Carl Kadie. "Empirical analysis of predictive algorithms for collaborative filtering." Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1998.

[14] Sarwar, Badrul, et al. "Item-based collaborative filtering recommendation algorithms." Proceedings of the 10th international conference on World Wide Web. ACM, 2001.

[15] Resnick, Paul, et al. "GroupLens: an open architecture for collaborative filtering of netnews." Proceedings of the 1994 ACM conference on Computer supported cooperative work. ACM, 1994.

[16] Linden, Greg, Brent Smith, and Jeremy York. "Amazon. com recommendations: Item-to-item collaborative filtering." Internet Computing, IEEE 7.1 (2003): 76-80.

[17] Herlocker, Jonathan L., et al. "Evaluating collaborative filtering recommender systems." ACM Transactions on Information Systems (TOIS) 22.1 (2004): 5-53.