Netflix Challenge: A Study in Recommender Systems

Wenbo Qiao
Department of Computer Science & Engineering, Faculty of Engineering, Michigan State University
517-896-4239, 48823
qiaowb@gmail.com

Yuan Zhang
Department of Computer Science & Engineering, Faculty of Engineering, Michigan State University
517-899-5310, 48823
nick.zhangyuan@gmail.com

Rujira Achawanantakun
Department of Computer Science & Engineering, Faculty of Engineering, Michigan State University
517-974-4431, 48823
rujira.a@gmail.com

ABSTRACT
The Netflix Prize is an open competition for the best system recommender algorithms to predict user ratings for movies, based on former ratings. The rating prediction is an important knowledge to recognize subscriber's favorite movie styles. Based on this information, the company can recommend new movies to users. In this paper, we propose several approaches to predict user ratings based on the well-known k-nearest neighbor and singular value decomposition algorithms. The time dependence factor and blending technique were also integrated with those algorithms to improve accuracy of predictions. By doing so, we have successfully achieved a RMSE of 0.8932, which is a 15% progress.

Keywords
Netflix prize, k-nearest neighbors (KNN), singular value decomposition (SVD), recommends system, RMSE

1. INTRODUCTION
In October, 2006 Netflix released this large movie rating dataset and challenged the data mining, machine learning and computer science communities to develop systems that could beat the user rating prediction of Cinematch. The Netflix company [1] has amassed a collection of 100,000 titles and approximately 10 million subscribers. The company has more than 55 million discs and, on average, ships 1.9 million DVDs to customers each day. The rating prediction is an important knowledge to recognize scriber's favorite movies style. Based on this information, the company can predict users' favorite movies, ans they also can provide the set of recommended movies when user sing in the website. The data mining task is necessary in this case, since the data is large and the correlation between attributes and records has to be considered in order to gain better results of the prediction.

In this paper, we propose some techniques to predict user ratings, using combined methods between time dependence factor and two well-known algorithms, the K-Nearest Neighbor (KNN) and the Singular Value Decomposition (SVM). First, we apply the KNN and the SVM individually. Then we integrate time factor with previous results and achieve an improved RMSE. Furthermore, we also apply blending techniques on KNN, SVM and both their time-dependence models to increase the prediction accuracy. The measurement we use to evaluate our empirical predictions is root mean square error (RMSE). The experiment outputs show a significant improvement in the accuracy by combining multiple models together.

2. PROBLEM STATEMENT
Recommender systems provide user-oriented suggestions about items (e.g., movies, books…) and are useful for both users, simplifying the research of items, and service providers, improving their services. Nowadays in the web-based e-commerce applications, recommender systems can be used to augment the visibility of items in very large catalogs by providing users with a short list of items which they are certain to enjoy.

In the past years, the Netflix contest has boosted the research on recommender systems, mainly because of the one million dollars prize and the challenging size of the dataset.

The original Netflix dataset consists of 5-star ratings on 17770 movies and 480189 anonymous users. It was collected by Netflix in a period of approximately 7 years. In total, the number of ratings is 100480507. This is for the training set.

To test your result of predictions of ratings, there are two sub datasets. One probe set whose size is 1408395 and the qualifying set (size: 2817131). To get qualified for the Grand prize, a RMSE of 0.8563 should be achieved on the quiz set. The quiz ser is an unknown 50% random subset of the qualifying set. The judging criteria for winning the Netflix Grand Prize is the four digits rounded RMSE score on the test set (remaining 50%). The probe set has equal statistical properties as the qualifying set. Furthermore it is used as a hold-out set during the competition.

In this project the biggest problem is the huge dataset we have to process. The original one requires about 2G uncompressed disk spaces. For this reason, we've created a sample of the data set including about 880,000 ratings. The Netflix Prize subset is distributed in exactly the same format as the original dataset. If we want apply the original larger dataset, we can easily implement that with our algorithm.

The second problem we are facing is that currently there is no software like WEKA or CLUTO that we can easily process our data with. We have to program our own application which will be suitable for our algorithms. Luckily our dataset is a whole double

sparse matrix and MATLAB can process these data efficiently. So we decided to use MATLAB as our main programming language.

The most widely used algorithm to solve this problem is Collaborative Filtering. Collaborative filtering (CF) is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc. Applications of collaborative filtering typically involve very large data sets.

In this project, we used two kinds of CF to process our data: K-Nearest Neighbors (KNN) and Singular Value Decomposition (SVD).

Neighborhood based approaches are very popular collaborative filtering methods. The Netflix Prize has clearly shown that these methods can easily be beaten in terms of speed and accuracy by simple factor. As individual models, they do not achieve outstanding low RMSE values, but in an ensemble of methods they work very well.

In this project we use the item-item correlation ρ_{ij} between item I and item j, based on the ratings from users who rated both movies. So all the ratings, where a user rated only one of the movies, are skipped. There are many ways to calculate the correlations of two different vectors. Pearson correlation, Spearman's rank correlation, Set correlation and MSE correlation. In this project, we used a Spearman correlation to build our correlation matrix. It was said in many research papers that Spearman and Pearson correlation produce the best results among all the correlations.

Singular Value Decomposition (SVD) is an important factorization of a rectangular real or complex matrix, with many applications in signal processing and statistics. In this Netflix Problem, we use SVD to factorize the data matrix, and then get the latent factor size l, to generate a new prediction.

Both KNN and SVD can be added with time dependence. In KNN we can simply put a weight on the rating. The more recent the rating time, the more weight we put on. In SVD, we added a feature offset when calculating the inner feature product.

After realization of those processes, we tried to blend all the models together to get a better result. It was found that for only one model it is so hard to get a good result. But by combing multiple models, better RMSE shall be achieved. By dividing the whole dataset into different parts, each part use different model to train, we finally get an improved result comparing to those single algorithms.

2.1 Related Work

The Netflix Prize has been there since 2006 and because of the one million dollar prize, there have been many talented researchers who have taken part in this problem and done a huge contribution. In Sep. of 2009, the 10% improvement has been achieved and the one million dollar prize was given to the winner BellKor's Pragmatic Chaos. This is a team combined with three teams. Each of them contributed their models to the final result.

It is lucky for us that there are many papers solving this problem. Some algorithms have been discussed for years and are quite mature. So there is barely research work about algorithms for us to work on. But we still have to read more paper to understand those algorithms and then started to implement it.

In [9] the author did an overview of this problem and then gave two common solutions about it. The two solutions are based on KNN and SVD separately. This is a paper we read at the beginning of this project. It does not have many academic terms, and is easy to understand for a beginner in this area.

Paper [3] and [5] were written by the winner team of the 2009 prize. Those two papers included all the models they combined to win the prize, also the results for each model and combined ones. These are the papers focusing on the introduction. They just told us what they have done, what methods they have used, but there is no information about how they did that. However, those papers gave us a whole picture on how to solve this problem by blending different models. Since then we started to get an idea of how we are going to do our data mining part. It also reminds us to read more about the details of each algorithm.

In [8] the author talked about the item-based KNN algorithm. The algorithm in this paper is basically how we did in our project, only with a little revise.

In paper [2] the authors talked about different kinds of CF algorithms including both KNN and SVD.

In [4] and [6] we found the basic idea of time dependence and how to import time effects into different algorithms.

[7] is a online webpage which is a home work page for a data retrieval class of CMU. It uses a KNN algorithm based on Euclidean Distance. We did not do as what they did bacause we find it's hard to calculate and Euclidean Distance's result is not as good as Spearman's Rank. But we do use their sampled dataset which they got permission from Netflix.

3. METHODOLOGY

There are three sub-modules in our works, preprocessing, data mining and visualization.

3.1 Preprocessing

There are about 480,000 customers in the dataset, each identified by a unique integer id. The title and release year for each movie is also provided. There are over 17,000 movies in the dataset, each identified by a unique integer id. The dataset contains over 100 million ratings. The ratings were collected between October 1998 and December 2005 and reflect the distribution of all ratings received during this period.

The movie attributes contains a movie id, year of movie, and movie title. The rating attributes have a customer id, a movie id, the date of the rating, and the value of the rating which is on an integer scale from 1 to 5. The example of Netflix data set is shown in figure 1.

1,2003,Dinosaur Planet
2,2004,Isle of Man
3,1997,Character
4,1994,Paula Abdul's
5,2004,The Rise
6,1997,Sick
7,1992,8 Man
8,2004,What the #$*!
9,1991,Class of Nuke
10,2001,Fighter

7:
529849,3,2001-03-05
1892469,3,2001-02-15

12:
1691746,3,2000-01-22
249198,4,2001-02-15
1452454,4,2000-04-07

(a) Movie data (b) Rating data

Figure 1 Example of Netflix data set

Our sampling strategy is to select 70 percent of movie data by random, then get rating values from rating data which is related to sample movies. After preprocessing, there are 5,392 movies 5,392 customers and 880,367 rating, and we use this data as our data set through data mining and visualization processes.

3.2 Data Mining

Data mining part is the most important part of this project. In this part we are going to introduce our full algorithms. First are the KNN algorithm and SVD algorithm. Secondly, we introduce how we import time dependence into those two models. In the end, we blend the four models together and finally have a better result.

3.2.1 KNN Model

K Nearest Neighbors (KNN)/Neighborhood based approaches are very popular collaborative filtering methods. The Netflix Prize has clearly shown that these methods can easily be beaten in terms of speed and accuracy by simple factor. As individual models, they do not achieve outstanding low RMSE values, but in an ensemble of methods they work very well.

In this problem, the algorithm works on the principle that a person tends to give similar ratings to similar movies. Joe likes three movies on the left, so to make a predication for him, we find users who also like those movies and see what other movies they like. Here the three other viewers all like Saving Private Ryan, so that is the top recommendation. Two of them like Dune, so that ranks second, and so on.

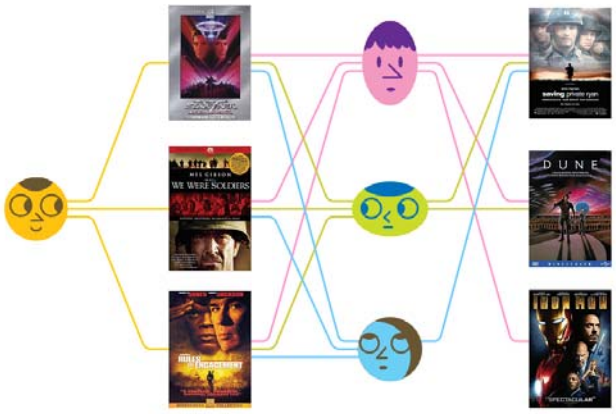


Figure 2 The KNN methodology example

The data matrix is an M×N matrix which has M number of users and N number of movies. We calculate the correlations between N columns which are the similarities between movies. This is so called "item-based" k nearest neighbors. There are also user-based algorithms. But in the recommender systems, it is widely recognized that "item-based" algorithms tend to have better results.

The item-item correlations have many different ways to calculate.

$$\rho_{ij} = 1 - \frac{6}{L \cdot (L^2 - 1)} \cdot \sum_{l=1}^{L} d_{ij}{}^2[l]$$

Pearson correlation, Spearman's rank correlation, set correlation, MSE and Ratio correlation. The project in our reference used a Euclidean distance to do the similarity. But on one hand it is source consuming to calculate; on the other hand the result using Euclidean distance is not good. So we used a Spearman's rank correlation which was recognized as one of the best correlation calculation algorithm in [3].

Each movie will be calculated with other M-1 movies using the equation. The correlation between movie i and movie j is denoted as ρ_{ij}. The range of ρ_{ij} is from 0 to 1(all the value smaller than 0 is set to 0).

One means the similarity between two movies is the closest. 0 means those two movies have no correlation. ρ_{ij} is equal to ρ_{ji}.

After this similarity calculation we have an M×M matrix whose elements are all similarity values. This matrix's size is 5392*5392.

Table 1 An example of correlation matrix

1	0.8	0.6	0	1
0.8	1	0.45	0.12	0
0.6	0.45	1	0.83	0.32
0	0.12	0.83	1	0
1	0	0.32	0	1

What is shown above is a sample similarity matrix of M×M. If we multiply a user row by this matrix, we will get a rating list of this user. If we order this list we will get the result of our rating levels. We can simply assign the top 20% of the list as five stars and the bottom 20% as one star.

3.2.2 SVD Models

Singular Value Decomposition (SVD) is an important factorization of a rectangular real or complex matrix, with many applications in signal processing and statistics. In this Netflix Problem, we use SVD to factorize the data matrix, and then get the latent factor size l, to generate a new prediction.

Collaborative algorithms based on dimensionality reduction techniques try to solve the sparse-matrix problem by reducing the problem dimensions, i.e., the number of features used to represent users and items. Let's suppose that the URM(user-rating matrix) has dimension $n \times m$ and rank p. Using SVD we can factorize this matrix into the product of three matrices, U, S and V, such that:

$$R = U \cdot S \cdot V^T$$

Where U is an $n \times p$ matrix which represents the users, V is a $m \times p$ matrix which represents the items, and S is a diagonal matrix which contains the p singular values of R, sorted from the greatest to the smallest. Usually only the first $l < p$ features are considered, because they well describe the most important information of the dataset. The prediction rating can be defined as

$$\hat{R}_l = U_l \cdot S_l \cdot V_l^T$$

Where U_l, S_l *and* V_l *are* the first l columns (i.e., features) of U, S and V, respectively. The optimal l value is referred to as the latent size. The latent factors inferred from the ratings are given to all the movies by all the reviewers.

The factors define a space that at once measures the characteristics of movies and the viewer's interest in those characteristics. Here we would expect the fellow in the southeast of the graph to love *Norbit*, to hate *Dreamgirls*, and, perhaps, to rate *Braveheart* about average.

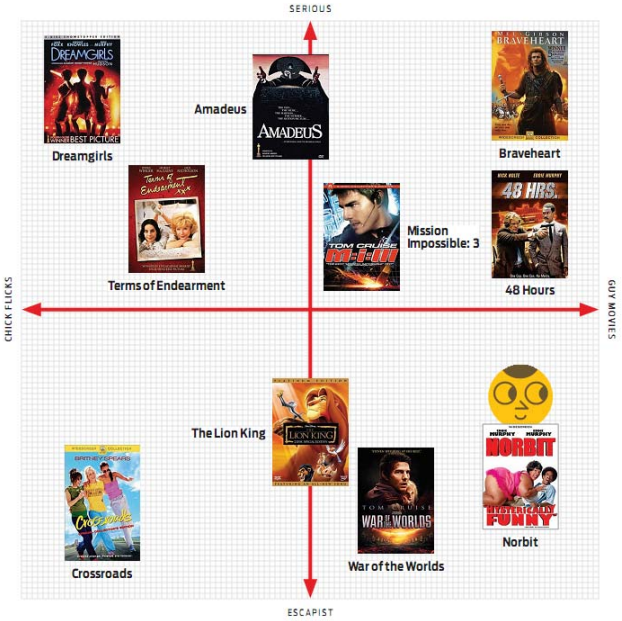


Figure 3 The SVD methodology example

This model scores both a given movie and a given viewer according to a set of factors, which are inferred from patterns in the ratings given to all the movies by all the viewers. Factors for movies may measure comedy versus drama, action versus romance, and orientation to children versus orientation to adults. Because the factors are determined automatically by algorithms, they may correspond to hard-to-describe concepts such as quirkiness, or they may not be interpretable by humans at all. Factors for viewers measure how much the viewer likes movies that score highly on the corresponding movie factor. Thus, a movie may be classified as a comedy lover.

The model may use 20 to 40 such factors to locate each movie and viewer in a multi dimensional space. It then predicts a viewer's rating of a movie according to the movie's score on the dimensions that person cares about most. We can put these judgments in quantitative terms by taking the dot (or scale) product of the locations of the viewer and the movie.

3.2.3 Time Dependence

It is widely recognized that time factor is important in user rating activities. But usually it is not that easy to add time influence into different models. In this project, we considered adding time factor into our basic models. So they are Time-KNN and Time-SVD two models on time influence.

- For Time-KNN, the authors of [2] believe that the more recent the rating is, the more trustable it is. So combined with KNN, we add a weight factor onto our KNN results. Since our ratings data was made between 1998~2005, so we just think the ratings made in 2005 has a bigger weight (at most 1). The ratings in 1998 have the lowest weight (at least 0.4). By doing so, we got a decrease on RMSE by 0.02. Obviously, time-factor will provide useful information to our KNN algorithm.
- For introducing time factor into SVD model we have a solution which is from [4] that brings a new factor into the whole equation. Basically, it is hard to directly incorporate time into a SVD model. The time-SVD model adds time information by adding a feature offset when calculating the inner feature product. Given a user feature and a movie feature vector, a rating of user u and movie i is predicted by :

$$\hat{r}_{ui} = \sum_{k=1}^{K} p_{u,(k+d_{ui})} \cdot q_{i,(k+d_{ui})}$$

This means that features are shifted, depending on the date of the rating. The discrete time offset d_{ui} are precalculated before the training starts. We divided the rating time span into T time slots per user. The first slot is left side open and the last slot is right side open. For example if a user u gives votes on 50 different days and T=10, then 5 days point to 1 slot. The offset d_{ui} can take values of *{0, 1,…, T-1}*.

3.2.4 Blending

Both collaborative-filtering techniques work even if you don't know anything about what's in the movies themselves. All you need to care is how the viewers rate the movies. However, neither approach is a panacea. We found that most nearest-neighbor techniques work best on 50 or fewer neighbors, which means these methods cannot exploit all the information a viewer's ratings may contain. SVD models have the opposite weakness: They are bad at detecting strong associations among a few closely related films, such as *The Lord of the Rings Trilogy (2001-2003).*

Because these two methods are complementary, we combined them, using many versions of each in what machine-learning experts call an ensemble approach. This allowed us to build systems that were simple and therefore easy to code and fast to run.

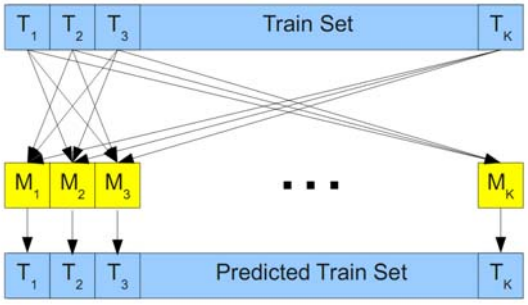


Figure 4 The blending technique

About how we did on the blending: like the figure above, we split the training set into K disjoint sets of equal size. And we train different models M_1 to M_k. The first model M_1 uses the ratings of the sets T_2 to T_k for training and generates predictions for the set T_1. The second model M_2 excludes the set T_2 in the training phase, and calculates predications for the set. Each rating in the training set is predicted by 1 model. Each training set in the test

set is predicted by 4 models. The predications for the probe and the qualifying set are linear blends of all K models.

3.3 Visualization

In order to show our experimental results as well as detailed information of our project, we build a project website which contains almost all information regarding the project. We use HTML for most pages and use PHP for search tools which enable visitors to conduct search of data in our database, which is calculated by our data mining methods.

3.1.1 HTML Pages

There are five main parts of the website: home page, introduction, user-movie search, performance demo and contact. And each part has links to other four parts as well as external websites containing background information.

In the Home page, a general introduction is given on our project, the Netflix Challenge. In the Introduction page, first we give an introduction to the Netflix Challenge and what we do to solve the problems. Then we give five passages which present detailed explanations about this project, such as the data set information, our data mining methodology and our experimental results. These five pages are linked to the introduction pages. In the User-Movie Search page, we give some explanations about the four types of search: movie-movie search, movie-user search, user-movie search and user-user search. The search tool is implemented by PHP, which we will explain in next part. In the Performance Demo page, we give presentation on the performance of our three algorithms and their combination: KNN, SVD and time-dependence. RMSE is used to measure the efficiency of their performance. In the contact part, we give contact information of us and enable visitors to leave a message.

3.1.2 PHP Pages

While HTML pages take a large part in our project website, PHP pages play an important role in processing information visitors enter and then show search results. On one side, we use PHP to include a text file which contains messages which we will show in the posts in every page. In order to show the same posts in every page, we include the post.txt file using PHP so that all pages will reflect content in that file. On the other, when user choose a search method and enter a movie or user ID, we use a form to convey this information to our query.php file. This file is connected to MySQL database.

We have put our experimental results into five CSV files. Then we load these files into MySQL database to create five tables: movie-movie, movie-user, user-movie, user-user and movie. The first 4 tables have 11 attributes. The first attribute is query ID. For example, in the movie-movie table, the first attribute is the query movie ID. And the 10 attributes left are 10 results. In movie-movie table, they are 10 movies ID which are most similar to the query movie. The last table, movie, stores the information of all 17,000 movies. The 3 attributes are movie ID, released year and movie title.

When visitors specify a search choice, e.g. movie-movie search, we find the query ID in the corresponding table, movie-movie table. And then we connect the resulted row with the movie table to get the movie title and the year it is released. Then we show the results in the website.

4. EXPERIMENTAL EVALUATION

This part has two sections. In section one， we presented how we did experimental setup in different part of our project. In section two we gave our experimental results including the webpage.

4.1 Experimental Setup

This section we gave some important information on the dataset and measurement result.

4.1.1 Data Set Information

There are two data types in the whole data set: movie and rating. The movie data contains a movie ID, a release year and a movie title. The rating data contains a movie ID, a user ID and the date of the rating and the value of the rating. The value of the rating is an integer scale from 1 to 5, inclusively. The original raw data are stored in plain text files. During preprocessing, we convert the raw data into matrix data set, which columns represent movie ID, customers represent in row, and the value store in the matrix are the rating users gave to each movie. We called this matrix as the user-movie_rating matrix. The example of this matrix is shown in table 2.

Table 2 The example of user-movie rating matrix

Movie ID/ User ID	7	14	23	42	76	79	85	93	97	...
433	3								2	
455			4							
12545					3					
23323	5								1	
...										

4.1.2 Evaluation Measure

We use RMSE as the evaluation measure:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{r}_{ij} - r_{ij})^2}$$

This measure shows how close our predictions are to the true ratings in the Netflix database.

4.1.3 Computing Platforms

Table 3 Computing Platforms of each module

Role	Platform	Programming Language
Preprocessing	IntelCore2Duo 1.66GHz, 4G memory	Java
Data Mining	IntelCore2Duo2.53 GHz, 4G memory	Matlab, C and C++
Visualization	IntelCore2Duo 2.10GHz, 4G memory	HTML, MySQL PHP

4.2 Experimental Results

Our project website: http://www.cse.msu.edu/~zhangy72/netflix/

4.2.1 Preprocessing

After preprocessing we store the data into the following structures (Note that we do not store the user data because it is easy to extract from movie-user data):

Movie (movieID, year, titile)

Movie-User (movieID, userID, rating, ratingDate)

4.2.2 Data Mining

After data mining, we generate two matrices. One is the user-movie matrix and the other is movie-movie matrix. In the user-movie matrix, each element is the rating the user in the row gives to the movie in the column. In the movie-movie matrix, each element is the similarity value between the movie in the row and the movie in the column.

The RMSE of the three methods and their combinations are described in figure

Method	KNN	SVD	Time-Dep	SVD-Time	Blending #1	Blending#2
RMSE	0.9251	0.9128	0.9061	0.9968	1.0475	0.8932

Figure 5. RMSE values of the tree methods and their combinations.

Blending#2 method has the minimum RMSE value. This shows that by adjusting our bending methods we can have improved results. This is also what we should focus on in our future work.

4.2.3 Visualization

The database schema is shown in Figure 6.

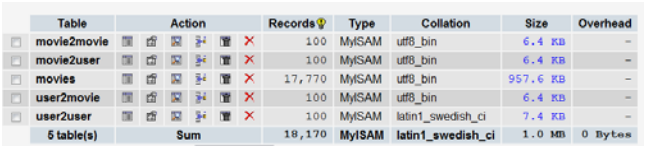
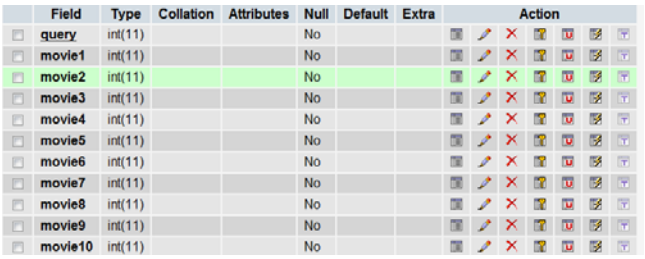


Table	Action	Records	Type	Collation	Size	Overhead
movie2movie		100	MyISAM	utf8_bin	6.4 KB	-
movie2user		100	MyISAM	utf8_bin	6.4 KB	-
movies		17,770	MyISAM	utf8_bin	957.6 KB	-
user2movie		100	MyISAM	utf8_bin	6.4 KB	-
user2user		100	MyISAM	latin1_swedish_ci	7.4 KB	-
5 table(s)	Sum	18,170	MyISAM	latin1_swedish_ci	1.0 MB	0 Bytes

Figure 6 The tables in MySQL database

Here is the movie-movie table.



Field	Type	Collation	Attributes	Null	Default	Extra	Action
query	int(11)			No			
movie1	int(11)			No			
movie2	int(11)			No			
movie3	int(11)			No			
movie4	int(11)			No			
movie5	int(11)			No			
movie6	int(11)			No			
movie7	int(11)			No			
movie8	int(11)			No			
movie9	int(11)			No			
movie10	int(11)			No			

Figure 7 The movie-movie table

The other 3 tables are similar to movie-movie table. Fig. is our search tool.

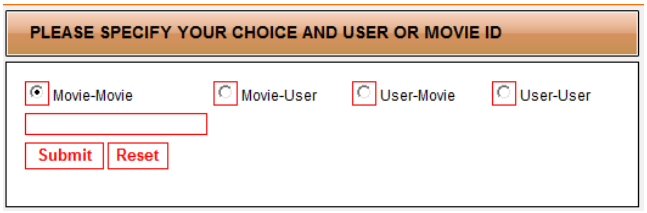


Figure 8 The search page

After entering a search choice and the query ID, we get the following results in Fig.

Figure 9 An example result of the movie-movie search.

4.3 Discussion

The final result of RMSE shows that single KNN and SVD can do a basic job on the predication. As a single model, SVD has a better result than KNN. However, KNN is more suitable for

blending algorithm. As soon as we put a time factor into KNN model, its RMSE is getting much better. What confuse us is that SVD plus time factor does not work very well. It gave a RMSE of 0.9968 at last, much higher than SVD model. Luckily, by blending those four models we get a RMSE of 0.8932, a very good progress compared to our former models.

5. CONCLUSIONS

In this project we start preprocessing by sampling the large dataset. Due to the huge disk space the original dataset would take, we would like to run our algorithm efficiently. The sampling gave us plenty of favor to speed up. Although we did not get very good result (RMSE below 0.88), but we have implemented four algorithms in a relatively short time.

In data mining we first studied all the basic ideas of most of the models, and then chose four models as our realization object. In the process of doing data mining, our ability of logic thinking has improved, skill of programming has sharpened and overall the techniques of solving academic problems have been gained. Netflix is a very interesting project and I really did have fun during the whole process. Recommender system is becoming more and more popular in nowadays Internet, and we have reason to believe those data mining techniques we have learnt in this project will be helpful.

In visualization part we use HTML and PHP to build our websites which contain both our methods and experimental results. In addition, the website enables visitors to conduct search so that visitors will have a direct understanding how the prediction methods work.

Since it is obvious that blending methods can improve our prediction accuracy in terms of RMSE value, our future work will focus on how to assign different weights to different subparts of the whole training set and then apply different algorithms accordingly. This requires insightful study of characteristics of training data. Moreover, we can further study how to assign time weight to enhance time dependence method.

In the end, we want to give our sincere thanks to Dr. Tan for his amazing classes of the whole semester and plenty of help on our project.

6. REFERENCES

[1] Netix Prize homepage. Website,2006. http://www.netflixprize.com.

[2] R.Bell and Y.Koren. "Scalable collaborative filtering with jointly derived neighborhood interpolation weights".In IEEE International Conferenceon DataMining.KDD-Cup07,2007.

[3] Andreas Taoscher and Michael Jahrer "The Big Chaos Solution to the Netfix Grand Prize“

[4] Liang Xiang and Qing Yang, "Time-dependent Models in Collaborative Filtering based Recommender System"

[5] Andreas Taoscher and Michael Jahrer "The Big Chaos Solution to the Netfix Prize 2008“

[6] Yi Ding and Xue Li, "Time Weight Collaborative Filtering"

[7] Jamie Callan and Yiming Yang, CMU 11-741-Information Retrieval homework-3

[8] MUKUNDDE SHPANDE and GEORGE KARYPIS, "Item-BasedTop-N Recommendation Algorithms" ACM Transactions on Information Systems,Vol.22,No.1,January2004,Pages143–177.

[9] Robert M.Bell,Jim Bennett and Chris Volinsky,"The Million Dollar Programming Prize", IEEE Spectrum, May 2009