Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 742341, 9 pages
http://dx.doi.org/10.1155/2014/742341

Research Article

Precomputed Clustering for Movie Recommendation System in Real Time

Bo Li,[1] Yibin Liao,[1] and Zheng Qin[2]

[1] *Department of Computer Science, University of Georgia, Athens, GA 30602, USA*
[2] *School of Software, Tsinghua University, Beijing 100084, China*

Correspondence should be addressed to Zheng Qin; qingzh@tsinghua.edu.cn

Received 7 February 2014; Accepted 12 May 2014; Published 25 June 2014

Academic Editor: Guiming Luo

A recommendation system delivers customized data (articles, news, images, music, movies, etc.) to its users. As the interest of recommendation systems grows, we started working on the movie recommendation systems. Most research efforts in the fields of movie recommendation system are focusing on discovering the most relevant features from users, or seeking out users who share same tastes as that of the given user as well as recommending the movies according to the liking of these sought users or seeking out users who share a connection with other people (friends, classmates, colleagues, etc.) and make recommendations based on those related people's tastes. However, little research has focused on recommending movies based on the movie's features. In this paper, we present a novel idea that applies machine learning techniques to construct a cluster for the movie by implementing a distance matrix based on the movie features and then make movie recommendation in real time. We implement some different clustering methods and evaluate their performance in a real movie forum website owned by one of our authors. This idea can also be used in other types of recommendation systems such as music, news, and articles.

1. Introduction

Movie recommendation systems suggest movies to user that he/she might be interested in. The generated suggestions are obtained from the consideration of many aspects. The suggestion can be based on the tastes, interests, goals, people connections, and so forth.

In general, a movie recommendation system compares user's profile or usage data to some reference characteristics and combines the user's social environment to make movie recommendations. This type of recommendations is based on user. However, this type of recommendations may not work or make inaccurate recommendation in the following situations. The user does not have strong profile setting in the system. There are many users who do not want to set their profile due to laziness or privacy concerns. In this case, most recommendation systems consider the user's social connection in the system such as friends, classmates, families, and colleagues. However, the tastes of the movies may be various even among best friends. What is worse, the "friends" that the user has added in the website may not be people with the same interest. For example, in a community network or local network such as a university or college, like our experimental environment, users' social connections are built mainly because they are in the same university with the same major and similar age; however, their tastes towards movies may be totally different, which fails the fundamental bias in the recommendation system mentioned above.

Because of the situation stated above, we propose the use of machine learning (ML) techniques (clustering) to analyze the movie features and system logs (user's voting logs) to make correct recommendations more adequately. In this proposed approach, we compute a distance matrix for the movie features and apply the clustering techniques to classify movies into different areas off-line. For every user logged on the system, we recommend movies from the clusters combined with the user's majority voting result in real time.

In order to compare the accuracy and efficiency, we have implemented different clustering techniques as follows:

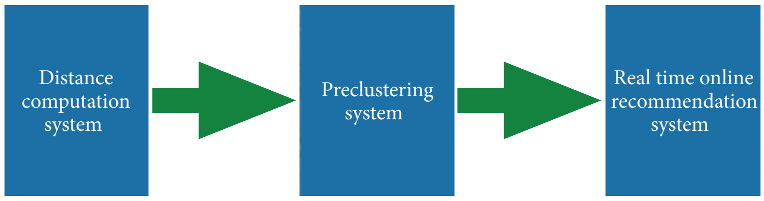

Distance computation system
Preclustering system
Real time online recommendation system

Figure 1: System overview.

DBSCAN (density-based clustering), affinity propagation, hierarchical clustering, and random clustering as a base line.

We have tested this proposal with all the clustering techniques in a university social website owned by one of our authors. There is a subsystem with movie service in the website. We add a recommendation function in a banner and put it on top of the section to let user make votes. We offer them all the clustering results from four cluster algorithms shuffled together and record the voting results separately.

The rest of the paper is organized as follows. Section 2 gives an overview of related works in which machine learning techniques have been applied to recommendation systems. Section 3 is the system overview of our recommendation system. Section 4 illuminates the distance functions we use to compute the distance matrix. Section 5 introduces all the clustering techniques used in our preclustering system. Section 6 includes the experimental setup and evaluation results. Finally, Section 7 concludes with discussions and future research.

2. Related Work

Machine learning techniques are very useful when huge amounts of data have to be classified and analyzed, which nowadays is a very common situation in many scenarios, especially in the recommendation system. There are a lot of different machine learning techniques used in different recommendation systems, such as Naive Bayes classification [1], decision tree [2], k-means clustering and improvement [3–6], and so forth.

Generally, the recommendation systems are divided into two major categories such as collaborative recommendation system and content based recommendation system [7]. In case of collaborative recommendation systems, these try to seek out users who share same tastes as that of the given user as well as to recommend the movies according to the liking of these sought users. For example, Tatli and Birtürk described an approach for creating music recommendations based on user-supplied tags that are augmented with a hierarchical structure extracted for top level genres from Dbpedia. In this structure, each genre is represented by its stylistic origins, typical instruments, derivative forms, subgenres, and fusion genres. They use this well-organized structure in dimensionality reduction in user profiling [8]. Yang et al. proposed a personalized web page recommendation model called PIGEON (abbreviation for PersonalIzed web paGe rEcommendatiON) via collaborative filtering and a topic-aware Markov model and proposed a graph-based iteration algorithm to discover user's interested topics, based on which user similarities are measured [9]. Cai et al. also proposed a model that fully captures the bilateral role of user interactions within a social network and formulated collaborative filtering methods to enable people to people recommendation [10]. As I mentioned in Section 1, making recommendation based on user is not working properly in some particular areas such as a community network system or university network.

The content based recommendation systems try to recommend contents similar to those web sites the user has liked. Biancalana et al. proposed two different context-aware approaches for the movie recommendation task, one is a hybrid recommender that assesses available contextual factors related to time in order to increase the performance of traditional CF approaches, and the other one aims at identifying users in a household that submitted a given rating [11]. This latter approach is based on machine learning techniques, namely, neural networks and majority voting. In our paper, we focus on the content based recommendation and use the majority voting and clustering techniques. We obtained better results compared to their research, which will be shown in Section 6.

Some of the researchers proposed hybrid recommendation approaches by combining different approaches. Ghazanfar and Prügel-Bennett proposed a unique switching hybrid recommendation approach by combining a Naive Bayes classification approach with the collaborative filtering [1]. Ujwala et al. presented and investigated an approach based on weighted Association Rule Mining Algorithm and text mining [7]. Bellogín et al. implemented decision trees and attribute selections together to build the recommendation model to find the most relevant preferences of user and system [2]. The idea of hybrid recommendation approach gives us inspiration to combine different approaches to implement the movie recommendation for future work.

3. System Overview

Figure 1 provides a high-level overview of our new movie recommendation approach. As shown in Figure 1, the new movie recommendation system comes in three parts: distance computation system, preclustering system, and real time online recommendation system.

3.1. Distance Computation System. Distance computation system computes the distance between different movies based

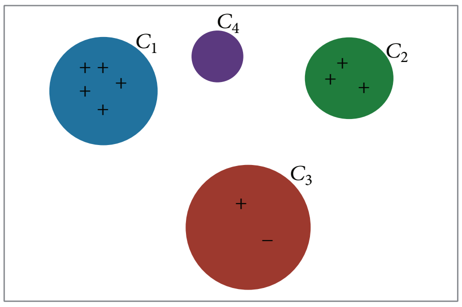


Figure 2: A demo of voting. The probability that recommended movies are in C_1 is 5/8.

on different properties of the movies including the movies types, publish year, countries of publishing companies, languages, directors, casts, and duration time. We first compute the Jaccard distance based on movies types, countries of publishing companies, languages, directors, and casts, and then the distance based on publish year and duration time by the distance we defined in Sections 4.2.2 and 4.2.3. We then compute the overall distances between each movie sample by summarizing them together with the weights which we obtained from the survey mentioned in Section 4.1.

3.2. Preclustering System. Preclustering system separates movies into different clusters before giving them to online recommendation system. In this system, we tried five kinds of different clustering algorithms: affinity propagation, DBSCAN, hierarchical clustering, spectral clustering, and random generator, among which spectral clustering is too slow to get the result in demanded time. Therefore, we sent the remaining four clustering results to the recommendation system with two data formats, one can get the cluster label by one movie's IMDB identifier and another one can drag all the movies' IMDB identifiers by one of cluster labels. In this way, the online recommendation system in Section 3.3 can quickly get the information it wants.

3.3. Real Time Online Recommendation System. After getting the preclustering results, the website read the history of users' votes on movies and used the majority of the voting to generate recommendations. In more detail, if one user U_i likes movie M_{j1} which belongs to cluster C_{k1}, U_i will give C_{k1} one *vote* on C_{k1}. On contract, if U_i dislikes movie M_{j2} in C_{k2}, U_i will cancel a *vote* for C_{k2}. If a cluster gets negative *vote* at the end, it will return to 0. After U_i votes all the movies he/she comments, different clusters will get different votes. The movie recommendation system will recommend movies according to the votes of clusters C_j as $V(C_j)$ as follows:

$$P(C_i) = \frac{V(C_i)}{\sum_{j=1}^{n} V(C_j)}, \tag{1}$$

where $P(C_i)$ is the probability that a movie should be recommended from cluster C_i and $V(C_i)$ is the number of votes in cluster C_i. n is the number of clusters. As shown in Figure 2.

To make sure that in our evaluation the reason that user votes a movie we recommended as dislike is not because the movie itself is really bad, we recommend movies with the IMDB score higher than 7 in the clusters which have already been voted out by the user.

4. Distance Computation System

4.1. Weight Survey. Movies' properties create a very special space, where the weights of each dimension are treated completely different. For example, the type of a movie is of course more important than the duration of it. Therefore, we cannot use the default distance function provided by Scikit-learn [18] package that we use. Instead, the feature selection like in [2] or starting a survey to figure out the weights to different dimensions should be implemented. In other research areas where we human beings do not know which feature is important, such as features in carcinogenic, we always use feature selection to decide the weights of different features. In movie recommendation area, however, we can investigate it by the survey shown in Figure 3, since people know why they like the movie. By carefully designing a survey and letting users vote the top three factors, we obtained a result shown in Figure 4.

From the survey result shown in Figure 4, we can figure out that the weight of publish year is 0.0915, weight of country is 0.147, weight of type is 0.4167, weight of language is 0.0545, weight of director is 0.0896, weight of casts is 0.1792, and weight of duration is 0.0214. These weights will be used in the overall distance computation in Section 4.3.

4.2. Distance Function

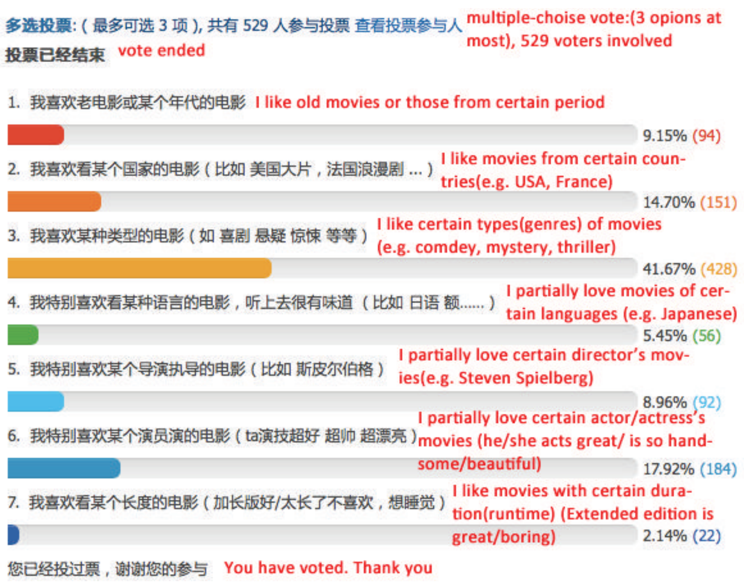
4.2.1. Jaccard Distance. Since for countries, languages, casts, and types of the movie, the distance is determined by how many same and different items there are between sets, we decided to use Jaccard distance proposed in [12]. Jaccard distance is a statistic used for measuring dissimilarity between sample sets. The formula is shown in formula (2). According to this formula, we use intersection and union to do the calculation and easily obtain the distance between each movie feature mentioned above as follows:

$$J(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}. \tag{2}$$

4.2.2. Publish Year Distance. The distance for publish year between movies is a very special case for the human being. To human's intuition, the distance between movies published in 1950s and 1960s seems much less than that between movies published in 2012 and 2013. In another word, the distance of publish year is not a linear function. One reason for such phenomenon is that there are more and more movies published recently. The statistic of number of movies in each year in our database is shown in Figure 5.

Figure 5 illuminates that the number of published movies in each year is similar to an exponential function. The trend line is shown as follows:

$$f(x) = 0.3584 \cdot \exp^{0.0598(x-1902)}. \tag{3}$$



多选投票:（最多可选 3 项）, 共有 529 人参与投票 查看投票参与人 multiple-choise vote:(3 opions at most), 529 voters involved

投票已经结束 vote ended

1. 我喜欢老电影或某个年代的电影 I like old movies or those from certain period — 9.15% (94)
2. 我喜欢看某个国家的电影（比如 美国大片，法国浪漫剧 ...） I like movies from certain countries(e.g. USA, France) — 14.70% (151)
3. 我喜欢某种类型的电影（如 喜剧 悬疑 惊悚 等等） I like certain types(genres) of movies (e.g. comdey, mystery, thriller) — 41.67% (428)
4. 我特别喜欢看某种语言的电影，听上去很有味道（比如 日语 额......） I partially love movies of certain languages (e.g. Japanese) — 5.45% (56)
5. 我特别喜欢某个导演执导的电影（比如 斯皮尔伯格） I partially love certain director's movies(e.g. Steven Spielberg) — 8.96% (92)
6. 我特别喜欢某个演员演的电影（ta演技超好 超帅 超漂亮） I partially love certain actor/actress's movies (he/she acts great/ is so handsome/beautiful) — 17.92% (184)
7. 我喜欢看某个长度的电影（加长版好/太长了不喜欢，想睡觉） I like movies with certain duration(runtime) (Extended edition is great/boring) — 2.14% (22)

您已经投过票，谢谢您的参与 You have voted. Thank you

Figure 3: Survey in our community.

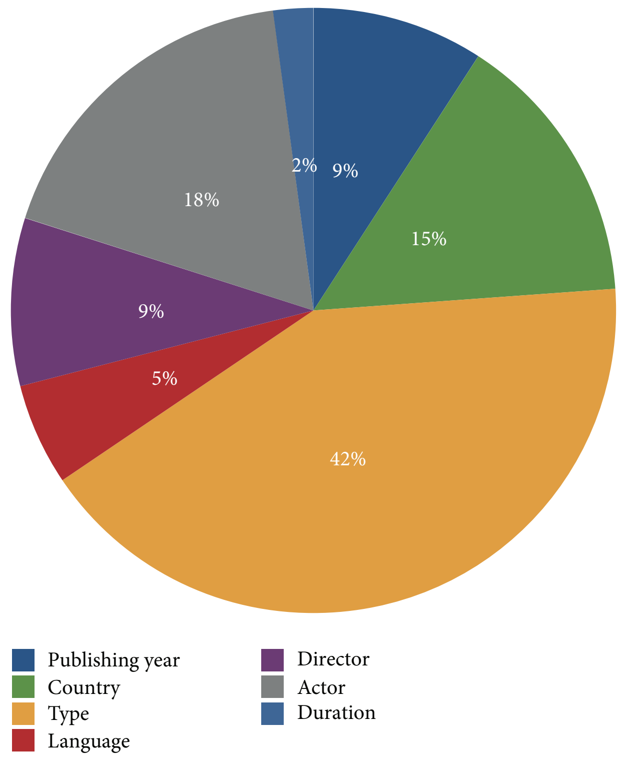


Figure 4: Survey of top three liked movie factors.

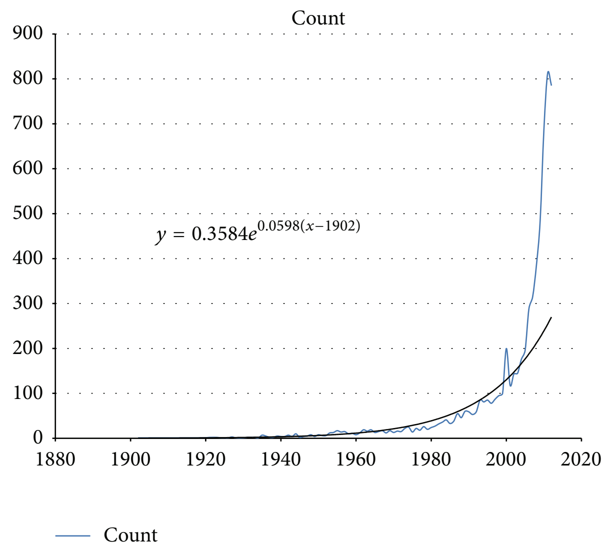


Figure 5: Statistic result of movie published in each year.

According to formula (3), we propose a new distance function for publish year $\mathrm{PD}(\mathrm{year}_i, \mathrm{year}_j)$ shown in the following:

$$\mathrm{PD}\left(\mathrm{year}_i, \mathrm{year}_j\right) = \frac{\int_{\min(\mathrm{year}_i,\mathrm{year}_j)}^{\max(\mathrm{year}_i,\mathrm{year}_j)} f(x)}{\int_{1902}^{2014} f(x)}. \tag{4}$$

4.2.3. Duration Distance. Human being is very sensitive for a small period of time, which means that half an hour is nearly as double time as a quarter of an hour in human's feeling. Therefore, we use a linear function shown in formula (5) to compute the duration distance $\mathrm{DUD}(\mathrm{dur}_i, \mathrm{dur}_j)$

$$\mathrm{DUD}\left(\mathrm{dur}_i, \mathrm{dur}_j\right) = \frac{\min\left(\left|\mathrm{dur}_i - \mathrm{dur}_j\right|, 200\right)}{200}. \tag{5}$$

Here, in order to normalize the distance, we consider 200 minutes as the maxima duration of a movie.

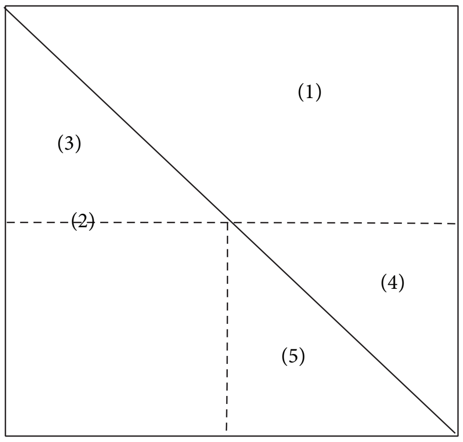


FIGURE 6: Matrix computation parallel demo.

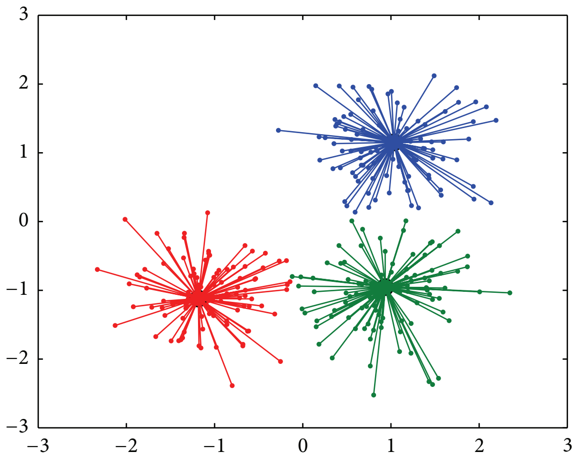


FIGURE 7: Affinity propagation demo [18].

4.3. Overall Distance. We build up a distance matrix $[n, n]$ between each movie in our database based on formula (6) as follows:

$$\mathrm{OD}\left(m_i, m_j\right) = \sum_{k \in \{\text{movie features}\}} \text{weight}_k * \text{distance}_k\left(i, j\right). \tag{6}$$

4.4. Parallel Distance Computation. Since our distance matrix is symmetric, we can use such property to do distance computation in parallel.

To do parallel computation, two conditions must be satisfied.

(1) Works assigned to different processing must be approximately equal.

(2) There is no interaction or data exchange between different working processing.

The distance matrix we need to build is a symmetric matrix in which all the elements on the main diagonal are equal to 0, formally defined as a matrix D, where $D = D^T$ and $D[i, i] = 0$, as follows:

$$D = \begin{bmatrix} 0 & \lambda_{0,1} & \lambda_{0,2} & \cdots & \lambda_{0,n-1} & \lambda_{0,n} \\ \lambda_{0,1} & 0 & \lambda_{1,2} & \cdots & \lambda_{1,n-1} & \lambda_{1,n} \\ \lambda_{0,2} & \lambda_{1,2} & 0 & \cdots & \lambda_{2,n-1} & \lambda_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \lambda_{0,n-1} & \lambda_{1,n-1} & \lambda_{2,n-1} & \cdots & 0 & \lambda_{n-1,n} \\ \lambda_{0,n} & \lambda_{1,n} & \lambda_{2,n} & \cdots & \lambda_{n-1,n} & 0 \end{bmatrix}. \tag{7}$$

The scratch of the distance matrix shown in formula (7) is shown in Figure 6. Since the distance matrix is symmetric, the data in area (1) and area (2) and that in area (4) and area (5) are correspondingly equal. Therefore instead of computing all of them, we just compute the upper triangle of the matrix, that is, the data in areas (1) and (5). However, if we separate the parallel tasks by rows of the matrix, the works (number of elements) in each task are (is) not equal (from $n - 1$ to 1). To overcome this problem, the works needed in area (4) are filled into area (3), so that we can group $n - 1$ unequal tasks into $\lceil n/2 \rceil$ equal tasks. In this way, a process pool where each process computes distance in one task can be used to parallel these $\lceil n/2 \rceil$ tasks.

More specifically, we firstly assign the main diagonal line to 0. There is no need to compute the main diagonal, since we know that the distance between one movie and itself is 0. Then for each row i, we combine the distance computations of $D[n-1-i, n-i]$, $D[n-1-i, n-i+1]$,. . .,$D[n-1-i, n-2]$, $D[n-1-i, n-1]$, and those of $D[i, i+1]$, $D[i, i+2]$,. . .,$D[i, n-2]$, $D[i, n-1]$ together as a task. After computation, we get an upper triangular matrix, that is, area (1) and area (4). Then fill the lower triangle with the upper triangle; that is, let $D = D + D^T$.

5. Preclustering System

In this section, we use four clustering methods to investigate the best cluster method which can be used in the final movie recommendation system. The four clustering algorithms are affinity propagation [13], DBSCAN [14], hierarchical clustering [15], and random clustering for base line usage.

5.1. Clustering Method

5.1.1. Affinity Propagation. Affinity propagation which was first proposed in [13] generates clusters by sending messages between pairs of samples until convergence or changes falling below a threshold. A dataset is then described using a small number of exemplars, which are identified as those most representative of other samples. The messages sent between pairs represent the suitability for one sample to be the exemplar of the other, which is updated in response to the values from other pairs. This updating happens iteratively until convergence, at which point the final exemplars are chosen, and hence the final clustering is given as demonstrated in Figure 7.

5.1.2. DBSCAN. The DBSCAN algorithm [14] views clusters as areas of high density separated by areas of low density.

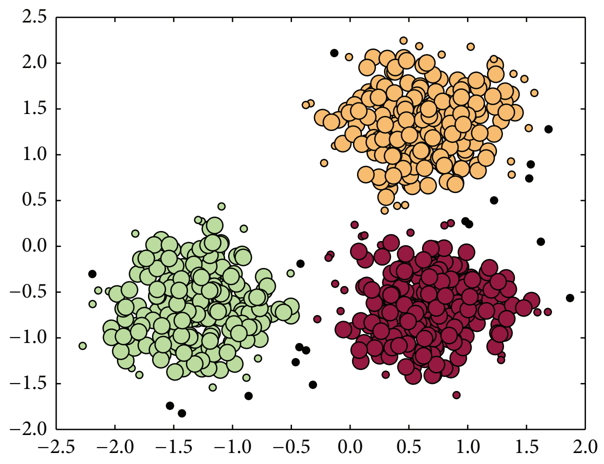


Figure 8: DBSCAN demo [18].

Due to this rather generic view, clusters found by DBSCAN can be in any shape, as opposed to k-means which assumes that clusters are convex shaped. The central component to the DBSCAN is the concept of core samples, which are samples that are in areas of high density. A cluster is therefore a set of core samples, each close to each other (measured by some distance measure) and a set of non-core samples that are close to a core sample (but are not themselves core samples) as demonstrated in Figure 8.

5.1.3. Hierarchical Clustering. Hierarchical clustering [15] is a general family of clustering algorithms that build nested clusters by merging them successively. This hierarchy of clusters is represented as a tree. The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample.

The beauty of this clustering method is that we have no need to provide the size of clustering ahead. Instead, we can initialize with a large cut height and decrease the cut height when the user has voted most movies in one cluster. In this way, we can provide a larger cluster to fit user's interests. As shown in Figure 9.

5.1.4. Random Clustering. We first fix the size of clustering to 60 and then for each movie, we randomly pick the cluster it belongs to by generating a random number between 1 and 60.

5.2. Preclustering System Output. Because of seeking efficiency, we output the clustering label results in two formats. One can be used to check clustering labels by movie ID and another is used to check all the movies' IDs in one cluster given the cluster label. By outputting the file in JSON, we can easily transfer data between preclustering system in Python and real time online recommendation system in PHP.

6. Evaluation

Before the evaluation starts, we created a survey and asked users to select the top three movie features they care about as mentioned in Section 4.1. After one week of data collection, we have the result which is shown in Figure 4 as mentioned.

Based on the result, we calculate the distance matrix and make it as an input to the different clusters. We implemented the four clustering techniques by using python library Scikit-learn.

Table 1: Comparison of results between different clustering methods in first evaluation method.

	True positive	False positive	Accuracy rate
AF	239	98	70.920%
DB	174	77	69.323%
HC	**228**	**88**	**72.152%**
RA	125	104	54.585%

6.1. Experimental Setup. The evaluation has been conducted using a university social website in China since the website is owned by one of our authors. We add the recommendation function in a banner and put it on top of the movie service to let users vote. The overview voting component of the website is showed in Figure 10.

There are three voting choices: like, do not like, and unseen for each recommended movie. Voting like scores 1, voting do not like scores −1, and voting unseen scores 0. Because of the limited time of data collection, we cannot wait for users to watch the recommended movies. Therefore, we add the unseen choices for users to select if they have not seen the recommended movies and only consider the votes which are like or do not like to evaluate our approach.

6.2. Result. After two weeks voting for the movie recommendation, we have collected 168,424 total votes, 132,360 votes are collected in 12 days and used for recommendation majority voting, as mentioned in Section 3.3. The remaining 6064 votes are collected in two days after the recommendation system is deployed. Since only the like vote and do not like vote are considered illegible, we generated the results in the following subsections.

6.2.1. First Evaluation Result. In the first evaluation result, listed in Table 1, we calculated the true positive, false positive, and accuracy for each clustering techniques. True positive equals the number of like votes, and false positive equals the number of do not like votes. Because most of the recommended movies are voted unseen, there are a total of 1,133 votes counted, and 216 users participated in the voting. Consider the following:

$$\text{Accuracy} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}. \tag{8}$$

As shown by the result in Table 1, the hierarchical clustering approach is the best. However, the accuracy of every clustering approach is not satisfied. Therefore, we extracted the users' voting history from our server database and analyzed those data. We found that some of the users made less than 5 votes, some of them made between 5 to 10 votes.

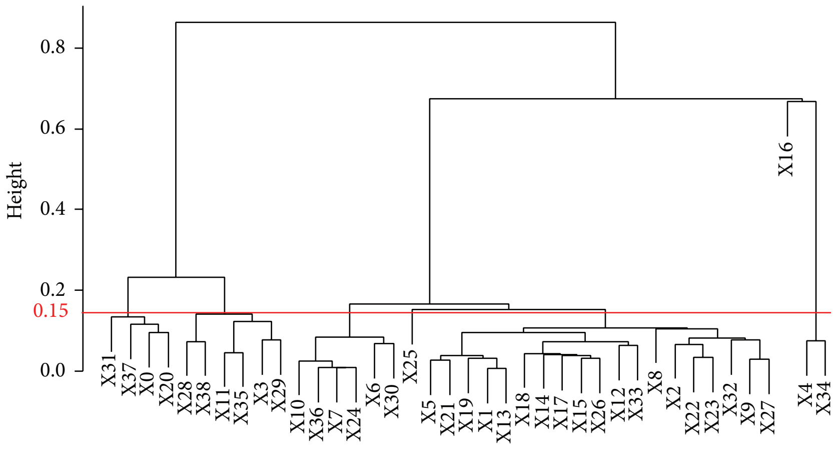


FIGURE 9: Hierarchical clustering demo.

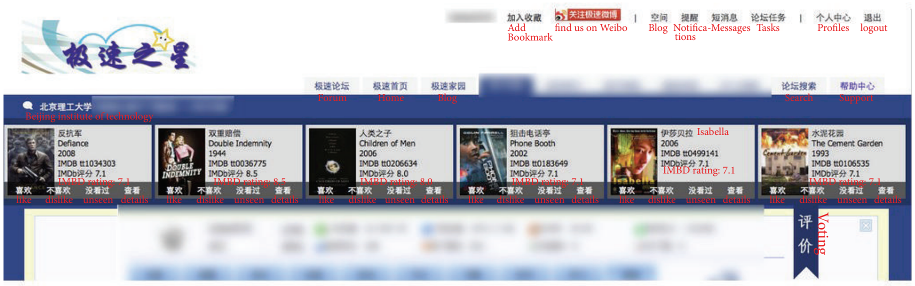

加入收藏 Add Bookmark
关注极速微博 find us on Weibo
空间 Blog
提醒 Notifica-tions
短消息 Messages
论坛任务 Tasks
个人中心 Profiles
退出 logout
极速之星
极速论坛 Forum
极速首页 Home
极速家园 Blog
论坛搜索 Search
帮助中心 Support
北京理工大学 Beijing institute of technology
反抗军 Defiance 2008 IMDB tt1034303 IMDb评分 7.1 IMBD rating: 7.1
双重赔偿 Double Indemnity 1944 IMDB tt0036775 IMDb评分 8.5 IMBD rating: 8.5
人类之子 Children of Men 2006 IMDB tt0206634 IMDb评分 8.0 IMBD rating: 8.0
狙击电话亭 Phone Booth 2002 IMDB tt0183649 IMDb评分 7.1 IMBD rating: 7.1
伊莎贝拉 Isabella 2006 IMDB tt0499141 IMDb评分 7.1 IMBD rating: 7.1
水泥花园 The Cement Garden 1993 IMDB tt0106535 IMDb评分 7.1 IMBD rating: 7.1
喜欢 like
不喜欢 dislike
没看过 unseen
查看 details
评价 Voting

FIGURE 10: Website overview.

There are a few users who made more than 20 votes. In those people who made more than 20 votes, we found that 4 users made all their votes do not like. Therefore, almost one third of the false positive votes were made by only 4 users over 216 users. Table 3 shows one example from one typical user. The first column is the movie ID from imdb, the second column is the voting results, and the third column is the voting time. We think maybe this user dislikes our recommendation idea so he made all the votes negative.

There is another small portion of users made more than 20 votes. Most of the votes are negative. Only a few votes are positive, and no zero or unseen votes. Table 4 shows one of such examples. It is probably because this user is voting negative instead of the unseen choices for the unseen movies, since most of the users were voting unseen more than like and do not like.

6.2.2. Second Evaluation Result. These two situations mentioned in Section 6.2.1 effect our result a lot in the first evaluation. Therefore, we made a second evaluation by calculating each user's accuracy in formula (8) and then computing the average of all user's accuracy. Consider the following:

$$\text{Accuracy} = \frac{\sum \text{accuracy}}{\text{Total Number of Users}}. \qquad (9)$$

TABLE 2: Comparison of results between different clustering methods in second evaluation method.

	Accuracy rate
AF	80.95%
DB	**84.71%**
HC	82.03%
RA	63.51%

In the second evaluation, we wait for another 6 hours to collect more data. There are 7,100 votes in total, 1420 votes are considered and 242 users participated in voting. Based on the second evaluation, we obtained much better result as shown in Table 2. DBSCAN (density-based clustering), affinity propagation, and hierarchical clustering obtained over 80% accuracy. DBSCAN reached to 84.71%.

6.3. Comparison with Related Work. In this section, we compared our result with those of with several other movie recommendation approach/systems. Pomerantz and Dudek [16] used a hierarchical Bayesian approach combined with the item similarity of the content for the movie recommender and achieved the classifiers of 70.8% as the best result. In Biancalana et al.'s paper [11], the best result is 82.4% obtained by combining three classifiers in a neural network. Basu et al.

Table 3: Example of all negative votes.

imdb	Vote	Last update
tt1716777	−1	2013-12-06 23:36:55
tt0183649	−1	2013-12-06 23:36:56
tt1728196	−1	2013-12-06 23:36:58
tt0014429	−1	2013-12-06 23:36:59
tt0034248	−1	2013-12-06 23:37:01
tt0017136	−1	2013-12-06 23:37:02
tt0035169	−1	2013-12-06 23:37:03
tt0023940	−1	2013-12-06 23:37:04
tt0035209	−1	2013-12-06 23:37:05
tt0047876	−1	2013-12-06 23:37:23
tt0106535	−1	2013-12-06 23:37:24
tt0121164	−1	2013-12-06 23:37:27
tt0006864	−1	2013-12-06 23:37:28
tt0499141	−1	2013-12-06 23:37:30
tt0049010	−1	2013-12-06 23:37:32
tt1298650	−1	2013-12-06 23:40:39
tt0408236	−1	2013-12-06 23:40:40
tt0162661	−1	2013-12-06 23:40:41
tt0349903	−1	2013-12-06 23:40:43
tt0114369	−1	2013-12-06 23:40:44
tt0096895	−1	2013-12-06 23:40:45
tt0111161	−1	2013-12-06 23:40:49
tt2992146	−1	2013-12-06 23:40:53
tt0082971	−1	2013-12-06 23:40:56
tt1075419	−1	2013-12-06 23:40:58

Table 4: Example of nonzero votes.

imdb	Vote	Last update
tt0068767	−1	2013-12-06 22:33:30
tt0109958	−1	2013-12-06 22:33:32
tt0047034	−1	2013-12-06 22:33:33
tt0397892	−1	2013-12-06 22:33:35
tt0032551	−1	2013-12-06 22:33:36
tt0110008	1	2013-12-06 22:33:38
tt0416881	−1	2013-12-06 22:33:40
tt1233499	−1	2013-12-06 22:33:42
tt1754691	−1	2013-12-06 22:33:45
tt1034303	−1	2013-12-06 22:33:47
tt0015324	−1	2013-12-06 22:33:48
tt0049408	−1	2013-12-06 22:33:50
tt0033467	−1	2013-12-06 22:33:51
tt0000417	−1	2013-12-06 22:33:54
tt1437358	−1	2013-12-06 22:33:56
tt0063105	−1	2013-12-06 22:33:57
tt0253474	−1	2013-12-06 22:33:58
tt0034248	1	2013-12-06 22:34:00
tt0018578	1	2013-12-06 22:34:04
tt0335345	−1	2013-12-06 22:34:07
tt0028445	1	2013-12-06 22:34:10
tt0040536	1	2013-12-06 22:34:16
tt0332379	−1	2013-12-06 22:34:19
tt1728196	−1	2013-12-06 22:34:21

[17] presented an inductive learning approach to recommendation and achieved an averaged precision of 83%.

Our best result (84.71%) achieved by DBSCAN [14] as shown in Table 2 improves 1.71% compared to best result of related works mentioned above. Meanwhile all these related works are computed offline and they use the stored models when testing, and these models cannot be updated in time since it takes too long to redo the learning process. Our approach, on the other hand, does not need to store user models and it updates the recommendation result in real time by preclustering movies and using a very simple learning approach (majority voting) to provide movie recommendation. Since movies are updated much less frequently than users' votes, our approach is much more time efficient than related works. To sum up, our research becomes significantly meaningful as it realizes the real time model updating while it does not compromise the accuracy.

7. Conclusion and Future Work

In this paper, we proposed a new distance function for publish year (year related) to compute the distance matrix, and then implemented a real time movie recommendation system based on content (movie) using preclustering and majority voting. We implemented four different clustering techniques in the preclustering process and obtained 84.71% accuracy as the best result, which is better than some of the other research papers' approaches. We realized a real time updating model with no compromises on accuracy. Our approach can even work better with special user groups such as people in colleges, universities, or professional communities.

The next step in this research is to make the system adaptive to be widely used by many other types of groups such as articles, news, and music, and to obtain better accuracy. One of the limitations in our approach is that if there is a cluster that contains a lot of positive votes as well as a lot of negative ones, we will not recommend movies in this cluster. However, this situation may happen because of two totally different reasons. One is that the user does not care about the movies in this cluster. In this case, no recommendation in this cluster is a wise choice. However, if the user really likes the movies in this cluster and he/she watched a lot of movies in this cluster, and some movies he/she likes while some he/she dislikes, ignoring this cluster, in such situation, will dissatisfy the user. To conquer this limitation, we can give weights to the votes from users. In the movie which is voted positive by the user and is voted positive by many other people as well, the weight of such vote will be decreased, while if the movie is voted positive by the user but is voted negative by most of others, this means such vote reveals the special taste from this user, and gains higher weight and vice versa. In a more specific way, we can set the weight of positive as $N/(P + N)$ and the weight of negative as $P/(P + N)$, where P is the total positive votes from all users towards this movie and N is the total negative votes from all users towards this movie. In this way, we can still ignore the cluster in first situation, while overcoming the limitation we had in the second situation.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors especially thank Dr. Khaled Rasheed for providing them with suggestions and information regarding their research. The authors also thank Mr. Hongfei Yan for the related work and discussions.

References

[1] M. A. Ghazanfar and A. Prügel-Bennett, “An improved switching hybrid recommender system using Naive Bayes classifier and collaborative filtering,” in *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS '10)*, pp. 493–502, March 2010.

[2] A. Bellogín, I. Cantador, P. Castells, and A. Ortigosa, “Discovering relevant preferences in a personalised recommender system using machine learning techniques,” in *Proceedings of the ECML-PKDD Workshop on Preference Learning*, 2008.

[3] F.-C. Lin, H.-W. Yu, C.-H. Hsu, and T.-C. Weng, “Recommendation system for localized products in vending machines,” *Expert Systems with Applications*, vol. 38, no. 8, pp. 9129–9138, 2011.

[4] K.-J. Kim and H. Ahn, “A recommender system using GA K-means clustering in an online shopping market,” *Expert Systems with Applications*, vol. 34, no. 2, pp. 1200–1209, 2008.

[5] S. B. Aher and L. M. R. J. Lobo, “Combination of machine learning algorithms for recommendation of courses in E-Learning System based on historical data,” *Knowledge-Based Systems*, vol. 51, pp. 1–14, 2013.

[6] G. M. Dakhel and M. Mahdavi, “Providing an effective collaborative filtering algorithm based on distance measures and neighbors' voting,” *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 5, pp. 524–531, 2013.

[7] W. H. Ujwala, V. R. Sheetal, and M. Debajyoti, “A hybrid web recommendation system based on the improved association rule mining algorithm,” *Journal of Software Engineering and Applications*, vol. 6, article 396, 2013.

[8] I. Tatli and A. Birtürk, “A tag-based hybrid music recommendation system using semantic relations and multi-domain information,” in *Proceedings of the 11th IEEE International Conference on Data Mining Workshops (ICDMW '11)*, pp. 548–554, December 2011.

[9] Q. Yang, J. Fan, J. Wang, and L. Zhou, “Personalizing web page recommendation via collaborative filtering and topic-aware Markov model,” in *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM '10)*, pp. 1145–1150, December 2010.

[10] X. Cai, M. Bain, A. Krzywicki et al., “Collaborative filtering for people to people recommendation in social networks,” in *AI 2010: Advances in Artificial Intelligence*, vol. 6464, pp. 476–485, Springer, Berlin, Germany, 2010.

[11] C. Biancalana, F. Gasparetti, A. Micarelli, A. Miola, and G. Sansonetti, “Context-aware movie recommendation based on signal processing and machine learning,” in *Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation*, pp. 5–10, October 2011.

[12] P. Jaccard, “The distribution of the flora in the alpine zone. 1,” *The New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[13] B. J. Frey and D. Dueck, “Clustering by passing messages between data points,” *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

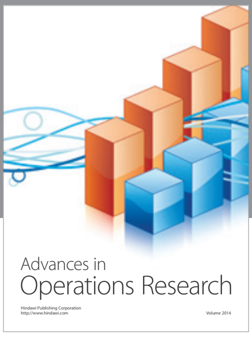
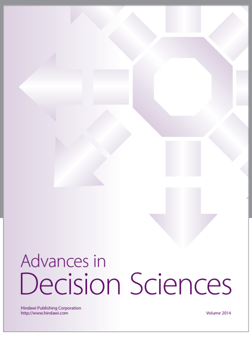
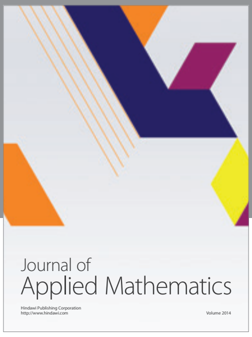
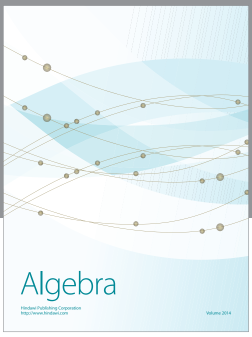
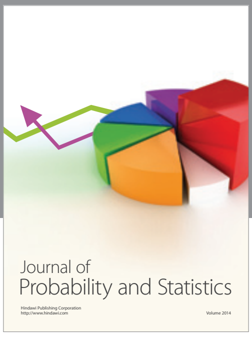
[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD '96)*, vol. 96, pp. 226–231, 1996.

[15] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Upper Saddle River, NJ, USA, 1988.

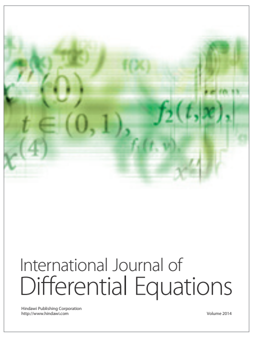
[16] D. Pomerantz and G. Dudek, “Context dependent movie recommendations using a hierarchical Bayesian model,” in *Advances in artificial intelligence*, vol. 5549 of *Lecture Notes in Computer Science*, pp. 98–109, Springer, Berlin, Germany, 2009.

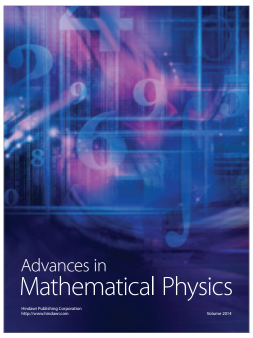
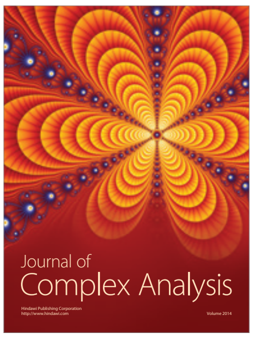
[17] C. Basu, H. Hirsh, and W. Cohen, “Recommendation as classification: using social and content-based information in recommendation,” in *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI '98)*, pp. 714–720, July 1998.

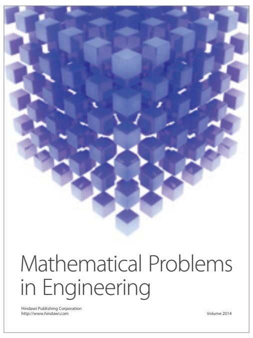
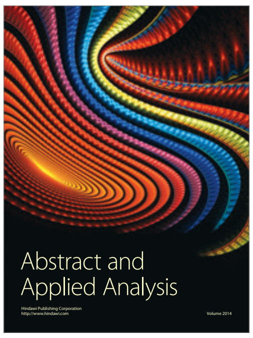
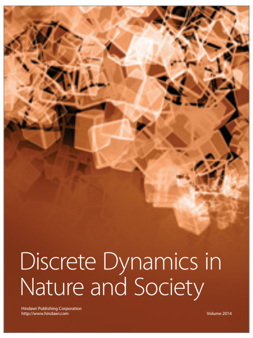
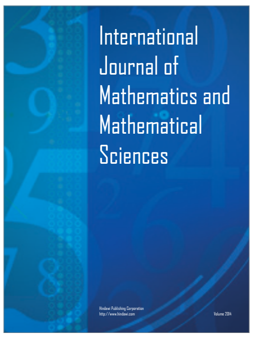
[18] “2.3. clustering—scikit-learn 0.14 documentation,” http://scikit-learn.org/stable/modules/clustering.html#clustering.


Advances in
Operations Research

Advances in
Decision Sciences

Journal of
Applied Mathematics

Algebra

Journal of
Probability and Statistics

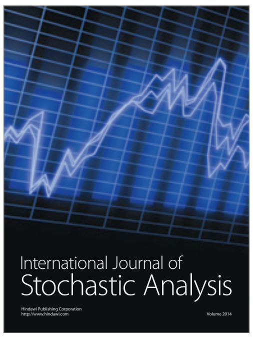
The Scientific
World Journal


International Journal of
Differential Equations

International Journal of
Combinatorics

Advances in
Mathematical Physics

Journal of
Complex Analysis

Journal of
Mathematics

Mathematical Problems
in Engineering

Abstract and
Applied Analysis

Discrete Dynamics in
Nature and Society

International
Journal of
Mathematics and
Mathematical
Sciences

Journal of
Discrete Mathematics

Journal of
Function Spaces

International Journal of
Stochastic Analysis

Journal of
Optimization