

A Survey Paper on Recommender Systems

Dhoha Almazro
Concordia University
d_almaz@encs.concordia.ca

Ghadeer Shahatah
Concordia University
g_shaha@encs.concordia.ca

Lamia Albdulkarim
Concordia University
l_alabd@encs.concordia.ca

Mona Kherees
Concordia University
m_khere@encs.concordia.ca

Romy Martinez
Ecole Polytechnique
romy.martinez@polymtl.ca

William Nzoukou
Concordia University
w_nzouko@encs.concordia.ca

ABSTRACT

Recommender systems apply data mining techniques and prediction algorithms to predict users' interest on information, products and services among the tremendous amount of available items. The vast growth of information on the Internet as well as number of visitors to websites add some key challenges to recommender systems. These are: producing accurate recommendation, handling many recommendations efficiently and coping with the vast growth of number of participants in the system. Therefore, new recommender system technologies are needed that can quickly produce high quality recommendations even for huge data sets.

To address these issues we have explored several collaborative filtering techniques such as the item based approach, which identify relationship between items and indirectly compute recommendations for users based on these relationships. The user based approach was also studied, it identifies relationships between users of similar tastes and computes recommendations based on these relationships.

In this paper, we introduce the topic of recommender system. It provides ways to evaluate efficiency, scalability and accuracy of recommender system. The paper also analyzes different algorithms of user based and item based techniques for recommendation generation. Moreover, a simple experiment was conducted using a data mining application - *Weka* - to apply data mining algorithms to recommender system. We conclude by proposing our approach that might enhance the quality of recommender systems.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval: Clustering

Keywords

Recommender Systems, Collaborative filtering, User Based, Item Based

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 2010 Team #6 : INSE6180 .

1. INTRODUCTION

Nowadays the amount of information we are retrieving have become increasingly enormous. Back in 1982, John Naisbitt observed that: "we are drowning in information but starved for knowledge." [13]. This "starvation" caused by having many ways people pour data into the Internet but not many techniques to process the data to knowledge. For example, digital libraries contain tens of thousands of journals and articles. However, it is difficult for users to pick the valuable resources they want. What we really need is new technologies that can assist us find resources of interest among the overwhelming items available.

One of the most successful such technologies is the Recommender system; as defined by M. Deshpande and G. Karypis: "a personalized information filtering technology used to either predict whether a particular user will like a particular item (prediction problem) or to identify a set of N items that will be of interest to a certain user (top-N recommendation problem)" [1].

Over the years, various approaches for building recommender systems have been created [3]; collaborative filtering has been a very successful approach in both research and practice, and in information filtering and e-commerce applications [22]. Collaborative filtering works by creating a matrix of all items and users' preferences. In order to recommend items for the target user, similarities between him and other users are computed based on their common taste. This approach is called user-based approach. A different way to recommend items is by computing the similarities between items in the matrix. This approach is called item based approach.

1.1 Background

Recommender system can be built with many approaches. Below are some of them:

- *Random prediction algorithm* is an algorithm that randomly chooses items from the set of available items and recommends them to the user. Since the item's selection is done randomly, the accuracy of the algorithm is based on luck; the greater the number of items is, the chance of good selection lowers. Random prediction has a great probability of failure. Thus, it has never been taken seriously by any researcher or vendor and only serves as reference point¹, helping to compare the

¹A similar algorithm to the Random-based algorithm is the

quality of the results obtained by the utilization of a more sophisticated algorithm [16].

- *Frequent sequences* can help build recommender systems. For example, if a customer frequently rates items we can use the frequent pattern to recommend other items to him. The only problem is that this method will only be efficient after the customer makes minimum purchases.
- *Collaborative filtering algorithms (CF)* are algorithms that require the recommendation seekers to express their preferences by rating items. In this algorithm, the roles of recommendation seeker (a user) and preference provider² are merged; the more users rate items (or categories), the more accurate the recommendation becomes.
In most CF approaches, there is a list of users $U = u_1, u_2, \dots, u_m$ and a list of items $I = i_1, i_2, \dots, i_n$. Each user u_i has a list of item I_{u_i} on which he has expressed his opinion [19].
- *Content based algorithms* are algorithms that attempt to recommend items that are similar to items the user liked in the past. They treat the recommendation's problem as a search for related items. Information about each item is stored and used for the recommendations. Items selected for recommendation are items that content correlates the most with the user's preferences [22]. For example, whenever a user rated an item, the algorithm constructs a search query to find other popular items by the same author, artist, or director, or with similar keywords or subjects [14]. Content based algorithms analyze item descriptions to identify items that are of particular interest to the user [18].

Many others approaches for recommender system exist. However, collaborative filtering algorithms have come to be the best of recommendation algorithms. As stated by Papageilis, collaborative filtering algorithms have “been extensively adopted by both research and e-commerce recommendation systems in order to provide an intelligent mechanism to filter out the excess of information available and to provide customers with the prospect to effortlessly find out items that they will probably like according to their logged history of prior transactions” [16]. *CF* algorithms have significant advantages over traditional content-based filtering; they can filter any type of content, e.g. text, artwork, music, mutual funds. They can also filter based on complex and hard to represent concepts such as taste and quality. They can to make serendipitous³ recommendations. *CF* algorithms do not depend on error-prone machine analysis of content [8].

1.2 Contribution

This paper has three primary research contributions:

1. Analysis of the user-based and item-based prediction algorithms.

so-called Rating-based Algorithm where instead of recommending items randomly, the system will recommend only the most popular items

²It is users who provide ratings for items

³serendipity: the lucky tendency to find interesting or valuable things by chance (Cambridge Advanced Learner's Dictionary, 2010)

2. Formulation of a hybrid model that uses both item based and user algorithm for more accurate prediction.
3. An experiment to show how from data mining we can deduce rules and make predictions.

1.3 Organization

Our work will be primarily based on *CF* algorithms. First, we introduce and describe collaborative filtering. Afterward, we talk two of the most used collaborative filtering algorithms; user-based and items-based algorithms. We continue by giving an example of an algorithm that is used in a commercial recommender system: the Amazon.com *item-to-item CF* algorithm. Following that, we discuss about some of the privacy and security issues related to recommender systems and also describe the metrics used to evaluate recommender. Subsequently, we continue by describing our own hybrid method. Finally we present the results of the experiments we did.

2. COLLABORATIVE FILTERING ALGORITHMS

The term “collaborative filtering” was first coined by Goldberg to describe an email filtering system called *Tapestry*⁴. *Tapestry* was an electronic messaging system that allowed users to rate messages (“good” or “bad”) or associate text annotations with those messages. Annotations and ratings could then be shared between users. Afterward a user could write some queries (on the annotations and on the ratings) to filter his messages. Although *Tapestry* provided good recommendations, it had one major drawback; the user was required to write complicated queries [3]. The first system to generate automated recommendations was the GroupLens⁵ system (Resnick et al. 1994; Konstan et al. 1997). The GroupLens system provided users with personalized recommendation on Usenet⁶ postings. It recommended articles found interesting by users similar to the target user.

Most of the *CF* algorithms are based on the concept of similarity. Some algorithms (like the GroupLens system) compute the similarity between users, others look at the similarity between items, others at the similarity between categories of items. Before we can understand how *CF* algorithms work, we need to understand this *similarity*.

2.1 Similarity

Similarity (closeness) is define by data analysis in a term of a distance⁷ function such as the Euclidean (Equation 1) and the Manhattan (Equation 2).

$$d(i, j) = \sqrt{(x_{i1} - x_{j1})^2 + \dots + (x_{in} - x_{jn})^2} \quad (1)$$

$$d(i, j) = |x_{i1} - x_{j1}| + \dots + |x_{in} - x_{jn}| \quad (2)$$

In those distance functions, the difference between corresponding values of attributes in tuples i and j are taken.

⁴Tapestry was the first recommendation support system to be made. It was build at Xerox®Parc which also famous for inventing graphical operating system[21]

⁵GroupLens Research is a research lab at the University of Minnesota that specialize in recommender systems.(Wikipedia , 2010)

⁶Usenet is a worldwide distributed Internet discussion system where users read and post public messages to one or more categories, known as newsgroups. (Wikipedia 2010)

⁷the more distant objects are, the less similar they become

Typically attributes are normalized so that attributes with larger values do not outweigh attributes with smaller values.

The Euclidean and the Manhattan distance trivially work well and can help us compute the similarity for tuples that have attributes with numerical values. However if the attribute is categorical such as color, we need more sophisticated methods to differentiate the grading (for example color *blue* vs *black*) [7]. Some of those methods are cosine-based similarity, Conditional Probability-Based Similarity and Pearson correlation Similarity.

2.1.1 Cosine-Based Similarity

In this approach, items are thought of as vectors in the m dimensional user-space where the dimension is the attribute by which the item are rated. The cosine of the angle between the vectors that represent two items is their similarity (see Figure 1). We know from calculus the dot-product formula:

$$\vec{i} \cdot \vec{j} = \|\vec{i}\| \cdot \|\vec{j}\| \cdot \cos \Theta$$

$$\Rightarrow \text{sim}(i, j) = \cos \Theta = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\| \cdot \|\vec{j}\|}$$

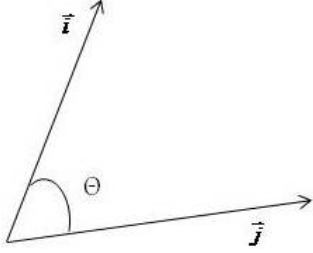


Figure 1: Vector representation of items. The more Θ is small, the more similar are the items

2.1.2 Conditional Probability-Based Similarity

Another way to compute the similarity is to use a measure that is based on the conditional probability of liking (or rating) an item given that the user already showed his interest for another item. If an item i has a good chance of being purchased after an item j was purchased then i and j are similar. The similarity is given by $\text{sim}(i, j) = P(i|j) \times \alpha$ where α is a factor dependent on the problem [3].

2.1.3 Pearson correlation Similarity

The similarity is given by the amount of correlation between the items or users. The correlation is computed with the Pearson formula (equation (3)). If the set of users who both rated i and j are denoted by U then the correlation similarity is given by:

$$\text{sim}(i, j) = \text{corr}_{ij} \quad (3)$$

$$= \frac{\sum_{u \in U} (R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U} (R_{u,i} - \bar{R}_i)^2} \sqrt{\sum_{u \in U} (R_{u,j} - \bar{R}_j)^2}}$$

Experimentations have shown that Pearson correlation function performs better than cosine vector similarity (Breese et al. 1998), Spearman correlation, entropy-based uncertainty (Herlock et al. 1999). Pearson correlation is the most

used similarity function in the two approaches of *CF* based recommender; user-based or memory-based and item-based or model based [3].

2.2 User-Based algorithms

User based algorithms are *CF* algorithms that work on the assumption that each user belongs to a group of similar behaving users. The basis for the recommendation is composed by items that are liked by users. Items are recommended based on users tastes (in term of their preference on items). The algorithm considers that users who are similar (have similar attributes) will be interested on same items [4]. User based algorithms are three steps algorithm; the first step is to profile every user in order to find which ones are similar to the target user, the second step is to compute the union of the items selected by these users and associate a weight with each item based on its importance in the set and the third and final step is to select and recommend items that have the highest weight and have not been already selected by the active user [3]. The most important step is the first one; creating the union of items liked by others or selecting the most important of them is easily done when the set of similar users is known [6]. Thus the overall performance of the algorithm will depend on the method used to find users that are similar to the target user. There are many methods by which it can be done. the k -Nearest Neighbors algorithm is the most used because of its efficiency [3].

k -Nearest Neighbors algorithm is a lazy learner classification algorithm. The algorithm requires to be provided with a training data set; a set of users who are well categorized. Then for a given user, it will compare that user's attribute with all the user in the training data set to find which ones are similar to him. The similarity between users can then be calculated using Pearson correlation (see the above sections to see why Pearson correlation is better than cosine-based and other similarity functions). Two approaches can be used to compute the similarity between users; explicitly and implicitly.

2.2.1 Prediction based on explicit ratings

In this case, users are required to express their ratings on items. This process sometimes happens through a form or a control panel. Let $I' = i_x : x = 1, 2, \dots, n' \wedge n' \leq n$ where n is the total number of items in the database the set of items that users u_x and u_y have both rated. The similarity between u_x and u_y is given by [16]:

$$\kappa_{x,y} = \text{sim}(u_x, u_y)$$

$$= \frac{\sum_{h=1}^{n'} (r_{u_x, i_h} - \bar{r}_{u_x})(r_{u_y, i_h} - \bar{r}_{u_y})}{\sqrt{\sum_{h=1}^{n'} (r_{u_x, i_h} - \bar{r}_{u_x})^2} \sqrt{\sum_{h=1}^{n'} (r_{u_y, i_h} - \bar{r}_{u_y})^2}}$$

2.2.2 Prediction based on implicit ratings

Implicit rating does not mean that a user will not show his appreciation toward an item, it simply means that he does not do it directly or explicitly as with the preceding approach. The rating of each item is captured implicitly. For example, if a user spend more time looking on an item, the item get an high rating. Another example is that an item will also get a high rating if an user repeatedly come look it.

M. Papagelis and D. Plexousakis define a Pearson Correlation function for a recommender where the item

rating is captured by looking at the explicit rating the users gave to the categories.

Let $C' = c_x : x = 1, 2, \dots, n' \wedge n' \leq n$ where n is the total number of categories in the database that users u_x and u_y have both rated. The similarity between u_x and u_y is given by [16]:

$$\lambda_{x,y} = \text{sim}(u_x, u_y) = \frac{\sum_{h=1}^{n'} (r_{u_x, c_h} - \bar{r}_{u_x})(r_{u_y, c_h} - \bar{r}_{u_y})}{\sqrt{\sum_{h=1}^{n'} (r_{u_x, c_h} - \bar{r}_{u_x})^2} \sqrt{\sum_{h=1}^{n'} (r_{u_y, c_h} - \bar{r}_{u_y})^2}}$$

where $c_x; x = 1, 2, \dots, p$ are the available categories.

After users have been clustered, the algorithms pursue by finding popular items between those users and recommend them[5].

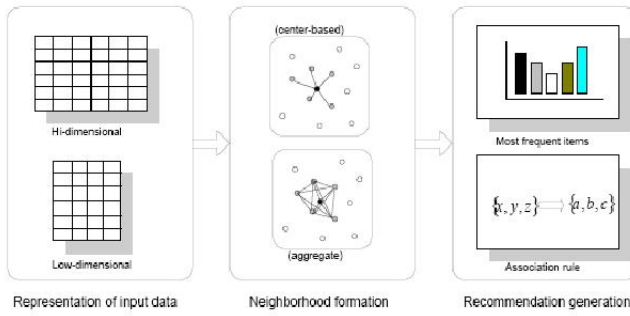


Figure 2: The user-based collaborative filtering process. Image taken from *Analysis of Recommendation Algorithms for ECommerce* by B. Sarwar, Ge. Karypis, J. Konstan, and J. Riedl

Although user based algorithms are very efficient and give good results, they suffer some drawbacks;

- *sparsity*: The number of users and items in major e-commerce website is very large. Most of the users however only rated a small portion of the total items available; even very popular items result in having been rated by only a few of the total number of users. This means that the user-item matrix is very sparse and has a lot of 0 element. Because of that it is possible that the similarity between two users cannot be defined thus making the algorithm useless [17]. Even when the evaluation of similarity is possible, it may not be very reliable, because of insufficient information processed.
- *scalability*: Because finding the optimal clusters of users over large data sets is impractical. Most user-based recommenders use various forms of greedy cluster generation algorithms such as Lazy learner k-nearest neighbors. These cluster generation algorithms require a lot of computations⁸ that grow linearly with the numbers of users and cannot be precomputed because users and items are changing over time in the database. Moreover, since the user based algorithms must compute

⁸For example in the k-nearest neighbors algorithm, finding the optimal k requires a large amount of computations

the k-nearest neighbors for every users browsing the system, the latency (waiting time) for each recommendation will increase and may it affect real-time performance of the system [3]. The conclusion is that user-based algorithm do not scale well and are not suitable for large databases of users and items.

2.3 Item-Based algorithms

Because of the problems mentioned above with the user-based recommender systems, item-based (or model-based) recommender were developed.

Item-based recommender are a type of *collaboration filtering* (CF) algorithms that look at the similarity between items to make a prediction. The idea is that a user is most likely to purchase items that are similar to the one he already bought in the past; so by analyzing the purchasing information we can have an idea about what he may want in the future (Deshpande, Karypis 2004). Analyzing the historical information can be done explicitly (by looking at the explicit ratings users made on the items) or implicitly (for example through the user browsing information or the rating on categories of item).

Item-based algorithms are two steps algorithms; in the first step, the algorithms scan the past informations of the users; the ratings they gave to items are collected during this step. From these ratings, similarities between items are built and inserted into an item-to-item matrix M . The element $x_{i,j}$ of the matrix M represents the similarity between the item in row i and the item in column j . Afterward, in the final step, the algorithms selects items that are most similar to the particular item a user is rating. Deshpande and Karypis give a method to construct M (Algorithm 1) after computing the similarities between the items. For each

Algorithm 1

```

for  $j \rightarrow 1$  to  $m$  do
  for  $i \rightarrow 1$  to  $m$  do
    if  $i \neq j$  then
       $M_{i,j} \rightarrow \text{sim}(R_{*,j}, R_{*,i})$ 
    else
       $M_{i,j} \rightarrow 0$ 
    end if
  end for
  for  $i \rightarrow 1$  to  $m$  do
    if  $i \neq$  among the  $k$  largest values in  $M_{*,j}$  then
       $M_{i,j} \rightarrow 0$ 
    end if
  end for
end for

```

item j , the algorithm computes the similarity between j and the other items and stores the results in the j^{th} column of M (line 1). After that it zero-all the entries in M that less similarity than the k^{th} largest similarity. The second inner for-loop makes sure that an item does not recommend itself.

Similarity in item based collaborative filtering can also be computed following two approaches: implicit or explicit.

2.3.1 Prediction based on explicit ratings

As stated before, this approach requires users to specifically rate (give their opinion) on items.

Let $U' = u_x : x = 1, 2, \dots, m' \wedge m' \leq m$ where m is the total number of users in database, the set of users that have

both rated item i and item j , the Pearson correlation coefficient of their associated columns in the user-item matrix and is given by the following formula [16].

$$\text{sim}(i, j) = \frac{\sum_{h=1}^{m'} (R_{u_h, i} - \bar{R}_i)(R_{u_h, j} - \bar{R}_j)}{\sqrt{\sum_{h=1}^{m'} (R_{u_h, i} - \bar{R}_i)^2} \sqrt{\sum_{h=1}^{m'} (R_{u_h, j} - \bar{R}_j)^2}}$$

$R_{u_h, i}$ is the explicit rating given by a user u_h to an item i . And \bar{R}_i is the average of the ratings given on item i .

2.3.2 Prediction based on implicit ratings

As with the implicit user based algorithm (see section 2.2.2), the ratings given to items can be implicitly captured.

M. Papagelis and D. Plexousakis computes the similarity between two items as the Pearson correlation coefficient of their associated rows in the item-category bitmap matrix⁹.

$$\text{sim}(i, j) = \frac{\sum_{h=1}^p (v_{c_h, i} - \bar{v}_i)(v_{c_h, j} - \bar{v}_j)}{\sqrt{\sum_{h=1}^p (v_{c_h, i} - \bar{v}_i)^2} \sqrt{\sum_{h=1}^p (v_{c_h, j} - \bar{v}_j)^2}}$$

p is the number of categories and $v_{c_h, i}$ is a Boolean value that equals to 1 if the item i belongs to the category h or equals to 0 otherwise.

Compared to the user-based algorithms, item-based algorithms sparse better and scale well. Their major disadvantage is the cost to build the item-to-item matrix M . If we recall section 2.3, then we see that in order to construct M , we need to compute the similarity between every pair of items. Once this is done, item-based algorithms perform more rapidly and scale better than the user-based algorithms. Despite their slowness, experiments have shown that user-based algorithm produce more accurate recommendation than item-based algorithms [3].

The choice of the algorithm will then be based on how much trade-off can be made between the prediction performance and the scalability.

2.4 The Amazon.com example

Amazon.com is a e-commerce website in which users can buy books, music and others goods. It has a databases containing more than 29 million customers and several million catalog items.

Amazon.com use a algorithm based on item-based collaborative filtering to make their recommendations. Their algorithm, called *item-to-item* collaborative filtering, works by first matching each of the user's purchased and rated items to similar items (as with the item based *CF*, this is use to create an item-to-item matrix where elements are the similarities between items). Afterward, it combines those similar items into a recommendation list [14]. The most similar items are found using algorithm 2.4 (G. Linden, B.Smith, and J. York, 2003).

To improve the scalability and the performance, Amazon.com has built its recommender as two components. An offline component that creates the expensive and costly item-to-item matrix offline. The other component is the online component that look at the item-to-item matrix to produce the recommendations. The online component is dependent only on how many titles the user has purchased or rated [14].

⁹“item-category bitmap matrix is a matrix of items against categories that have as elements the value 1 if the item belongs to the specific category and the value 0 otherwise.” (M. Papagelis, D. Plexousakis, 2005)

Algorithm 2 The most similar items algorithm. *Amazon.com computes the similarity using cosine measure*

```

for each item in product catalog,  $I_1$  do
  for each customer  $C$  who purchased  $I_1$  do
    for each item  $I_2$  purchased by customer  $C$  do
      Record that a customer purchased  $I_1$  and  $I_2$ 
    end for
  end for
  for each item  $I_2$  do
    Compute the similarity between  $I_1$  and  $I_2$ 
  end for
end for

```

3. EVALUATION

User satisfaction is the most important factor of the success of a recommender system which is an accurate recommendation within a reasonable time. In commercial systems, it is measured by number of recommended items that has been bought (and of course not returned!)[9]. For non-commercial systems, it is measured by asking for users' feedback. To properly employ a recommender system, it is important to study the domain for which it is being used [9].

This section will focus on evaluating recommender systems for different systems. We will then introduce three important metrics for evaluating the quality of recommender systems. Finally, we will address the challenges of employing recommender systems.

3.1 Different Systems, Different Algorithms

Recommender systems differ based on the type of application used. Therefore, a certain algorithm may work very well on a dataset and work poorly on different data set. In other words, some algorithms work well in situations where items are more than users (e.g. a recommender system that suggest tens of thousands of research papers to thousands of users). Other algorithms are designed for the opposite situation where users are more than items (e.g. MovieLens, a system for recommending movies, has a data set of 65000 users and 5000 movies) [9]. Furthermore, Recommender Systems varies according to the nature of data sets. The static nature of items allows us to pre compute and store some of the values of the algorithm. However, the same technique is not efficient for items with a dynamic nature [19]. In some cases where similarities are way more than the dissimilarity, it is efficient to compute the dissimilarity and extract the similarity afterwards [6].

3.2 Recommendation Metrics

Items and users are getting increased in systems where recommender systems utilization is crucial. To ensure user satisfaction all the time, algorithms must not work on thousands, but millions of item within reasonable time [23]. Therefore, recommender systems must cope with the growth by making the suggestions more accurate, efficient and scalable.

Accuracy

This is measured by how close the result of a recommendation matches a user's preference. Accuracy is the most important metric in evaluating the quality if a recommender system because this is what all is about: understanding the user and suggesting what he really likes

or what he is looking for precisely to gain the user's trust [23].

There are two measures for evaluating the accuracy of a recommender system:

- **Statistical Accuracy Metric:** This compares the numerical recommendation scores against the actual user rating. One of the widely used metrics is the Mean Absolute Error (MAE); the lower the value of MAE the more accurate the result is [19].
- **Decision Support Accuracy:** which measure how effective the prediction engine is at helping a user selecting high- quality item from the set of all items. Receiver Operating Characteristic (ROC) is one of the metrics that help assessing the accuracy of predictions [19].

An interesting point regarding accuracy was pointed out by many researchers. Very accurate recommender systems are not always good! An example is an online travel agency that recommends destinations that has already been visited by a user. Yes the recommendation was accurate enough but it was not useful she already visited these places [15]. This means that recommendation must be accurate in predicting the upcoming actions of a user not only knowing him. Moreover, the recommendations that are most accurate according to the standards metrics are sometimes not the recommendations that are useful to users [15].

Efficiency

In order for a recommender system to be reliable, not only it must be accurate, but also it must process within a reasonable time, make good use of the available resources, and handle hundred requests per second [23]. Memory and Computation time are two important metrics that evaluate the efficiency of a recommender system.

Algorithms that work with item sets that has a static nature tend to pre-compute item similarity and stores a matrix of similarities. The more the items, the bigger the matrix will grow. Therefore, we will end up with a quick look up table that speeds up the recommendation process; however, an $O(n^2)$ space is needed for n items [19]. Because of the space problem we may not consider all the n items of a system. Instead, we only consider a small fraction of the most similar items k where ($k < n$). This attempt will reduce the size of the lookup table but we will have a trade-off: smaller model size means a reduced quality [19]. Another approach to efficiently allocate space needed is to give each item a space according to the amount of rating. In other words, the more an item has ratings, the more space I allocate [10].

In some situation, the knowledge of customer preferences changes, memory consumption reduces and the time used for computation increases, therefore the efficiency of the recommender system in dynamic datasets depends on the amount of calculation required in an algorithm [23]. In this situation, two calculations must be performed: learning time and running time. In some cases, running time was fast but learning time was 8 hours. To speed up the calculation, we consider a relevant dataset rather than the whole database; again, a trade-off between the accuracy and efficiency. Another approach to speed up the calculation is to use data

structure or other data mining techniques such as hierarchical clustering since searching for neighbors is faster than scanning the whole tree [23].

Scalability

A good recommendation algorithm that handles thousands of request, must also handle hundred of thousand requests in the future. Despite the accuracy and efficiency of many algorithms, they are not coping with the growth of data sets. Therefore, in order to manage the vast increase in number of users and items, a trade-off between the prediction performance and scalability is inevitable. Again, this is done by considering a portion of the whole dataset with similar characteristics.

One of the best approaches for maintaining accuracy, efficiency, and scalability is to use hashing techniques. It compresses large data sets, scale very large number of users, and obtain a good performance within a reasonable time [10].

3.3 Challenges of Recommender Systems

If recommender systems rely only on items that have been rated, then it is missing a lot of good items for recommendation that are hidden because no one has rated them. This is called the Coverage metrics which is the percentage of items for which a recommender agent can provide predictions [16]. This is one of the problems that face systems that employ recommender systems. Another challenge is the sparsity issue which is rating few of the total number of items [16]. For systems that has just established, they are facing the cold start problem where the recommender system is unable to accurately recommend items due to the fact that only few rating has been performed on items. Noise, data redundancy, and overfitting are also other challenges of recommendation agent [23].

In order to reduce the sparsity problem, some researchers have proposed a compensation system by which users are rewarded for providing ratings to items. Others have proposed to capture the ratings by implicitly look at the user's behavior [20]. Another approach to solve the sparsity problem is to rely filtering agent called *filterbots* or dynamic agents to automatically rate items [20].

4. SECURITY AND PRIVACY ISSUES

Collaborative filtering *CF* recommender requires personal information from a user to give personalized recommendations. The more users express their preferences on items, the more accurate the recommendation they receive become. As with any data mining systems, users must trust the recommender to protect their information appropriately. Moreover, since the user does not know how the recommendation is performed, he should trust the accuracy of the recommender[11]. The recommender should not violate the trust of the users.

4.1 Privacy Risks

In most systems, users need to register before they can enjoy personalized recommendation. The registration process often requires them to provide some personal informations like their names, birth dates, postal code and email. Combinations of those required fields (attributes) may be highly identifying (*Quasi-identifier*¹⁰). Personal preferences

¹⁰Quasi-identifier: "Variable values or combinations of vari-

like those expressed to many recommender systems may become quasi-identifier, especially if some users express unusual preferences (S. Lam, D. Frankowski, and J. Riedl, 2006). User's preferences could then be used to re-identify him in another system. For example, a company like Netflix could use the preferences some users saved in its system to find them on a competitor website.

Since not every users want their information to be disclosed or misused, the recommender should then protect itself against exposition of users informations or misuse of those informations. Recommender systems are also confronted with other type of problems such as security's related problem. Since being recommended is often promise of good selling, recommender are often target of manipulation from producers or malicious users [2]. For example, a book writer may try to alter the recommendation so that his book get recommended. Recent research by Dellarocas and others have shown that even popular systems such as Amazon and eBay have (and are) being manipulated [12]. *Shilling attacks* are one of the most discussed method by which the prediction of a recommender can be bias.

4.2 Shilling Attacks

A shilling attack is an attack in which the system's recommendations for a particular item is manipulated by submitting misrepresented opinions to the system (S. Lam, D. Frankowski, and J. Riedl, 2006). The attack can have two objectives: decrease the ratings of all the items outside its target item-set (*push attack*) to make them more recommended. He may also increase the ratings (*nuke attack*) of other items to make its target item-set less recommended. Two simple types of shilling attacks are *RandomBot* and *AverageBot*.

- A *RandomBot* is filterbot who randomly rate items outside of the target item-set with either the minimum rating (for nuke attack) or maximum rating (for push attack).
- An *AverageBot* is a filterbot where the rating is based on the average rating of each item following a normal distribution with a mean equal to the average rating for that item.

Another type of attack that may affect recommender are the so called *Sybil attack* in which a dishonest user may create multiples users account in other to improve the recommendation of another user or another item.

Recommender shall then provides ways to protect itself against those attacks since they are well known. Some systems provided CAPTCHA¹¹ to stop *filterbots* from corrupting the ratings.

5. OUR APPROACH

After studying collaborative filtering with its both approaches item-based and user-based, we found that each approach has its advantages and disadvantages. Thus, we are able values within a dataset that are not structural uniques but might be empirically unique and therefore in principle uniquely identify a population unit.”(OECD, Glossary of statistical term, 2010)

¹¹“A CAPTCHA or Captcha is a type of challenge-response test used in computing to ensure that the response is not generated by a computer” (Wikipedia, 2010)

proposing a new hybrid technique that combines the two approaches and trying to come up with a new approach that is more accurate and efficient.

The proposed approach starts by clustering all items and users based on demographic information. In other word, items and users will be categorized based on users personal attributes and make recommendations based on demographic categorization. Clustering techniques work by identifying groups of users and groups of items which appear to have similar preferences.

After applying the clustering technique, the next step is to extract the suitable clusters for both item based algorithm and user based algorithm. The item based algorithm will measure the similarities between a target user's preference and the items we have in the cluster. The user based algorithm will measure the similarities between the target user and other users in its cluster.

The results of both algorithms are listed it terms of items. These items will be ranked from the most appropriate to the least appropriate for the target user. Then, the items in both item sets will be merged in one item set also depending on the rank that each item got in the step before. Finally, the recommendation of top-k items will be generated to the user. Figure 3 illustrates the new hybrid approach.

As mentioned earlier, recommender systems must cope with the growth of items and users by making suggestions more accurate, efficient and scalable. Hopefully our approach is able to handle the massive growth in a way that ensures user satisfaction.

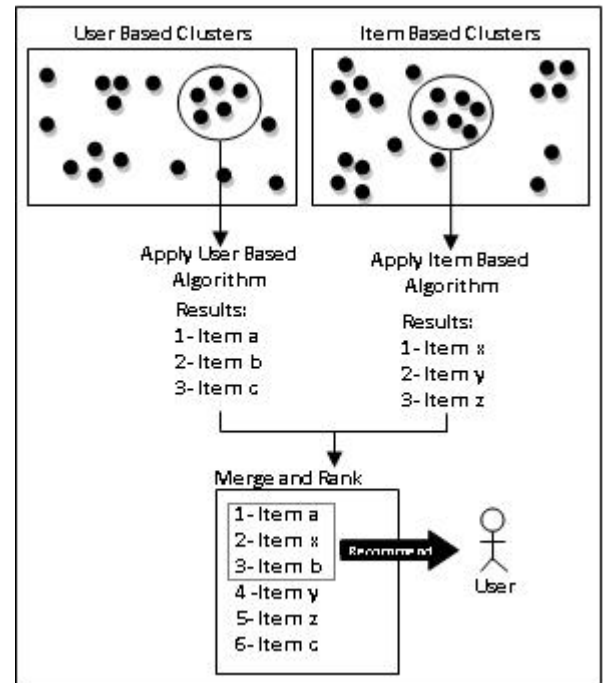


Figure 3: Our hybrid approach

In terms of **accuracy**, by employing item based and user based together and ranking both results, we are extracting

the best of both methods and suggesting the most accurate items to users.

In terms of **efficiency**, the proposed approach will not go to deal with the whole database and will only deal with a portion of it due to the clustering technique that will be implemented before applying the proposed technique. Therefore, the amount of computation and memory will be much less, and it will speed up the calculation of the recommendation.

In terms of **scalability**, the proposed approach will not have a problem with scalability since item based algorithms is still going to be implemented and is able to handle the scalability issue. Moreover, applying a hashing technique will make the proposed system able to absorb the growth of users and items.

6. EXPERIMENTS

We were able to implement a recommender system based on a user's profile as well as on an item based profile. To do so, we used the Java open-source program named *Weka*. *Weka* provides environment for comparing learning algorithms, graphical user interface, comprehensive set of data pre-processing tools, learning algorithms and evaluation methods. Furthermore *Weka* provides implementation of Regression, Clustering, Classification, Association rules and feature selection¹². As part of our experiment we used the classification algorithm *J48* which is an open source Java implementation of the C4.5¹³ algorithm in *Weka* (Wikipedia,2010). *Weka* also provides many methods for loading data such as (ARFF) or (CSV) file, in our experiment we use a file in CSV format.

6.1 Dataset

In the first part of our experiment, we inputted a ".csv" file containing the following parameters: UserID, Age, Gender, Student, Have children, Movie category. The table below (Table 1) provides further details of each parameter.

In the second part , the ".csv" file contained the following parameters: UserID, Movie title, Movie categories: Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War. Table 2 provides further details of each parameter.

6.2 Data cleaning and preparation

Both datasets were populated using the spawner¹⁴ program. After establishing which parameters we wanted to generate, we inputted them in spawner and we got some sample datasets. Each datasets contained 300 data. Each datasets were modified manually in order to make sure that the dataset were coherent and logical. The movies titles in the item datasets were identical to titles provided by the MovieLens website. It is important to note that the same movie title can be viewed and rated by different users.

¹²IBM@developerworks website

¹³C4.5 is an algorithm used to generate a decision tree that was developed by Ross Quinlan from his earlier ID3 algorithm. (Wikipedia,2010)

¹⁴Spawner is a software that can generate sample based on certain criteria

Once we generated the datasets, user and item sets, we made sure that some entries were blanks in order for weka to perform some predictions. For the user data set, weka will predict a movie category whereas for the item data set, it will predict a movie title according to which category the movie belong to. Once this step was performed, the pre-processing steps consisted of three main steps: opening the file with weka, selecting all the other attributes and finally choosing the attribute as a class attribute. These pre-processing steps were important in order to insure that the appropriate data were used in our experiments.

6.3 Results

The implementation of the recommender system was done using two different input files. Indeed, our objective was to recommend a movie title to a user according to his profile and also based on the movie category that movie belong to.

Using this result (Figure 4), we are able in the first part of the experiment, to predict what kind of movie a certain user with specific characteristics would like. For instance, an adult, who is also a female student with children would most likely like an animation type movie. On the other hand, a male teenager who is a student without children would rather an action movie. Therefore, we are able to recommend a certain movie category based on a user's profile. We used a user based algorithm based on demographics to generate our data. The accuracy for this experiment was 61.43% of correctly classified.

Using this result (Figure 5), we are able in the second part of the experiment, to recommend a movie title according to what type of movie category that movie belongs to. For instance, if a user normally selects movies that are comedies, animations and a children's movies, we would recommend "Alladin and the king of thieves". Furthermore, if a user rated a movie as drama and science fiction but not as a comedy and adventure, we would recommend "Twelve Monkeys". Again, here we were able to recommend movie titles based on what users normally selects. The accuracy for this experiment was 66.01%.

With these experiments we were able to demonstrate recommender systems using weka and trying to reproduce the user based algorithm (based on demographics) and item based algorithms.

7. CONCLUSION

This report presented some of the algorithms used to build recommender systems. Each of these algorithm has its advantages and disadvantages; user-based algorithms are accurate but not scalable, item-based algorithms are scalable but not precise as the user-based. Research on recommender system is mainly focused on finding ways to improve the performance, scalability or accuracy of the algorithms. Hybrid algorithms that combine features of user-based and item-based algorithms have been created. Other approaches using Rough Set Prediction, Slope One Scheme Smoothing and other methods have been developed to build item-based and user-based algorithms. As with any systems that contains data on users, recommender systems have some privacy and security issues to deal with.

In conclusion, recommender systems are powerful systems that give an added-value to business and corporation. They are a relatively recent technology and they will only keep improving in the future.

8. ACKNOWLEDGMENTS

The authors would like to thank Dr. Benjamin Fung for his insightful help during the writing of this paper. This paper was produced using the ACM SAC 2010 L^AT_EX₂ ϵ template.

9. REFERENCES

- [1] BIGDELI, E., AND BAHMANI, Z. Comparing accuracy of cosine-based similarity and correlation-based similarity algorithms in tourism recommender systems. In *Management of Innovation and Technology, 2008. ICMIT 2008. 4th IEEE International Conference on* (21-24 2008), pp. 469–474.
- [2] CHIRITA, P.-A., NEJDL, W., AND ZAMFIR, C. Preventing shilling attacks in online recommender systems. In *WIDM '05: Proceedings of the 7th annual ACM international workshop on Web information and data management* (New York, NY, USA, 2005), ACM, pp. 67–74.
- [3] DESHPANDE, M., AND KARYPIS, G. Item-based top- n recommendation algorithms. *ACM Trans. Inf. Syst.* 22, 1 (2004), 143–177.
- [4] GHAZANFAR, M. A., AND PRUGEL-BENNETT, A. A scalable, accurate hybrid recommender system. *International Workshop on Knowledge Discovery and Data Mining 0* (2010), 94–98.
- [5] GODOY, D., AND AMANDI, A. An agent-based recommender system to support collaborative web search based on shared user interests. In *CRIWG* (2007), pp. 303–318.
- [6] GROUP, G. R., SARWAR, B., KARYPIS, G., KONSTAN, J., AND RIEDL, J. Analysis of recommendation algorithms for e-commerce. ACM Press, pp. 158–167.
- [7] HAN, J., AND KAMBER, M. *Data Mining: Concepts and Techniques*, second ed. Elsevier, 2006.
- [8] HERLOCKER, J. L., KONSTAN, J. A., AND RIEDL, J. Explaining collaborative filtering recommendations. In *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work* (New York, NY, USA, 2000), ACM, pp. 241–250.
- [9] HERLOCKER, J. L., KONSTAN, J. A., TERVEEN, L. G., AND RIEDL, J. T. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* 22, 1 (2004), 5–53.
- [10] KARATZOGLOU, A., SMOLA, A., AND WEIMER, M. Collaborative filtering on a budget, 2010.
- [11] LAM, S. K., FRANKOWSKI, D., AND RIEDL, J. Do you trust your recommendations? an exploration of security and privacy issues in recommender systems. In *ETRICS* (2006), pp. 14–29.
- [12] LAM, S. K., AND RIEDL, J. Shilling recommender systems for fun and profit. In *WWW '04: Proceedings of the 13th international conference on World Wide Web* (New York, NY, USA, 2004), ACM, pp. 393–402.
- [13] LAROSE, D. *Discovering knowledge in Data*. Wiley Interscience, 2008.
- [14] LINDEN, G., SMITH, B., AND YORK, J. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7 (2003), 76–80.
- [15] MCNEE, S. M., RIEDL, J., AND KONSTAN, J. A. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems* (New York, NY, USA, 2006), ACM, pp. 1097–1101.
- [16] PAPAGELIS, M., AND PLEXOUSAKIS, D. Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents. *Engineering Applications of Artificial Intelligence* 18, 7 (2005), 781 – 789.
- [17] PAPAGELIS, M., PLEXOUSAKIS, D., AND KUTSURAS, T. Alleviating the sparsity problem of collaborative filtering using trust inferences. 2005, pp. 224–239.
- [18] PAZZANI, M., AND BILLISUS, D. Content-based recommendation systems. 2007, pp. 325–341.
- [19] SARWAR, B., KARYPIS, G., KONSTAN, J., AND REIDL, J. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web* (New York, NY, USA, 2001), ACM, pp. 285–295.
- [20] SARWAR, B. M. Using semi-intelligent filtering agents to improve prediction quality in collaborative filtering systems, 1998.
- [21] TERVEEN, L., AND HILL, W. Beyond recommender systems: Helping people help each other, 2001.
- [22] VAN METEREN, R., AND VAN SOMEREN, M. Using content-based filtering for recommendation.
- [23] YU, K., XU, X., TAO, J., ESTER, M., AND KRIEGLER, H.-P. Instance selection techniques for memory-based collaborative filtering, 2002.

Parameter	Description
UserID	User who provided his favorite movie category, denoted by a numerical value
Age	denoted by Kid, Teenager and Adult
Gender	denoted by a "M" for male and "F" for female
Student	denoted by a "Y" for Yes and "N" for No
Having Children	denoted by a "Y" for Yes and "N" for No
Movie Categories	Preferred by a user Comedy, Drama, Romance, Action, Science Fiction, Crime, Documentary, Fantasy, Horror

Table 1: Schema of the dataset we used. Note: The attribute UserID was not used in the computation of the tree.

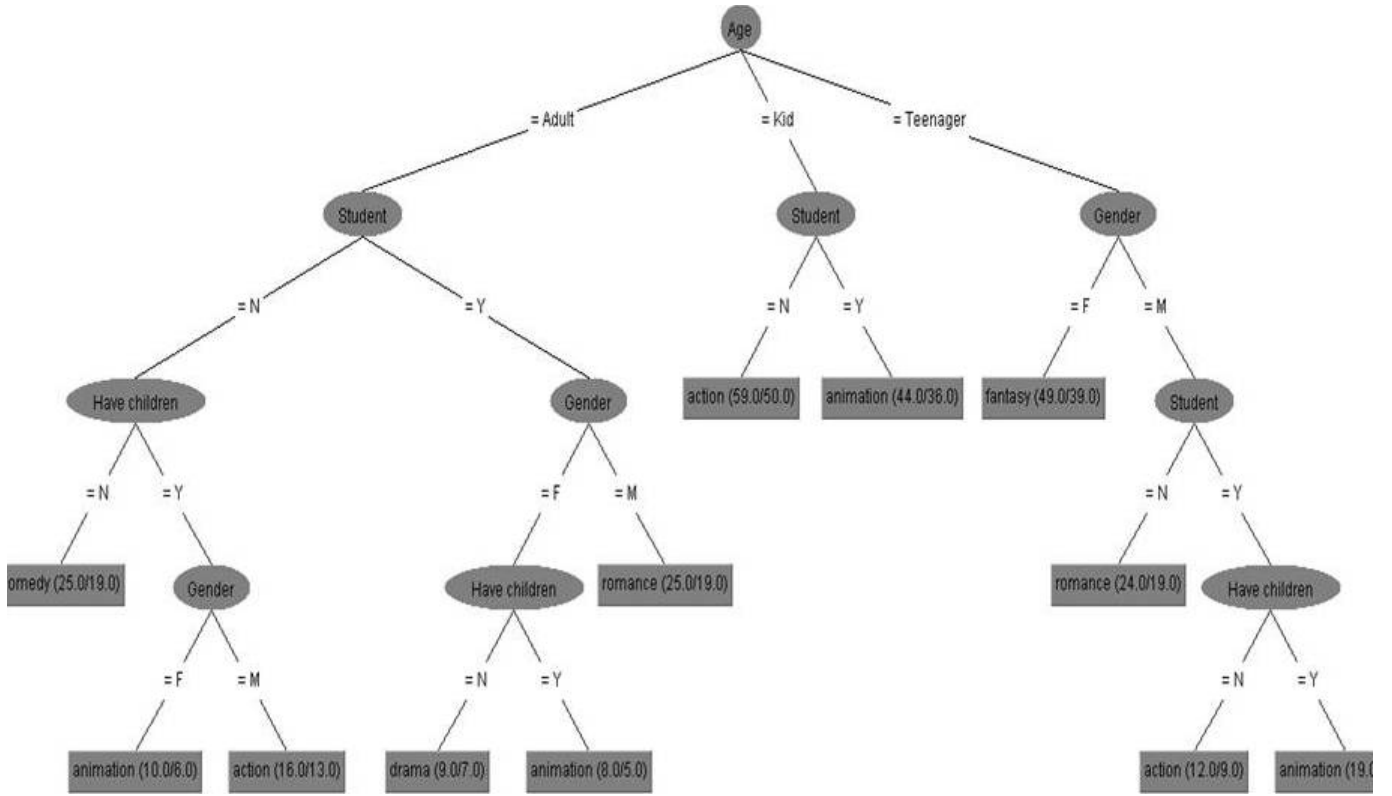


Figure 4: Decision built using our training data with the schema in table 1

Parameter	Description
UserID	User who provided his favorite movie category, denoted by a numerical value
Movie title	Name of the movie
Movie Categories	denoted by 17 fields of categories where "yes" indicates if the movie is of that genre and "No" which indicates that it is not; movies can be in several genres at followed once.

Table 2: Schema of the dataset

=== Classifier model (full training set) ===

J48 pruned tree

```
-----
Comedy = yes
| Animation = yes
| | Children's = yes: Aladdin and the King of Thieves (1996) (5.0/3.0)
| | Children's = no: Close Shave, A (1995) (2.0/1.0)
| Animation = no
| | Children's = yes
| | | Drama = no: Apple Dumpling Gang, The (1975) (2.0/1.0)
| | | Drama = yes: Babe (1995) (2.0)
| | Children's = no
| | | Action = no
| | | | Drama = no
| | | | | Fantasy = no: Mighty Aphrodite (1995) (10.0/8.0)
| | | | | Fantasy = yes: Blue Chips (1994) (2.0/1.0)
| | | | Drama = yes: Jack (1996) (2.0)
| | | Action = yes: Get Shorty (1995) (3.0/1.0)
Comedy = no
| Adventure = no
| | Drama = no
| | | Crime = no
| | | | Animation = yes
| | | | | Musical = no: Mark of Zorro, The (1940) (3.0/2.0)
| | | | | Musical = yes: Alice in Wonderland (1951) (3.0/1.0)
| | | | Animation = no
| | | | | Fantasy = no
| | | | | | Thriller = no
| | | | | | | War = no
| | | | | | | | Horror = no: Commandments (1997) (2.0/1.0)
| | | | | | | | Horror = yes: Children of the Corn: The Gathering (1996) (2.0/1.0)
| | | | | | | War = yes: Innocents, The (1961) (6.0/4.0)
| | | | | | | Thriller = yes: Four Rooms (1995) (4.0/1.0)
| | | | | Fantasy = yes
| | | | | Sci-Fi = no
| | | | | | Documentary = no
| | | | | | | Mystery = no: Double Happiness (1994) (20.0/18.0)
| | | | | | | Mystery = yes: Doors, The (1991) (2.0/1.0)
| | | | | | | Documentary = yes: Newton Boys, The (1998) (2.0/1.0)
| | | | | | | Sci-Fi = yes: Beautiful Thing (1996) (4.0/3.0)
| | | | Crime = yes
| | | | | Fantasy = no
| | | | | | Thriller = no: Road to Wellville, The (1994) (11.0/9.0)
| | | | | | Thriller = yes: Seven (Se7en) (1995) (4.0/2.0)
| | | | | Fantasy = yes
| | | | | | Sci-Fi = no: C'est arrive pr is de chez vous (1992) (2.0/1.0)
| | | | | | Sci-Fi = yes: Twelfth Night (1996) (3.0/1.0)
| | Drama = yes
| | | Sci-Fi = no
| | | | War = no
| | | | | Children's = yes: Old Yeller (1957) (2.0/1.0)
| | | | | Children's = no
| | | | | | Crime = no
| | | | | | | Romance = no: Jupiter's Wife (1994) (18.0/15.0)
| | | | | | | Romance = yes: Dead Man Walking (1995) (2.0/1.0)
| | | | | | Crime = yes: Copycat (1995) (2.0/1.0)
| | | | | War = yes: Richard III (1995) (2.0)
| | | Sci-Fi = yes: Twelve Monkeys (1995) (4.0/2.0)
| Adventure = yes
| | Action = no
| | | Crime = no
| | | | Documentary = no: Double Team (1997) (3.0/2.0)
| | | | Documentary = yes: Hackers (1995) (2.0)
| | | Crime = yes
| | | | Documentary = no: Tank Girl (1995) (2.0)
| | | | Documentary = yes: Best Men (1997) (2.0)
| | Action = yes
| | | Thriller = no: Highlander (1986) (2.0/1.0)
| | | Thriller = yes: GoldenEye (1995) (2.0)
Comedy = on: White Balloon, The (1995) (1.0)
```

Number of Leaves : 35

Size of the tree : 68

Figure 5: Decision tree built using our training data with the schema in table 2