**INDIAN INSTITUTE OF MANAGEMENT CALCUTTA**

**WORKING PAPER SERIES**

**WPS No. 654/ April 2010**

# Drifting Preferences in Recommender Systems

**by**

**Sourav Saha**
Doctoral student, Indian Institute of Management Calcutta, Joka
Diamond Harbour Road, Kolkata 700104, India

**Sandipan Majumdar**
Research Assistant, Indian Institute of Management Calcutta, Joka
Diamond Harbour Road, Kolkata 700104, India

**Sanjog Ray**
Fellow, Indian Institute of Management Calcutta, Joka
Diamond Harbour Road, Kolkata 700104, India

**&**

**Ambuj Mahanti**
Professor, Indian Institute of Management Calcutta, Joka
Diamond Harbour Road, Kolkata 700104, India

# Drifting Preferences in Recommender Systems*

**by**

**Sourav Saha**
Doctoral Student, Indian Institute of Management Calcutta
Joka, Diamond Harbour Road, Kolkata 700 104, India
sourav.saha@gmail.com

**Sandipan Majumdar**
Research Assistant, Indian Institute of Management Calcutta
Joka, Diamond Harbour Road, Kolkata 700 104, India
sandipan.compengg@gmail.com

**Sanjog Ray**
Fellow, Indian Institute of Management Calcutta
Joka, Diamond Harbour Road, Kolkata 700 104, India
sanjogray@gmail.com

**&**

**Ambuj Mahanti**
Professor, Indian Institute of Management Calcutta
Joka, Diamond Harbour Road, Kolkata 700 104, India
am@iimcal.ac.in

*This research has been supported by the IIMC research project W.O. No. 019/RP: SIRS//3397/2008-09

## Abstract

Recommender systems are increasingly becoming popular with the enormous choice that the online virtual marts present. Collaborative filtering is one of the most popular techniques to generate recommendation by means of collaboration among multiple information agents. It uses past transactions to gather critical information and then extracts knowledge by means of filtering.

One of the major issues in collaborative filtering is the sparsity problem, wherein the data is sparse in nature and carries only partial information or misses out information totally. Another issue is that in reality, collaborative filtering is characterized by the recency effect wherein recent items tend to speak volumes about user preferences than past data. This concept, sometimes called the drift effect is absent in the traditional collaborative filtering algorithm.

In this paper, an attempt has been made to come up with a novel approach that would try to address the sparsity problem and would take the drifting effect into consideration. This algorithm uses minimal information to make predictions and takes the drifting effect fully into consideration. Some newer algorithms do make use of a decreasing time function that assigns a maximal weight to the recent data and a minimal weight to past data. However, if the time-frame from which the data is constructed is not continuous in nature, this approach has the possibility of assigning differential weights to a data cluster that belongs to the same time-frame. Here, a sliding window approach has been taken that views cluster of data in consecutive frames of time in a window. The algorithm applies a decaying or decreasing function only when a particular trait falls outside the window by means of consecutive incidence of non-occurrences. The user-preference is clustered under five distinct heads, namely "Sporadic", "New", "Regular", "Old" and "Past". Initial occurrence places a particular trait into "Sporadic" category. Repeated occurrence of this particular trait to fill a window of defined size promotes this trait to the "New" category. Repeated occurrence of this particular trait to satisfy a bigger reference window of given size promotes this particular trait into the "Regular" category. Conversely, repeated non-occurrences of the particular trait in the "New" category to void a particular window of pre-determined size would demote the trait to the "Sporadic" category. Further non-occurrence of the particular trait would pull itself out of our reference heads completely. Conversely, repeated occurrences of the particular trait would promote it to the "New" category once again and subsequently back to the "Regular" category. Once a trait is in the "Regular" category, several repeated non-occurrence would put it into an intermediate category called "Past" that denotes that the interest has started fading off and then moves to "Old" and then moves away.

The algorithm has been tested on a sample of Yahoo! Movies community's preferences for various movies.

Keywords: Collaborative filtering, drifting preference, sliding window.

## Introduction to Recommender System:

The internet has given a totally new platform for people to explore newer trends in processes and businesses. With the advent of Web 2.0, social media and online social networking, there has been an explosion of information. It has been found that Facebook itself is only pushing 10 Gigabyte of data per hour into the system. This vast amount of information has made the world smaller. However, a problem of adverse selection has also stemmed up. With such a sea of knowledge, the average user’s capacity to process information has reached its threshold. Earlier, one had to do with only the small information to which he/she had access to. Now the problem is to reach the information that is only required filtering out everything around. For example, earlier in the ages of brick and mortar store, user had limited reach in-terms of the number of stores that he could visit and the number of items on display. With the current information age, firstly the same user has access to the many online stores. So the first problem is that which store to select among the so many stores. One can select the store based on price advantage, based on the speed of delivery, based on variety, based on fewer errors made or based on overall user satisfaction. Now this task of comparison itself is resource intensive. Say for example the user wants to buy a specific item from the online store. To do so, the following set of steps must be done:

- Firstly, the user needs to make a list of all the stores where the exact or at-least a comparable product could be found
- Second, the user needs to check if the item exists in the store
- Third, the user would like to see if the price at which the item is offered to him suits his budget
- Fourth, the user would like to find out that how fast the item can be delivered
- Fifth, the return policy of the item should be checked
- Sixth, a check on satisfied customers should be found out
- Seventh, he needs to find out if the online store accepts standardized payment modes
- Finally, he needs to order the item through the portal

It’s evident that at every step, the user had to browse though a lot of information before he could move onto the next step. Hence the advantage the online store gives is almost ruined by the several choices that have been presented and the vastness of information. The user had a lot of unnecessary or redundant information to process before the final objective was reached.

The idea of recommender system stems from our social systems. We tend to have recommendations from our friends, our peers, our families before we would like to make a purchase. All these peers summarize their experience with a particular store or a particular item and gives in their viewpoint. Some of the views focus on the cost advantage, some on the quality advantage, some the service advantage and some on the speed advantage. Based on all of the above recommendations, the user makes a conscious choice or decision that whether he would like to buy the item or he should let go. This decision accepting a recommendation sits over some of the following factors:

- What is the recommendation

- How authentic is the source of recommendation
- How knowledgeable the recommender is
- How trustworthy the recommender is
- How valuable has his recommendations been in the past
- What's the risk in accepting such recommendation

This exercise is being done by the user for a limited number of recommenders and then finally the user takes a call. Thus when there is a pool of dedicated recommenders who would satisfy the above criterion, the user would be happy to take their advice. However, in the situation that no such recommendation is made available to the user, he has to depend on his luck or gut feeling.

These days' collaborative recommendation and trust based recommendation have helped the recommender systems gain more prominence and widespread acceptance. These automatic recommender system works on the similar principle and logic as discussed above. By means of collaborative recommendation, all the recommendations about a particular subject or element of interest are amassed. The trust component in the recommendation takes care of the fact that the recommendations that are generated in the process are genuine. User's trust can be again decomposed into a two-fold process, a context specific interpersonal trust and a system/impersonal trust. In case of the context specific interpersonal trust, the user is more comfortable in getting recommendations from personal acquaintances. However, in case of the system/impersonal trust, the user is basically putting his trust on the system as a whole. This latter area has been of more interest to the researchers. The line of reasoning in which a recommender system is constructed is similar to what a typical user employs when he tries to make use of recommendations. However there is one major difference – while a typical user mostly depends on logic and gut feeling while accepting a recommendation, the online recommender uses trusted source of data and algorithms that helps them come with the recommendations that they make. In an online system, the major advantage is that the database of good recommendations and good recommenders is huge. The asset of so-many recommenders and so many recommendations is by itself a virtue that would often beset the best networked individuals. The recommenders are constantly monitored based on the acceptability of their recommendations. In turn their recommendations are also rated. This creates a huge pool of information which is difficult to monitor and process when even the most resourceful individual is concerned. Some big organizations that have recommender system at the core of their success are Amazon and NetFlix.

Largescale commercial applications of the recommender systems can be found at many e-commerce sites, such as Amazon, CDNow, and MovieFinder. These commercial systems recommend products to potential consumers based on previous transactions and feedback. Examples of research prototypes of recommender systems are: PHOAKS [Terveen et al. 1997], Syskills andWebert [Pazzani and Billsus 1997], Fab [Balabanovic and Shoham 1997], and GroupLens [Konstan et al. 1997; Sarwar et al. 1998]. These systems recommend various types of Web resources, online news, movies, among others, to potentially interested parties. Recommender system is becoming a standard for e-commerce businesses that aims at providing the content that the user wants with an aim of converting the browsing experience to the buying experience. With good recommendations, the customer base turns loyal as well since they don’t have to take the pain of getting recommendations from various other trusted sources.

The idea of recommendation has resulted in the emergence of new business concepts as well. Some example that can be cited here is that of Google News. Google here acts as a simple aggregator, which takes the recommendation of the best news from among the various sites in the world based on the content and user preference. It then combines such recommendation with its award winning search technology to enable users search the news that they like or accept feeds. This has resulted in the popular Google News that has been increasing its customer base every day.

Networking and networked devices are increasing manifold with every passing year. As more people become networked, the values and use of recommendation also amplifies. Recommender systems are here to stay.

## Literature Survey

Recommender systems are technology based systems that provide personalized recommendations to users. They generate recommendations by profiling each user. Profiling is done by observing each user's interests, online behavior and transaction history. Recommendations can also take into account opinions and actions of other users with similar tastes. Recommender systems algorithms can be classified into two major categories, namely content based and collaborative filtering based. A third approach called hybrid approach combines both content based and collaborative filtering based methods. In content based recommendations, a user is recommended items similar to the items he preferred in the past. For content based recommendations, each item in the items domain must be characterized by a set of keywords that describe it. In content-based recommendations, a profile of a user is first created by keyword analysis of the items previously seen and rated by him. On the basis of the profile created, the user is then recommended new items. Content based systems are not very popular as they lag in recommendation quality when the user's tastes change. Also, in many domains getting high quality content data is a major issue. In collaborative filtering, the user is recommended items that people with similar tastes and preferences liked in the past. This technique mainly relies on explicit ratings given by the user and is the most successful and widely used technique. There are two primary approaches that are used to build collaborative filtering memory based recommender systems: user based collaborative filtering [1] and item based collaborative filtering [2]. In user based collaborative filtering, a neighborhood of k similar users is found for a particular user. Items are then recommended to the user from the set of items seen by its k nearest neighbors but not yet seen by him. In item based collaborative filtering, similarities between the various items are computed. Items are then recommended to the user from the set of k items that have highest similarity to the items already present in the user's profile.

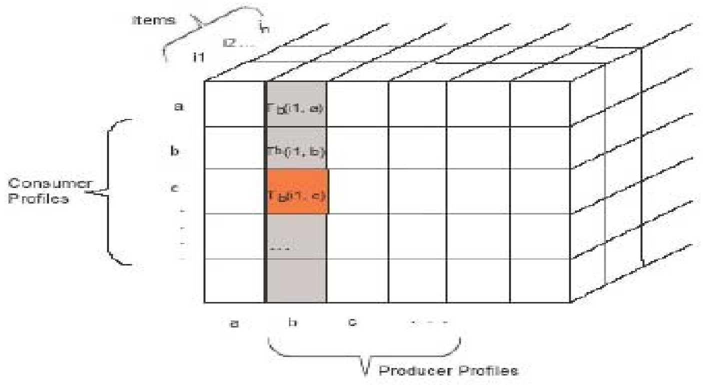

**Figure 1: Calculation of Trust Scores from Rating Data**

One drawback of traditional recommendation systems are that they assume user interests to be static, which is not the case as preferences change with time. This change in preference can occur in two ways. The first kind of change that is most widely seen occurs when a user develops a new interest. Some common examples of this kind change are a movie viewer acquiring a recent liking for western movies, a book reader developing a new interest for books by an author or on a particular subject. Examples of users developing new interests are prevalent in most domains where recommender systems are extensively used like books, movies, music, research papers, television etc. Popular recommender system algorithms, like user based collaborative filtering and item based collaborative filtering algorithms, use only ratings data to generate recommendation. As context data like genre, subject and keywords are not used, the recommendations generated by these systems are not specifically customized for the user's current interests, and as a result there is lag before they can detect change in his preferences. This inability of systems to quickly recognize a shift in interest is also known as the concept drift problem.

The second kind of change occurs when user preference for a particular item changes over time. When asked to re-rate movies, users may re-rate the same item differently. User preferences evolve over time: a user who has rated an item highly five years back may not hold the same opinion about the item currently. Existing algorithms analyze a user's complete movie rating data to create a profile. As equal weights are given to all the items in the user's profile, the profile created may not give the true picture of the user's current preferences. These drawbacks of existing algorithms can hamper the effectiveness of the recommender systems, which can lead to missed opportunities to market products to a user and can affect user confidence towards the recommender system.

Most of the previous research on the problem of concept drift has focused on applying weights to time series data. The weights applied are based on a time decaying function with higher weights applied to new items. Older data points are removed or very low weights are applied to them so as to decrease their influence on recommendations generated. This approach can be helpful in solving the problem of concept drift when the second kind of interest change occurs, i.e. when the user's preference for a particular item changes over time and the present data points present a truer picture of the user's interest. But this approach of using only current data points further magnifies the problem of data sparsity. While previous research on concept drift in recommender systems have tried to tackle the problem by using weighted data, none of them have tried to use content data like genre, keywords etc.

In this paper, we propose an approach to categorize user preferences from its transaction history in a recommender system. Our approach can be used by context based, collaborative filtering based and hybrid recommender system algorithms to make better predictions. Our approach of categorizing user interests can be used as one component of a hybrid algorithm designed for solving the concept drift problem.

# Existing Algorithms

In this section, we would define a few popular existing algorithms in this domain for the sake of completeness.

## Item Based Collaborative Filtering

Collaborative Filtering problem can be defined as follows (adapted from Time Weight Collaborative Filtering [4]) :

Given a database D as a tuple < Ui; Ij ;Oij ; Tij >, where Ui identifies the i-th user of the system, Ij identifies the j-th items of the system, Oij represents the i-th user's opinion on the j-th item and Tij represents producing time of the opinion, find a list of k recommended items for each user U.

In item-based collaborative filtering algorithms, an item is regarded as a vector in the user space. The whole process is divided into two phases:

### Phase 1: Similarity Computation

There are three main approaches to compute similarity between two items.

#### *Cosine Similarity*

An item is considered as a vector in the m dimensional user-space. The similarity between different items is measured by computing the cosine of the angle between different vectors as:

$$sim(I_a, I_b) = \cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} = \frac{\Sigma_i^m O_{ia} \times O_{ib}}{\sqrt{\Sigma_i^m O_{ia}^2}\sqrt{\Sigma_i^m O_{ib}^2}}$$

Where $I_a$ identifies the a-th item of the system. $O_{ia}$ represents the i-th user opinion on the a-th item.

#### *Pearson Correlation Coefficient*

The similarity between different items is measure as follows:

$$sim(I_a, I_b) = \frac{\Sigma_i^m (O_{ia} - \overline{O_i}) \times (O_{ib} - \overline{O_i})}{\sqrt{\Sigma_i^m (O_{ia} - \overline{O_i})^2}\sqrt{\Sigma_i^m (O_{ib} - \overline{O_i})^2}}$$

Where $\overline{O_i}$ is the average user’s rating.

### *Conditional Probability Based Similarity*

An alternate way of computing the similarity between different items is to use a measure that is based on the conditional probability of selecting one of the items, given that the other item has already been selected.

$$P(j|i) = \frac{Freq(ij)}{Freq(i)}$$ ,

Where $Freq(i)$ is the number of users that have already selected the i-th item.

## Phase 2: Preference Prediction

The prediction of the preference for a given object can be computed by using the sum of the ratings of the user to items weighted by the similarity between different items as

$$O_{ij} = \frac{\Sigma_{c=1}^{k} O_{ic} \cdot sim(I_j, I_c)}{\Sigma_{c=1}^{k} sim(I_j, I_c)}$$

Where, $I_j$ identifies the j-th item, $I_c$ identifies the nearest neighbor of the j-th item, $O_{ij}$ represents the i-th user's opinion on the j-th item.

In the case where some weights $W_c$ needs to be assigned to the item $I_c$, the modified equation is

$$O_{ij} = \frac{\Sigma_{c=1}^{k} O_{ic} \cdot sim(I_j, I_c) \cdot W_c}{\Sigma_{c=1}^{k} sim(I_j, I_c) \cdot W_c}$$

Here, the weight can be assigned to the item based on many parameters. Some of the typical cases, where the weights are assigned may be listed as follows:

- A weightage to the time of occurrence of the observation
- A weightage based on familiarity with the recommender
- A weightage based on the item of recommendation itself

The accuracy of the prediction is given by the following formulae for "Mean Absolute Error"

$$MAE = \frac{\Sigma_{i=1}^{N} |p_i - q_i|}{N}$$

, where N is the number of user rating, $p_i$ is the predicted rating for the i-th item and $q_i$ is the actual rating given by the user.

# The Proposed New Algorithm

In the new algorithm, we have tried to imitate the human behavior by describing it through a set of logical steps. We have made some very basic but simple observations while deriving the particular algorithm. Some of those observations are listed as below:

- The interests of a human being changes with age, time and responsibilities
- Interest is drastically influenced by the peer-group to which a person belongs
- The change of interest is not instantaneous but takes some time to stabilize
- When a change is initiated, several alternatives are explored
- The suitable alternative(s) repeatedly tested till one is confirmed that it suits his/her interest or liking
- Once an interest becomes regular, it recurs frequently or at least after regular intervals
- When interest changes, there is one dominant trait which brings about the change and the user adjusts himself to the other attributes
- A user can be found to regularly deviate from his existing interests. Then such interests are no longer appealing to the user. Such interests then slowly moves out of the users' interest domain

With such simple assumptions, we try to look into the existing recommender systems algorithm and try to find the issues with the existing algorithms.

## Issues with already existing algorithms:

The existing algorithms generally have the following few common shortcomings:

- **Sparsity Problem:** The collaborative filtering techniques take a heavy blow if the data presented is not complete or not accurate. Like the algorithms that heavily rely on user feedbacks, if the rating is not present then the accuracy of the algorithm is doubted. The algorithm presented here needs minimal information and hence the sparsity problem can be done away with.
- **Drifting Effect:** Most of the traditional algorithms don't consider the time-value of the information that they have. It's apparent that information received today about a user preference is much more valuable than the information received say one year back. Our algorithm accurately considers the drifting effect.
    - Example: Based on the user preference data of the last year, we found that the user liked cartoon movies. This year however the user has gone to high school and maybe he likes musical more than the cartoons. When he goes to college, then probably he would prefer action movies more.
- **Halo Effect:** Even with the latest data, the user rating can be biased depending on the mood the user is at the moment.

- Example: A user may like romantic movies. But say right now he is in terrible grief. In this scenario, a romantic movie might not appeal to him at all and he may end up giving a terrible rating to the romantic movie. In this scenario, though he likes the kind of movies, the ratings given by him are not indicative of his actual preferences.

## Establishing the Algorithm

### Methodology

The designing of the new algorithm came through a series of logical activities. We try to undermine some of the activities that let to the construction of this algorithm.

- **Identification of Scope:** We initially started with the identification of the older algorithms. Our observation was that most of the previous algorithms relied too much on data mining and existing statistical procedures. Most of the algorithms tried to bring in some newer unit of analysis to account for a newer data mining technique or statistical procedure. We tried to break free from the paradigm. Recommendation is a human phenomenon and though statistics is important, it’s not the only element that one should look for. We have typically looked at the places where the existing techniques are suffering and tried to fit in human behavior in place.
- **Definitions:** We have derived the ideas from traditional networks. Every actor in a network has a state, a space and a transition. In our example, the interests can be identified to belong to a particular state, the domain of the interest is the space in which it belongs and the transition is the movement by which such interests are placed in higher or lower levels of preferences
- **Identification of Scenarios:** To establish our hypothesis, we needed concrete real-life data where such system of analysis can be established. We were looking for datasets that can confirm to real-life behavior. Our initial choice was the food segment. However we realized that there is no valid source which can give us individual user data on their food preferences. We also realized that even if food preference information has been recorded, chances are that such information doesn’t reflect the reality. This is because people when takes food outside in restaurants, they are looking for a change. The best we could have is data on such changing patterns. Further, food is more of a custom rather than interest. Hence we dropped the idea of food preference analysis. Next was holidays but here also the same issues were of prime concern. Moreover, holidays are a rarer phenomenon. We were hence on the lookout for something, that the user can enjoy sitting in the comfort of home. Movies seemed an excellent choice that satisfies the entire criterion.
- **Identification of Element of Analysis:** The movie data is a rich data that contains a range of information. However, we required something that would remain invariant over time and place. The choice of movie genre suited the criterion. It was decomposable as well allowing further analysis.

- **Time to Execution:** We realized that human beings take decisions on the fly for non-trivial activities like that of movie watching. Hence, developing an extremely complicated algorithm was out of question. We tried an intuitive approach that would appeal even to the layman. The calculations can be done easily and quickly facilitating real-time decision making.
- **Developing the Algorithm:** To reduce any bias, the algorithm has been developed from bottoms up. The new algorithm has been developed keeping human behavior as the centre of focus. We have had the algorithm analyzed from the eyes of an user and the eyes of a critique to the user's interests.

## Definitions

Along with this, we define 5 classes under which we can classify the user preference:

- **Sporadic:** This is some sort of temporary unstable category, where the user's current interests are stored. In our example, the movie categories are stored here. If the user keeps on watching movies that belong to this genre, it is promoted to the next level after it satisfies a particular pre-defined threshold. If the movie of this particular kind of genre doesn't repeat within a given sequence-frame, then it leaves the system.
- **New:** User interests, that keeps on recurring in the "Sporadic" category gets promoted to the "New" category. These are the new interests of the user and there is a chance that items from this particular category would convert to permanent interest of the user if the occurrence exceeds a particular threshold. Successive non-occurrences puts it back to the Sporadic category when it breaches the threshold of "New".
- **Regular:** Once the user has been subscribing to a particular type of interest considerable number of times while the interest is resident in "New", it gets converted to "Regular". This is the category which has items that attracts attention from the user as evident from his affinity pattern towards such interest. Successive occurrence just reinforces the belief and increases the occurrence counter.
- **Old:** Repeated occurrence of an attribute of interest increases its counter while non-occurrence decreases the counter. The unit of decay and time of activation depends on the category. For example, if a movie in the "Regular" category is short of the "Regular" threshold due to successive non-occurrence, then its place in the "Old" category. With increased occurrence once again, the "Old" interest can convert back to "Regular".
- **Past:** Successive non-occurrence to breach threshold for "Old" category puts it to "Past" category. Further non-occurrence of interests in past category puts them out of the system altogether.

Define the thresholds set T = {$I_S$, $I_N$, $I_R$, $O_R$, $O_o$, $O_P$} for "Sporadic", "New", "Regular", "Regular", "Old" and "Past" respectively, where I is the incoming threshold and O is the outgoing threshold. In most practical cases and the example we have considered, the incoming ($I_{suffix}$) threshold and outgoing ($O_{suffix}$) threshold is similar. Let there be a total of N users denoted by U1, U2, Ui…Un where each Ui user has ei observations. Thus, the total number of records in the system is given by $\sum_{i=1}^{N} ei = E = \sum_{j=1}^{M} Rj$ . Define the weight function f(c) where c is the category in which the attribute of interest belongs currently. The h(c,w) is the binary activation function where w is the window size for successive non-occurrence of a interests that currently belong to the particular category c (here sporadic, new, old, regular and past) after which the decaying function gets activated . Hence the decaying function d(c,w) = f(c)*h(c,w). Some temporary registers keeps a note of the interests, the category to which they belongs and the corresponding score for the user Ui. The designing of the algorithm here takes care of the drifting effect. Here we have taken the example of movies and the genres to they belong. Thus if the users liked movies of a particular genre some times back and is not subscribed to it anymore, the preference fades away and finally goes off.

## Assumptions

The inputs to the algorithm are the user “identifier” (unique user identification) and the element of analysis “identifier”, grouped by user. Our algorithm makes the following assumptions:

- The data is chronological with the last entry signifying the latest element that the user has evaluated (here in our example, we consider the last movie that the user has seen)
- If the user has been evaluating elements with particular attributes repeatedly, then such an attribute is an attribute of interest (here in our example if an user has been viewing movie of a particular genre say comedy repeatedly, then the user has an affinity for the kind of the movie)
- We assume that repeated instances of a particular attribute of interest are not by chance but via a conscious decision. (For example, the user selects movie that matches his taste. So we exclude the idea of a user making random selection)
- When the user is trying new interests, we give a high value to the decay since chances are that such interests don’t recur. Once when we understand that the user interest has slowly stabilized, we put in lower values to the decay to hold onto the particular interest
- As and when the interests moves to higher levels of preferences, we understand that there is a chance that the user might get “bored” with such interests. However, such interests still remain at the top of his preference list. Hence along with lesser penalization for non-occurrences, we also define a window or a “safe-zone” at every levels of preference. When the repeated non-occurrences breach the “safe-zone” threshold, then only the decay function gets activated.

## Algorithm

```
Set J=1, initialize temporary registers of Sporadic, New, Old, Past and Regular to 0
    While $J \neq M$
        Decompose the record R𝑱 into UJ (the user) and E𝑱, (the element analyzed)
        If U𝑱 ≠ U𝒊 (this implies that a new series of user record has started)
            Write the following values into the database, where G=interest list and c = count
            <Ui, Sporadic (G,c), New (G,c), Regular (G,c), Old(G,c), Past (G,c)>
            Reset temporary register of Sporadic, New, Regular, Old and Past
        End-if;
        Set Ui = U𝑱
        Decompose E𝑱 into C𝑱 where C𝑱 = {g𝟏, g𝟐 ,..., g𝒑), i.e. the attributes for the elements  in record R𝑱.
        Let A𝑱 be the attribute set that occur in Sporadic, New, Regular, Old and Past currently for user Ui
        Let B𝑱 be the set of attributes in A𝑱 but not in C𝑱 for the user Ui.  B𝑱 = A𝑱 - C𝑱
        Set k = 0. Do the following for all elements in the set C𝑱 for the user Ui
        For $k \neq p$
            Check if gk in Old (Ui),Past (Ui), Regular (Ui), New (Ui) or Sporadic (Ui).
            If g𝒌 exists, then
                Retrieve earlier value of the attribute of interest g𝒌,
                Increase the counter of g𝒌 by 1
                Check set T for appropriate threshold
                Move to the higher category if applicable and remove from previous category
                Update current list of interests and count in Sporadic, New, Regular, Old, Past
                Note: $\text{external} \xrightarrow{Is} \text{Sporadic} \xrightarrow{In} \text{New} \xrightarrow{Ir} \text{Regular}$
            Else
                Put g𝒌 in Sporadic (i) with a value of 1
            End-If;
            Increase counter of k by 1
        End-For;
        Let b𝟏,b𝟐,…b𝒒 be the attributes list in B𝑱. Set k = 0
        For $k \neq q$
            Retrieve the earlier value of b𝒌
            Set b𝒌 = b𝒌 - d(c,w)
            Check the set T for appropriate threshold if b𝒌 has a change in value
            Move b𝒌 to a lower category if it breaches any given threshold
            Remove from the previous category and update the category list and corresponding counts
            Note: $\text{external} \xleftarrow{Op} \text{Past} \xleftarrow{Oo} \text{Old} \xleftarrow{Or} \text{Regular}$
            Increase counter of k by 1
        End-For;
        Update counter of J
```

# Results

To test the algorithm we used data from the Yahoo Webscope R4 that has details of Yahoo Movie user ratings and Movie descriptive content information. Here, the user identifier maybe masked but the movie identifier is accurately required so that one can match the identifier with the IMDB database. The IMDB can give vital information about the movie but here we are concerned only with the movie genre and nothing else.

The "Yahoo! Webscope™ Program" is a reference library of interesting and scientifically useful datasets for non-commercial use by academics and other scientists. All datasets have been reviewed to conform to Yahoo!'s data protection standards, including strict controls on privacy. Data may be used only for academic use by faculty and other University researchers who agree to and sign the Data Sharing Agreement.

The Yahoo Webscope R4 gave a set of training files that contained the user identifier, the movie identifier and the corresponding details like movie ratings. From the movie identifier, one can get the movie details like the Name of the movie, Year of release, Movie Genre, Actors, Directors, etc. In total the training set had 174130 records, 6719 distinct users and an average movie viewership of 25.92 per user. The max no. of movies seen by a particular user was 1388 while the minimum was 1.

To conduct the evaluation, we used the test-dataset once again provided by the Yahoo Webscope R4. The test dataset had a total of 8358 records, 1675 distinct users an average of 5 records (4.9899 to be exact) per user. The min test-record for a single user was 1 while the max was 145. We simply evaluated the test-data from the tables constructed with the training-data.

## Experimental Setup

From the training data set, the genre list that belongs to New, Sporadic, Regular, Old and Past has been constructed for each of the users. In total the training set had 6719 users. In our case, we have

***f(c)  = 0.25 for "Regular" category***

***= 0.50 for "New" and "Old"***

***= 0.75 for "Sporadic" and "Past"***

The activation function for decaying has the following values

***h(c,w)  = 4 for "Regular" category***

***= 3 for "New" and "Old"***

***= 2 for "Sporadic" and "Past"***

To exemplify, once a particular genre is into the "Regular" category, even if it doesn't occur for 3 successive times, it still remains in the said category. Only after the 4th consecutive non-occurrence, the value starts decaying by an absolute amount of 0.25

The threshold values kept for the dataset is as follows T = {0, 3, 5, 2, -0.5} for Sporadic, New, Regular, Old and Past respectively – the values of incoming and outgoing threshold being identical for "Regular". That means when the score of a particular genre is more than 3, then it finds its place in “New” and when it’s 5 or more, then it finds its place in “Regular”. If the score of a genre falls below 5 after it has reached “Regular”, then we place it to “Old”. If it falls below 2 then it finds its place in “Past”. If score is less than -0.5 it moves out of the system.

With the above values, the algorithm is run on the training dataset to generate the list of “Sporadic”, “New”, “Regular” “Old” and “Past” for the 6719 users. Once that’s been done, this result is used for further prediction. The “Regular” category once constructed is an identification of a user’s “Core” tastes. The area of interest is the “New” and “Old”. The “New” categories are those that have a potential of becoming “Regular” if suitable interest elements that would appeal to the user’s taste be provided. Similarly, the “Old” category is the one from which the user is drifting away. Suitable interest elements to appeal to a user’s taste can once again get him interested in the attributes currently in “Old” category and might convert it back to “Regular”.

## Illustrations from the sample data

| Transaction | Movie Genre | Sporadic | New | Regular | Old | Past |
|---|---|---|---|---|---|---|
| T1 | Thriller, Science Fiction/Fantasy, Action/Adventure | Thriller(1), Science Fiction/Fantasy(1), Action/Adventure(1) | - | - | - | - |
| T2 | Action/Adventure, Science Fiction/Fantasy, | Thriller, Science Fiction/Fantasy(2), Action/Adventure(2) | - | - | - | - |
| T3 | Action/Adventure | Thriller(0.25), Science Fiction/Fantasy(2) | Action/Adventure(3) | - | - | - |

| | | | | | | |
|---|---|---|---|---|---|---|
| T4 | Comedy, Romance, Kids/Family, Animation | Science Fiction/Fantasy(1.25), Comedy(1), Romance(1), Kids/Family(1), Animation(1) | Action/Adventure(3) | - | - | - |
| T5 | Science Fiction/Fantasy, Action/Adventure | Comedy(1), Romance(1), Kids/Family(1), Animation(1), Science Fiction/Fantasy(2.25) | Action/Adventure(4) | - | - | - |
| T6 | Action/Adventure, Drama, Science Fiction/Fantasy | Comedy(0.25), Romance(0.25), Kids/Family, Animation, Drama(1) | Science Fiction/Fantasy(3.25) | Action/Adventure (5) | - | - |
| T7 | Romance | Romance(1.25), Drama(1) | Science Fiction/Fantasy(3.25) | Action/Adventure(5) | - | - |
| T8 | Romance | Romance(2.25), Drama(0.75) | Science Fiction/Fantasy(3.25) | Action/Adventure(5) | - | - |
| T9 | Romance | Science Fiction/Fantasy(2.75) | Romance(3.25) | Action/Adventure(5) | - | |
| T10 | Romance | Science Fiction/Fantasy(2) | Romance(4.25) | - | Action/Adventure(4.75) | - |

## Testing with the New Algorithm

The result that has been generated from section 7.1.1 has been used on the test data for "Yahoo Webscope R4". For each of the user in the test set, some movies have been provided. The movies were decomposed to their genre classifications. If the same genres appeared in the "New", "Regular" or "Old" category for the particular user in the constructed result-set, then we say a hit has been made and assign a value of 1 to the particular movie for the given user else we call it a miss and assign a value 0. This has been done for all users in the test result set, and finally the average score has been taken over all the users for which this exercise has been done. The result was a hit ratio of 85.0009% in terms of percentage.

## Benchmarking against existing algorithms

The benchmarking of the algorithm has been done against the sliding window algorithm, whereby the movies that appeared in the last n transactions have been taken and their genres computed. These lists of genres that occurred in the last n transactions have been considered to check against the movies in the test dataset. Following is the result:

| *Size of Sliding Window* | *Hit ratio as a percentage* |
|---|---|
| ***3*** | 53% |
| ***4*** | 61% |
| ***5*** | 68% |

Thus, if all the genres in the last 3 movies are considered, we get a hit ratio of 53% while with all the genres in the last 5 transactions; we get a hit ratio of 68%. The maximum size of the sliding window which we kept was 5. The logic for keeping the value as 5 stemmed from two observations. Firstly, we found that the average value of movies seen by a typical user is 25.92 and the root of the value is very close to 5. Secondly, the threshold for the "Regular" category in our algorithm has been kept at 5. Hence, computing the genres of the last 5 movies would be a fair indication of the case where we have taken care of most of the elements corresponding to the "Regular" category. We also ran our tests with a sliding window of 10 and 20. The advantages were definitely a better hit ratio. However on the downside, we were moving to a scenario, where we were attempting to include every genres watched by the user making a very big set of user preference resulting a narrow set of non-preferences. Hence we limited the results to a max window size of 5 only.

It is to be noted that the sliding window of size 5 had on-average 4.65 attributes of interest while the combined "Regular", "New" and "Old" had 4.98 attributes of interest. We would take up the matter in more details in the discussions part.

## Other Benchmarking

We were just curious to find out the genre ratio by finding out the total occurrence of a given genre dividing it by the total number of movies seen by the user. We then put them into categories called “Very Frequent”, “Frequent” and “Least Frequent” based on certain threshold that we define and then use the genres occurring in “Very Frequent” ( an indicative of “Regular” category) and “Frequent” (an indicative of the “New” and “Old” category) to compute the hit ratio similar to the above. Let’s consider the first row. Here more than 75% means “Very Frequent”, between 75 and 50 is the “Frequent” category and less than 50 till 25 is the  is the “Least Frequent” category. From the test result-set, if the movies for a given user are decomposed and the corresponding genres occur in “Very-Frequent” and “Frequent” category, we say that a “hit” has been made; else we say that it’s a “miss”.

| Serial No. | Very-Frequent Threshold Percent | Frequent Threshold Percent | Least-Frequent Threshold Percent | Hit Ratio % |
|---|---|---|---|---|
| 1 | 75 | 50 | 25 | 14.69% |
| 2 | 70 | 40 | 15 | 28.14% |
| 3 | 66 | 33 | 16 | 43.20% |
| 4 | 60 | 35 | 20 | 38.56% |
| 5 | 50 | 25 | 10 | 58.35% |

As evident from the above results, even in the best case the hit ratio has not been able to cross 60%. And this was taking into consideration all the genre categories, which occurred in 25% or more cases in the user’s total movie viewer ships. This signifies that absolute measurement of the interest expressed as a percentage is not a very good indicator of the fluctuating tastes of a user. When the absolute tastes are considered, there is a chance that the drift component is left unattended.

## Discussion

A simple intuitive algorithm like this accounting for more than 85% prediction is definitely a hint that the algorithm is not totally without any ground. One argument that definitely comes is that if the preference set contains more number of elements, then the chances of good predictions or “hits” are higher. We would like to counter this point by putting forward the argument that the combined number of elements in “Regular”, “New” and “Old” account for only about 27% of the total number of genres available (18 in no).

We have indicated earlier, that the sliding window with last 5 transactions had lesser elements than that of our algorithm. However, with only 6% more elements, our algorithm accounted for more than 25% more accuracy compared to the sliding window. From section 6.1.5, we saw that even if we assigned a higher preference for genres that occurred more frequently, still they have been unable to provide reasonable enough results.

We have however not tried to match our algorithm with the existing popular techniques like that of collaborative filtering and context-based filtering. This is because all those techniques are resource intensive. The time and computing power that they require would not lent itself to be used for real-time stream data. A more useful technique would be to use the existing algorithms with this new one. Once our algorithm has identified a “New” interest of the user then the ideal approach would be to provide very good recommendations of the similar type so that such interests can be converted to “Regular” ones. Similarly, if the user is drifting away from his “Regular” interests, then this can be attributed to two factors. Firstly, the user is saturated and bored with his current “Regular” interest and hence looking for a change. Secondly, the user is not finding enough elements of interest that could make him stick to his “Regular” interests.

Our algorithm is capable of identifying interests that are of similar kind, but whether the identified element is good or bad – that’s totally outside the purview of our algorithm. The strength of data-mining techniques can help to come up with excellent recommendations to bridge this particular gap.

Another issue that we face was giving suitable values to the threshold and defining the window size. For our dataset, we have experimentally found out the values of the threshold as well for the window, taking in the average number of transactions carried out by the users. However, there is an inherent problem associated with this approach. Firstly, there would always be users, who have evaluated very few elements and there would be users who have made large number of evaluations. Making thresholds based on average values would make both the users vulnerable to errors. In our example, users with small number of movie viewer ship would have difficulty in breaching the threshold values in the first place. Hence too high thresholds would inhibit such user to have interests that can make it to the “New” or “Regular” category. Similarly, there are users who have very high viewer ship. A moderate threshold would make the score or value of “Regular” so high that even after prolonged non-occurrence of such interests; it would persist in the “Regular” category and would “age” very slowly to be termed as “Old”.

Thus we summarize our discussions into the following

### Merits of the New Algorithm

- Minimal information required hence sparsity problem shouldn’t occur

- Takes the drift information into consideration
- Easy to compute and speedy execution
- Minimal storage required
- Can be applied in the case of stream data as well
- Flexible and amenable to be applied in diverse scenarios

## Pitfalls of the New Algorithm

- Since uses minimal information, hence misses out vital dimensions
- Assumes that the user selected a movie implied he is making conscious decision regarding the movie genre that he wants to view. This may not be true
- Doesn't take the user rating of the movie hence misses out the vital scenario when the user doesn't like the movie
- Take information as a stream flow without taking into consideration the time gap between the last movie seen and the current movie seen. With a long time gap, the user would tend to forget what he saw last time and how much did he enjoy
- We haven't yet had any provisions where the aspect of collaboration and trust could be made useful in the algorithm
- The values of the thresholds have been experimentally determined
- Threshold values have been kept uniform across all the users but in reality it might be different owing to the difference in the user characteristics and movie watching pattern.

# Conclusions and future scope of work

Our discussion section provides the hints to the future works that are possible if we can extend our algorithm. In isolation, this algorithm would find restricted success in finding out just the user interest. However, the domain of element that matches the user interests is also plenty. Hence, to keep such interests alive, the user should be able to identify good elements so that his interests are reinforced. This can be done with the help of the popular collaborative-filtering algorithms.

We have zeroed on a single attribute from which we are trying to predict the user interest. This might hold good for some given element, but in reality, most of the elements of interests have multiple attributes. Hence, we need to first identify the attributes that would be analyzed. Then suitable thresholds for each of the attributes need to be established. Finally, a corresponding weightage for each of the identified attributes should account for the overall accuracy of the prediction. This would definitely add more value to the algorithm but the inherent simplicity would be lost. If the elements that are considered did not have enough valid data, the sparsity problem would arise once again.

The enhancement of the score of a particular attribute of interest and similarly the corresponding decay takes place in a simple linear fashion. The activation function only delays the process of decay. In reality, such decay and enhancement are not linear. It has been found out that if interests can be identified and good elements can be recommended, then the enhancement of interest takes place in an exponential fashion. But in the initial phase of interest generation, if the user gets recommended with inferior elements (say bad quality movies but matching an users taste of comedy movies), then the interests enhancement remains constant and then quickly starts decreasing. The algorithm can be enhanced to take care of such variations. The user ratings can be very useful in this regard. Thus a bad rating to movies which match the interest domain of the user may give dual indications; firstly the user is losing interest and secondly the recommendation is of very poor quality. In such cases, it’s advisable to recommend movies that match his interest and more importantly that have been rated favourably by users of similar taste and preference.

In our discussions section, we have already identified that having a uniform threshold across all the users may create biased results for many users. There is no foolproof way to counter this. The only logical solution might be to group the users based on their movie viewing pattern and then establish the thresholds. There is one problem with this approach as well. A frequent watcher of movies tomorrow might turn an irregular watcher and correspondingly an irregular movie viewer might turn to a frequent viewer. Hence this strategy is also prone to be a failure. A more balanced approach would be to consider the last few transactions in predefined window size, analyze the characteristics and then readjust the thresholds for the given set of users. So time plays a very crucial factor here. Data streams can be weighted by a time-factor, adjusting for the frequency at which the data arrives. Thus data which occurred at the similar time-instance might hold different weightage or value for different users based on their nature of viewership and the frequency at which the user views movies.

Finally, when we are talking of collaboration, our discussion would be incomplete if we don't mention social networking. Networks are important in our life. The social capital theory introduces us to the huge knowledge, which lies in such a network. Whenever we are talking of interests, it is definitely going to be affected by the peer group of a particular user. Thus if the user can be found to tightly coupled  to a particular network, then such user's interests are invariably affected by the peer group. Hence we can add another dimension to the user preference list, namely the effect of the peer group. A user is most likely to favour a movie that has been rated favourably by his peer group. We have already mentioned earlier that every year, more devices are going online. This has created a huge pool of information to be processed and the opportunity of ample recommendations to be generated. We can think of a future, where recommendations wouldn't be a feature but a necessity in every online system.

Here, we have identified so many future possibilities that the algorithm holds. Nevertheless, we have proposed a very simple but solid approach in this particular paper and small variations of the basic approach can result in finding its use to recommendation problems of diverse nature.

## References

*This document is based on and refers to the following documents and websites:*

[1] Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., and Riedl, J. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40, 3(1997), 77–87., 1997

[2] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. Item-based collaborative filtering of recommendation algorithms. *In Proceedings of the 10th International WWW Conference*, 2001

[3] Adomavicius, G., Tuzilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No 6, June 2005.

[4] Ding, Y., Li, X. Time weight collaborative filtering. *In Proceedings of the 14th ACM International Conference on Information and Knowledge Management* , 485-492, 2005.

[5] Zhan , S., Gao, F., Xing, C., and Zohu, L. Addressing concept drift problem in collaborative filtering systems . *In proceedings of ECAI Workshop on Recommender Systems*, 2006.

[6] Ding, Y., Li, X., and Orlowska, M. Recency-based collaborative filtering. *In Proceedings of the 17th Australasian Database Conference* , 2006.

[7] Herlocker, J., Konstan, J., Terveen, L., and Riedl, J Evaluating Collaborative Filtering Recommender Systems. Transactions on Information Systems, Vol. 22, ACM Press (2004), 5-53, 2004.

[8] www.grouplens.org

[9] www.netflixprize.com

[10] Huang, Z., Chen, H., and Zeng, D. (2004). Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering. In ACM Transactions on Information Systems, Vol. 22, No. 1, January 2004, pp 116-142

[11] Sarda, K., Gupta, P., Mukherjee, D., Padhy, S. and Saran, H. A Distributed Trust-based Recommendation System on Social Networks

[12] P. Massa, A Survey of Trust Use and Modeling in Real Online Systems, In Trust in E-services: Technologies, Practices and Challenges, Idea Group, Inc. 2007.

[13] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative Filtering to weave an information tapestry. In Communications of the ACM, 35:61-70,1992.

[14] Sarwar, B., Karypis, G., Konstan, J. & Riedl, J.(2001), Item-based collaborative Filtering recommendation algorithms, in Proceedings of International Conference on World Wide Web, pp. 285- 295.

[15] L., Maritza, C. & Perez-Alcazar, J. d. J. (2004), A comparison of several predictive algorithms for collaborative Filtering on multi-valued ratings, in `ACM symposium on Applied computing', pp. 1033 - 1039.

# Appendix

An enormous task in hand was getting the relevant data. We considered the “Yahoo Webscope R4” dataset that has been made available for the research community.

## The Details of the Dataset

- The following is a sample data from “YAHOO MOVIES”

*1800019565 The King and I (1999)A brand new full-length animated feature film based on the musical. An all-star cast of voices introduces us to the beloved characters of THE KING AND I. Miranda Richardson is Anna (with her songs sung by Christiane Noll); Martin Vidnovic (Lun Tha in the 1977 Broadway revival) is the voice of the King; Saturday Night Live¹s Darrell Hammond plays the all-new character Master Little and Ian Richardson is the voice of the dastardly Kralahome. Under the direction of Disney veteran animator Richard Rich, the animated version of THE KING AND I represents a delightful marriage of the most innovative animation technology available with the timeless songs considered to be the finest Richard Rodgers and Oscar Hammerstein II ever wrote. 1 hr. 30 min. G \N March 19, 1999 19990319 Warner Brothers guide/00709501.jpg Kids/Family|Musical/Performing Arts Richard Rich*

- The data consists of :
  - MOVIE_ID (the above example shows 1800019565 as Movie ID)
  - Short Description of the Movie
  - Production House Details
  - Director’s Name
  - Casting Details
  - Duration of the Movie
  - Release Data
  - Movie Genre

- The data was in a flat file and 106959 records were extracted from the data

## Data Cleaning

The data we derived from the flat files had the following issues:

- BLANK SPACES(HERE AND THERE)
- UNRELATED DATA
- SPECIAL CHARACTERS
- REDUNDANT INFORMATION

We had to clean the data properly before we could use them to run the algorithm.

## Data Processing

The following tables were created.

I. **MOVIE_NAME_GENRE_TBL**
  - MOVIE_ID,
  - MOVIE_NAME,
  - GENRE

The above table contains the movie id, movie name and the corresponding genre as is evident from the table definition.

II. **YAHOO_USER_TRAIN_DATA**
  - USER_ID,
  - USER_SEQ,
  - MOVIE_ID,
  - GENRE

This is the training table which derives Genre corresponding to a given movie for a user from the already provided data containing the User_ID and Movie_ID

III. **USER_MOVIE_RATING_TBL**
  - USER_ID,
  - USER_SEQ,
  - MOVIE_ID,
  - RATING

We haven't made use of this particular data. But this table contains a very vital information, which is the rating the user has given to the movie. This can be used in future works.

IV. **USER_GENRE_TRAIN_TBL**
  - USER_ID,
  - USER_SEQ,
  - GENRE

This view is constructed from Table II above. This table has been used during the training procedure.

V. **YAHOO_USER_GENRE_PREFERENCE_TBL**

- USER_ID,
- SPORADIC
- NEW
- REGULAR
- OLD
- PAST

This is the outcome of the algorithm and categorizes the movie genres in the corresponding classes of “Sporadic”, “New”, “Regular”, “Old” and “Past”. In the code in annexure, we have actually combined “New” with “Old” and “Sporadic” with “Past”.

**VI. YAHOO_USER_TEST_DATA**

- USER_ID,
- USER_SEQ,
- MOVIE_ID
- SPORADIC_MATCH,
- NEW_MATCH
- REGULAR_MATCH
- OLD_MATCH
- PAST_MATCH
- COMPOSITE_SCORE
- PROCESSED_FLAG

This table once again is constructed in two phases. In the first phase we derive the genres from the MOVIE_ID given. In the second stage, we match the GENRES with that of table V in the “Yahoo_User_Genre_Preference_Tbl”. A corresponding value of hit or miss are updated in the “Sporadic_Match”, “New_Match”, “Regular_Match”, “Old_Match” and “Past_Match”. The composite score is arrived after giving a suitable weightage to each of the scores. The “Processed_Flag” indicates if such record has been processed or not.