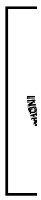

INDIAN INSTITUTE OF MANAGI

WORKING PAPER S

WPS No. 654/ April

Drifting Preferences in Recom

by

Sourav Saha
Doctoral studen, Indian Institute of Ma
Diamond HarbourRoad, Kolka

Sandipan Majum
Research Assistat, Indian Institute of M
Diamond HarbourRoad, Kolka

Sanjog Ray
Fellow, Indian Institute of Manage
Diamond HarbourRoad, Kolka

&

Ambuj Mahant
Professor, Indian Institute of Manag
Diamond HarbourR

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 construc

- 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 numbe

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 led to 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 system 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 base

change are a movie viewer acquiring a recent liking for western movies, a book reader developing a new interest for

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:

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

Where identifies the a-th item of the system. represents the i-th user opinion on the a-th item.

Pearson Correlation Coefficient

The similarity between different items is measure as follows:

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

Where 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.

,

Where 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

Where, identifies the j-th item, identifies the nearest neighbor of the j-th item, represents the i-th user's opinion on the j-th item.

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

Here, the weight can be assigned to the item based on any 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"

, where N is the number of user ratings, is the predicted rating for the i-th item and q

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 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

- **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

the window size for successive non-occurrence of a interests 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 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 on to 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 ≤ 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 of C for the user Ui

For k ≤ 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, g

Increase the counter of g 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 counts in Sporadic, New, Regular, Old, Past

Note: external $\xrightarrow{Is}$ Sporadic $\xrightarrow{In}$ New $\xrightarrow{Ir}$ 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 ≤ q

Retrieve the earlier value of b

Set bk = bk - d(c,w)

Check the set T for appropriate threshold if there 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: external $\xrightarrow{On}$ Past $\xrightarrow{Og}$ Old $\xrightarrow{Or}$ 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

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 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 genre 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 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 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

Discussion

- 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 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 users 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 a predefined window size, analyze the characteristics and then readjust the thresholds for the 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

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, 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 fu

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