Collaborative Filtering Recommender Systems using Association Rule Mining

Abstract: Recommender systems are quite famous in any business happening in the web. Few classical examples are Amazon's recommender system, Netflix movie recommender systems, Orkut's community recommender systems etc. As part of this report we are trying to explore the traditional recommender systems, current recommender systems, and extend to our model of exploring FP-Tree for generating association rules. We are also presenting our experimental analysis and giving the future directions in this area.

I. CURRENT LITERATURE

We try to explore few of the current literature in recommender systems.

A. *Traditional Collaborative Filtering*

A traditional collaborative filtering algorithm represents a customer as an N-dimensional vector of items, where N is the number of distinct catalog items. The components of the vector are positive for purchased or positively rated items and negative for negatively rated items. To compensate for best-selling items, the algorithm typically multiplies the vector components by the inverse frequency (the inverse of the number of customers who have purchased or rated the item), making less well-known items much more relevant. For almost all customers, this vector is extremely sparse.

The algorithm generates recommendations based on a few customers who are most similar to the user. It can measure the similarity of two customers, A and B, in various ways; a common method is to measure the cosine of the angle between the two vectors

B. *Cluster Models*

To find customers who are similar to the user, cluster models divide the customer base into many segments and treat the task as a classification problem. The algorithm's goal is to assign the user to the segment containing the most similar customers. It then uses the purchases and ratings of the customers in the segment to generate recommendations. The segments typically are created using a clustering or other unsupervised learning algorithm, although some applications use manually determined segments. Using a similarity metric, a clustering algorithm groups the most similar customers together to form clusters or segments. Because optimal Clustering over large data sets is impractical, most applications use various forms of greedy cluster generation. These algorithms typically start with an initial set of segments, which often contain one randomly selected customer each. They then repeatedly match customers to the existing segments, usually with some provision for creating new or merging existing segments. For very large data sets — especially those with high dimensionality — sampling or dimensionality reduction is also necessary.

Once the algorithm generates the segments, it computes the user's similarity to vectors that summarize each segment, then chooses the segment with the strongest similarity and classifies the user accordingly. Some algorithms classify users into multiple segments and describe the strength of each relationship

C. *Search-Based Methods*

Search- or content-based methods treat the recommendations problem as a search for related items. Given the user's purchased and rated items, the algorithm constructs a search query to find other popular items by the same author, artist, or director, or with similar keywords or subjects. If a customer buys the Godfather DVD Collection, for example, the system might recommend other crime drama titles, other titles starring Marlon Brando, or other movies directed by Francis Ford Coppola.

If the user has few purchases or ratings, search based recommendation algorithms scale and performs well. For users with thousands of purchases, however, it's impractical to base a query on all the items. The algorithm must use a subset or summary of the data, reducing quality. In all cases, recommendation quality is relatively poor. The recommendations are often either too general (such as best-selling drama DVD titles) or too narrow (such as all books by the same author). Recommendations should help a customer find and discover new, relevant, and interesting items. Popular items by the same author or in the same subject category fail to achieve this goal.

D. *Item-to-Item Collaborative Filtering (Amazon's Recommendation Model) [1]*

Rather than matching the user to similar customers, item-to-item collaborative filtering matches each of the user's purchased and rated items to similar items, then combines those similar items into a recommendation list. To determine the most-similar match for a given item, the algorithm builds a

similar-items table by finding items that customers tend to purchase together. Build a product-to-product matrix by iterating through all item pairs and computing a similarity metric for each pair. However, many product pairs have no common customers, and thus the approach is inefficient in terms of processing time and memory usage. The following iterative algorithm provides a better approach by calculating the similarity between a single product and all related products:

```
For each item in product catalog, I1
   For each customer C who purchased I1
      For each item I2 purchased by customer C
         Record that a customer purchased I1 and I2
For each item I2
   Compute the similarity between I1 and I2
```

It's possible to compute the similarity between two items in various ways, but a common method is to use the cosine measure described earlier, in which each vector corresponds to an item rather than a customer, and the vector's M dimensions correspond to customers who have purchased that item.

E. Recommendation using association rules [2]

Recommendation using association rules is to predict preference for item k when the user preferred item i and j, by adding confidence of the association rules that have k in the result part and i or j in the condition part. Sarwar[2] used the rule with the maximum confidence, but we used the sum of confidence of all rules in order to give more weight to the item that is associated with more rules.

Recommendation using association rules describes as follows. Let P be the preference matrix of n users on m items. In this matrix, pij is 1 if the user i has preference for the item j, and 0 otherwise. Let A be an association matrix containing confidence of association rules of m items to each other. The matrix A is computed from P. In this matrix, aij is confidence of association rule $i \Rightarrow j$. Then the recommendation vector r for the target user can be computed from the association matrix A and the preference vector u of the target user as equation (1). The top-N items are recommended to the target user based on the values in r.

$$r = u \cdot A .$$

The above discussed algorithms does not address sparsity in the recommender systems.

II. PROBLEM DESCRIPTION

Recommendation systems can be classified based on what characteristics of the data they are considering. Typically, they can be categorized into three groups:

1. Pure Static Recommendation System: Here only the rating triplet (user, item, rating) or count data (user, item, count) or just binary transaction data (user, item) is considered. No additional information is made use of.
2. Hybrid Static Recommendation System: Here apart from the above mentioned data, the recommendation system also utilized additional features related to the user or the item itself.
3. Dynamic Recommendation System: Here the recommendation system assumes that the user's preferences change over time and hence processes the data in chronological order.

There can be some more variations based on how the algorithm itself processes the data, for example, some recommendation systems assume hidden variables, for example, User's Personality profile as a hidden variable and try to estimate this from the given data and use it for prediction.

Our problem description corresponds to the pure static recommendation problem, which is most common in literature. However, instead of using the rating triplet as it is we convert the data into binary transaction format (user, item) and work on this data. We do this by converting the ordinal rating data into a binary format (likes, dislikes). If the user had rated an item as 4 or 5 the item is marked as *likes (1)* and if it is less than 4 it is marked as *dislikes (0)*. Thus, the problem is reduced to an association rule mining problem, where each row represents a user and the columns represent the movies that the user likes/dislikes.

Unlike what is found in literature, we do not use the all-but-1 approach for testing or prediction. In the all-but-1 approach, the algorithm is trained on all but one of the ratings given by a user and the algorithm is expected to predict the rating for the single item that is held out.

In our approach we assume that only one item rating is available for the user under consideration and try to predict the remaining items that the user would have rated high.

III. DATA

We planned to try our approach on very large datasets. Couple of datasets that we experimented with are: (a) Netflix data and (b) Movie Lens Data.

NETFLIX DATA

The Netflix data consists of 1GB of data contained in

17770 files, one per movie. The first line of each file contains the movie id followed by a colon. Each subsequent line in the file corresponds to a rating from a customer and its date in the following format:

CustomerID, Rating, Date

- MovieIDs range from 1 to 17770 sequentially.
- CustomerIDs range from 1 to 2649429, with gaps. There are 480189 users.
- Ratings are on a five star (integral) scale from 1 to 5.
- Dates have the format YYYY-MM-DD.

The training data consists of 100 million ratings and the test data consists of 3 million ratings.

MOVIE LENS DATA

The movie lens data consists of 15MB of data. The data is in the format below:

UserID, MovieID, Rating, TimeStamp

- UserID ranges from 1 to 943 sequentially without any gaps.
- MovieID ranges from 1 to 1682 without any gaps.
- Ratings are on a five star (integral) scale from 1 to 4.
- The time stamps are Unix seconds since 1/1/1970 UTC.

The total data comprises of 100,000 ratings. The data set is split into five 80%/20% disjoint splits into training and test data. This is used for 5-fold cross validation.

IV. Assumptions

The algorithm would depend on the following assumptions:

- Rating is ordinal and not continuous. However, it is ok to have ordinal or continuous rating prediction.
- A user rates an item only once.
- A user may not rate all the items and might be rating only a very small subset of the entire list of items.
- Items not rated by a user are given an ordinal rating of 0.
- Data preprocessing is not done. Assumption is that the data is not skewed, i.e., users have rated the items impartially.
- Typical cases, assume all-but-1 approach where all but one rating of the user is given and we have to predict the rating for the left one. In our approach we assume that only one rating has been observed and we have to predict the rating for rest of the items.

V. Algorithm

We utilized the FP-Growth algorithm to construct and mine the association rules. However, the mining time and classification time using the standard algorithm grew exponentially as can be seen from the table below.

% Support	% Conf	Mining Time (secs)	# of Rules	% Accuracy	Classfn Time (secs)
4	100	1.97	610	2.03	1.06
4	90	2.33	13019	5.51	29.27
4	80	9.7	39328	12.5	178.66
4	70	88.31	79030	18.5	667.63
4	60	361.97	135074	24.59	2074.94
4	50	1018.9	201756	32.14	5759.14

VI. Contribution & Observations

The key contribution in our approach is to make the mining and classification time to be a constant, O(1), for a given number of ratings. This is done based on the assumption that the in association rule such as, {X} -> {Y} the subsets in {X} are independent of each other and each subset of {X} -> {Y}. This assumption can be shown to be true in all cases, when the confidence level is set to 0%.

In the original approach the rules are stored in a linked list which required lot of memory and processing time as and when a rule in inserted. By the assumption made above we can do rule mining and classification using a two-dimensional square matrix structure such as (movie x movie), which reduces the space and time requirements drastically.

Other contributions and observations based on our approach is listed below:

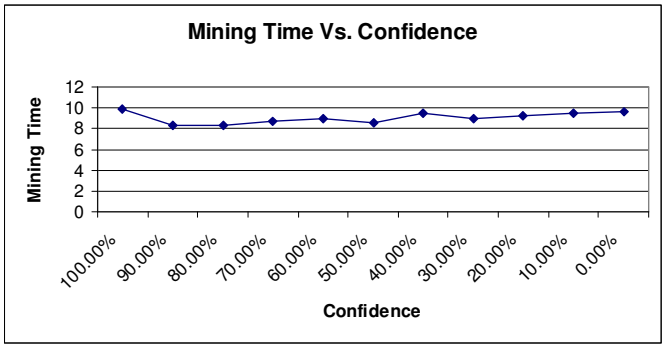
- Our approach requires a user to just rate one movie and based on this it can predict other movies that the user would like or dislike. However, typical recommender systems require a minimum number of ratings to be done by a user for example, a user has to rate at least 20 movies to make any prediction for the user. However,
- Our recommendation is based on impersonal querying or universal querying. Since user similarities aren't computed. Whereas typical recommender systems are specific to a user.
- If the prediction is correct increase the reward, if the prediction doesn't match penalize the score.
- There is a tradeoff between quality of prediction and accuracy of prediction. As confidence is lowered, the quality of prediction goes down, but the accuracy of prediction goes up.

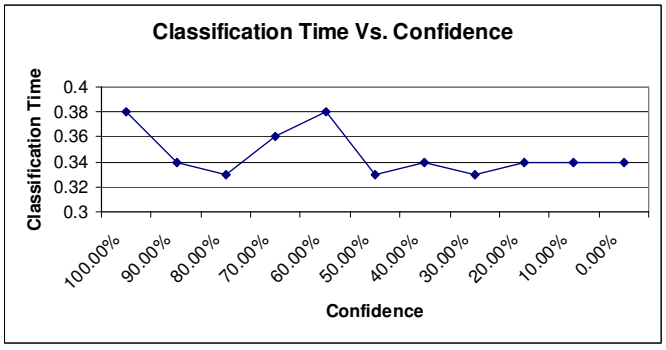
VII. Experimental Analysis

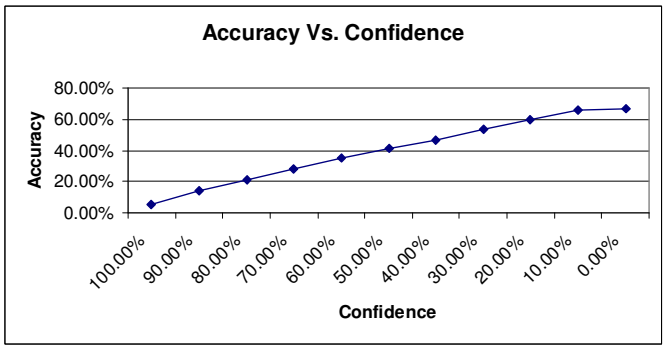
Using divide and conquer strategy, we converted the 1GB of Netflix data into a format that can be used by the association rule mining algorithm. However, due to limitations in Java Virtual Machine, we couldn't even build the FP-tree. Hence, we worked on the Movie Lens Data and the results of which are given in the figures below.

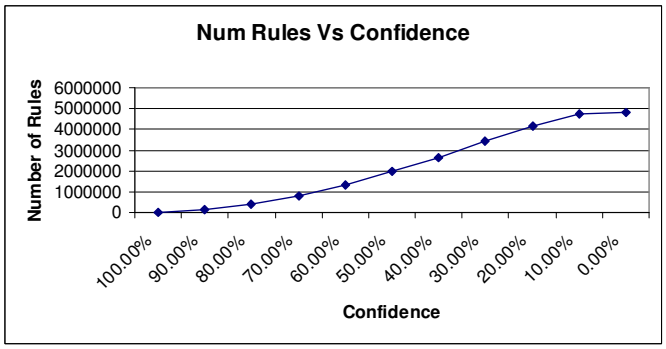
Considering the fact that almost all the recommender systems in the literature doesn't provide any measure on the quality of prediction, the best result that we got using our approach with 0% confidence and 3% support is 64% (on average using 5-fold cross validation). A comparison with results (based on Movie Lens data) declared in literature is given below:

	% Accuracy
ARM Based Approach	**64%**
NNC	55%
Naïve Bayes Classifier	52%
K-Medians Clustering	56%









VIII. Disadvantages of this Approach

ARM based recommendation system is not equivalent to collaborative filtering based recommendation system. Specifically, co-occurrence or transaction based recommendation system is not equivalent to rating triplet based recommendation system. The reason being, not all the three parameters are utilized in the transaction based system. In ARM based approach only two-dimensions (user, rating) or (movie, rating) are utilized. In our approach we have utilized only the (movie, rating) pair.

Accuracy is improved by lowering the confidence. This result in generation of exponential number of rules most of those are often not interesting.

IX. Future Work

We are classifying the future work in two directions: Future work on the application and data mining techniques to be probed.

A. *Application specific tasks*

- Currently, we only ran on (user, likes) pair, we can use the same approach and try to predict what movies the user might dislike using the (user, dislikes) data.
- Learning based on feedback – increase or decrease the weight of a rule based on whether it correctly predicts or not.
- Personified recommendation by including user

details in the algorithm, i.e., by considering the complete data, the rating triplet given by (user, item, rating).

- Do dynamic profiling of users through sequential processing by utilizing the time of rating. This would be required in cases where a user initially liked cartoon movies and later developed a liking to action movies (as the user's age increases).
- Preprocessing of data to eliminate cases that could skew the data or doesn't add value. For example, we need not consider the data related to a user who always gives a rating of 3. In other words, retain only those data that has variance or interestingness.
- Recommending a sequence of items rather than just predicting the rating for a single item. For example, recommend a sequence of research articles (introduction, survey, etc.) to learn about a field.

B. *Data Mining Techniques to be explored*

- Prune rules during generation itself based on novelty or some other measure of interestingness.
- Improve accuracy by considering all the (preprocessed) data by having a support of 0%. To do this, utilize compressed data representation, to overcome the memory constraints.
- To handle missing data, generate synthetic patterns, by utilizing additional features such as genre of movies, age of user, etc.
- Use an ensemble algorithm similar to weighted majority voting.
- Use SVD to reduce the dimensionality of the ratings matrix.
- Use subspace clustering to reduce the dimensionality.

REFERENCES

[1] G Linden, B Smith, J York, "Amazon. com recommendations: item-to-item collaborative filtering", Internet Computing, IEEE, 2003

[2] B.M. Sarwar et al., "Item-Based Collaborative Filtering Recommendation Algorithms," 10th Int'l World Wide Web Conference, ACM Press, 2001, pp. 285-295.

[3] L. Ungar and D. Foster, "Clustering Methods for Collaborative Filtering," Proc. Workshop on Recommendation Systems, AAAI Press, 1998

[4] W Lin, SA Alvarez, C Ruiz, " Efficient Adaptive-Support Association Rule Mining for Recommender Systems", Data Mining and Knowledge Discovery, 2002

[5] D Fisher et al., "SWAMI: a framework for collaborative filtering algorithm development and evaluation", CS Report Berkley available at http://guir.berkeley.edu/projects/swami/swami-paper/paper2.html

[6] Movielens data - http://www.grouplens.org/

[7] Neflix prize - http://www.netflixprize.com/

[8] B. Marlin. Collaborative filtering: A machine learning perspective. Master's thesis, University of Toronto, 2004.