Towards Generating Recommendations on Large Dynamically Growing Domains

Nargis Pervin  
*National University of Singapore*, nargisp@comp.nus.edu.sg

Anindya Datta  
*National University of Singapore*, datta@comp.nus.edu.sg

Kaushik Dutta  
*National University of Singapore*, dutta@comp.nus.edu.sg

Follow this and additional works at: [http://aisel.aisnet.org/pacis2013](http://aisel.aisnet.org/pacis2013)

Recommended Citation

[http://aisel.aisnet.org/pacis2013/29](http://aisel.aisnet.org/pacis2013/29)

This material is brought to you by the Pacific Asia Conference on Information Systems (PACIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in PACIS 2013 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
TOWARDS GENERATING RECOMMENDATIONS ON LARGE DYNAMICALLY GROWING DOMAINS

Nargis Pervin, School of Computing, National University of Singapore, Singapore, nargisp@comp.nus.edu.sg
Anindya Datta, School of Computing, National University of Singapore, Singapore, datta@comp.nus.edu.sg
Kaushik Dutta, School of Computing, National University of Singapore, Singapore, dutta@comp.nus.edu.sg

Abstract

Overwhelming increase in the amount of information raise a requirement of personalized recommendation system. A vast amount of studies have applied traditional collaborative filtering (CF) techniques for generating recommendations. However, well-known scalability issue imposes a limit on its general application. Association Rule based CF techniques where the association rules are primarily generated on items have also been considered but the approaches turned out inefficient with the rapid growth of item space. We anticipated a promising solution of these issues could be diminution in the cardinality of the large user-item rating matrix. Thus, instead of generating associations among the dynamically growing items, generation of associations among the categories, which is quasi-static in practice, could be a convenient route. Herein, we have proposed an elegant approach to generate recommendations using association rule based CF approach on categories. To evaluate the method, we have experimented with real world mobile application user data from MobileWalla (MobileWalla is a venture capital backed company which accumulates data for mobile applications from four major platforms, Apple, Android, Windows, and Blackberry). Two new measures are introduced for calculating accuracy. Findings show that our method scales well on dynamic mobile application domain with legitimate accuracy.

Keywords: Dynamic Domain, Mobile Applications, Association Rule, Collaborative Filtering.
1 INTRODUCTION

Recommendation technology has been around for a long time and is quite well understood. A review of the recommendation literature demonstrates its use in certain classes of products such as books (Linden et al. 2003), movies (Lekakos et al. 2008), music (Davidson et al. 2010) etc. Among various existing approaches collaborative filtering technique (CF) continues to be most favoured, where items have been recommended considering either similar items rated by other users or items from users sharing similar rating pattern for different items. The main stream researches for generating “good recommendation” have been engaged to improve the accuracy of exact item prediction by reducing the Root Mean Square Error (RMSE) or the Mean Absolute Error (MAE). Recently, methods for non-monotonous predictions have also been addressed (Ziegler et al. 2005; Zhang et al. 2008; Zhang et al. 2009; Vargas et al. 2011). However, the issues of scalability, data sparseness (Sarwar et al. 2000), and association problems (Kim and Yum 2011) remain vastly underdeveloped and are challenging till date. In fact, these general recommendation methods (e.g., user based CF, item based CF, and content-based technique) are quite computationally intensive and when new products or reviews come in, the system has to be re-run to factor in their effects.

Situation becomes worse when one considers other classes of products (e.g., digital music, apps) whose injection volume is orders of magnitude higher than products like movies, books etc. For instance, the mobile applications, a domain of digital goods, have enormous growth of its app-space. While on average over 15,000 new apps are launched weekly, only 100 new movies and 250 new books are released worldwide (Datta et al. 2011). In fact, currently there are over 1.2 million apps on the Apple, Android, Blackberry, and Microsoft native app markets. In addition, in these cases the number of reviews also concomitantly grows in massive numbers. So the scale problem arises both from the volume of products as well as reviews. Further, the rapid growth of products like mobile apps gives birth to the problem of “item-discovery” in its extremely sparse domain and the sparsity grows exponentially with time. Therefore, it is highly competitive for the new items to reach the users in fast growing digital domain. Though in iTunes store, a popular mobile app domain, it is possible to navigate the popular apps so called ‘hot apps’ and ‘new apps’, but it is still hard for the mobile app users to find their preferred apps manually from extensive amount of apps. For these products, existing mechanisms will take very long time to run and by the time new input has been factored in, it will have grown old.

Attempts have also been made to generate recommendations in the area of Association Rule (ARM) based CF techniques. Similar to traditional CF methods, application of ARM based CF techniques also turned out inefficient for the rapidly growing itemspace. We reasoned the failure of this approach arises due to generation of rules on items which are highly dynamic in nature. We anticipated that a promising solution of these issues could be diminution in the cardinality of the large user-item rating matrix. Thus, instead of generating associations among the items, generation of associations among the categories, which is quasi-static in practice, could be a convenient route.

Our study tackles with the scalability issue of the recommendation algorithm while introducing diversity and maintaining an acceptable degree of accuracy. To address the problem of scalability, sparse user-item rating matrix has been converted to denser user-category rating matrix. Proposed framework for recommendation uses the co-liked categories by several users derived from user-category rating matrix. To show the utility of our approach in practical scenario, we have implemented as well as experimented the algorithm using real world mobile application user data from Mobilewalla (Mobilewalla is a venture capital backed company which accumulates data for mobile applications from four major platforms Apple, Android, Windows, and Blackberry). Additionally, two new metrics have been proposed to evaluate recommendations dealing accuracy and diversity concurrently. It is important to mention that though the association rules have been formulated using the entire database, it can be done off-line and need not to be computed during recommendation generation process.
Our finding shows that this method can be used to predict items with a legitimate recall value in scalable fashion. At this moment, comparatively the precision is not that high; however, the approach also inherently covers diversity as our algorithm generates recommendation using association rules on diverse categories. System scalability has been verified by measuring both offline and online time spent in the process. The result demonstrates the superiority of our approach over traditional CF techniques.

Rest of the paper is organized as follows: the immediate section following this discusses the brief overview of the related literature followed by the problem formulation and our proposed approach. After presenting our empirical results, we discuss our findings and finally conclude with future work recommendations.

2 RELATED WORK

Overwhelming increase in the amount of information over internet raise a requirement of personalized recommendation system for filtering the abundant information. Traditional recommender system predicts a list of recommendations based on two well-studied approaches, collaborative filtering and content-based techniques (Golderberg et al. 1997; Herlocker et al. 2004; Miller et al. 1997). ‘Collaborative filtering’ (CF) concept was pioneered by Goldberg et al. (1992) that uses the historical records of users’ behaviour, either the items previously purchased or the numerical ratings provided by them. Similar users are mined and their known preferences are used to make recommendations or predictions of the unknown preferences for other users (Miller et al. 1997). There are several CF techniques known in literature which can be broadly classified into user based and item based CF technique (Herlocker et al. 2004).

Though traditional CF techniques are adapted by many e-commerce portal, Amazon (Linden et al. 2003), youtube (Davidson et al. 2010), Netflix (Bennet and Andlanning 2007), it has few fundamental drawbacks pointed out earlier and the most important one is scalability issue. CF technique is very much computation expensive and the computational cost grow polynomially with the number of users and items in a system leaving the system ineffective in practice. Recently attempts have been made by several research groups to improve the efficiency of collaborative filtering techniques in different domains. Takács et al. (2009) have employed Matrix Factorization method on Netflix dataset and showed that their method is scalable for large dataset. The efficiency of the method was also verified on MovieLens and Jester dataset. Koren (2010) introduced a new neighbourhood model with an improved accuracy on par with recent latent factor models, and it is more scalable than previous methods without compromising its accuracy. Several incremental CF algorithms are designed (Papagelis et al 2005; Khoshneshin et al. 2010; Yang et al. 2012) to handle the scalability issue. Papagelis et al. (2005) proposed an incremental CF method which updates the user-to-user similarities incrementally and hence suitable for online application. Khoshneshin et al. (2010) proposed an evolutionary co-clustering technique that improves predictive performance while maintaining the scalability of co-clustering in the online phase. Yang et al. (2012) have also proposed incremental item based CF technique for continuously changing data and insufficient neighbourhood problem is handled based on a graph-based representation of item similarity.

Another drawback of CF technique is the data sparsity problem. Because of the fact that in practical scenario most of the users rate only a few number of items, a very sparse user-item rating matrix is generated and the sparsity increases with the growth of item space resulting low accuracy of the system. Cross-domain mediation can be used to address sparsity problem as well as to widen and diversify the recommendation list. In Li et al. (2009), sparsity problem is addressed by transferring a dense user-item rating matrix to target domain. The basic assumption here was that related domains (e.g., books and movies share similar genres) share similar rating patterns and hence can be transferred from one domain to target domain. Ziegler et al. (2004) have proposed a hybrid approach that exploits taxonomic information designed for exact product classification to address the product classification problem. They have constructed user profiles with hierarchical taxonomic score for super and subtopic.
rather than an individual item. This method tried to overcome the sparsity problem in CF techniques and contributed toward generating novel recommendations by topic diversification. However because of the fact that one item may be present in more than one super or sub topic, the structure became more complicated. Later, Ziegler et al. (2005) have also proposed to diversify the topic and return items to the end user by topic diversification, but still these generated recommendations are from the same domain.

Association rule (Agrawal et al. 1993; Agrawal and Srikant 1994) mining technique has also been applied to CF for mining interesting rules for recommendation generation (Kim and Yum 2011; Yu et al. 2005). In Sarwar et al. (2000) the top-N items are generated by simply choosing all the association rules that meet the predefined thresholds for support and confidence, and the rules having the higher confidence value (sorted and top N are chosen finally) have been selected as the recommended items. To address data sparseness and non-transitive associations Leung et al. (2006) proposed a collaborative filtering framework using fuzzy association rules and multilevel similarity.

Applying associations among the items have some shortcomings as it suffers from the problem of ‘new item discovery’. The new item will less likely to be purchased by the users and hence, will not come in the rules unless it is rated by many users. As mentioned earlier, user-item rating matrix is often very sparse, and therefore getting interesting rules is hard.

In all these studies, the authors tried to find the associations among the items and the consequent items in the rules are the candidates for recommendations. In contrast, we have used the association rules to find the association pattern in the category chosen by the users. Since the rules are generated offline, it does not add to computational complexity.

3 SOLUTION INTUITION

Recommendation generation is a single step process that works on item set and user set. However, when both users and items are large and dynamically growing, the scalability, accuracy, and diversity of the recommendation become challenging. In this research, we follow a two-step process where in the first step we focus on the category of items rather than items itself. This reduces the scalability issue significantly, because the number of categories is far less than the number of items. In the next step, we follow an approach that is very similar to the item based recommendation technique. This allows us to focus on much smaller set of items in the second step than typically used in the existing recommendation technique.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U = {u_1, u_2, \ldots, u_n}$</td>
<td>Set of n users</td>
</tr>
<tr>
<td>$I = {I_1, I_2, \ldots, I_l}$</td>
<td>Set of l items in the itemspace</td>
</tr>
<tr>
<td>$D = {d_1, d_2, \ldots, d_m}$</td>
<td>Set of m categories items belong to</td>
</tr>
<tr>
<td>$I_{u_i}$</td>
<td>Items perceived by user $u_i$, where $u_i \in U$</td>
</tr>
<tr>
<td>$I_{u_i}(d_k)$</td>
<td>Items perceived by user $u_i$, from category $d_k$</td>
</tr>
<tr>
<td>$D_{u_i}$</td>
<td>Category set of items perceived by $u_i$</td>
</tr>
<tr>
<td>$C^{d}_{u_i}$</td>
<td>Category interest vector of user of dimension $d =</td>
</tr>
<tr>
<td>$r^{d}_{u_i}$</td>
<td>Rating of $u_i$ for category $d$</td>
</tr>
</tbody>
</table>

Table 1. Notation Table

4 SOLUTION DETAILS

Our proposed system has two main components: (a) Global Knowledge Acquisition Module (GKA) and (b) Recommendation Engine (RE) (See Figure 1). Prior one is mostly done offline while the later one is an online process. At a high level, GKA identifies the categories the user has interest in and also pre-computes the item-item similarity based on meta-data information of the items. The online
component, RE operates on the output of GKA, i.e., the association rules of the categories and item-item similarity matrix to create a profile vector for each user. The generated profile vector is then used to compute the similarity across users and recommend new items accordingly. Next, we describe the details of GKA and RE. Notations used are shown in Table 1.

![Architecture of Recommender System](image)

4.1 Global Knowledge Acquisition Module (GKA)

Input to GKA is the meta-data (name, description, categories) of existing items users have used. Symbolically if \( I_{uj} \) is the set of items user \( u_j \) has used such that \( I_{uj} \subset I \), the input to GKA are \( I_{uj} \forall u_j \in U \). The GKA consists of two main sub-components, category association rule generator and item-item similarity generator.

*Generating Transactional Data on Category Choices:* The goal of this task is to transfer this user-item matrix to denser user-category matrix. Thus, each record in the new matrix corresponds to the transactional information on category for a user. Consider, for each of \( n \) number of users, we have the set of item \( I_{uj} \) used by \( u_j \). Using the item-category mapping, from \( I_{uj} \) we can derive a set of categories \( D_{uj} \) used by user \( u_j \).

*Association Rule Generator for Categories*

In this work we have employed association rule mining (ARM) on the transactional data on categories to find the associations of different categories. Analogous to ‘Market Basket Analysis’, we identify the usage pattern in various categories simply by finding the ‘togetherness’ of the categories in the data with a support and confidence value chosen experimentally. The calculated confidence for each rule is used as the ‘score of closeness’ of the categories. To illustrate, if a user likes a ‘Travel’ application, he might be interested in ‘Restaurant’ applications in that area.

It is worthy to emphasize that in practical scenario item space is more dynamic in nature compared to category taxonomy. As a result frequent re-evaluation of the rule set in ARM based CF on items is inevitable and retards the system proficiency.
**Item-Item Similarity Generator**

In item-based CF techniques, similarities among the items are computed by exploiting the similar rating pattern by the users. In contrast, in this work semantic similarity has been pre-computed for an item pair using item-information meta-data, “Info = [Description, Name]”, i.e., the description of the item and the name of the item. Apache Lucene is used to first index the items and then compute the item-item similarity score based on Cosine similarity. It is independent of the previous module and hence can be done in parallel.

Let us assume a dummy example shown in Table. 2. Say, there are 15 mobile apps are available in the iTunes store from 6 different categories namely ‘Book’, ‘Action Games’, ‘Arcade Games’, ‘Entertainment’, ‘News’, and ‘Classical Music’. Descriptions and names of these 15 mobile apps are crawled and indexed using Lucene indexer. With these two information meta-data, Lucene similarity score has been calculated among these apps. So at most 210 app-pair will have similarity score which is then stored in a knowledge base. In practice, very few app-pair will have non-zero similarity score.

Assume there are 6 users who have used those 15 apps, \( \{a_1, a_2, ..., a_{15}\} \) as shown in Table. 2(A). Say, \( \{a_1, a_2, a_3\} \in \text{‘Books’}, \{a_4, a_5\} \in \text{‘Action Games’}, \{a_6, a_7, a_9\} \in \text{‘Arcade Games’}, \{a_9, a_{10}\} \in \text{‘Entertainment’}, \{a_{11}, a_{12}\} \in \text{‘News’}, \) and \( \{a_{13}, a_{14}, a_{15}\} \in \text{‘Classical Music’} \) (See Table. 2(B)). From the dataset total of 8 association rules are mined (Table. 2(D)) using minimum support = 0.2 and minimum confidence = 0.65. Each of these rules is of the form [rule.antecedent → rule.consequent, confidence].

Once the offline processes are done, they are fed to the recommendation system through the central knowledge base which comprises of the association rules on categories and up-to-date item-item similarity score. This information is accessed each time recommendation is generated for a user.

<table>
<thead>
<tr>
<th>User</th>
<th>Item Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_1 )</td>
<td>{a_1, a_2, a_{13}}</td>
</tr>
<tr>
<td>( u_2 )</td>
<td>{a_2, a_3, a_4, a_{15}}</td>
</tr>
<tr>
<td>( u_3 )</td>
<td>{a_3, a_4, a_9}</td>
</tr>
<tr>
<td>( u_4 )</td>
<td>{a_2, a_3, a_4, a_5, a_7, a_{13}}</td>
</tr>
<tr>
<td>( u_5 )</td>
<td>{a_9, a_9, a_{12}, a_{13}}</td>
</tr>
<tr>
<td>( u_6 )</td>
<td>{a_1, a_9, a_{12}, a_{15}}</td>
</tr>
</tbody>
</table>

**Table 2. Association Rule Generation Process**

<table>
<thead>
<tr>
<th>Association Rules</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Classical Music → Book</td>
<td>1</td>
</tr>
<tr>
<td>2 Book → Classical Music</td>
<td>0.833</td>
</tr>
<tr>
<td>3 [Book, Action Games] → Arcade Games</td>
<td>0.67</td>
</tr>
<tr>
<td>4 [Book, Arcade Games] → Action Games</td>
<td>1</td>
</tr>
<tr>
<td>5 [Arcade Games, Action Games] → Book</td>
<td>1</td>
</tr>
<tr>
<td>6 [Book, Entertainment] → News</td>
<td>1</td>
</tr>
<tr>
<td>7 [Entertainment, News] → Book</td>
<td>1</td>
</tr>
<tr>
<td>8 [Book, News] → Entertainment</td>
<td>1</td>
</tr>
</tbody>
</table>

Diagram representation of user and item set association with category and item set association.
4.2 Recommendation Generation Module

Recommendation generation module which is core to generate online recommendations consists of 4 sub-modules. The first step is to generate the profile for the users using pre-computed category association rules and item-item similarity matrix. Afterwards the generated user profile is updated in the profile database. Next, the neighbourhood of the active user \( u_t \) is formed. Finally, the recommendations are generated from the \( u_t \)'s top \( N \)-similar users' item list. Next follows the detailed discussion of these four steps.

4.2.1 User Profile Generator

User profile comprises of two features, namely category affinity vector and item feature. Below, we define the category score and category affinity vector for a user.

**Definition: (Category Score)** For a user \( u_j \) category score for category \( d_k \) is the fraction of items \( u_j \) own from category \( d_k \)

\[
Score_{u_j}^{d_k} = \left| I_{u_j}(d_k) \right| / \left| I_{u_j} \right|
\]

If the user does not have any item from category \( d_k \) then \( Score_{u_j}^{d_k} = 0 \).

**Definition: (Category Affinity Vector)** For \( n \) categories \( \{d_1, d_2, \ldots, d_n\} \) for user \( u_j \) category affinity vector is an \( n \) dimensional vector with each entity being the category score defined as above, i.e.,

\[
C_u^d = \{Score_{u_j}^{d_1}, Score_{u_j}^{d_2}, \ldots, Score_{u_j}^{d_n}\}
\]

For a user, the category affinity vector defines the preference of the user over different categories in the application domain.

<table>
<thead>
<tr>
<th>Initial Category Affinity Vector</th>
<th>User Profile</th>
<th>Updated Category Affinity Vector</th>
<th>Item List</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(After normalization)</td>
<td></td>
</tr>
<tr>
<td>( u_1 )</td>
<td>0.667</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( u_2 )</td>
<td>0.5</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>( u_3 )</td>
<td>0.25</td>
<td>0.25</td>
<td>0.5</td>
</tr>
<tr>
<td>( u_4 )</td>
<td>0.333</td>
<td>0.333</td>
<td>0.1</td>
</tr>
<tr>
<td>( u_5 )</td>
<td>0.25</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>( u_6 )</td>
<td>0.25</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 3. Category Affinity Vector Calculation**

Say, user \( u_2 \) has 4 mobile apps \{a_2, a_3, a_4, a_{15}\} installed in his cell phone (Table 3), 2 from ‘Books’ category, 1 from ‘Action Games’ category, and 1 from ‘Classical Music’ then initial \( C_{u_2}^d = [0.5, 0.25, 0, 0, 0, 0.25] \).

**Updating category affinity vector:** Once initial \( C_{u_j}^d \) is calculated for each user, ARM generated rules are injected to update \( C_{u_j}^d \). If the user has expressed interest in categories in the antecedent part of the
rule, then the categories in the consequent part of the rule are updated with an average score of the antecedent categories weighted by the confidence of the rule.

From the dataset 8 association rules are mined as mentioned earlier (Table 2(D)). Extending the previous example, ‘Arcade Games’ is added to \( C_{u_2}^d \) with score \( (0.5 + 0.25)/2 \times 0.67 = 0.25125 \) (using association rule \{Book’, ‘Action Games’\} \( \rightarrow \) ‘Arcade Games’, 0.67 ) respectively. Thus, the category affinity vector reduces to \([0.5,0.25,0.25125,0,0,0.25]\). Next, the vector is normalized to unity. Note that, the other rules were skipped as \( u_2 \) does not own ‘Entertainment’ or ‘News’ app resulting category affinity vector as \([0.399,0.1998,0.201,0,0.199]\). Similarly, the category vectors are updated for other users (See Table. 3).

**Item Feature:** Once category affinity vector is calculated for the users, items set \((I_{u_j})\) for a user is added to the profile to find the semantic similarity of items in \( I_{u_j} \) with \( I_{u_k} \forall I_{u_k} \not\in I_{u_j} \) in later phase. For user \( u_2 \) reference of 4 apps \( \{a_2,a_3,a_4,a_5\} \) installed in his cell phone is added to his profile.

### 4.2.2 User Profile Database

For the existing users, profile generation can be done offline and stored in the database, whereas for the new users, the generated profile can be updated for future reference.

### 4.2.3 Neighbourhood Formation

For an active user \( u_i \) we need to find the similar peers using the well-known proximity measure described below:

**Proximity Measurement.** For similarity computation there are several measures exist in the literature. In this work, we have used Pearson correlation to measure the similarity between two users defined as follows:

\[
r(u_i, u_j) = \frac{\sum_{k=1}^{n}(v_{ik} - \bar{v}_i)(v_{jk} - \bar{v}_j)}{\sqrt{\sum_{k=1}^{n}(v_{ik} - \bar{v}_i)^2(v_{jk} - \bar{v}_j)^2}}
\]

**Table 4. Calculation of Category Score**

<table>
<thead>
<tr>
<th>ScoreCategory((u_i, u_j))</th>
<th>(u_1)</th>
<th>(u_2)</th>
<th>(u_3)</th>
<th>(u_4)</th>
<th>(u_5)</th>
<th>(u_6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(u_1)</td>
<td>1</td>
<td>0.793</td>
<td>0.159</td>
<td>0.535</td>
<td>0.463</td>
<td>0.463</td>
</tr>
<tr>
<td>(u_2)</td>
<td>1</td>
<td>0.649</td>
<td>0.8905</td>
<td>-0.1736</td>
<td>-0.1736</td>
<td></td>
</tr>
<tr>
<td>(u_3)</td>
<td>1</td>
<td>0.595</td>
<td>-0.684</td>
<td>-0.684</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(u_4)</td>
<td>1</td>
<td>-0.433</td>
<td>-0.433</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(u_5)</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(u_6)</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For two different features the similarity score is calculated.

- **Category Feature:** For user pair \((u_i, u_j)\), Pearson correlation has been computed between \( C_{u_i}^d \) and \( C_{u_j}^d \) using Eq. 1 and is denoted by \( \text{Score}_{\text{Category}} \).

In Table 4, for each pair of users, the category score has been calculated using Eq. 1.

- **Item Feature:** For \( u_i \) and \( u_j \), item similarity score \( \text{Score}_{\text{Item}} \) has been computed as the average semantic similarity of the pair of item set \( I_{u_i} \) and \( I_{u_j} \) normalized by \(|I_{u_i}| \times |I_{u_j}|\). Here, by semantic similarity of two sets we mean semantic similarity of the elements of the two sets.
\[ Score_{item}(u_i, u_j) = \frac{1}{|I_{u_i}| |I_{u_j}|} \sum_{p \in I_{u_i}, q \in I_{u_j}} \Sim(p, q) \]

Eq. 2

\[ Score_{item}(u_{12}) \] with all the other items are calculated using Eq. 2 as shown in Table 5.

Further, a weighted score for these two features has been calculated and the final score is computed as

\[ \text{Score}(u_i, u_j) = w_1 \cdot \text{Score}_{item}(u_i, u_j) + w_2 \cdot \text{Score}_{category}(u_i, u_j) \]

where \( w_1 + w_2 = 1 \). \( w_1 \) and \( w_2 \) are decided experimentally, though we gave higher weight to \( w_2 \) compared to \( w_1 \).

With \( u_2 \)'s category affinity vector, Pearson correlation has been computed with all the other users \( (u_1, u_3, u_4, u_5, u_6) \) category affinity vectors which comes out to be 0.793, 0.649, 0.8905, -0.1736, -0.1736 respectively.

Assume \( w_1 = 0.4 \) and \( w_2 = 0.6 \) and the score for each pair of users is calculated.

<table>
<thead>
<tr>
<th>Score</th>
<th>( u_1 )</th>
<th>( u_3 )</th>
<th>( u_4 )</th>
<th>( u_5 )</th>
<th>( u_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Score}<em>{item}(u_i, u</em>{12}) )</td>
<td>0.7</td>
<td>0.7</td>
<td>0.9</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>( \text{Score}<em>{category}(u_i, u</em>{12}) )</td>
<td>0.793</td>
<td>0.649</td>
<td>0.8905</td>
<td>-0.1736</td>
<td>-0.1736</td>
</tr>
<tr>
<td>( \text{Score}(u_i, u_{12}) )</td>
<td>0.4758</td>
<td>0.3894</td>
<td>0.8943</td>
<td>0.17584</td>
<td>0.17584</td>
</tr>
</tbody>
</table>

Table 5. Score calculation among the items

Neighbour Selection. To find the top neighbours, two approaches can be used: by either setting a threshold value above which the peers are considered as similar users, or selecting top-N users similar to active user \( u_{12} \). In this work, we have chosen the top-N users as neighbours.

For user \( u_2 \) if we take top-1 neighbour, then \( u_4 \) becomes the selected one (similarity score = 0.8943).

4.2.4 Recommendation Generation

Once the neighbourhood is generated from the set of users, recommendations are generated from the items of the top-N users’ list. From the top-N neighbours we find the items those are not in item \( u_2 \)'s list. These items are ranked and assigned the similar users’ similarity value as the score. If one item is recommended from many users in top-N user list, then the user’s score is added up to assign a higher score to that item. Finally the top-k items have been recommended to the active user. Recommendations are generated from \( u_4 \)'s item list where only three apps \( (a_5, a_7, a_{13}) \) can be recommended to user \( u_2 \) as he owns all the other apps from \( u_4 \)'s list. For \( a_5, a_7, a_{13} \) score has been assigned as 0.8943 since \( \text{Score}(u_4, u_{12}) = 0.8943 \) (Table 5).

5 EXPERIMENTAL RESULTS

In this section we will discuss the experimental results evaluating the accuracy and scalability of the recommendation system. Experimental settings are described first followed by the findings of the proposed algorithm.
5.1 Experimental Settings

All modules are implemented in Java 1.6 and MySQL v5.1 was employed as database back-end. All modules and the database reside in the same computer (a server equipped with a 2.33 GHz quad-core CPU and 8 GB RAM, and running on Linux operating system).

5.1.1 Data Acquisition

Experiment has been conducted with a real world data of mobile app users’ reviews as a surrogate of installed apps in user’s mobile phone. A sample of user review of Apple app users and the corresponding app information have been collected from Mobilewalla\textsuperscript{1}, which contains the following dimensions depicted in Table 6. Total of 1744811 users’ information have been collected, out of which only 22213 users who have rated more than 5 apps are considered in this study.

Getting the real mobile app usage data is hard in real scenario. Thus, to evaluate the effectiveness of our method, review data for the mobile app users have been used as surrogate for usage data. Descriptive statistics of the data is given in Table 7 and on average there are 3 reviews per app. Each mobile application belongs to one or more categories. The generation of association rules among these categories is discussed below. With support threshold of 0.1 and confidence threshold of 0.7, total 72407 frequent item sets are mined generating 977678 rules.

To find the similar apps first they are indexed using Lucene and later the similarity score (in [0,1]) has been calculated for each pair of apps\textsuperscript{2}. Data is randomly divided into training ($A_{tr}$) (50%), and test set ($A_{ts}$) (50%), where both the datasets contain mutually exclusive items as well as categories at user level. To illustrate, say a user has four apps from two categories, Games and Entertainment ($a_1, a_2 \in \text{‘Games’}, a_3, a_4 \in \text{‘Entertainment’}$. Then we keep $a_1, a_2$ in $A_{tr}$ and $a_3, a_4$ in $A_{ts}$. Experiment is conducted for 5000 users and collaborative filtering technique (user based and item based CF method) and content based method are used to compare the result with the proposed one. While recommendations are generated using the training set data, test set data is used for evaluation.

<table>
<thead>
<tr>
<th>App and User Details</th>
<th>Variables</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>iTunes ID of the author (unique)</td>
<td>Total Number of Users</td>
<td>22213</td>
</tr>
<tr>
<td>Application ID</td>
<td>Total Number of Products</td>
<td>66137</td>
</tr>
<tr>
<td>Name of the app</td>
<td>Average Number of Products Rated per User</td>
<td>3</td>
</tr>
<tr>
<td>Description of the app</td>
<td>Total Number of Categories</td>
<td>194</td>
</tr>
</tbody>
</table>

Table 6. App and User Details

<table>
<thead>
<tr>
<th>Variables</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of Users</td>
<td>22213</td>
</tr>
<tr>
<td>Total Number of Products</td>
<td>66137</td>
</tr>
<tr>
<td>Average Number of Products Rated per User</td>
<td>3</td>
</tr>
<tr>
<td>Total Number of Categories</td>
<td>194</td>
</tr>
</tbody>
</table>

Table 7. Descriptive Statistics

5.2 Evaluation Metrics

In traditional recommendation systems, performance (precision and recall) is measured by calculating the exact match of the items in the test set with that of generated recommendation. While exact match would be preferable, but the similar predictions should not be overlooked. Thus, instead of evaluating the predictions against the exact item set, we have examined the closeness of user’s actual taste and generated recommendations. To evaluate recommendation performance, we have each algorithm

\textsuperscript{1} Mobilewalla is a venture capital backed company for Mobile app search http://mobilewalla.com/, which accumulates data for mobile applications from four major platforms Apple, Android, Windows and Blackberry.

\textsuperscript{2} Apache, Lucene, http://lucene.apache.org/core/
generate a ranked list of recommended items for each user and then the recommended items are compared with the actual transactions in test data. Measures used for evaluation are discussed in turns.

- **Binary Precision and Binary Recall**: In binary precision and recall we assume if two items are similar with respect to a predefined threshold \( \beta_{th} \), then the two items are same. “Binary Precision” and “Binary Recall” are formulated similar to the standard precision and recall. The only difference in our metrics is that two items are considered same when they are similar for a threshold \( \beta_{th} \). We define “Binary Precision” and “Binary Recall” as follows:

\[
\text{Binary Precision}(\beta_{th}) = \frac{|A_{ts} \cap A_o|}{|A_o|} \quad \text{and} \quad \text{Binary Recall}(\beta_{th}) = \frac{|A_{ts} \cap A_o|}{|A_{ts}|}
\]

\[l_i = l_j \text{ if } \text{Sim}(l_i, l_j) > \beta_{th} \text{ where } l_i \in A_o, l_j \in A_{ts}\]

\(\text{Binary Precision}(\beta_{th})\) depicts the fraction of items in the recommendation list similar with the expected ones with a similarity threshold \( \beta_{th} \). On the other hand \(\text{Binary Recall}(\beta_{th})\) explains the fraction of items in the expected list similar with the recommended ones for a similarity threshold \( \beta_{th} \).

- **Fuzzy Precision and Fuzzy Recall**: Fuzzy precision and fuzzy recall (BartosZiolko et al. 2007) is defined with a membership function of an element \( l_i \) in a set \( A_k \) by the maximum similarity score of \( l_i \) with all the remaining elements in \( A_k \).

\[
\text{Fuzzy Precision} = \frac{\sum_{l_j \in A_o} f(A_{ts}\{l_i\})}{|A_o|} \quad \text{and} \quad \text{Fuzzy Recall} = \frac{\sum_{l_j \in A_{ts}} f(A_o\{l_i\})}{|A_{ts}|}
\]

where membership function \( f(A_k\{l_i\}) = \max_{l_j \in A_k} \text{Sim}(l_i, l_j) \)

It is worthy to mention, binary precision and recall is a special case of fuzzy precision and recall with a membership value of 1.

- **Entropy of Recommendation List**: Entropy of a recommendation list is defined as \( H = -\sum_{i=1}^{n} p(i) \log p(i) \) where \( p(i) \) is the probability of occurrence of item \( i \) in the recommendation list and is calculated based on popularity fraction of that item. Higher entropy denotes that the distribution is less biased to only popular items.

5.3 Experimental Findings

The proposed method (ADR) has been compared with two traditional collaborative filtering techniques (Item-based CF (ICF), User-based CF (UCF)) and Content-based technique (CR). In CR the content similarity of two items is computed based on the tags extracted from item descriptions. The comparative results of three aforementioned techniques with the proposed one are discussed in terms of accuracy, scalability, and entropy of recommendation in the following section.

5.3.1 Accuracy

From Figure 2 and Figure 3 it is clear that our algorithm (ADR) has very high recall value for small \( \beta_{th} \) and it reduces with increase of \( \beta_{th} \). For \( \beta_{th} = 0.8 \) the recall value coincides with other three methods. On the other hand, precision value for ADR is higher than that of ICF and UCF but lower than CR for all values of \( \beta_{th} \). Fuzzy precision and recall has also been compared where fuzziness gives the true average closeness of test set and recommended set (Figure 4 and Figure 5). Similar to binary recall, ADR outperforms all three methods in terms of fuzzy recall. However, for fuzzy precision ADR is comparable to UCF but lower than CR.

5.3.2 Scalability

To measure the scalability of the system, time spent in offline and online recommendation generation processes have been plotted for all the algorithms (Figure 6, Figure 7) where user-base has been
increased keeping the number of items fixed (66,137). From Figure 6 it is clear that scalability in offline process for ICF, UCF, and ADR are comparable and superior to CR. Figure 7 shows that in online process time spent in ICF and UCF increases rapidly with increasing user-base. In comparison, under the same conditions, time spent in ADR is acceptable. CR consumes minimum time because the item-item similarity has already been calculated during the offline process and stored in the database for online reference.

5.3.3 Entropy of Recommendation List

For each user top 40 recommendations are generated and entropy for the recommendation list is calculated using entropy formula discussed earlier. The highest value of entropy for ADR indicates that the generated recommendations are not biased to only popular items (Figure 8).
CONCLUSION AND FUTURE WORK

In this paper we have described a novel approach to generate item recommendations for users in a scalable fashion. Association rule mining approach is used to generate rules for interrelated categories from users’ transactions and following which user’s profile is updated using pre-computed rules to redefine his category interest. In distinct contrast to traditional approach where association rules are structured among the items (dynamic in nature), we have generated these rules among the categories (quasi-static in practice). Our findings show that this method can be used to predict items with a legitimate recall value though comparatively the precision is not that high. System scalability has been verified by measuring both offline and online time spent in the process. Two measures have been proposed to find the modified precision and recall value when recommendation evaluation is an issue. For future work, it would be interesting to investigate how social information can be integrated to the user profiles to understand their product preference. This would lead us to find the users in the community who share similar taste with the active user, for which there are now limited methods available, but will be very important in future.
References


