Using Groups of Friends to Help Reduce the Cold-Start Problem

STEPHEN TSOUROUNIS
University of Toronto

Keywords: Collaborative Filtering, Cold-Start Problem

1. INTRODUCTION

In everyday life, people rely on recommendations from other people by spoken words, reference letters, news reports from the news media, general surveys, travel guides, and so forth [1]. Recommender systems have been developed to address this complex problem of matching users to items of potential interest. As [2] explains, a good recommender system will recommend items that fall into the users interests. A great recommender system will expand a users interests into neighboring areas. Such a system will not only recommend items that fall into the users interests, but will identify new potential interests from which to suggest items. In the music domain, people in the article from [3] are skeptical about the effectiveness of algorithms to recommend songs:

> However sophisticated the algorithms are, they will not be able to take into account the random ways in which we discover music and this method of filtering music for us to listen to, is limiting, wrote one commenter.

In order to improve these algorithms, researchers need to take into account the many different ways that users get songs recommended to them. One user from [3] said "word of mouth has a much greater effect than some computer generated recommendation." To my knowledge, limited work has investigated music recommendation that takes into account a user's group of friends. This could be due to the limited amount of that has been made available to researchers. Some researchers [4, 5] have tried to make up for that by creating arbitrary groups in two ways:

(1) Random: stochastic groups makes the assumption that users can meet completely randomly.
(2) Similarity Threshold: making the assumption that friends share some sort of similarity in music preferences.

The problem is that this is not a true measure of real social networks and we cannot get a sense of the true information-value by making these assumptions.

By using motivation from assumption (2), that friends share some sort of similarity in music preferences, this paper attempts to look at the information value of social networking information (i.e., bi-directional friend relations) in order to solve the cold-start problem.

1.1 The Cold-Start Problem

The cold-start problem is when the system cannot draw any inferences for users or items that have not yet gathered sufficient amount of information. This usually occurs when a user or item is first added to the system. That is, recommending a new item that no one has yet rated, or providing recommendations for new users that you have no preference information for.

1.1.1 *Content-based approach.* In the content-based approach, the system must be capable of matching the characteristics of an item against relevant features int he user's profile. In order to do this, it must first construct a sufficiently-detailed model of the user's tastes and preferences through preference elicitation. This may be done either explicitly or implicitly. In both cases, the cold start problem would imply that the user has to dedicate an amount of effort using the system in its 'dumb' state - contributing to the construction of their user profile - before the system can start providing any intelligent recommendations.

1.1.2 *Collaborative filtering approach.* In the collaborative filtering approach, the recommender would identify users who share the same preferences (e.g., rating patterns) with the active users, and propose items which the like-minded users favoured (and the active user has not yet seen). Due to the cold start problem, this approach would fail to consider items which no-one in the community has rated previously.

1.1.3 *Current Solutions.* In recommender systems, the cold-start problem is often reduced by adopting a hybrid approach between content-based matching and collaborative filtering. New items (which have not yet received any ratings in the community) would be assigned a rating automatically, based on the ratings assigned by the community to other similar items. Item similarity would be determined according to the items' content-based characteristics [6].

The construction of the user's profile may be automated by integrating information from other user activities, such as browsing histories. If, for example, a user has been reading information about a particular music artist from a media portal, then the associated recommender system would automatically propose the artist's releases when the user visits the music store [7].

In this paper, we look at the collaborative filtering approach.

2. COLLABORATIVE FILTERING

Collaborative Filtering (CF) is a method of making predictions about an individual's preferences based on the preference information from many users. In a typical CF scenario, there is a list of n users $\{u_1, u_2, ..., u_n\}$ and a list of m items $\{i_1, i_2, ..., i_m\}$. Let $I(u_i)$ be the set of items which user u_i has rated (or their preferences that was inferred implicitly through their behaviours). Also, the set, $U(v_i)$, represents the users have have rated item v_i. CF techniques use a database in the form of a user-item matrix R of preferences, where $R = \{r_{ui}\} \in \mathbb{R}^{n \times m}$ represent how much user u likes item i. In this paper, we assume $r_{ui} \in \{0, 1\}$. Entries r_{ui} represent the fact that user u have listened to (or would like to listen to) song i. In this paper we refer to items or songs interchangeably.

2.1 Memory-Based Collaborative Filtering Techniques

In memory-based CF algorithms, the entire user-item matrix is used to generate a prediction. Generally, given a new user for which we want to obtain the prediction, the set of items to suggest are computed by looking at similar users. This strategy is typically referred to as *user-based recommendation*. Alternatively, int he *item-based recommendation* strategy, one computes the most similar items for the items that have been already purchased by the active user, and then aggregates those items to form the final recommendation.

Neighbourhood-based CF algorithms use the following steps [1]:

(1) Calculate the similarity, $w_{i,j}$, which reflects distance, correlation, or weight, between two users (or items), i and j.
(2) Produce a prediction for the active user by taking the weighted average of all the ratings of the user (or item) on a certain item (or user), or using a simple weighted average.
(3) When the task is to generate a top-N recommendation, we need to find k most similar users or items (nearest neighbours) after computing the similarities, then aggregate the neighbours to get the top-N most frequent items as the recommendation.

In this paper I focus on the simple weighted-sum strategy used by [8]. In the *user-based* type of recommendation, the scoring function, on the basis of which recommendation is made, is computed by:

$$h_{ui} = \sum_{v \in U} f(w_{ij}) r_{vi} = \sum_{v \in U(i)} f(w_{uv}). \tag{1}$$

That is, the score obtained on an item for a target user is proportional to the similarities between the target user u and the other users v that have listened to the item $i (v \in U(i))$. This score will be higher for songs which are often rated by similar users.

On the other hand, within an *item-based* type of recommendation, the target item i is associated with a score:

$$h_{ui} = \sum_{j \in I} f(w_{ij}) r_{uj} = \sum_{j \in I(u)} f(w_{ij}). \tag{2}$$

Hence, the score is proportional to the similarities between item i and the other items already purchased by the user $u (j \in I(u))$.

The function $f(w)$ can be assumed monotonic not decreasing and its role is to emphasize similarity contributions in such a way to adjust the *locality* of the scoring function, that is how many of the nearest users/items really matter in the computation.

We can think of the user-item matrix $R = \{r_{ui}\}$ as a bipartite graph where the set of nodes is $N = U \cup I$, and there exist only edges from $u \in U$ to $i \in I$ whenever $r_{ui} = 1$. Now, the user based recommendation strategy corresponds to the intuition that if a user tends to link to the same set of items as the active users, then this gives us some evidence that there can exist a link between the active user and the item i. Dually, in item-based recommendation, the intuition is that, if the active user links to one out of two items which tend to be linked by the same users, then, we can infer that the active users will probably like the other item as well.

2.2 Similarity Measure

A central component of these neighbourhood-based methods is the similarity function used to compute the neighbours [1]. Several functions have been proposed including Cosine Similarity [9]. In the case of binary grades, the cosine similarity can be simplified as in the following. The cosine similarity between two users u, v will be defined as:

$$w_{uv} = \frac{|I(u) \cap I(v)|}{|I(u)|^{\frac{1}{2}} |I(v)|^{\frac{1}{2}}} \tag{3}$$

Similarly for items i, j, we have:

$$w_{ij} = \frac{|U(i) \cap U(j)|}{|U(i)|^{\frac{1}{2}} |U(j)|^{\frac{1}{2}}} \tag{4}$$

The cosine similarity has the nice property to be symmetric but it might not be the better choice. As pointed out by [8], in fact, especially for the item case, we are more interested in computing how likely it is that an item will be appreciate by a user when we *already* know that the same user likes another item. It is clear that this definition is not symmetric. As an alternative to the cosine similarity is to take the conditional probability measure which can be estimated with the following formulas:

$$w_{uv} = P(u|v) = \frac{|I(u) \cap I(v)|}{|I(v)|} \tag{5}$$

and

$$w_{ij} = P(i|j) = \frac{|U(i) \cap U(j)|}{|U(j)|} \tag{6}$$

[10] pointed out that the conditional probability measure of similarity, $P(i|j)$, will tend to have higher values for more popular items, which could be a nice property for a recommender system.

3. EXPERIMENTS

We make use of the Last.FM dataset [11] (described in the next section) which contains social network information that we can exploit in order to attempt to find the information-value of groups of friends and song recommendations. We take the following steps in our method:

(1) Find groups by analyzing the bi-directional friend relations of users and find fixed-size, k, cliques. We use max-k cliques (with $k = 4$), so that all users with friends can be used (i.e., 2-cliques represent bi-directional friend relations).
(2) Rank each clique (by taking the average user-similarity with everyone in the clique) and take the highest ranking clique for each user.
(3) Take items from each user in the clique and add them to the active user's preferences for recommendation.

By ranking each clique, the performance *should* improve as the user rates more items and adds more friends.

I hypothesize that, at first, this clique-method will do worse than baseline but better than user/item neighbourhood-based approaches (i.e., when we have little or no information about the user). As the active user adds friends and item ratings (preferences), I hypothesize that this approach will do better than both the baseline and user/item neighbourhood-based approaches until a certain point. After that the user/item neighbourhood-based approach for a single user will probably outperform the clique/group-based predictions. This is illustrated in Figure 1.

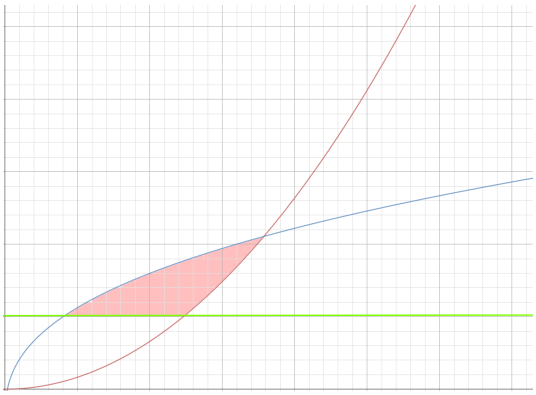

Fig. 1. Prediction of the experiments of the clique-based (blue), item-based (red), and baseline (green) predictions. The predicted information-gain that the clique based recommendations is shaded in pink.

3.1 Last.FM Dataset

The Last.FM dataset [11] contains:

(1) 1892 users
(2) 17632 artists
(3) 12717 bi-directional user friend relations
—Avg. 13.443 friend relations per user
(4) 92835 user-listened artist relations
—Avg. 49.067 artists listened by each user
—Avg. 5.265 users who listened to each artist

The test sets are created and cross-validated by randomly taking 25% of users and removing an arbitrary amount of user-preferences from the training data (depending on the experiment).

3.2 Truncated Mean Average Precision

I used the truncated mAP (mean average precision) as the evaluation metric. Let y denote a ranking over items, where $y(p) = i$ means that item i is ranked at position p. The mAP metric emphasizes the top recommendations. For any $k \leq \tau$, the *precision at k* (π_k) is defined as the proportion of correct recommendations within the top-k of the predicted ranking (assuming the ranking of y does not contain the visible songs).

$$\pi_k(u, y) = \frac{1}{k} \sum_{p=1}^{k} r_{uy(p)} \tag{7}$$

For each user, the truncated average precision is the average precision at each recall point:

$$AP(u, y) = \frac{1}{\tau_u} \sum_{p=1}^{\tau} \pi_k(u, y) r_{uy(p)} \tag{8}$$

where τ_u is the smaller between τ and the number of the user u's positively associated songs. The average of $AP(u, y_u)$'s over all users give the mean average precision (mAP).

4. RESULTS

As a baseline, we use recommendations by popularity. With this strategy, each song i simply gets a score proportional to the number of users $|U(i)|$ which listened to the song. The baseline is shown in the following table:

Method	mAP@100
Baseline (recommendation by popularity)	0.105893

For both item- and clique-based methods, we took the mAP@100 when giving recommendations, given a variable number of item-ratings for the active users in the test sets. The results from the *item-based* neighbourhood approach is shown in the following table:

Number of rated items	Item-based mAP@100
0	0.0
5	0.00880
10	0.02660
15	0.05063
20	0.07714
25	0.10182
30	0.12235
35	0.14242
40	0.16158

The following table shows the results for the clique-based method. To recall, for each active user with friends, items from users of high-ranking cliques (i.e., cliques that contain users most similar to the active user) were added to the user's item preference when doing recommendation:

Number of rated items	Clique-based mAP@100
0	0.01010
5	0.02948
10	0.04417
15	0.05932
20	0.07474
25	0.09013
30	0.10427
35	0.12280
40	0.12184

As predicted, baseline (popularity-based) methods outperformed both item-based and clique-based recommendation when the active user had little or no item preferences. Also, the clique-based recommendations outperformed the item-based until enough items were rated by the active user. The item- and clique-based recommendations outperform the baseline measure after a certain number of items have been rated by the active user (25 and 30, respectively). What was not anticipated was that the item-based recommendations would outrank the the baseline popularity measure before the clique-based. This could be because the items in the data represent artists (as opposed to songs), which in some ways represent a category as opposed to a particular item. A comparison of methods are shown in a graph in Figure 2.

5. CONCLUSIONS AND FUTURE WORK

Even though this experiment did not work as expected, the expected shape of the prediction accuracies was more or less correct (which was pretty cool to find out). In the future, I would like to try this method with actual song data (as opposed to artists) because the popularity baseline may be weaker than in this last.FM dataset.

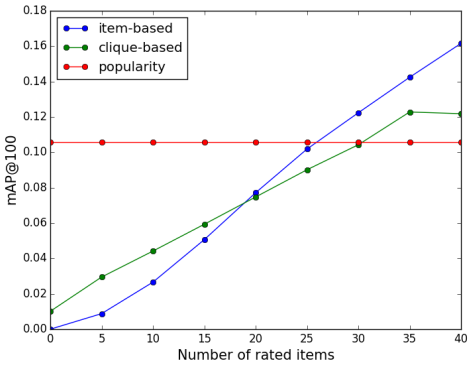

Fig. 2. Graphic comparison of the mAP for the item-based neighbourhood recommender (blue), the proposed clique-based neighbourhood recommender (green), and the baseline (red), for different numbers of rated items for the active user.

6. REFERENCES

(1) Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. Advances in artificial intelligence, 2009.

(2) Christodoulakis, C. (2014). Design Considerations for an Effective Recommendation Manager in the Context of the IBM LabBook Distributed Collaborative Analytics Platform.

(3) The Gaurdian, ”5 Types of Music Discovery”. http://www.theguardian.com/technology/2014/mar/19/music-discovery-spotify-apps-facebook

(4) Roy, S. B., Lakshmanan, L. V., & Liu, R. (2015). From Group Recommendations to Group Formation. arXiv preprint arXiv:1503.03753.

(5) Jameson, A. (2004, May). More than the sum of its members: challenges for group recommender systems. In Proceedings of the working conference on Advanced visual interfaces (pp. 48-54). ACM.

(6) Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, David M. Pennock (2002). Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002). Methods and Metrics for Cold-Start Recommendations. New York City, New York: ACM. pp. 253260. ISBN 1-58113-561-0.

(7) Xiam (2007-06-29). ”Vendor attempts to crack cold start problem in content recommendations”. Mobile Media (PDF) (United Kingdom: Informa Telecoms & Media): 18.

(8) F. Aiolli, A Preliminary Study on a Recommender System for MSD Challenge Preference Learning: Problems and Applications in AI (PL-12).

(9) Tan, P. N., Steinbach, M., & Kumar, V. (2006). Introduction to data mining (Vol. 1). Boston: Pearson Addison Wesley.

(10) George Karypis, Evaluation of item-based top-n recommendation algorithms, in CIKM, pp. 247254, (2001).

(11) Last.FM dataset. HetRec 2011, GroupLens. http://grouplens.org/datasets/hetrec-2011.