Top-N Recommendation for Shared Accounts

Koen Verstrepen Bart Goethals

University of Antwerp
Antwerp, Belgium
{koen.verstrepen,bart.goethals}@uantwerp.be

ABSTRACT

Standard collaborative filtering recommender systems assume that every account in the training data represents a single user. However, multiple users often share a single account. A typical example is a single shopping account for the whole family. Traditional recommender systems fail in this situation. If contextual information is available, context aware recommender systems are the state-of-the-art solution. Yet, often no contextual information is available. Therefore, we introduce the challenge of recommending to shared accounts in the absence of contextual information. We propose a solution to this challenge for all cases in which the reference recommender system is an item-based top-N collaborative filtering recommender system, generating recommendations based on binary, positive-only feedback. We experimentally show the advantages of our proposed solution for tackling the problems that arise from the existence of shared accounts on multiple datasets.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information filtering

Keywords: Collaborative filtering, recommender systems, nearest neighbors, explaining recommendations, shared account.

1. INTRODUCTION

Typical recommender systems assume that every user- account represents a single user. However, multiple users often share a single account. An example is a household in which all people share one video-streaming account, one music-streaming account, one online-shopping account, one loyalty card for a store they make purchases in, etc.

Three problems arise when multiple users share one account. First, the *dominance problem* arises when all recommendations are relevant to only some of the users that share the account and at least one user does not get any relevant recommendation. We say that these few users dominate the account. Consider, for example, a family that often purchases household items. Once in a while they also purchase toys for the children together with the household items. Now, it is likely that all recommendations will be based on the numerous household items and the recommender system will be essentially useless for the children.

Second, the *generality problem* arises when the recommendations are only a little bit relevant to all users in the shared account, but are not really appealing to any of them. When the diverse tastes of multiple users are merged into one account, the recommender system is more likely to recommend overly general items that are preferred by most people, regardless their individual tastes.

Third, if the recommender system would be able to generate relevant recommendations for every user in the shared account, how does every user know which recommendation is meant for her? We call this the *presentation problem*.

If contextual information such as time, location, buying intent, item content, session logs, etc. is available, context aware recommender systems are the state-of-the-art solution to split accounts into multiple users and detect the identity of the active user at recommendation time.

However, often no contextual information is available for splitting the accounts. A first example concerns the case of the numerous organizations that simply did not keep records of any contextual information in the past, not even time stamps. A second example are families that shop together in a hypermarket: they have one loyalty card account and bundle their purchases when they visit the store. In this case, the context is exactly the same for every family member and cannot be used to split the family account into its members. Therefore, we introduce the challenge of *top-N recommendation for shared accounts in the absence of contextual information*, in which the above three shared account problems are tackled without using any contextual information.

Formally, we consider the setting of collaborative filtering with binary, positive-only preference feedback. We represent the available data as a preference matrix in which the rows represent the users and the columns represent the items. Every value in this preference matrix is 1 or 0, with 1 representing a known preference and 0 representing the unknown. Pan et al. [12] call this setting one-class collaborative filtering (OCCF) but it is also referred to as top-N recommendation based on binary, positive-only preference data [2]. This kind of data is typically associated with implicit feedback [5]. However, it can also be the result of explicit feedback. Likes on social networking sites for example, are explicit, binary and positive-only. Other applications that correspond to this version of the collaborative

RecSys'15, September 16–20, 2015, Vienna, Austria.

ACM 978-1-4503-3692-5/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2792838.2800170.

filtering problem are tags for photo's, words for documents, articles bought by a customer etc.

Despite the significance of *top-N recommendation for shared accounts in the absence of contextual information*, we know of no prior research on tackling this challenge. We give a start to filling this gap by proposing a solution for all cases in which the reference recommender system is an item-based top-N collaborative filtering recommender system, generating recommendations based on binary, positive-only feedback [2]. In this way, we cover a large number of applications since item-based top-N collaborative filtering recommender systems are very popular. Multiple authors attribute this popularity to the combination of favorable properties such as simplicity, stability, efficiency, reasonable accuracy, the ability for intuitively explaining their recommendations, and the ability for immediately taking into account newly entered feedback [3, 8, 6, 9].

Central to our approach, we show a property of item-based top-N collaborative filtering recommender systems that allows us to compute a recommendation score in $\mathcal{O}(n \log n)$ instead of exponential time.

The main contributions of this work are:

- We formally introduce the challenge of *top-N recommendation for shared accounts in the absence of contextual information* (Sec. 2).
- We propose a solution to this challenge for all cases in which the reference recommender system is an item-based top-N collaborative filtering recommender system, generating recommendations based on binary, positive-only feedback [2] (Sec. 5-8).
- Most importantly, we show an essential property of item-based top-N collaborative filtering recommender systems that allows us to keep the time complexity of our proposed solution within practically feasible limits (Sec. 6).
- We experimentally show on multiple datasets that our proposed solution is able to detect preferences of individual users in shared accounts and has therefore significant advantages for tackling the dominance, generality and presentation problems (Sec. 9).

After formalizing the definitions of the challenge (Sec. 2) and the reference recommender system (Sec. 3) we first give further insight in how the reference recommender system suffers from the shared account problems (Sec. 4). Afterwards we sequentially solve the generality problem (Sec. 5), the dominance problem (Sec. 7) and the presentation problem (Sec. 8). Furthermore, we inserted a section on the efficient computation of our solution to the generality problem (Sec. 6). Finally, we discuss the experimental evaluation of our proposed solution (Sec. 9).

2. PROBLEM DEFINITION

Let $\mathcal{U}$ be the set of users, $\mathcal{A}$ the set of accounts and $\mathcal{I}$ the set of items. Furthermore, let $U(a) \subseteq \mathcal{U}$ be the set of users that share account a, i.e. the userset of account a, and let $a(u) \in \mathcal{A}$ be the account that user u belongs to. Notice that in this problem setting every user belongs to exactly one account.

First, consider the user-rating-matrix $\mathbf{T} \in \{0,1\}^{|\mathcal{U}| \times |\mathcal{I}|}$. $\mathbf{T}_{ui} = 1$ indicates that there exists a preference of user $u \in \mathcal{U}$ for item $i \in \mathcal{I}$. $\mathbf{T}_{ui} = 0$ indicates that there is no such preference.

We are given a reference recommender system R_{ref} that produces the desired recommendations given $\mathbf{T}$. Consequently, we say that an item i is **relevant** to a user u if i is in the top-N recommendations for u as computed by the reference recommender system $R_{ref}(\mathbf{T})$ on the user-rating-matrix $\mathbf{T}$.

Unfortunately, in our problem setting $\mathbf{T}$ is unknown. Instead we are given the account-rating-matrix $\mathbf{R} \in \{0,1\}^{|\mathcal{A}| \times |\mathcal{I}|}$. $\mathbf{R}_{ai} = 1$ indicates that there is a known preference of account $a \in \mathcal{A}$ for item $i \in \mathcal{I}$. $\mathbf{R}_{ai} = 0$ indicates that there is no such information.

Now, the challenge of *top-N recommendation for shared accounts in the absence of contextual information* is to devise a shared account recommender system $R_{sa}(\mathbf{R})$ that, based on the account-rating-matrix $\mathbf{R}$, computes for every account a the top N_a recommendations such that:

- Ideally, this top-N_a contains all top-N items for every user in the userset of a, with $N = \frac{N_a}{|U(a)|}$. Practically, the goal is to avoid the dominance and the generality problem by maximizing the number of users with at least one item from its top-N.
- It is clear for a user in the userset of a which items in the top-N_a are meant for her, i.e. the presentation problem gets solved.

Notice that in the above definition, the shared account recommender system does **not** get the number of users sharing every account as an input. Furthermore, no assumption is made about the shared interests of the users sharing an account. They can have totally different interests, partially overlapping interests or fully overlapping interests.

Finally, notice that this problem definition is orthogonal to a typical group recommendation problem [10]. First, in group recommendation, the individual profiles of the users in the shared account are typically known. Here, they are unknown. Second, in group recommendation, it is typically assumed that the recommendations will be consumed by all users in the shared account together. Here, it is assumed that every user in the shared account can identify the recommendations meant for her and consumes these recommendations individually.

3. THE REFERENCE RECOMMENDER SYSTEM

Typically, recommender systems find the top-N recommendations for a user u by first computing the recommendation scores $s(u,i)$ for every candidate recommendation i and afterwards selecting the N recommendations i for which $s(u,i)$ is the highest.

One of the most popular classes of recommender systems for binary, positive-only feedback are the item-based collaborative filtering recommender systems which Deshpande et al. discussed in detail [2]. These item-based recommender systems are rooted in the intuition that good recommendations are similar to the items already preferred by the target user, where the similarity between two items is measured using any set similarity measure between the respective sets of users preferring the items. Thus, for a target user u, this recommender system first finds $KNN(j)$, the k most similar

items to j, for every preferred item j ($\mathbf{T}_{uj} = 1$) by using a similarity measure $sim(j, i)$. Next, every preferred item independently increases the recommendation score for its k most similar items $i \in KNN(j)$ with the similarity value $sim(j, i)$. Thus, the item-based recommendation score of a candidate recommendation i for user u is given by [2]:

$$s_{IB}(u, i) = s_{IB}(I(u), i) = \sum_{j \in I(u)} sim(j, i) \cdot |KNN(j) \cap \{i\}|, \quad (1)$$

with $I(u) = \{j \in \mathcal{I} \mid \mathbf{T}_{uj} = 1\}$, the set of items preferred by u.

A typical choice for $sim(j, i)$ is the cosine similarity. Furthermore, Deshpande et al. report that normalizing the similarity scores improves the performance [2]. This comes down to defining $sim(j, i)$ in Equation 1 as:

$$sim(j, i) = \frac{cos(j, i)}{\sum_{l \in KNN(j)} cos(j, l)}.$$

We will use this recommender system as the reference recommender system R_{ref}.

4. SHARED ACCOUNT PROBLEMS OF THE REFERENCE RECOMMENDER SYSTEM

Simply applying the reference recommender system (Sec. 3) to the account-rating-matrix $\mathbf{R}$ leads to inferior results because the reference recommender system suffers from all three shared account problems. We illustrate this with two toy examples. In both examples we consider the two users u_a and u_b that share the account s. User u_a has a known preference for the items a_1 and a_2 and user u_b has a known preference for the items b_1, b_2 and b_3. There are five candidate recommendations: $r_a^1, r_a^2, r_b^1, r_b^2$ and r_g. r_a^1 and r_a^2 are good recommendations for u_a. r_b^1 and r_b^2 are good recommendations for u_b. r_g is an overly general recommendation to which both users feel neutral.

Tables 1 and 2 summarize some intermediate computations on the first and second example respectively. The left hand side of both tables lists for every candidate recommendation (rows) the similarity to the known preferences of the shared account s (columns). The right hand side of both tables lists for every candidate recommendation (rows) three recommendation scores (columns). These scores are computed using Equation 1 and the similarity values on the left hand side of the respective row. The first two scores are for u_a and u_b respectively if they would not share an account. The third score is for s, the account shared by u_a and u_b.

The first example, corresponding to Table 1, illustrates that the item-based reference recommender system can suffer from the generality problem. From Table 1 we learn that if u_a would not share an account with u_b, the item-based reference recommender system would correctly assign the highest scores to r_a^1 and r_a^2 for u_a and to r_b^1 and r_b^2 for u_b. However, if u_a and u_b share the account s, the overly general item r_g receives the highest score. In this case, the item-based reference recommender system suffers from the generality problem because it does not discriminate between a recommendation score that is the sum of a few large contributions and a recommendation score that is the sum of many small contributions.

Table 1: Item similarities and resulting scores for Example 1.

	sim					s_{IB}		
	$(a_1, *)$	$(a_2, *)$	$(b_1, *)$	$(b_2, *)$	$(b_3, *)$	$(u_a, *)$	$(u_b, *)$	$(s, *)$
r_a^1	5	5	1	0	0	10	1	11
r_a^2	4	4	1	0	0	8	1	9
r_b^1	1	0	5	5	2	1	12	13
r_b^2	1	0	4	4	2	1	10	11
r_g	3	3	3	3	3	6	9	15

Table 2: Item similarities and resulting scores for Example 2.

	sim					s_{IB}		
	$(a_1, *)$	$(a_2, *)$	$(b_1, *)$	$(b_2, *)$	$(b_3, *)$	$(u_a, *)$	$(u_b, *)$	$(s, *)$
r_a^1	5	5	0	0	1	10	1	11
r_a^2	4	4	1	0	0	8	1	9
r_b^1	0	1	5	5	5	1	15	16
r_b^2	1	0	4	4	4	1	12	13
r_g	2	1	2	1	2	3	5	8

The second example, corresponding to Table 2, illustrates that the item-based reference recommender system can suffer from the dominance problem. From Table 2 we learn that if u_a would not share an account with u_b, the item-based reference recommender system would correctly assign the highest scores to r_a^1 and r_a^2 for u_a and to r_b^1 and r_b^2 for u_b. However, if u_a and u_b share the account s, all recommendations for u_b receive a higher score than any recommendation for u_a. Hence, the recommendations for u_b dominate the account at the expense of u_a. In this case, the item-based reference recommender system suffers from the dominance problem because it does not take into account that u_b has more known preferences than u_a (3 vs. 2).

Both examples are suitable to illustrate that the reference recommender system suffers from the presentation problem. As an example, consider the first row of Table 1. The recommendation score $s(s, r_a^1) = 11$ is the sum of $sim(a_1, r_a^1) = 5$, $sim(a_2, r_a^1) = 5$ and $sim(b_1, r_a^1) = 1$. Therefore, it can be explained by a_1, a_2 and b_1. This is however a bad explanation because due to the presence of b_1, u_a will have difficulties to identify with the explanations and u_b might wrongly conclude that the recommendation is meant for her.

In our experimental evaluation (Sec. 9), we show that similar problems also arise for multiple large, real-life datasets.

5. SOLVING THE GENERALITY PROBLEM

The previous section showed that the generality problem arises because the item-based reference recommender system (Eq. 1) does not discriminate between a score that is the sum of a few large similarities and a score that is the sum of many small similarities. Therefore, our first step is to adapt the item-based recommendation score (Eq. 1) into the *length-adjusted* item-based recommendation score:

$$s_{LIB}(u, i) = s_{LIB}(I(u), i) = \frac{1}{|I(u)|^p} \cdot s_{IB}(I(u), i), \quad (2)$$

with the hyperparameter $p \in [0, 1]$. Although this adjustment does not immediately solve the generality problem, it does provide a way to differentiate between the sum of a few large similarities and the sum of many small similarities. By choosing $p > 0$, we create a bias in favor of the sum of a few large similarities. The larger p, the larger the bias.

Since the factor $\frac{1}{|I(u)|^p}$ is the same for all candidate recommendations i, the top N items for user u according to s_{LIB} and s_{IB} are the same. However, when we compare the scores of two different users, s_{LIB} also takes into account the the total amount of items preferred by the user.

To avoid the generality problem we ideally want to recommend an item i if it is highly relevant to one of the users in the userset of the shared account a. Hence, we want to compute the recommendation score of an item i for every individual user $u \in U(a)$, and use the highest one. Formally, we want to rank all items i according to their ideal recommendation score

$$\max_{u \in U(a)} s_{LIB}(I(u), i).$$

Unfortunately, we cannot compute this ideal recommendation score because $U(a)$ and consequently $I(u)$ are unknown. Instead, we only know $I(a) = \{j \in \mathcal{I} \mid \mathbf{R}_{aj} = 1\}$, the set of items preferred by account a.

We can, however, approximate the ideal recommendation score with its upper bound:

$$\max_{S \in 2^{I(a)}} s_{LIB}(S, i) \geq \max_{u \in U(a)} s_{LIB}(I(u), i),$$

in which $2^{I(a)}$ is the powerset of $I(a)$, i.e. the set containing all possible subsets of $I(a)$. The proposed approximation is an upper bound of the ideal score because every set of items $I(u)$ for which $u \in U(a)$ is also an element of $2^{I(a)}$. This approximation is based on the assumption that of all possible subsets of $I(a)$, the ones that correspond to users are more likely to result into the highest recommendation scores than the ones put together at random.

Consequently, we propose to solve the generality problem with the *disambiguating* item-based (DAMIB) recommender system, according to which the DAMIB recommendation score of an account a for an item i is given by:

$$s_{DAMIB}(a, i) = \max_{S \in 2^{I(a)}} s_{LIB}(S, i). \tag{3}$$

Every score $s_{DAMIB}(a, i)$ corresponds to an optimal subset $S_i^* \subseteq I(a)$:

$$S_i^* = \operatorname*{argmax}_{S \in 2^{I(a)}} s_{LIB}(S, i). \tag{4}$$

Hence, $s_{DAMIB}(a, i) = s_{LIB}(S_i^*, i)$. As such, the DAMIB recommender system not only computes the recommendation scores, but also finds the subset S_i^* that maximizes the length-adjusted item-based recommendation score of a for i. This subset serves as the sharply defined, intuitive explanation for recommending i to a.

In other words, the DAMIB-recommender system implicitly splits the shared account a into (possibly overlapping) subsets S_i^* based on the intuitive and task-specific criterium that every S_i^* maximizes s_{LIB} for one of the candidate recommendations i. When $s_{LIB}(S_i^*, i)$ is high, we expect that S_i^* corresponds well to an individual user. When $s_{LIB}(S_i^*, i)$ is low, there is no user in the shared account for whom i is a strong recommendation and we expect S_i^* to be a random subset. As such, we avoid the error prone task of estimating the number of users in the shared account and explicitly splitting the account a into its alleged users, based on a general clustering criterium [15].

Furthermore, since subsets can potentially overlap, the DAMIB recommender system does not care whether the known preferences of the users in a shared account are strongly, slightly or not at all overlapping.

Finally, notice that for $p = 0$ it always holds that $s_{DAMIB} = s_{LIB} = s_{IB}$. Hence, the item based recommender system is a special case of the DAMIB recommender system.

6. EFFICIENT COMPUTATION

Finding the maximum in Equation 3 in a direct way requires to compute s_{LIB} an exponential number of times, namely $2^{|I(a)|}$. Consequently, computing s_{DAMIB} in a direct way is intractable.

Fortunately, we are able to show a property of s_{LIB} that allows us to compute s_{DAMIB} in $\mathcal{O}(n \log n)$ time, with $n = |I(a)|$. This property is given by Theorem 6.1.

THEOREM 6.1. *Let a be an account that prefers the set of items $I(a)$. Furthermore, let i be a candidate recommendation. If we rank all items $j, l \in I(a)$ such that $rank(j) < rank(l) \iff sim(j, i) > sim(l, i)$, then the subset $S_i^* \subseteq I(a)$ that maximizes $s_{LIB}(S, i)$ over all $S \in 2^{I(a)}$ is a prefix of that ranking.*

PROOF. Given any $S \subseteq I(a)$. Initialize $P = S$. While P is not a prefix, remove r, the worst ranked item from P, and add a, the best ranked item that is not in P to P. As long as P is not yet a prefix, it holds that $sim(a, i) \geq sim(r, i)$. Therefore, every such item replacement increases (or keeps equal at least) $s_{LIB}(P, i)$ since the factor $1/|I(a)|^p$ does not change and a smaller term in the sum $\sum_{j \in I(a)} sim(j, i) \cdot |KNN(j) \cap \{i\}|$ is replaced by a larger term. Hence, for every $S \subseteq I(a)$ that is not a prefix of the ranking, we can always find a prefix $P \subseteq I(a)$ for which $s_{LIB}(P, i) \geq s_{LIB}(S, i)$. Therefore, the subset S_i^* that maximizes $s_{LIB}(S, i)$ over all $S \in 2^{I(a)}$ must always be a prefix of the ranking. □

Since the optimal subset is a prefix, we can find it with one scan over the ranked items of $I(a)$ in linear time. The logarithmic factor in the time complexity comes from ranking the $|I(a)|$ items.

This theorem is central to our approach because it allows us to compute s_{DAMIB} in $\mathcal{O}(n \log n)$ instead of exponential time.

7. SOLVING THE DOMINANCE PROBLEM

The DAMIB recommender system allows us to detect when the dominance problem arises. This is because every recommendation i provided by DAMIB comes with a clear explanation in the form of the optimal subset $S_i^* \subseteq I(a)$. Therefore, if the union $\bigcup_{i \in top\text{-}N_a} S_i^*$ is only a small subset of $I(a)$, we know for sure that this small subset dominates the generation of the top N_a recommendations for account a.

Solving the dominance problem is done by choosing $ALG =$ DAMIB in Algorithm 1, called COVER. As such, our final algorithm for recommending to shared accounts is DAMIB-COVER, with DAMIB-COVER$(a) =$ COVER$(a,$ DAMIB$)$.

The DAMIB-COVER algorithm uses the DAMIB scores to find the N_a highest scoring candidate recommendations

Algorithm 1: COVER(a,ALG)

input : $a \in \mathcal{A}$, ALG
output: top-N_a recommendations for account a

Compute $s_{ALG}(a,i)$ for all $i \in \mathcal{I} \setminus I(a)$
Rank all $i \in \mathcal{I} \setminus I(a)$ according to $s_{ALG}(a,i)$ in descending order with $t_a[r]$ the item at position r in the tuple of ranked items t_a
$C(a) \leftarrow \{\}$
$r \leftarrow 1$
$top\text{-}N_a \leftarrow \{\}$
while $|top\text{-}N_a| < N_a$ **do**
  $c \leftarrow t_a[r]$
  compute S_c^*
  if $D(S_c^*, C(a)) \geq \theta_D$ **then**
    $top\text{-}N_a \leftarrow top\text{-}N_a \cup \{c\}$
    $C(a) \leftarrow C(a) \cup S_c^*$
    remove c from t_a
    if $C(a) = I(a)$ **then**
      $C(a) \leftarrow \{\}$
      $r \leftarrow 1$
  else
    $r \leftarrow r + 1$
    if $r > |t_a|$ **then**
      $C(a) \leftarrow \{\}$
      $r \leftarrow 1$

and removes a candidate recommendation c from the top N_a if its explanation S_c^* is not sufficiently different from the explanations of the higher ranked candidates. The explanation-difference condition $D(S_c^*, C(a)) \geq \theta_D$ measures whether the explanation of a candidate (S_c^*) and the union of the explanations of the higher ranked candidates ($C(a)$) are sufficiently different.

Possible heuristic definitions of the explanation-difference condition are $|S_c^* \setminus C(a)| \geq 0.5 \cdot |S_c^*|$, and $|S_c^* \setminus C(a)| = |S_c^*|$. However, our experiments showed that $|S_c^* \setminus C(a)| \geq 1$ works better than the other two. We therefore use the latter heuristic in the remainder of this work.

8. SOLVING THE PRESENTATION PROBLEM

Generating the $top\text{-}N_a$ recommendations for a shared account a with DAMIB-COVER is insufficient because the users that share the account don't know which recommendation belongs to which user. This is the presentation problem.

Our solution to the presentation problem is to present every recommendation $i \in top\text{-}N_a$ together with its explanation S_i^* as defined by Equation 4. We expect that for a large majority of the items i in the $top\text{-}N_a$, the explanation S_i^* is a subset of the preferences $I(u)$ of u, one of the user that shares the account a. We empirically validate this hypothesis in the experimental section (Sec. 9).

Hence, we can present the recommendations as *the item r is recommended to the person that prefers the items s_1, s_2 and s_3*. Then, a user will recognize s_1, s_2 and s_3 as her preferences, and know that r is recommended to her.

9. EXPERIMENTAL EVALUATION

All datasets used are publicly available, readily or upon request to the owner. Furthermore, both the source code of our algorithms and links to the datasets are available on `https://bitbucket.org/BlindReview/rsa`. Besides, this website contains scripts to automatically run every experiment in this section after compiling our source code and retrieving the datasets. As such, all our results can be reproduced with minimal effort, and the way in which we obtained the results can be thoroughly inspected by inspecting the scripts.

9.1 Datasets

Ideally, we would use a dataset that contains real life shared account information. The CAMRa 2011 dataset, for example, contains household membership information for a subset of the users that rated movies [15]. As such we could construct realistic shared accounts with this dataset. Unfortunately, the owner did not wish to distribute the dataset anymore and we have no knowledge of other datasets that contain shared account information. However, from the CAMRa 2011 dataset we learn that most household accounts consist of two users (272 out of 290 households) and some consist of three (14 out of 290) or four users (4 out of 290). Therefore, we will follow the approach of Zhang et al. and create 'synthetic' shared accounts by randomly grouping users in groups of two, three or four [15]. Although this approach is not perfect, Zhang et al. showed that the properties of the 'synthetic' shared accounts were similar to the properties of the real shared accounts from the CAMRa 2011 dataset [15].

We evaluated our proposed solution on four datasets: the *Yahoo!Music* [13], *Movielens1M* [4], *Book-Crossing* [16] and the *Wiki10+* [17] datasets.

The *Yahoo!Music* dataset contains ratings of 14382 users on 1000 songs on a 1 to 5 scale [13]. Since we consider the problem setting with binary, positive-only data we binarize the ratings. We convert the ratings 4 and 5 to preferences and ignore all other ratings. On average, a user has 8.7 preferences.

The *Movielens1M* dataset contains ratings of 6038 users on 3533 movies on a 1 to 5 scale [4]. Again, we convert the ratings 4 and 5 to preferences and ignore all other ratings. On average, a user has 95.3 preferences.

The *Book-Crossing* dataset contains two sorts of information [16]. First, there are ratings of users for books on a 1 to 10 scale. Analogously to the previous two datasets, we convert the ratings 8,9 and 10 to preferences and ignore all other ratings. Secondly, there are also binary preferences that we simply add to our list of preferences. In total, there are 87 835 users, 300695 books and every user has on average 11 preferences.

The *Wiki10+* dataset contains 99162 tags assigned to 20 751 Wikipedia articles [17]. In this case we consider the recommendation of tags to articles, hence the articles take the role of 'users' and the tags take the role of 'items'. If an article a was tagged at least once with a tag t, we consider a 'preference' of article a for tag t. In this context, a shared account is a big article on a wider topic containing multiple smaller 'articles' on subtopics. On average, every article has 22.1 tags.

Due to space restrictions we only show numerical results for the *Yahoo!Music* dataset. However, the results for the three other datasets can be consulted on `https://bitbucket.org/BlindReview/rsa`, and lead to the same conclusions.

9.2 Competitive Algorithms

We compare our novel algorithm, DAMIB-COVER, with two competitive algorithms. The first one is IB, simply the item-based reference recommender system applied to the account-rating-matrix, essentially ignoring the existence of the shared account problems. This is our baseline. The second competitive algorithm is IB-COVER, which is defined as IB-COVER(a) = COVER$(a,$ IB$)$. IB-COVER is similar to one of the algorithms already proposed by Yu et al. in a different context [14].

9.3 Performance

First, consider the recall of a user that shares an account a with $|U(a)|$ other user. This is the percentage of its individual top-5 recommendations that is also present in the top-N_a recommendations for its shared account, with $N_a = 5 \cdot |U(a)|$. Formally, we define the recall of user u as:

$$rec(u) = \frac{|top\text{-}5(u) \cap top\text{-}N_a(a)|}{5}.$$

Ideally, the recall of all users in a shared account is 1, meaning that the top-N_a for the shared account is the union of the individual top-5's of the $|U(a)|$ users sharing the account.

Now, to investigate how many users genuinely suffer from sharing an account, we measure the fraction of users that does not get any relevant recommendation, i.e. that does not find a single one of its top-5 individual recommendations in the top-N_a recommendations of the shared account it belongs to. We denote this number as $rec_0^{\mathcal{U}}$, the fraction of users for which the recall is zero. Formally, we define

$$rec_0^{\mathcal{U}} = \frac{|\{u \in \mathcal{U} \mid rec(u) = 0\}|}{|\mathcal{U}|}.$$

An illustrative example of a user that genuinely suffers from sharing an account is depicted in Table 3. This table shows two real users from the *Movielens1M* dataset with their respective known preferences $I(u)$ and item-based individual top-5 recommendations. Their item-based individual top-5 recommendations look reasonable given their known preferences and it is not unrealistic that these two users would be part of the same household and therefore share an account. Consequently, Table 3 also shows the recommendations for the 'synthetic' account shared by both users for two cases: R_{sa} = IB and R_{sa} = DAMIB-COVER. In case R_{sa} = IB, $rec(562) = 0$, i.e. user 562 does not get a single recommendation and genuinely suffers from sharing an account. In case R_{sa} = DAMIB-COVER, $rec(562) = 0.6$, i.e. user 562 gets 3 good recommendation and there is no serious problem. Obviously, this is just one example and we need to look at all users in the dataset for comparing the different algorithms.

Figure 1 displays $rec_0^{\mathcal{U}}$ for the *Yahoo!Music* dataset. The number of nearest neighbors, k, is a parameter of the item-based reference recommender system (Eq 1). There are multiple ways of choosing k. Amongst others, examples are accuracy in an off-line experiment, subjective quality judgment of the recommendations, accuracy in an on-line A/B test, computational efficiency, etc. Therefore, we present our results for a variation of reference recommender systems, i.e. item-based collaborative filtering recommender systems that differ in their choice of k. Consequently, every plot in Figure 1 shows the results for a different k.

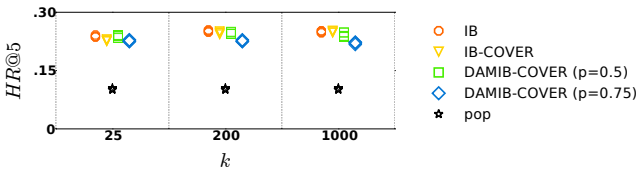

Figure 2: HR@5 as a function of k for different recommender systems. Higher is better.

For every choice of k and the individual top-5 recommendations corresponding to this choice we consider four experiments: an account shared by one, two, three or four users respectively. Notice that an account shared by one user is actually not shared. Every horizontal axis indicates the number of users that share the account, every vertical axis indicates the resulting $rec_0^{\mathcal{U}}$. The four different markers show the results for four different shared account recommender systems R_{sa}: the baseline algorithm IB, the competitor IB-COVER and two variations of the proposed DAMIB-COVER algorithm. These two variations differ in their choice of the parameter p (Eq. 2): $p = 0.5$ and $p = 0.75$. Since we repeat every experiment 5 times with other randomizations, every plot contains $5 \times 4 = 20$ markers of the same kind. However, because of the low spread, most markers are plotted on top of each other, forming dense marker clouds. Furthermore, since the 95% confidence intervals for the mean are more narrow than the marker-clouds of 5 datapoints, we do not draw them. Consequently, two marker clouds that are visually well separated, are also significantly different at the 5% significance level.

We make four observations from Figure 1. First, we observe that the baseline performance is not good. Up to 19% of the users get no relevant recommendation when they share their account with another user. This confirms that shared accounts can cause significant problems for recommender systems.

Secondly, our proposed solution, the DAMIB-COVER algorithm, can significantly improve $rec_0^{\mathcal{U}}$. In some cases the improvement is even drastic. One example is for the case that $|U(a)| = 2$ and that the individual top-5 is generated with $k = 200$. In this case, 12% of the users does not get any relevant recommendation when using the baseline algorithm IB. By using DAMIB-COVER ($p = 0.75$), this number is reduced with a factor four ($rec_0^{\mathcal{U}} = 0.03$).

Thirdly, sometimes IB-COVER already improves over IB. There are however multiple cases in which DAMIB-COVER further improves over IB-COVER. Furthermore, the advantages of DAMIB-COVER over IB-COVER will become even more evident in the evaluation of the presentation problem in Section 9.5.

Finally, when $|U(a)| = 1$, i.e. when the accounts are not shared, $rec_0^{\mathcal{U}} = 0$ by definition for the baseline algorithm IB. However, we observe that also for the IB-COVER and the variants of the DAMIB algorithms $rec_0^{\mathcal{U}}$ can be kept sufficiently low. Hence, the proposed DAMIB algorithm does not fail when accounts are not shared.

9.4 Limited Trade-Off

To emphasize the point that the DAMIB-COVER algorithm still performs well in a traditional setting when no accounts are shared, we also discuss the results of DAMIB-COVER on a more established experimental setup that was

Table 3: Example of user 562 suffering from sharing an account with user 4385.

user ID	562	4385
$I(u)$	Wes Craven's New Nightmare, The Exorcist III, Serial Mom, Scream, Scream 2, The Blair Witch Project, Good Will Hunting, Misery, Interview with the Vampire, Candyman, Freddy's Dead: The Final Nightmare	American Beauty, The Shawshank Redemption, Being John Malkovich, L.A. Confidential, Boys Don't Cry, Croupier, Dogma, Cider House Rules, Girl Interrupted, Saving Grace, The Talented Mr. Ripley
individual top-5: IB, $k = 25$	A Nightmare on Elm Street, Halloween, Halloween:H20, The Shining, Seven	Pulp Fiction, Fargo, The Sixth Sense, The Silence of the Lambs, Shindler's List
R_{sa} = IB	The Silence of the Lambs, Fargo, Pulp Fiction, The Sixth Sense, Saving Private Ryan, The Usual Suspects, Shindler's List, Shakespeare in Love, Star Wars: Episode V, The Matrix	
R_{sa} = DAMIB-COVER (p=0.75)	The Silence of the Lambs, Fargo, Schindler's List, A Nightmare on Elm Street, Halloween:H20, Pulp Fiction, Shakespeare in Love, The Shining, The Exorcist, Sleepy Hollow	

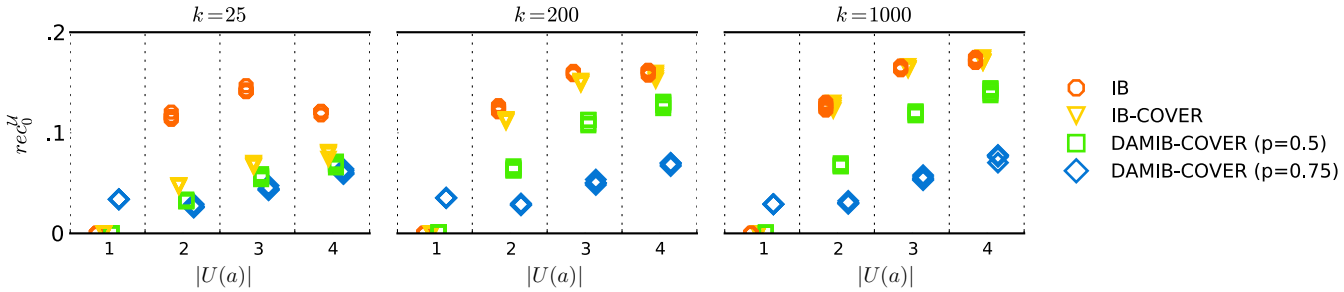

Figure 1: $rec_0^{\mathcal{U}}$ as a function of the number of merged users, $|U(a)|$, for different k and shared account recommender systems R_{sa}. Lower is better. 95% confidence intervals are more narrow than the marker clouds and are therefore not drawn.

used by Deshpande et al. [2], amongst many others. To avoid all confusion: this experimental setup has nothing to do with shared accounts. In this experimental setup, one preference of every user is randomly chosen to be the test preference h_u for that user. If a user has only one preference, no test preference is chosen. The remaining preferences are represented as a 1 in the training matrix $\mathbf{R}$ (which is in this case exactly the same as $\mathbf{T}$ because no accounts are shared). All other entries of $\mathbf{R}$ are zero. We define $\mathcal{U}_t$ as the set of users with a test preference. For every user $u \in \mathcal{U}_t$, every algorithm ranks the items $\{i \in \mathcal{I} \mid \mathbf{R}_{ui} = 0\}$ based on $\mathbf{R}$. Following Deshpande et al. we evaluate every ranking using hit rate at 5 [2]. Hit rate at 5 is given by

$$HR@5 = \frac{1}{|\mathcal{U}_t|} \sum_{u \in \mathcal{U}_t} |\{h_u\} \cap top5(u)|,$$

with $top5(u)$ the 5 highest ranked items for user u. Hence $HR@5$ gives the percentage of test users for which the test peference is in the top 5 recommendations. The results of the experiment for the *Yahoo!Music* dataset are shown in Figure 2. Additionally to the algorithms discussed earlier, Figure 2 also contains the results for the baseline-algorithm POP, the non-personalized algorithm that ranks all items according to their popularity, i.e. the number of users in the training set that prefer the item. Also in this case we repeated every experiment five times with a different randomization. Again, the five data points are often plotted on top of each other because of the low spread. Figure 2 shows that $HR@5$ is very similar for DAMIB-COVER and IB. Hence, there is almost no trade-off in terms of global accuracy measured as $HR@5$.

9.5 Presentation

In Section 8 we proposed to solve the presentation problem by presenting every recommendation together with its explanation. If then, a user in the shared account recognizes an explanation as a subset of her preferences, this user can identify with the recommendation and therefore knows the recommendation is meant for her. For this proposed solution to work, it is crucial that the recommendation is identifiable, i.e. that its explanation is a subset of the preferences of one of the users in the shared account. We quantify the identifiability of a recommendation i, with explanation S_i^*, for a shared account a as:

$$ident(S_i^*) = \max_{u \in U(a)} \frac{|S_i^* \cap I(u)|}{|S_i^*|}.$$

Ideally, $ident(S_i^*) = 1$, i.e. every item in the explanation is a preference of one and the same user. In the worst case, $ident(S_i^*) = 1/|U(a)|$, i.e. the explanation contains an equal amount of preferences from all users in the shared account and therefore none of the users can identify herself with the recommendation.

Figure 3 shows histograms of the identifiability of the top-10 recommendations for $|U(a)| = 2$ on the *Yahoo!Music* dataset for multiple shared account recommender systems R_{sa}. From Figure 3a we learn that if one simply applies the item-based reference algorithm to the shared account data of the *Yahoo!Music* dataset, the presentation problem arises: very few recommendations can be identified with one of the users in the shared account, i.e. $ident(S_i^*) = 1$ for only 10% of the explanations. Figure 3b shows that using IB-COVER instead of IB does not improve the situation. However, figure 3c shows that using DAMIB-COVER dras-

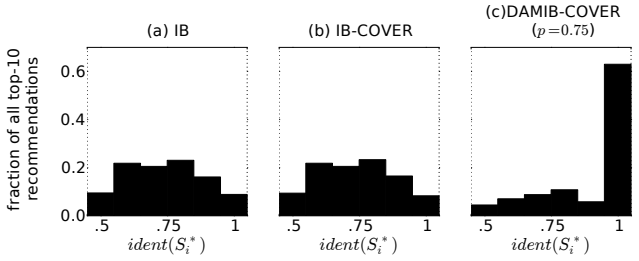

Figure 3: Histograms of identifiability of top-10 recommendations for $|U(a)| = 2$ on the *Yahoo!Music* dataset. $R_{ref} = \mathbf{IB}, k = 200$

tically increases the identifiability of the recommendations, i.e. $ident(S_i^*) = 1$ for approximately 60% of the explanations. Hence, the DAMIB explanations are superior to the item-based explanations.

10. RELATED WORK

Although we did not find prior art tackling the same challenges as we do, there are some works that have commonalities with ours.

First, Palmisano et al. [11] consider a problem setting in which the contextual information is *sometimes* missing. However, their proposed solution draws upon all training data for which they do know the context to devise a 'context predictor'. Hence, their solution relies on contextual information.

Second, Anand et al. [1] do not use explicit contextual information. However, their solution assumes that the preferences of every account are grouped into transactions. Our solution does not assume that this kind of extra data is available.

Third, Zhang et al. [15] study the extent to which it is possible to explicitly split shared accounts into their individual users without contextual information. They are able to split certain shared accounts very nicely, but find that in general, explicitly splitting accounts into their users is very error prone. Fortunately, by means of s_{DAMIB}, we are able to avoid this explicit split of accounts into users and perform a softer, implicit split instead.

Fourth, Yu et al. [14] propose to use the explanations of item-based recommendations to generate diversified top-N recommendation lists. Where they focus on the diversity of the explanations, we focus on covering all items preferred by the account with the different explanations. Furthermore, in our experimental evaluation (Sec. 9) we showed that the explanations provided by our DAMIB-COVER algorithm are superior to those that can be extracted from an item-based algorithm.

Finally, the NLMF algorithm might be an alternative to solve the generality problem [7]. However, in that case it is not clear how to solve the dominance an presentation problems.

11. CONCLUSIONS AND FUTURE WORK

We showed that the widely used item-based recommender systems fails when it makes recommendations for shared accounts. Therefore, we introduced the challenge of Top-N recommendation for shared accounts in the absence of contextual information. Furthermore, we proposed the DAMIB-COVER algorithm, our solution to this challenge. Central to our approach, we showed a theorem that allowed us to compute a recommendation score in $\mathcal{O}(n \log n)$ instead of exponential time. Finally, we experimentally validated that our proposed solution has important advantages.

As future work, we plan to generalize our proposed solution to a wider range of reference recommender systems.

12. REFERENCES

[1] S. Anand and B. Mobasher. Contextual recommendation. In *WebMine*, pages 142–160, 2006.

[2] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *TOIS*, 22(1):143–177, 2004.

[3] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. In F. Ricci, L. Rokach, B. Shapira, and P. Kantor, editors, *Recommender Systems Handbook*. Springer, Boston, MA, 2011.

[4] Grouplens. ml-1m.zip. http://grouplens.org/datasets/movielens/.

[5] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[6] D. Jannach, M. Zanker, A. Felfernig, and G. Frierich. *Recommender Systems: An Introduction*. Cambridge University Press, New York, NY, 2011.

[7] S. Kabbur and G. Karypis. Nlmf: Nonlinear matrix factorization methods for top-n recommender systems. In *ICDMW*, pages 167–174. IEEE, 2014.

[8] Y. Koren and R. Bell. Advances in collaborative filtering. In F. Ricci, L. Rokach, B. Shapira, and P. Kantor, editors, *Recommender Systems Handbook*. Springer, Boston, MA, 2011.

[9] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Comp.*, 7(1):76–80, 2003.

[10] J. Masthoff. Group recommender systems: Combining individual models. In F. Ricci, L. Rokach, B. Shapira, and P. Kantor, editors, *Recommender Systems Handbook*. Springer, Boston, MA, 2011.

[11] C. Palmisano, A. Tuzhilin, and M. Gorgolione. Using context to improve predictive modeling of customers in personalization applications. *TKDE*, 20(11):1535–1549, 2008.

[12] R. Pan, Y. Zhou, B. Cao, N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *ICDM*, pages 502–511, 2008.

[13] Yahoo!Research. Yahoo_webscope_r3.tgz. http://research.yahoo.com/Academic_Relations.

[14] C. Yu, L. Lakshmanan, and S. Amer-Yahia. Recommendation diversification using explanations. In *ICDE*, pages 1299–1302, 2009.

[15] A. Zhang, N. Fawaz, S. Ioannidis, and A. Montanari. Guess who rated this movie: Identifying users through subspace clustering. In *UAI*, pages 944–953, 2012.

[16] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW*, pages 22–32, 2005.

[17] A. Zubiaga. Enhancing navigation on wikipedia with social tags. arXiv:1202.5469, 2012.