Unifying Nearest Neighbors Collaborative Filtering

Koen Verstrepen Bart Goethals

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

ABSTRACT

We study collaborative filtering for applications in which there exists for every user a set of items about which the user has given binary, positive-only feedback (one-class collaborative filtering). Take for example an on-line store that knows all past purchases of every customer. An important class of algorithms for one-class collaborative filtering are the nearest neighbors algorithms, typically divided into user-based and item-based algorithms. We introduce a reformulation that unifies user- and item-based nearest neighbors algorithms and use this reformulation to propose a novel algorithm that incorporates the best of both worlds and outperforms state-of-the-art algorithms. Additionally, we propose a method for naturally explaining the recommendations made by our algorithm and show that this method is also applicable to existing user-based nearest neighbors methods.

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

Keywords: One-class collaborative filtering, nearest neighbors, explaining recommendations, top-N recommendation, recommender systems.

1. INTRODUCTION

Typically, the training data for collaborative filtering is represented by a matrix in which the rows represent the users and the columns represent the items. A value in this matrix can be unknown or can reflect the preference of the respective user for the respective item. Here, we consider the specific setting of binary, positive-only preference feedback. Hence, every value in the preference matrix is 1 or 0, with 1 representing a known preference and 0 representing the unknown. Pan et al. [10] call this setting one-class collaborative filtering (OCCF). Applications that correspond to this version of the collaborative filtering problem are likes on social networking sites, tags for photo's, websites visited during a surfing session, articles bought by a customer etc.

RecSys'14, October 6–10, 2014, Foster City, Silicon Valley, CA, USA.

http://dx.doi.org/10.1145/2645710.2645731.

In this setting, identifying the best recommendations can be formalized as ranking all user-item-pairs (u, i) according to the probability that u prefers i.

One class of algorithms for solving OCCF are the nearest neighbors algorithms. Sarwar et al. [13] proposed the well known user-based algorithm that uses cosine similarity and Deshpande et al. [2] proposed several widely used item-based algorithms.

Another class of algorithms for solving OCCF are matrix factorization algorithms. The state-of-the-art algorithms in this class are Weighted Rating Matrix Factorization [7, 10], Bayesian Personalized Ranking Matrix Factorization [12] and Collaborative Less is More Filtering [14].

This work builds upon the different item-based algorithms by Deshpande et al. [2] and the user-based algorithm by Sarwar et al. [13] that uses cosine similarity. We introduce a reformulation of these algorithms that unifies their existing formulations. From this reformulation, it becomes clear that the existing user- and item-based algorithms unnecessarily discard important parts of the available information. Therefore, we propose a novel algorithm that combines both user- and item-based information. Hence, this algorithm is neither user-based nor item-based, but *nearest-neighbors-based.* Our experiments show that our algorithm not only outperforms the individual nearest neighbors algorithms but also state-of-the-art matrix factorization algorithms.

Furthermore, it is well accepted that every recommendation should come with a short explanation to why it is recommended [7, 6]. Good explanations help users to put the recommendations in the right perspective [17]. Typically, item-based nearest neighbors algorithms are considered to be superior for this task [3, 7].

Thanks to our reformulation however, we are able to challenge this belief and show that also other nearest neighbors algorithms have a natural explanation.

The main contributions of this work are:

- We propose a reformulation that unifies user-and item-based nearest neighbors algorithms for OCCF (Sec. 3) [13, 2].
- We propose KUNN, a novel algorithm for OCCF that is grounded in our unifying reformulation (Sec. 4).
- We extensively evaluate our algorithm on real life data and show that it outperforms state-of-the-art algorithms (Sec. 5).
- We propose a method that naturally explains the recommendations by our novel algorithm and user-based algorithms (Sec. 6)

2. PRELIMINARIES

Let $\mathcal{U}$ be a set of users and $\mathcal{I}$ a set of items. We are given a matrix with training data $\mathbf{R} \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$. $\mathbf{R}_{ui} = 1$ indicates that there is a known preference of user $u \in \mathcal{U}$ for item $i \in \mathcal{I}$. $\mathbf{R}_{ui} = 0$ indicates that there is no such information.

Furthermore, $c(x)$ gives the count of x, meaning

$$c(x) = \begin{cases} \sum_{i\in\mathcal{I}} R_{xi} & \text{if } x \in \mathcal{U} \\ \sum_{u\in\mathcal{U}} R_{ux} & \text{if } x \in \mathcal{I}. \end{cases}$$

The goal of OCCF is to rank all user-item-pairs (u,i) for which $\mathbf{R}_{ui} = 0$ according to the likelihood of u preferring i. The most well known algorithms perform this task by computing for every user-item-pair (u,i) a score $s(u,i)$, by which the user-item-pairs are then ranked.

A key element of nearest neighbors algorithms is their definition of the neighborhood of an object, or more specifically, the similarity measure $sim()$ used for computing it. A typical choice for $sim()$ is the cosine similarity. The cosine similarity between two users u and v is given by

$$cos(u,v) = \frac{\sum_{j\in\mathcal{I}} \mathbf{R}_{uj}\mathbf{R}_{vj}}{\sqrt{c(u)c(v)}}. \tag{1}$$

Analogously, the cosine similarity between two items i and j is given by

$$cos(i,j) = \frac{\sum_{v\in\mathcal{U}} \mathbf{R}_{vi}\mathbf{R}_{vj}}{\sqrt{c(i)c(j)}}. \tag{2}$$

We denote the $k_\mathcal{U}$ ($k_\mathcal{I}$) nearest neighbors of a user u (an item i) by $KNN(u)$ ($KNN(i)$).

3. UNIFYING NEAREST NEIGHBORS

In this section we propose a novel formulation of nearest neighbors collaborative filtering that unifies the known formulations of the user-based algorithm by Sarwar et al. [13] and the different item-based algorithms by Deshpande et al. [2]. This formulation is given by

$$s(u,i) = \sum_{v\in\mathcal{U}}\sum_{j\in\mathcal{I}} (\mathcal{L}\cdot\mathcal{N}\cdot\mathcal{G}\cdot\mathcal{S})\big((u,i),(v,j)\big). \tag{3}$$

The score $s(u,i)$, that represents the likelihood that a user u prefers an item i, is computed as a weighted sum over all possible user-item pairs (v,j). The weight of every (v,j)-pair with respect to the pair (u,i) is a multiplication of four functions: the local function $\mathcal{L}\big((u,i),(v,j)\big)$, the neighborhood function $\mathcal{N}\big((u,i),(v,j)\big)$, the global function $\mathcal{G}\big((u,i),(v,j)\big)$ and the rescaling function $\mathcal{S}\big((u,i),(v,j)\big)$.

Before giving a general definition of $\mathcal{L},\mathcal{N},\mathcal{G}$ and $\mathcal{S}$, we first discuss the reformulation of the user-based algorithm by Sarwar et al. [13] and the different item-based algorithms by Deshpande et al. [2].

3.1 Item-Based

Deshpande et al. [2] proposed the now widely used class of item-based nearest neighbors algorithms. We discuss the variation that uses cosine similarity without similarity normalization (*cosine, SNorm-*) because of its clean formulation. The analysis for the other variations is analogous.

This algorithm first finds the neighborhood $KNN(j)$ for every preferred item j ($\mathbf{R}_{uj} = 1$) by using the cosine similarity $cos(j,i)$. Next, every preferred item independently increases the score for its $k_\mathcal{I}$ most similar items $i \in KNN(j)$ with the similarity value $cos(j,i)$. Thus, the score of a candidate recommendation i for user u is given by [2]:

$$s(u,i) = \sum_{\substack{j\in\mathcal{I} \\ \mathbf{R}_{uj}=1}} cos(j,i)\cdot|KNN(j)\cap\{i\}|. \tag{4}$$

We reformulate this algorithm by substituting $cos(j,i)$ by its definition (Eq. 2) and regrouping the terms. This gives

$$s(u,i) = \sum_{v\in\mathcal{U}}\sum_{j\in\mathcal{I}} \mathbf{R}_{uj}\mathbf{R}_{vj}\mathbf{R}_{vi}\cdot|KNN(j)\cap\{i\}|\cdot\frac{1}{\sqrt{c(j)c(i)}}.$$

This particular formulation now nicely fits our Equation 3: a weighted sum over all possible user-item pairs (v,j) in which the weight of every (v,j)-pair with respect to the pair (u,i) is determined by the functions $\mathcal{L},\mathcal{N},\mathcal{G}$ and $\mathcal{S}$.

The local function $\mathcal{L}$ selects the pairs (v,j) based on their direct relation to the pair (u,i). For the item-based algorithm, it is given by

$$\mathcal{L}\big((u,i),(v,j)\big) = \mathbf{R}_{uj}\mathbf{R}_{vj}\mathbf{R}_{vi}.$$

It thus only selects those pairs (v,j) such that v and u share a preference for j and both i and j are preferred by v.

Next, the neighborhood function $\mathcal{N}$ further selects the pairs (v,j) based on their neighborhood relations to the pair (u,i). For the item-based algorithm, it is given by

$$\mathcal{N}\big((u,i),(v,j)\big) = |KNN(j)\cap\{i\}|.$$

Thus, only those pairs (v,j) for which i is in the neighborhood of j are selected. Notice that the selection of the pair (v,j) by $\mathcal{N}$ is independent of v. As such, the item-based algorithm ignores half of the neighborhood information about the pair (v,j).

Then, the global function $\mathcal{G}$ weighs the selected pairs (v,j) based on global statistics of the items i and j. For the item-based algorithm, it is given by

$$\mathcal{G}\big((u,i),(v,j)\big) = 1/\sqrt{c(i)c(j)}.$$

It thus gives lower weights to those pairs (v,j) for which j has a higher count. Intuitively, if j is more popular, the evidence related to the pair (v,j) is considered less informative. Similarly, if i is more popular, all evidence with respect to $s(u,i)$ is considered less informative. Notice that, in this case, also the weight of the pair (v,j), as determined by $\mathcal{G}$, is independent of v. As such, the item-based algorithm ignores $c(v)$ and $c(u)$, the global information about v and u. Notice furthermore that $\mathcal{G}$ is also used in the definition of the cosine similarity measure (Eq. 2), used to determine $KNN(j)$.

As in this case no rescaling is applied, $\mathcal{S}$ is given by

$$\mathcal{S}\big((u,i),(v,j)\big) = 1.$$

3.2 User-Based

For a given user u, the user-based nearest neighbors algorithm by Sarwar et al. [13] first finds the neighborhood $KNN(u)$ using the cosine similarity. Next, each neighboring user $v \in KNN(u)$ increases the score of a candidate recommendation i, if i is preferred by v. Thus, the score of a candidate recommendation i for user u is given by [13]:

$$s(u,i) = \frac{1}{|KNN(u)|}\sum_{v\in KNN(u)} \mathbf{R}_{vi}. \tag{5}$$

Multiplying the above equation with the constant $|\mathcal{I}| = \sum_{j\in\mathcal{I}} 1$ does not change the ordering of the pairs (u,i) and allows us to rewrite it as

$$s(u,i) = \sum_{v\in\mathcal{U}} \sum_{j\in\mathcal{I}} \mathbf{R}_{vi} \cdot |KNN(u) \cap \{v\}| \cdot \frac{1}{|KNN(u)|}.$$

Hence we have reformulated also the user-based algorithm as a weighted sum over all possible user-item pairs (v,j) in which the weight of a (v,j)-pair with respect to the pair (u,i) is determined by the functions $\mathcal{L},\mathcal{N},\mathcal{G}$ and $\mathcal{S}$ (Eq. 3).

The local function $\mathcal{L}$, which selects the pairs (v,j) based on their direct relation to the pair (u,i), is for the user-based algorithm given by

$$\mathcal{L}\big((u,i),(v,j)\big) = \mathbf{R}_{vi}.$$

It thus selects those pairs (v,j) such that v prefers i. Unlike the item-based algorithm, it does not consider the information $\mathbf{R}_{uj}$ and $\mathbf{R}_{vj}$ to discriminate between different (v,j)-pairs. Hence, the selection of the pair (v,j) by $\mathcal{L}$ is independent of j. As such, the user-based algorithm ignores local information related to j when weighing the pair (v,j).

Next, the neighborhood function $\mathcal{N}$ further selects the pairs (v,j) based on their neighborhood relations to the pair (u,i). For the user-based algorithm, it is given by

$$\mathcal{N}\big((u,i),(v,j)\big) = |KNN(u) \cap \{v\}|.$$

Thus, only those pairs (v,j) for which v is in the neighborhood of u are selected. Notice that the selection of the pair (v,j) by $\mathcal{N}$ is independent of j. As such, the user-based algorithm ignores half of the neighborhood information about the pair (v,j).

Furthermore, since this algorithm does not weight the pairs (v,j) with any global statistic of u,i,v or j, the global function for the user-based algorithm is given by

$$\mathcal{G}\big((u,i),(v,j)\big) = 1.$$

Finally, for the user-based algorithm, the rescaling function $\mathcal{S}$ rescales the weight of the pairs (v,j) with the size of the neighborhood of u and is therefore given by

$$\mathcal{S}\big((u,i),(v,j)\big) = |KNN(u)|.$$

3.3 Generalization

Now we generalize the definitions of the four functions $\mathcal{L},\mathcal{N},\mathcal{G}$ and $\mathcal{S}$ such that our formulation covers the most well known user- and item-based algorithms. Table 1 gives an overview of how these functions are defined for both the existing algorithms and our novel algorithm, which we will propose in the next section.

First, the local function $\mathcal{L}$ selects the pairs (v,j) depending on the direct relations $\mathbf{R}_{ui}$, $\mathbf{R}_{vj}$ and $\mathbf{R}_{vi}$ between (v,j) and (u,i). The user-based algorithm (Sec. 3.2) considers only $\mathbf{R}_{vi}$ and ignores the other information. The item-based algorithm (Sec. 3.1) on the other hand, combines all three pieces of direct information in the multiplication $\mathbf{R}_{ui}\mathbf{R}_{vj}\mathbf{R}_{vi}$, and thus selects those pairs (v,j) such that v and u share a preference for j and both i and j are preferred by v. Essentially, any combination of these direct relationships between u,i,v and j is possible.

Secondly, the neighborhood function $\mathcal{N}$ weighs the pairs (v,j) depending on the neighborhoods $KNN(u)$, $KNN(v)$, $KNN(i)$ and $KNN(j)$. Existing algorithms for OCCF consider only one of the four neighborhoods, as shown in Sections 3.2 and 3.1. Consequently, the weighing function $\mathcal{N}$ reduces to a selection function for these algorithms . However, any function of these four neighborhoods can be used. For example, in our novel algorithm KUNN, which we will propose in Section 4, we use

$$\mathcal{N}\big((u,i),(v,j)\big) = |KNN(u) \cap \{v\}| + |KNN(i) \cap \{j\}|.$$

Both the user- and the item-based algorithm discussed in the previous two sections used the cosine similarity measure to compute $KNN(x)$ of an object x. Our formulation however, covers a broader range of similarity measures. Let $p_u,p_i,p_v,p_j \in \mathbb{R}$. For users $u,v \in \mathcal{U}$, we define the similarity measure to compute $KNN(u)$ as

$$sim(u,v) = \sum_{r_\mathcal{I}\in\mathcal{I}} \mathbf{R}_{ur_\mathcal{I}}\mathbf{R}_{vr_\mathcal{I}} \cdot c(u)^{p_u} c(v)^{p_v} c(r_\mathcal{I})^{p_j}. \quad (6)$$

Similarly, for items $i,j \in \mathcal{I}$, we define the similarity measure to compute $KNN(i)$ as

$$sim(i,j) = \sum_{r_\mathcal{U}\in\mathcal{U}} \mathbf{R}_{r_\mathcal{U}i}\mathbf{R}_{r_\mathcal{U}j} \cdot c(i)^{p_i} c(r_\mathcal{U})^{p_v} c(j)^{p_j}. \quad (7)$$

Notice that, in our formulation, the similarity between users (Eq. 6) and the similarity between items (Eq. 7) share the parameters p_v and p_j. Thus, choosing the user similarity limits the possibilities for choosing the item similarity and vice versa.

Thirdly, the global function $\mathcal{G}$ uses the global statistics $c(u)$, $c(i)$, $c(v)$ and $c(j)$ to weigh the pairs (v,j) with respect to the pair (u,i). It is given by

$$\mathcal{G}\big((u,i),(v,j)\big) = \Big(c(u)^{p_u} c(i)^{p_i} c(v)^{p_v} c(j)^{p_j}\Big)^{p_g}, \quad (8)$$

with $p_g \in \{0,1\}$ and p_u,p_i,p_v,p_j the same parameters as in Equations 6 and 7. Typically, these parameters are negative or zero. In that case, users u,v and items i,j with higher counts reduce the weight of the (v,j)-pairs with respect to (u,i). Intuitively, a more popular item is considered less informative for determining a taste, since this item is more likely preferred by diverse users. Similarly, a user that prefers many items is considered less informative for determining a taste, since this user's preferences are more likely to cover diverse items.

Notice that $\mathcal{G}$, $sim(u,v)$ and $sim(i,j)$ (Eq. 8, 7 and 6) share the factors $c(u)^{p_u}$, $c(i)^{p_i}$, $c(v)^{p_v}$ and $c(j)^{p_j}$. Therefore we introduce the notation

$$\mathcal{W}\big((u,i),(v,j)\big) = c(u)^{p_u} c(i)^{p_i} c(v)^{p_v} c(j)^{p_j},$$

which allows us to rewrite the global function (Eq. 8) as

$$\mathcal{G}\big((u,i),(v,j)\big) = \mathcal{W}\big((u,i),(v,j)\big)^{p_g}, \quad (9)$$

and the similarities (Eq. 6 and 7) as

$$sim(u,v) = \sum_{r_\mathcal{I}\in\mathcal{I}} \mathbf{R}_{ur_\mathcal{I}}\mathbf{R}_{vr_\mathcal{I}} \cdot \mathcal{W}\big((u,*),(v,r_\mathcal{I})\big),$$

$$sim(i,j) = \sum_{r_\mathcal{U}\in\mathcal{U}} \mathbf{R}_{r_\mathcal{U}i}\mathbf{R}_{r_\mathcal{U}j} \cdot \mathcal{W}\big((*,i),(r_\mathcal{U},j)\big),$$

with $c(*) = 1$. This definition of $\mathcal{W}$ covers both the user-based algorithm by Sarwar et al. [13] and the different item-based algorithms by Deshpande et al. [2]. A more general

Table 1: Function definitions of the unifying formulation for selected algorithms. Our novel algorithm is bold faced and marked with a ⋆.

Algorithm	$\mathcal{L}\big((u,i),(v,j)\big)$	$\mathcal{N}\big((u,i),(v,j)\big)$	$\mathcal{S}\big((u,i),(v,j)\big)$	$\mathcal{W}\big((u,i),(v,j)\big)$	p_g	$k_\mathcal{U}$	$k_\mathcal{I}$
user-based, cosine [13]	$\mathbf{R}_{vi}$	$\lvert KNN(u)\cap\{v\}\rvert$	$\mathcal{S}_u(u)$	$\frac{1}{\sqrt{c(u)c(v)}}$	0	$\leq\lvert\mathcal{U}\rvert$	0
item-based, cosine, SNorm- [2]	$\mathbf{R}_{uj}\mathbf{R}_{vj}\mathbf{R}_{vi}$	$\lvert KNN(j)\cap\{i\}\rvert$	1	$\frac{1}{\sqrt{c(i)c(j)}}$	1	0	$\leq\lvert\mathcal{I}\rvert$
item-based, cosine, SNorm+ [2]	$\mathbf{R}_{uj}\mathbf{R}_{vj}\mathbf{R}_{vi}$	$\lvert KNN(j)\cap\{i\}\rvert$	$\mathcal{S}_j(j)$	$\frac{1}{\sqrt{c(i)c(j)}}$	1	0	$\leq\lvert\mathcal{I}\rvert$
⋆ **KUNN**	$\mathbf{R}_{uj}\mathbf{R}_{vj}\mathbf{R}_{vi}$	$\lvert KNN(u)\cap\{v\}\rvert$ $+\lvert KNN(i)\cap\{j\}\rvert$	1	$\frac{1}{\sqrt{c(u)c(j)c(v)c(i)}}$	1	$\leq\lvert\mathcal{U}\rvert$	$\leq\lvert\mathcal{I}\rvert$

definition of $\mathcal{W}$ would cover a broader range of nearest neighbors algorithms. We choose this definition over a more general one to emphasize the strong similarity between the latter two algorithms.

Finally, some algorithms rescale the weights of the pairs (v,j) with the density of $KNN(u)$, $KNN(i)$, $KNN(v)$ or $KNN(j)$. A neighborhood $KNN(x)$ of x is denser if the total distance of x to its neighbors is smaller. In other words, if the sum of the similarities of x to its neighbors is higher. Depending on the choice for $KNN(u)$, $KNN(i)$, $KNN(v)$ or $KNN(j)$, the rescaling function is given by one of the four density functions

$$\mathcal{S}_u(u) = 1/\sum_{r_\mathcal{U}\in KNN(u)} sim(u,r_\mathcal{U})^{p_g},$$
$$\mathcal{S}_i(i) = 1/\sum_{r_\mathcal{I}\in KNN(i)} sim(i,r_\mathcal{I})^{p_g},$$
$$\mathcal{S}_v(v) = 1/\sum_{r_\mathcal{U}\in KNN(v)} sim(r_\mathcal{U},v)^{p_g},$$
$$\mathcal{S}_j(j) = 1/\sum_{r_\mathcal{I}\in KNN(j)} sim(r_\mathcal{I},j)^{p_g},$$

with p_g the same parameter as for the global function $\mathcal{G}$ (Eq. 9).

For an algorithm that does not apply rescaling,

$$\mathcal{S}\big((u,i),(v,j)\big) = 1$$

4. KUNN UNIFIED NEAREST NEIGHBORS

Looking at Table 1, we observe that the user-based algorithm by Sarwar et al. [13] ignores the information $\mathbf{R}_{uj}$, $\mathbf{R}_{vj}$, $c(i)$, $c(j)$, $KNN(i)$ and $KNN(j)$ for weighing the pairs (v,j) with respect to (u,i). Similarly, all item-based algorithms by Deshpande et al. [2] ignore the information $c(u)$, $c(v)$, $KNN(u)$ and $KNN(v)$ for weighing the pairs (v,j) with respect to (u,i). Thus, the existing algorithms ignore an important part of the available information. What is more, the information ignored by item-based algorithms is disjoint with the information ignored by the user-based algorithm. However, the fact that both user- and item-based algorithms generate acceptable results [13, 2], indicates that most likely both types of information are useful. Therefore, a novel algorithm that combines both types of information, KUNN[1], potentially leads to improved results. The experiments discussed in Section 5 confirm that this is indeed the case. It is not only possible to outperform the individual user- and item-based algorithms, but also to outperform state-of-the-art matrix factorization algorithms [12, 7, 10].

The definitions of $\mathcal{L}$, $\mathcal{N}$, $\mathcal{G}$ and $\mathcal{S}$ corresponding to KUNN are given on the last row of Table 1.

For KUNN, the local function is given by

$$\mathcal{L}\big((u,i),(v,j)\big) = \mathbf{R}_{uj}\mathbf{R}_{vj}\mathbf{R}_{vi}$$

and thus selects those pairs (v,j) such that v and u share a preference for j and both i and j are preferred by v. As such, KUNN does not discard any information about the direct relation between (v,j) and (u,i).

Next, the neighborhood function for KUNN is given by

$$\mathcal{N}\big((u,i),(v,j)\big) = \lvert KNN(u)\cap\{v\}\rvert + \lvert KNN(i)\cap\{j\}\rvert.$$

It thus selects those pairs (v,j) such that v is in the neighborhood of u or j is in the neighborhood of i. If both conditions are fulfilled, the weight of the pair (v,j) with respect to $s(u,i)$ is doubled. As such, KUNN uses neighborhood information of both v and j to weight (v,j) with respect to (u,i).

Furthermore, for KUNN, $\mathcal{W}$ is given by

$$\mathcal{W}\big((u,i),(v,j)\big) = \frac{1}{\sqrt{c(u)c(i)c(v)c(j)}}.$$

Consequently, the user-similarity results in

$$sim(u,v) = \sum_{r_\mathcal{I}\in\mathcal{I}} \frac{\mathbf{R}_{ur_\mathcal{I}}\mathbf{R}_{vr_\mathcal{I}}}{\sqrt{c(u)c(v)c(r_\mathcal{I})}},$$

and the item-similarity:

$$sim(i,j) = \sum_{r_\mathcal{U}\in\mathcal{U}} \frac{\mathbf{R}_{r_\mathcal{U}i}\mathbf{R}_{r_\mathcal{U}j}}{\sqrt{c(i)c(r_\mathcal{U})c(j)}}.$$

Intuitively, if u and v share a preference for an item $r_\mathcal{I}$, it is only weak evidence of their similarity if $r_\mathcal{I}$ is popular and both u and v have many preferences. Similarly, if i and j are both preferred by $r_\mathcal{U}$, it is only weak evidence of their similarity if $r_\mathcal{U}$ has many preferences and both i and j are popular items.

In addition, the global function for KUNN is given by

$$\mathcal{G}\big((u,i),(v,j)\big) = \mathcal{W}\big((u,i),(v,j)\big)^1 = \frac{1}{\sqrt{c(u)c(i)c(v)c(j)}}.$$

Intuitively, if the counts of u, i, v and j are higher, it is more likely that the direct relation between (v,j) and

[1]KUNN is a recursive acronym for KUNN Unified Nearest Neighbors

$(u,i)(\mathcal{L}\big((u,i),(v,j)\big)=1)$, exists by chance. Therefore, this direct relation is less informative.

Finally, we see no convincing arguments for making KUNN more complex by introducing a rescaling factor. Therefore we define

$$\mathcal{S}\big((u,i),(v,j)\big)=1.$$

To enhance the intuitive understanding of KUNN, we rewrite Equation 3 as

$$s(u,i)=\frac{s_{\mathcal{U}}(u,i)+s_{\mathcal{I}}(u,i)}{\sqrt{c(u)c(i)}}, \tag{10}$$

with

$$s_{\mathcal{U}}(u,i)=\sum_{v\in KNN(u)}\mathbf{R}_{vi}\frac{1}{\sqrt{c(v)}}\sum_{\substack{j\in\mathcal{I}\\ \mathbf{R}_{uj}=1\\ \mathbf{R}_{vj}=1}}\frac{1}{\sqrt{c(j)}}$$

and

$$s_{\mathcal{I}}(u,i)=\sum_{j\in KNN(i)}\mathbf{R}_{uj}\frac{1}{\sqrt{c(j)}}\sum_{\substack{v\in\mathcal{U}\\ \mathbf{R}_{vi}=1\\ \mathbf{R}_{vj}=1}}\frac{1}{\sqrt{c(v)}}.$$

Thus, we can decompose $s(u,i)$ in a user-based part $s_{\mathcal{U}}(u,i)$ and an item-based part $s_{\mathcal{I}}(u,i)$. Notice that these two parts cannot be reduced to any existing user- or item-based algorithm.

The user-based part $s_{\mathcal{U}}(u,i)$ is a weighted sum over the neighbors of u in which the weight of a neighbor v is proportional to:

- $\mathbf{R}_{vi}$: v has a known preference for i,
- $1/\sqrt{c(v)}$: if v prefers many items, her known preference for i becomes less informative,
- $\sum_{j\in\mathcal{I},\mathbf{R}_{uj}=1,\mathbf{R}_{vj}=1}$: every preference that v shares with u increases the weight of v for recommending items to u,
- $1/\sqrt{c(j)}$: if v and u share a preference for j, it is less informative if j is a more popular item.

A similar intuition holds for the item-based part $s_{\mathcal{I}}(u,i)$.

Finally, the denominator of Equation 10, reduces the strength of the evidence if u prefers many items and i is popular.

5. EXPERIMENTAL EVALUATION

We experimentally evaluate the accuracy of KUNN on three datasets: the *Movielens*, the *Yahoo!Music*^{user} and the *Yahoo!Music*^{random} datasets [19, 5]. These datasets contain ratings of users for movies and songs respectively. The ratings are on a 1 to 5 scale with 5 expressing the highest preference. We convert these datasets to binary, positive-only datasets. Following Pradel et al. [11], we convert the ratings 4 and 5 to preferences and the ratings 1 and 2 to dislikes. Furthermore, we ignore the ratings 3, effectively converting them to unknowns. As such, we obtain a buffer between preferences and dislikes. Since our setting presumes binary, positive-only data, both the unknowns and the dislikes are represented by zeros in the training data. For evaluating the recommendations however, we are allowed to distinguish between unknowns and dislikes.

In our evaluation we compare *KUNN* with five other algorithms. As a baseline, we select *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. Next, we select the *user-based* nearest neighbors algorithm with cosine similarity by Sarwar et al. [13] and the widely used *item-based* nearest neighbors algorithm with cosine similarity and similarity normalization (*SNorm+*) by Deshpande et al. [2]. We choose this item-based algorithm because it performed well in comparison to other item-based algorithms [2]. Furthermore, we also compare with *UB+IB*, a linear ensemble that computes a recommendation score as

$$s(u,i)=\lambda s_{UB}(u,i)+(1-\lambda)s_{IB}(u,i),$$

with $s_{UB}(u,i)$ the user-based score from the algorithm by Sarwar et al. and $s_{IB}(u,i)$ the item-based score from the algorithm by Deshpande et al. Finally, we compare with two state-of-the-art matrix factorization algorithms for OCCF: the *BPRMF* algorithm by Rendle et al. and the *WRMF* algorithm by Hu et al. [7]. For *WRMF* and *BPRMF* we used the MyMediaLite implementation [4]. For all other algorithms, we used our own implementation, which is available at `https://bitbucket.org/KVs/unncf_submit`.

To thoroughly evaluate the performance of KUNN, we use evaluation measures from multiple previous works [2, 12, 11, 13]. The experimental evaluation consists of two experimental setups. In the *user selected* setup (Sec. 5.1), the users selected which items they rated. In the *random selected* setup (Sec. 5.2), users were asked to rate randomly chosen items.

5.1 User Selected Setup

This experimental setup is applicable to the *Movielens* and the *Yahoo!Music*^{user} datasets. In both datasets, users chose themselves which items they rated. Following Deshpande et al. [2] and Rendle et al. [12], one preference of every user is randomly chosen to be the test preference 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 $\mathbf{R}$. All other entries of $\mathbf{R}$ are zero. We define the hit set $\mathcal{H}_u$ of a user u as the set containing all test preferences of that user. For this setup, this is a singleton, denoted as $\{h_u\}$, or the empty set if no test preference is chosen. Furthermore, we define $\mathcal{U}_t$ as the set of users with a test preference, i.e. $\mathcal{U}_t=\{u\in\mathcal{U}\mid|\mathcal{H}_u|>0\}$. Table 2 summarizes some characteristics of the datasets.

For every user $u\in\mathcal{U}_t$, every algorithm ranks the items $\{i\in\mathcal{I}\mid\mathbf{R}_{ui}=0\}$ based on $\mathbf{R}$. We denote the rank of the test preference h_u in such a ranking as $r(h_u)$.

Then, every set of rankings is evaluated using three measures. Following Deshpande et al. [2] we use hit rate at 10 and average reciprocal hit rate at 10. In general, 10 can be replaced by any natural number $N\leq|\mathcal{I}|$. We follow Deshpande et al. [2] and choose N=10.

Hit rate at 10 is given by

$$HR@10=\frac{1}{|\mathcal{U}_t|}\sum_{u\in\mathcal{U}_t}|\mathcal{H}_u\cap top10(u)|,$$

with $top10(u)$ the 10 highest ranked items for user u. Hence, $HR@10$ gives the percentage of test users for which the test preference is in the top 10 recommendations.

Average reciprocal hit rate at 10 is given by

$$ARHR@10=\frac{1}{|\mathcal{U}_t|}\sum_{u\in\mathcal{U}_t}|\mathcal{H}_u\cap top10(u)|\cdot\frac{1}{r(h_u)}.$$

Table 2: Dataset characteristics after transformation to binary, positive-only data.

				$c(u)$		$\lvert\mathcal{H}_u\rvert$		$\lvert\mathcal{M}_u\rvert$	
Dataset	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{U}_t\rvert$	$\lvert\mathcal{I}\rvert$	mean	std	mean	std	mean	std
Movielens	6040	6037	3706	94.3	105.0	1	0	NA	NA
Yahoo!Music^{user}	15400	13204	1000	7.70	11.57	1	0	NA	NA
Yahoo!Music^{random}	15400	2401	1000	8.70	11.57	1.89	1.23	6.17	2.09

Unlike hit rate, average reciprocal hit rate takes into account the rank of the test preference in the top 10 of a user.

Following Rendle et al. [12], we also use the AMAN version of area under the curve, which is for this experimental setup given by

$$AUC^{AMAN} = \frac{1}{|\mathcal{U}_t|} \sum_{u \in \mathcal{U}_t} \frac{|\mathcal{I}| - r(h_u)}{|\mathcal{I}| - 1}.$$

AMAN stands for All Missing As Negative, meaning that a missing preference is evaluated as a dislike. Like average reciprocal hit rate, the area under the curve takes into account the rank of the test preference in the recommendation list for a user. However, AUC^{AMAN} decreases slower than $ARHR@10$ when $r(h_u)$ increases.

We repeat all experiments five times, drawing a different random sample of test preferences every time.

5.2 Random Selected Setup

The *user selected* experimental setup introduces two biases in the evaluation. Firstly, popular items get more ratings. Secondly, the majority of the ratings is positive. These two biases can have strong influences on the results and are thoroughly discussed by Pradel et al. [11]. The *random selected* test setup avoids these biases.

Following Pradel et al. [11], the training dataset is constructed in the *user selected* way: users chose to rate a number of items they selected themselves. The test dataset however, is the result of a voluntary survey in which random items were presented to the users and a rating was asked. In this way, both the popularity and the positivity bias are avoided.

This experimental setup is applicable to the dataset *Yahoo!Music*^{random}. The training data, $\mathbf{R}$, of this dataset is identical to the full *Yahoo!Music*^{user} dataset. Additionally, this dataset includes a test dataset in which the rated items were randomly selected. For a given user u, the hit set $\mathcal{H}_u$ contains all preferences of this user which are present in the test dataset. The set of dislikes $\mathcal{M}_u$ contains all dislikes of this user which are present in the test dataset. We define $\mathcal{U}_t = \{u \in \mathcal{U} \mid |\mathcal{H}_u| > 0, |\mathcal{M}_u| > 0\}$, i.e. all users with both preferences and dislikes in the test dataset. Table 2 summarizes some characteristics of the dataset.

For every user $u \in \mathcal{U}_t$, every algorithm ranks the items in $\mathcal{H}_u \cup \mathcal{M}_u$ based on the training data $\mathbf{R}$. The rank of an item i in such a ranking is denoted $r(i)$.

Then, following Pradel et al. [11], we evaluate every set of rankings with the AMAU version of area under the curve, which is given by

$$AUC^{AMAU} = \frac{1}{|\mathcal{U}_t|} \sum_{u \in \mathcal{U}_t} AUC^{AMAU}(u),$$

with

$$AUC^{AMAU}(u) = \sum_{h \in \mathcal{H}_u} \frac{|\{m \in \mathcal{M}_u \mid r(m) > r(h)\}|}{|\mathcal{H}_u||\mathcal{M}_u|}.$$

AMAU stands for All Missing As Unknown, meaning that a missing preference does not influence the evaluation. Hence, $AUC^{AMAU}(u)$ measures, for a user u, the fraction of dislikes that is, on average, ranked behind the preferences. A big advantage of this measure is that it only relies on known preferences and known dislikes. Unlike the other measures, it does not make the bold assumption that items not preferred by u in the past are disliked by u.

Because of the natural split in a test and training dataset, repeating the experiment with different test preferences is not applicable.

5.3 Parameter Selection

Every personalization algorithm in the experimental evaluation has at least one parameter. For every experiment we try to find the best set of parameters using grid search. An experiment is defined by (1) the outer training dataset $\mathbf{R}$, (2) the outer hitset $\bigcup_{u \in \mathcal{U}_t} \mathcal{H}_u$, (3) the outer dislikes $\bigcup_{u \in \mathcal{U}_t} \mathcal{M}_u$, (4) the evaluation measure, and (5) the algorithm. Applying grid search, we first choose a finite number of parameter sets. Secondly, from the training data $\mathbf{R}$, we create five inner data splits, defined by $\mathbf{R}^k$, $\bigcup_{u \in \mathcal{U}_t} \mathcal{H}_u^k$, and $\bigcup_{u \in \mathcal{U}_t} \mathcal{M}_u$ for $k \in \{1, 2, 3, 4, 5\}$. Then, for every parameter set, we rerun the algorithm on all five inner training datasets $\mathbf{R}^k$ and evaluate them on the corresponding inner test datasets with the chosen evaluation measure. Finally, the parameter set with the best average score over the five inner data splits, is chosen to be the best parameter set for the outer training dataset $\mathbf{R}$ with respect to the chosen evaluation measure. Notice that, given an outer training dataset, the best parameters with respect to one evaluation measure, can differ from the best parameters with respect to another evaluation measure.

5.4 Results and Discussion

Table 3 shows the evaluation of the considered algorithms on different datasets, with different measures. The experiments belonging to the user selected setup were repeated 5 times, drawing a different random sample of test preferences every time. Therefore, we report both the mean and the standard deviation for these experiments. The experiments belonging to the random selected setup use a natural split between the test and training dataset. Therefore, randomizing the test set choice is not applicable and only one value is reported for every experiment. Scripts for automatically repeating all experiments are available at `https://bitbucket.org/KVs/unncf_submit`. The exact paramter

Table 3: Selected algorithms evaluated with multiple experiments. Our novel algorithm is marked with a ⋆

dataset	measure		⋆ KUNN	WRMF [7, 10]	BPRMF [12]	UB+IB	item-based, cosine, SNorm+ [2]	user-based, cosine [13]	pop
Movielens	HR@10	mean	**.217**	**.217**	.174	.209	.165	.209	.063
		std	.002	.001	.004	.002	.002	.002	.002
	$ARHR$@10	mean	**.093**	.091	.071	.090	.073	.090	.022
		std	.003	.002	.001	.002	.002	.002	.002
	AUC^{AMAN}	mean	.941	**.944**	.934	.927	.916	.927	.865
		std	.001	.001	.002	.001	.001	.001	.001
Yahoo!Music^{user}	HR@10	mean	**.378**	.338	.285	.365	.364	.316	.187
		std	.009	.007	.009	.003	.012	.012	.004
	$ARHR$@10	mean	**.163**	.145	.102	.160	.160	.127	.062
		std	.003	.007	.006	.006	.007	.003	.002
	AUC^{AMAN}	mean	**.926**	.910	.891	.910	.911	.910	.841
		std	.002	.002	.003	.003	.003	.003	.001
Yahoo!Music^{random}	AUC^{AMAU}		**.788**	.770	.755	.768	.768	.761	.670

combinations explored by the grid search procedure (Sec. 5.3) and other details can be inspected in these scripts.

From Table 3 we can make several observations. First of all, KUNN outperforms every other algorithm five out of seven times, shares one best performance with WRMF, and is one time outperformed by WRMF.

Secondly, the user-based algorithm clearly outperforms the item-based algorithm on the *Movielens* dataset. On the *Yahoo!Music*^{user} dataset on the other hand, the item-based algorithm clearly outperforms the user-based algorithm. For UB+IB, the grid search procedure (Sec. 5.3) is successful in choosing λ such that the best of both algorithms in the ensemble gets the highest weight. However, UB+IB cannot outperform the best of both individual algorithms. KUNN, on the other hand, successfully combines the user- and item-based information and consistently outperforms both individual algorithms.

Thirdly, the rather disappointing performance of BPRMF stands out. A possible explanation lies in the choice of the parameters. Following Rendle et al. [12], we used grid search (Sec. 5.3) to choose the best out of 267 parameter combinations for BPRMF in every experiment. We can however not rule out that there exist parameter combinations outside our 267 possibilities, for which BPRMF performs better. Finding a good set of parameters is harder for BPRMF than for the other algorithms because BPRMF uses 7 parameters that can all take an infinite number of values. The parameters $k_{\mathcal{U}}$ and $k_{\mathcal{I}}$ of KUNN, on the other hand, are integers within the bounds $[0, |\mathcal{U}|]$ and $[0, |\mathcal{I}|]$ respectively. Therefore, we can find a good set of parameters among only 25 possibilities.

Finally, all personalized algorithms perform much better than the non personalized baseline *pop*.

6. EXPLAINABILITY

The consideration that explanations of item-based algorithms are superior comes from observing the formulas for computing the recommendation scores [3, 7]. For item-based algorithms on the one hand, this formula is given by Equation 4 in which every term can be attributed to one of the known preferences of the target user. Therefore the known preferences related to the biggest terms can naturally serve as an explanation for the recommendation. For user-based algorithms on the other hand, the formula is given by Equation 5 in which every term can be attributed to one of the collaborative users. This is much less useful because the most similar users give no intuitive explanation for the recommendation as they are probably strangers to the target user and the same for every recommendation. Furthermore, this kind of explanation would also be an invasion on the privacy of these collaborative users.

However, this difference is only artificial. Thanks to our reformulation, we can write both the user-based algorithm by Sarwar et al. [13] (Eq. 5) and KUNN as a sum over the known preferences of the target user.

We start with the user-based algorithm (Eq. 5). This algorithm can be rewritten as

$$s(u,i) = \frac{1}{|KNN(u)|} \sum_{v \in KNN(u)} \mathbf{R}_{vi} \left(\frac{\sum_{l_1 \in \mathcal{I}} \mathbf{R}_{ul_1} \mathbf{R}_{vl_1}}{\sum_{l_2 \in \mathcal{I}} \mathbf{R}_{ul_2} \mathbf{R}_{vl_2}} \right).$$

Notice that the last factor in the summation is simply 1, but allows us to rewrite the equation as

$$s(u,i) = \frac{1}{|KNN(u)|} \sum_{\substack{l_1 \in \mathcal{I} \\ \mathbf{R}_{ul_1}}} \left(\sum_{v \in KNN(u)} \frac{\mathbf{R}_{vl_1} \mathbf{R}_{vi}}{\sum_{l_2 \in \mathcal{I}} \mathbf{R}_{ul_2} \mathbf{R}_{vl_2}} \right),$$

In the above equation, we have written the user based score $s(u,i)$ as a weighted sum over the known preferences of u.

The known preferences l_1 with the biggest weights, serve as a natural explanation for recommending i to u. Hence, we have naturally explained the user-based recommendation of i for u.

Next, we consider KUNN, which can be rewritten as

$$s(u,i) = \frac{1}{\sqrt{c(u)c(i)}} \sum_{\substack{j \in \mathcal{I} \\ \mathbf{R}_{uj}=1}} \left(\frac{1}{\sqrt{c(j)}} \sum_{\substack{v \in \mathcal{U} \\ \mathbf{R}_{vi}=1 \\ \mathbf{R}_{vj}=1}} \frac{\mathcal{N}\big((u,i),(v,j)\big)}{\sqrt{c(v)}} \right),$$

by regrouping Equation 3. Thus, also KUNN computes $s(u,i)$ as a weighted sum over the known preferences of u.

Again, the known preferences with the biggest weights serve as a natural explanation for recommending i to u. Hence, we have naturally explained recommendations made by KUNN.

7. RELATED WORK

The majority of the work on collaborative filtering presumes rating data. This setting is significantly different from our setting with binary, positive-only data. Therefore, the algorithms for rating based collaborative filtering differ on important aspects such as how they measure similarity, how their performance is measured and how they handle missing ratings, user- and item-biases.

Hence, specific algorithms have been proposed for OCCF. First, an analysis of user-based nearest neighbors algorithms for OCCF was given by Sarwar et al. [13]. We discussed their work in Section 3.2 and compared experimentally to it in Section 5.

Later, Deshpande et al. [2] introduced the item-based nearest neighbors approach for OCCF. We discussed their work in Section 3.1 and experimentally compared to it in Section 5. Afterwards, similar approaches were suggested [15, 1]. Recently, multiple authors proposed to minimize a global cost function for computing the item similarities [12, 9, 8].

Furthermore, Symeonidis et al. [16] recognized that combining user- and item-based approaches could be beneficial. Additionally, Wang et al. [18] proposed a unification of user- and item-based algorithms. However, their work presumes rating data.

Finally, there exist also matrix factorization algorithms for OCCF [7, 10, 12, 14]. In our experimental evaluation (Sec. 5), we compared with two state-of-the-art algorithms of this class: WRMF by Hu et al. [7] and BPRMF by Rendle et al. [12].

8. CONCLUSIONS AND FUTURE WORK

We proposed KUNN, a novel algorithm for one class collaborative filtering, a setting that covers many applications.

KUNN originates from a reformulation that unifies user- and item-based nearest neighbors algorithms. Thanks to this reformulation, it becomes clear that user- and item-based nearest neighbors algorithms discard important parts of the available information.

KUNN improves upon these existing nearest neighbors algorithms by actually using more of the available information. Our experimental evaluation shows that KUNN not only outperforms existing nearest neighbors algorithms, but also state-of-the-art matrix factorization algorithms.

Finally, we challenged the well accepted belief that item-based algorithms are superior for explaining the recommendations they produce. Thanks to our reformulation, we were able to show that also recommendations by KUNN and the traditional user-based algorithm come with a natural explanation.

We see research on novel definitions of the functions $\mathcal{L}$, $\mathcal{N}$, $\mathcal{G}$ and $\mathcal{S}$ as the most important direction for future work.

9. REFERENCES

[1] F. Aiolli. Efficient top-n recommendation for very large scale binary rated datasets. In *RecSys*, pages 273–280, 2013.

[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] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Mymedialite: A free recommender system library. In *RecSys*, pages 305–308, 2011.

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

[6] J. Herlocker, J. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *CSCW*, pages 241–250, 2000.

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

[8] S. Kabbur, X. Ning, and G. Karypis. Fism: Factored item similarity models for top-n recommender systems. In *KDD*, pages 659–667, 2013.

[9] X. Ning and G. Karypis. Slim: Sparse linear methods for top-n recommender systems. In *ICDM*, pages 497–506, 2011.

[10] 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.

[11] B. Pradel, N. Usunier, and P. Gallinari. Ranking with non-random missing ratings: Influence of popularity and positivity on evaluation metrics. In *RecSys*, pages 147–154, 2012.

[12] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[13] B. Sarwar, G. Karypis, J. Kostan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *EC*, pages 158–167, 2000.

[14] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *RecSys*, pages 139–146, 2012.

[15] B. Sigurbjornsson and R. van Zwol. Flickr tag recommendation based on collective knowledge. In *WWW*, pages 327–336, 2008.

[16] P. Symeonidis, A. Nanopoulos, A. Papadopoulos, and Y. Manolopoulos. Nearest-biclusters collaborative filtering based on constant and coherent values. *Inf. Retr.*, 11(1):51–75, 2008.

[17] N. Tintarev. Explanations of recommendations. In *RecSys*, pages 203–206, 2007.

[18] J. Wang, A. P. de Vries, and M. J. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR*, pages 501–508, 2006.

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