Unifying user-based and item-based collaborative filtering approaches by similarity fusion

Conference Paper · August 2006

DOI: 10.1145/1148170.1148257 · Source: DBLP

3 authors:

Jun Wang
University College London
163 PUBLICATIONS **4,762** CITATIONS

SEE PROFILE

Arjen P. de Vries
Radboud University
353 PUBLICATIONS **6,938** CITATIONS

SEE PROFILE

Marcel J T Reinders
Delft University of Technology
699 PUBLICATIONS **17,874** CITATIONS

SEE PROFILE

Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion

Jun Wang[1], Arjen P. de Vries[1,2], Marcel J.T. Reinders[1]
Information and Communication Theory Group[1],
Faculty of Electrical Engineering, Mathematics and Computer Science,
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands
CWI[2], Amsterdam, The Netherlands
{jun.wang,m.j.t.reinders}@tudelft.nl, arjen@acm.org

ABSTRACT

Memory-based methods for collaborative filtering predict new ratings by averaging (weighted) ratings between, respectively, pairs of similar users *or* items. In practice, a large number of ratings from similar users or similar items are not available, due to the sparsity inherent to rating data. Consequently, prediction quality can be poor. This paper reformulates the memory-based collaborative filtering problem in a generative probabilistic framework, treating individual user-item ratings as predictors of missing ratings. The final rating is estimated by fusing predictions from three sources: predictions based on ratings of the same item by other users, predictions based on different item ratings made by the same user, and, third, ratings predicted based on data from other but similar users rating other but similar items. Existing user-based and item-based approaches correspond to the two simple cases of our framework. The complete model is however more robust to data sparsity, because the different types of ratings are used in concert, while additional ratings from similar users towards similar items are employed as a background model to smooth the predictions. Experiments demonstrate that the proposed methods are indeed more robust against data sparsity and give better recommendations.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - *Information Filtering*

General Terms

Algorithms, Performance, Experimentation

Keywords

Recommender Systems, Collaborative Filtering, Smoothing, Similarity Fusion

SIGIR'06, August 6–11, 2006, Seattle, Washington, USA.

1. INTRODUCTION

Collaborative filtering aims at predicting the user interest for a given item based on a collection of user profiles. Commonly, these profiles either result from asking users explicitly to rate items or are inferred from log-archives ([7]). Research started with memory-based approaches to collaborative filtering, that can be divided in user-based approaches like [1, 5, 9, 14] and item-based approaches like [3, 15]. The former approaches form a heuristic implementation of the "Word of Mouth" phenomenon. Memory-based approaches are widely used in practice, e.g., [5, 11].

Given an unknown test rating (of a test item by a test user) to be estimated, memory-based collaborative filtering first measures similarities between test user and other users (user-based), or, between test item and other items (item-based). Then, the unknown rating is predicted by averaging the (weighted) known ratings of the test item by similar users (user-based), or the (weighted) known ratings of similar items by the test user (item-based).

In both cases, only partial information from the data embedded in the user-item matrix is employed to predict unknown ratings (using either correlation between user data or correlation between item data). Because of the sparsity of user profile data however, many related ratings will not be available for the prediction, Therefore, it seems intuitively desirable to fuse the ratings from both similar users and similar items, to reduce the dependency on often missing data. Also, methods known previously ignore the information that can be obtained from ratings made by other but similar users to the test user on other but similar items. Not using such ratings causes the *data sparsity problem* of memory-based approaches to collaborative filtering: for many users and items, no reliable recommendation can be made because of a lack of similar ratings.

This paper sets up a generative probabilistic framework to exploit more of the data available in the *user-item matrix*, by fusing all ratings with predictive value for a recommendation to be made. Each individual rating in the user-item matrix is treated as a separate prediction for the unknown test rating (of a test item from a test user). The confidence of each individual prediction can be estimated by considering both its similarity towards the test user and that towards the test item. The overall prediction is made by averaging the individual ratings weighted by their confidence. The more similar a rating towards the test rating, the higher the weight assigned to that rating to make the prediction. Under this

framework, the item-based and user-based approaches are two special cases, and these can be systematically combined. By doing this, our approach allows us to take advantage of user correlations *and* item correlations embedded in the user-item matrix. Besides, smoothing from a background model (estimated from known ratings of similar items by similar users) is naturally integrated into our framework to improve probability estimation and counter the problem of data sparsity.

The remainder of the paper is organized as follows. We first summarize related work, introduce notation, and present additional background information for the two main memory-based approaches, i.e., user-based and item-based collaborative filtering. We then introduce our similarity fusion method to unify user-based and item-based approaches. We provide an empirical evaluation of the relationship between data sparsity and the different models resulting from our framework, and finally conclude our work.

2. RELATED WORK

Collaborative filtering approaches are often classified as memory-based or model-based. In the memory-based approach, all rating examples are stored *as-is* into memory (in contrast to learning an abstraction). In the prediction phase, similar users or items are sorted based on the memorized ratings. Based on the ratings of these similar users or items, a recommendation for the test user can be generated. Examples of memory-based collaborative filtering include user-based methods [1, 5, 9, 14] and item-based methods [3, 15]. The advantage of the memory-based methods over their model-based alternatives is that less parameters have to be tuned; however, the data sparsity problem is not handled in a principled manner.

In the model-based approach, training examples are used to generate a model that is able to predict the ratings for items that a test user has not rated before. Examples include decision trees [1], aspect models [7, 17] and latent factor models [2]. The resulting 'compact' models solve the data sparsity problem to a certain extent. However, the need to tune an often significant number of parameters has prevented these methods from practical usage. Lately, researchers have introduced dimensionality reduction techniques to address data sparsity [4, 13, 16]. However, as pointed out in [8, 19], some useful information may be discarded during the reduction.

Recently, [8] has explored a graph-based method to deal with data sparsity, using transitive associations between user and items in the bipartite user item graph. [18] has extended the probabilistic relevance model in text retrieval ([6]) to the problem of collaborative filtering and a linear interpolation smoothing has been adopted. These approaches are however limited to binary rating data. Another recent direction in collaborative filtering research combines memory-based and model-based approaches [12, 19]. For example, [19] clusters the user data and applies intra-cluster smoothing to reduce sparsity. The framework proposed in our paper extends this idea to include item-based recommendations into the final prediction, and does not require to cluster the data set a priori.

3. BACKGROUND

This section introduces briefly the user- and item-based approaches to collaborative filtering [5, 15]. For M items and K users, the user profiles are represented in a $K \times M$ user-item matrix $\mathbf{X}$ (Fig. 1(a)). Each element $x_{k,m} = r$ indicates that user k rated item m by r, where $r \in \{1, ..., |r|\}$ if the item has been rated, and $x_{k,m} = \emptyset$ means that the rating is unknown.

The user-item matrix can be decomposed into row vectors:

$$\mathbf{X} = [\mathbf{u}_1, \ldots, \mathbf{u}_K]^T, \mathbf{u}_k = [x_{k,1}, \ldots, x_{k,M}]^T, \ k = 1, \ldots, K$$

where T denotes transpose. Each row vector $\mathbf{u}_k^T$ corresponds to a user profile and represents a particular user's item ratings. As discussed below, this decomposition leads to user-based collaborative filtering.

Alternatively, the matrix can also be represented by its column vectors:

$$\mathbf{X} = [\mathbf{i}_1, ..., \mathbf{i}_M], \mathbf{i}_m = [x_{1,m}, ..., i_{K,m}]^T, \ m = 1, ..., M$$

where each column vector $\mathbf{i}_m$ corresponds to a specific item's ratings by all K users. This representation results in item-based recommendation algorithms.

3.1 User-based Collaborative Filtering

User-based collaborative filtering predicts a test user's interest in a test item based on rating information from similar user profiles [1, 5, 14]. As illustrated in Fig. 1(b), each user profile (row vector) is sorted by its dis-similarity towards the test user's profile. Ratings by more similar users contribute more to predicting the test item rating. The set of similar users can be identified by employing a threshold or selecting top-N. In the top-N case, a set of top-N similar users $\mathcal{S}_{\mathbf{u}}(\mathbf{u}_k)$ towards user k can be generated according to:

$$\mathcal{S}_{\mathbf{u}}(\mathbf{u}_k) = \{\mathbf{u}_a | rank \ s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a) \leq N, x_{a,m} \neq \emptyset\} \qquad (1)$$

where $|\mathcal{S}_{\mathbf{u}}(\mathbf{u}_k)| = N$. $s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)$ is the similarity between users k and a. Cosine similarity and Pearson's correlation are popular similarity measures in collaborative filtering, see e.g. [1, 5]. The similarity could also be learnt from training data [9]. This paper adopts the cosine similarity measure, comparing two user profiles by the cosine of the angle between the corresponding row vectors.

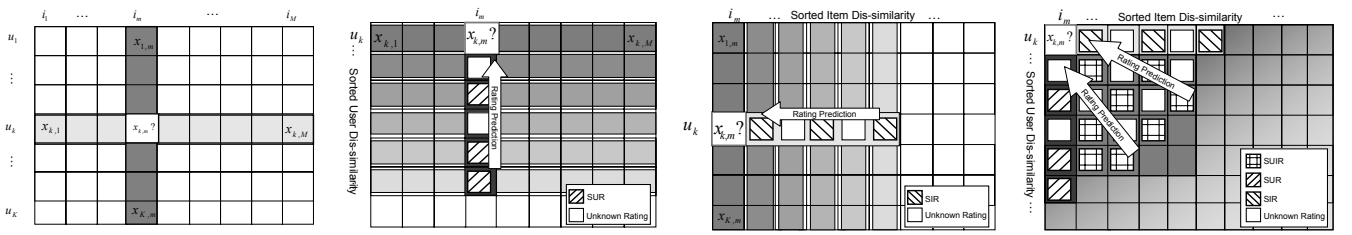
Consequently, the predicted rating $\widehat{x}_{k,m}$ of test item m by test user k is computed as (see also [1, 5])

$$\widehat{x}_{k,m} = \overline{u}_k + \frac{\sum_{\mathbf{u}_a \in \mathcal{S}_{\mathbf{u}}(\mathbf{u}_k)} s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)(x_{a,m} - \overline{u}_a)}{\sum_{\mathbf{u}_a \in \mathcal{S}_{\mathbf{u}}(\mathbf{u}_k)} s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)} \qquad (2)$$

where $\overline{u}_k$ and $\overline{u}_a$ denote the average rating made by users k and a, respectively.

Existing methods differ in their treatment of unknown ratings from similar users ($x_{a,m} = \emptyset$). Missing ratings can be replaced by a 0 score, which lowers the prediction, or the average rating of that similar user could be used [1, 5]. Alternatively, [19] replaces missing ratings by an interpolation of the user's average rating and the average rating of his or her cluster.

Before we discuss its dual method, notice in Eq. 2 and the illustration in Fig. 1(b) how user-based collaborative filtering takes only a small proportion of the user-item matrix into consideration for recommendation. Only the *known test item ratings by similar users* are used. We refer to these ratings as the set of 'similar user ratings' (the blocks



(a) (b) (c) (d)

Figure 1: (a) The user-item matrix (b) Rating prediction based on user similarity (c) Rating prediction based on item similarity (d) Rating prediction based on rating similarity.

with upward diagonal pattern in Fig. 1(b)): $SUR_{k,m} = \{x_{a,m} | \mathbf{u}_a \in \mathcal{S}_\mathbf{u}(\mathbf{u}_k)\}$. For simplicity, we drop the subscript k, m of $SUR_{k,m}$ in the remainder of the paper.

3.2 Item-based Collaborative Filtering

Item-based approaches such as [3, 11, 15] apply the same idea, but use similarity between items instead of users. As illustrated in Fig. 1(c), the unknown rating of a test item by a test user can be predicted by averaging the ratings of other similar items rated by this test user [15]. Again, each item (column vector) is sorted and re-indexed according to its dis-similarity towards the test item in the user-item matrix, and, ratings from more similar items are weighted stronger. Formally (see also [15]),

$$\widehat{x}_{k,m} = \frac{\sum_{\mathbf{i}_b \in \mathcal{S}_\mathbf{i}(\mathbf{i}_m)} s_\mathbf{i}(\mathbf{i}_m, \mathbf{i}_b)(x_{k,b})}{\sum_{\mathbf{i}_b \in \mathcal{S}_\mathbf{i}(\mathbf{i}_m)} s_\mathbf{i}(\mathbf{i}_m, \mathbf{i}_b)} \quad (3)$$

Where item similarity $s_\mathbf{i}(\mathbf{i}_m, \mathbf{i}_b)$ can be approximated by the cosine measure or Pearson correlation [11, 15]. To remove the difference in rating scale between users when computing the similarity, [15] has proposed to adjust the cosine similarity by subtracting the user's average rating from each co-rated pair beforehand. We adopt this similarity measure in this paper. Like the top-N similar users, a set of top-N similar items towards item m, denoted as $\mathcal{S}_\mathbf{i}(\mathbf{i}_m)$, can be generated according to:

$$\mathcal{S}_\mathbf{i}(\mathbf{i}_m) = \{\mathbf{i}_b | rank\ s_\mathbf{i}(\mathbf{i}_m, \mathbf{i}_b) \leq N, x_{k,b} \neq \emptyset\} \quad (4)$$

Fig. 1(c) illustrates how Eq. 3 takes only the *known similar item ratings by the test user* into account for prediction. We refer to these ratings as the set of 'similar item ratings' (the blocks with downward diagonal pattern in Fig. 1(c)): $SIR_{k,m} = \{x_{k,b} | \mathbf{i}_b \in \mathcal{S}_\mathbf{i}(\mathbf{i}_m)\}$. Again, for simplicity, we drop the subscript k, m of $SIR_{k,m}$ in the remainder of the paper.

4. SIMILARITY FUSION

Relying on *SUR or SIR* data only is undesirable, especially when the ratings from these two sources are quite often not available. Consequently, predictions are often made by averaging ratings from 'not-so-similar' users or items. We propose to improve the accuracy of prediction by fusing the *SUR* and *SIR* data, to complement each other under the missing data problem.

Additionally, we point out that the user-item matrix contains useful data beyond the previously used *SUR* and *SIR* ratings. As illustrated in Fig. 1 (d), the *similar item ratings made by similar users* may provide an extra source for prediction. They are obtained by sorting and re-indexing rows and columns according to their dis-similarities towards the test user and the test item respectively. In the remainder, this part of the matrix is referred to as 'similar user item ratings' (the grid blocks in Fig. 1(d)): $SUIR_{k,m} = \{x_{a,b} | \mathbf{u}_a \in \mathcal{S}_\mathbf{u}(\mathbf{u}_k), \mathbf{i}_b \in \mathcal{S}_\mathbf{i}(\mathbf{i}_m), a \neq k, b \neq m\}$. The subscript k, m of $SUIR_{k,m}$ is dropped.

Combining these three types of ratings in a single collaborative filtering method is non-trivial. We propose to treat each element of the user-item matrix as a separate predictor. Its reliability or *confidence* is then estimated based upon its similarity towards the test rating. We then predict the test rating by averaging the individual predictions weighted by their confidence. The remainder of the section gives a probabilistic formulation for the proposed method.

4.1 Individual Predictors

Users rate items differently. Some users have a preference for the extreme values of the rating scale, while others rarely deviate from the median. Likewise, items may be rated by different types of users. Some items get higher ratings than their 'true' value, simply because they have been rated by a positive audience. Addressing the differences in rating behavior, we first normalize the user-item matrix before making predictions.

Removing the mean ratings per user and item gives individual predictions as

$$p_{k,m}(x_{a,b}) = x_{a,b} - (\bar{x}_a - \bar{x}_k) - (\bar{x}_b - \bar{x}_m) \quad (5)$$

where $p_{k,m}(x_{a,b})$ is the prediction function for the test item k rating made by test user m, $\bar{x}_a$ and $\bar{x}_k$ are the average ratings by user a and k, and $\bar{x}_b$ and $\bar{x}_m$ are the average ratings of item b and m. Appendix A derives that normalizing the matrix by independently subtracting the row and column means gives the same result.

4.2 Probabilistic Fusion Framework

Let us first define the sample space of ratings as $\Phi_r = \{\emptyset, 1, ..., |r|\}$ (like before, $\emptyset$ denotes the unknown rating). Let $x_{a,b}$ be a random variable over the sample space Φ_r, captured in the user-item matrix, $a \in \{1, \ldots, K\}$ and $b \in \{1, \ldots, M\}$. Collaborative filtering then corresponds to estimating conditional probability $P(x_{k,m} | \mathcal{P}_{k,m})$, for an unknown test rating $x_{k,m}$, given a pool of individual predictors

$$\mathcal{P}_{k,m} = \{p_{k,m}(x_{a,b}) | x_{a,b} \neq \emptyset\}.$$

Consider first a pool that consists of *SUR* and *SIR* ratings only (i.e., $x_{a,b} \in (SUR \cup SIR)$).

$$P(x_{k,m} | SUR, SIR) \equiv P(x_{k,m} | \{p_{k,m}(x_{a,b}) | x_{a,b} \in SUR \cup SIR\}) \quad (6)$$

We write $P(x_{k,m}|SUR, SIR)$ for the conditional probability depending on the predictors originating from SUR and SIR. Likewise, $P(x_{k,m}|SUR)$ and $P(x_{k,m}|SIR)$ specify a pool consisting of SUR or SIR predictors only.

Now introduce a binary variable I_1, that corresponds to the relative importance of SUR and SIR. This hidden variable plays the same role as the prior introduced in [6] to capture the importance of a query term in information retrieval. $I_1 = 1$ states that $x_{k,m}$ depends completely upon ratings from SUR, while $I_1 = 0$ corresponds to full dependency on SIR. Under these assumptions, the conditional probability can be obtained by marginalization of variable I_1:

$$\begin{aligned} & P(x_{k,m}|SUR, SIR) \\ &= \sum_{I_1} P(x_{k,m}|SUR, SIR, I_1)P(I_1|SUR, SIR) \\ &= P(x_{k,m}|SUR, SIR, I_1 = 1)P(I_1 = 1|SUR, SIR)+ \\ &\quad P(x_{k,m}|SUR, SIR, I_1 = 0)P(I_1 = 0|SUR, SIR) \end{aligned} \tag{7}$$

By definition, $x_{k,m}$ is independent from SUR when $I_1 = 1$, so $P(x_{k,m}|SUR, SIR, I_1 = 1) = P(x_{k,m}|SUR)$. Similarly, $P(x_{k,m}|SUR, SIR, I_1 = 0) = P(x_{k,m}, |SIR)$. If we provide a parameter λ as shorthand for $P(I_1 = 1|SUR, SIR)$, we have

$$\begin{aligned} & P(x_{k,m}|SUR, SIR) \\ &= P(x_{k,m}|SUR)\lambda + P(x_{k,m}|SIR)(1 - \lambda) \end{aligned} \tag{8}$$

Next, we extend the model to take into account the $SUIR$ ratings:

$$\begin{aligned} & P(x_{k,m}|\mathrm{SUR}, SIR, SUIR) \\ &\equiv P(x_{k,m}|\{p_{k,m}(x_{k,m})|x_{a,b} \in SUR \cup SIR \cup SUIR\}) \end{aligned} \tag{9}$$

We introduce a second binary random variable I_2, that corresponds to the relative importance of the $SUIR$ predictors. $I_2 = 1$ specifies that the unknown rating depends on ratings from $SUIR$ only and $I_2 = 0$ that it depends on the ratings from SIR and SUR instead. Marginalization on variable I_2 gives:

$$\begin{aligned} & P(x_{k,m}|SUR, SIR, SUIR) \\ &= \sum_{I_2} P(x_{k,m}|SUR, SIR, SUIR, I_2)P(I_2|SUR, SIR, SUIR) \\ &= P(x_{k,m}|SUR, SIR, SUIR, I_2 = 1)\cdot \\ &\quad P(I_2 = 1|SUR, SIR, SUIR)+ \\ &\quad P(x_{k,m}|SUR, SIR, SUIR, I_2 = 0)\cdot \\ &\quad (1 - P(I_2 = 1|SUR, SIR, SUIR)) \end{aligned} \tag{10}$$

Following the argument from above and providing a parameter δ as shorthand for $P(I_2 = 1|SUR, SIR, SUIR)$, we have

$$\begin{aligned} & P(x_{k,m}|SUR, SIR, SUIR) \\ &= P(x_{k,m}|SUR, SIR)(1 - \delta) + P(x_{k,m}|SUIR)\delta \end{aligned} \tag{11}$$

Substitution of Eq. 8 then gives:

$$\begin{aligned} & P(x_{k,m}|SUR, SIR, SUIR) \\ &= \Big(P(x_{k,m}|SUR)\lambda + P(x_{k,m}|SIR)(1 - \lambda)\Big)(1 - \delta)+ \\ &\quad P(x_{k,m}|SUIR)\delta \end{aligned} \tag{12}$$

Finally, the following equation gives the expected value of the unknown test rating:

$$\begin{aligned} \hat{x}_{k,m} &= \sum_{r=1}^{|r|} rP(x_{k,m} = r|SUR, SIR, SUIR) \\ &= \big(\sum_{r=1}^{|r|} rP(x_{k,m} = r|SUIR)\delta\big)+ \\ &\quad \big(\sum_{r=1}^{|r|} rP(x_{k,m} = r|SUR)\lambda(1 - \delta)\big)+ \\ &\quad \big(\sum_{r=1}^{|r|} rP(x_{k,m} = r|SIR)(1 - \lambda)(1 - \delta)\big) \end{aligned} \tag{13}$$

The resulting model can be viewed as using importance sampling of the neighborhood ratings as predictors. λ and δ control the selection (sampling) of data from the three different sources.

4.3 Probability Estimation

The next step is to estimate the probabilities in the fusion framework expressed in Eq. 13.

λ and δ are determined experimentally by using the cross-validation, for example following the methodology of Section 5.3. The three remaining probabilities can be viewed as estimates of the likelihood of a rating $x_{a,b}$ from SIR, SUR, or $SUIR$, to be similar to the test rating $x_{k,m}$. We assume that the probability estimates for SUR and SIR are proportional to the similarity between row vectors $s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)$ (Section 3.1) and column vectors $s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)$ (Section 3.2), respectively. For $SUIR$ ratings, we assume the probability estimate to be proportional to the combination of $s_{\mathbf{u}}$ and $s_{\mathbf{i}}$. To combine them, we use a Euclidean dis-similarity space such that the resulting combined similarity is lower than either of them.

$$s_{\mathbf{ui}}(x_{k,m}, x_{a,b}) = \frac{1}{\sqrt{(1/s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a))^2 + (1/s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b))^2}} \tag{14}$$

This results in the following conditional probability estimates:

$$\begin{aligned} & P(x_{k,m} = r|SUR) \\ &= \frac{\sum_{\forall x_{a,b}:(x_{a,b} \in SUR) \wedge (p_{k,m}(x_{a,b}) = r)} s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)}{\sum_{\forall x_{a,b}:x_{a,b} \in SUR} s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)} \end{aligned}$$

$$\begin{aligned} & P(x_{k,m} = r|SIR) \\ &= \frac{\sum_{\forall x_{a,b}:(x_{a,b} \in SIR) \wedge (p_{k,m}(x_{a,b}) = r)} s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)}{\sum_{\forall x_{a,b}:x_{a,b} \in SIR} s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)} \end{aligned}$$

$$\begin{aligned} & P(x_{k,m} = r|SUIR) \\ &= \frac{\sum_{\forall x_{a,b}:(x_{a,b} \in SUIR) \wedge (p_{k,m}(x_{a,b}) = r)} s_{\mathbf{ui}}(x_{k,m}, x_{a,b})}{\sum_{\forall x_{a,b}:x_{a,b} \in SUIR} s_{\mathbf{ui}}(x_{k,m}, x_{a,b})} \end{aligned} \tag{15}$$

Table 1: Percentage of the ratings that are available ($\neq \emptyset$).

	test item	1st most sim. item	2nd most sim item	3rd most sim. item	4th most sim. item
test user	-	0.58	0.56	0.55	0.54
1st most sim. user	0.54	0.58	0.58	0.58	0.57
2nd most sim. user	0.51	0.56	0.56	0.56	0.56
3rd most sim. user	0.51	0.57	0.57	0.57	0.56
4th most sim. user	0.49	0.55	0.55	0.56	0.55

Table 2: Mean Absolute Err (MAE) of individual predictions.

	test item	1st most sim. item	2nd most sim. item	3rd most sim. item	4th most sim. item
test user	-	0.824	0.840	0.866	0.871
1st most sim. user	0.914	0.925	0.927	0.942	0.933
2nd most sim. user	0.917	0.921	0.931	0.935	0.927
3rd most sim. user	0.927	0.947	0.952	0.953	0.945
4th most sim. user	0.928	0.929	0.939	0.946	0.932

After substitution from Eq. 15 (for readability, we put the detailed derivations in Appendix B), Eq. 13 results in:

$$\widehat{x}_{k,m} = \sum_{x_{a,b}} p_{k,m}(x_{a,b}) W^{a,b}_{k,m} \tag{16}$$

where

$$W^{a,b}_{k,m} = \begin{cases} \frac{s_{\mathbf{u}}(\mathbf{u}_k,\mathbf{u}_a)}{\sum_{x_{a,b}\in SUR} s_{\mathbf{u}}(\mathbf{u}_k,\mathbf{u}_a)}\lambda(1-\delta) & x_{a,b} \in SUR \\ \frac{s_{\mathbf{i}}(\mathbf{i}_m,\mathbf{i}_b)}{\sum_{x_{a,b}\in SIR} s_{\mathbf{i}}(\mathbf{i}_m,\mathbf{i}_b)}(1-\lambda)(1-\delta) & x_{a,b} \in SIR \\ \frac{s_{\mathbf{ui}}(x_{k,m},x_{a,b})}{\sum_{x_{a,b}\in SUIR} s_{\mathbf{ui}}(x_{k,m},x_{a,b})}\delta & x_{a,b} \in SUIR \\ 0 & otherwise \end{cases} \tag{17}$$

It is easy to prove that $\sum_{x_{a,b}} W^{a,b}_{k,m} = 1$. $W^{a,b}_{k,m}$ acts as a unified weight matrix to combine the predictors from the three different sources.

4.4 Discussion

Sum as Combination Rule λ and δ control the importance of the different rating sources. Their introduction results in a sum rule for fusing the individual predictors (Eq. 12 and 16.). Using the independence assumption on the three types of ratings and the Bayes' rule, one can easily derive a product combination from the conditional probability ([10]). However, the high sensitivity to estimation errors makes this approach less attractive in practice. We refer to [10] for a more detailed discussion of using a sum rule vs. the product rule for combing classifiers.

Unified Weights The unified weights in Eq. 17 provide a generative framework for memory-based collaborative filtering.

Eq. 17 shows how our scheme can be considered as two subsequent steps of linear interpolation. First, predictions from *SUR* ratings are interpolated with *SIR* ratings, controlled by λ. Next, the intermediate prediction is interpolated with predictions from the *SUIR* data, controlled by δ. Viewing the *SUIR* ratings as a background model, the second interpolation corresponds to smoothing the *SIR* and *SUR* predictions from the background model.

A bigger λ emphasizes user correlations, while smaller λ emphasizes item correlations. When λ equals one, our algorithm corresponds to a user-based approach, while λ equal to zero results in an item-based approach.

Tuning parameter δ controls the impact of smoothing from the background model (i.e. *SUIR*). When δ approaches zero, the fusion framework becomes the mere combination of user-based and item-based approaches without smoothing from the background model.

5. EMPIRICAL EVALUATION

5.1 Experimental Setup

We experimented with the MovieLens[1], EachMovie[2], and book-crossing[3] data sets. While we report only the MovieLens results (out of space considerations), the model behaves consistently across the three data sets.

The MovieLens data set contains 100,000 ratings (1-5 scales) from 943 users on 1682 movies (items), where each user has rated at least 20 items. To test on different number of training users, we selected the users in the data set at random into a training user set (100, 200, 300 training users, respectively) and the remaining users into a test user set. Users in the training set are only used for making predictions, while test users are the basis for measuring prediction accuracy. Each test user's ratings have been split into a set of *observed items* and one of *held-out items*. The ratings of observed items are input for predicting the ratings of held-out items.

We are specifically interested in the relationship between the density of the user-item matrix and the collaborative filtering performance. Consequently, we set up the following configurations:

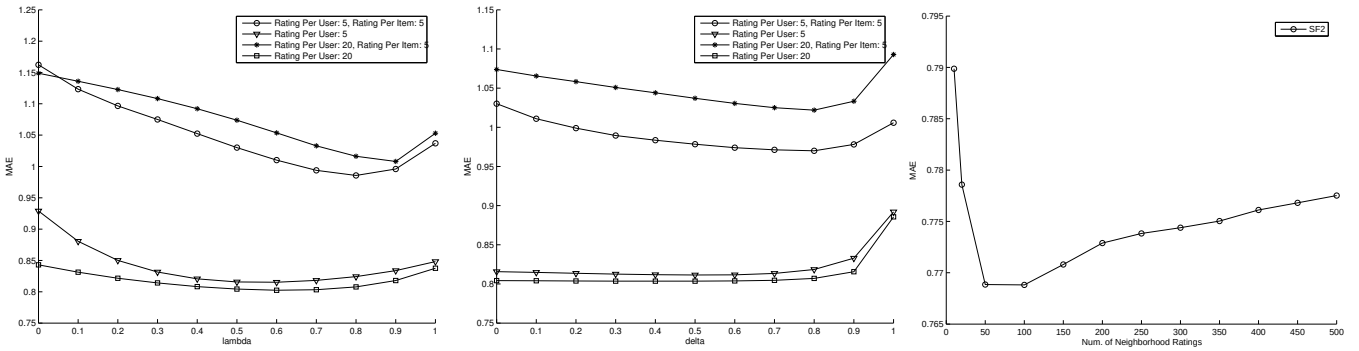
- **Test User Sparsity** Vary the number of items rated by test users in the observed set, e.g., 5, 10, or 20 ratings per user.
- **Test Item Sparsity** Vary the number of users who have rated test items in the held-out set; less than 5, 10, or 20 (denoted as '$<$ 5', '$<$ 10', or '$<$ 20'), or, unconstrained (denoted as 'No constraint').
- **Overall Training User Sparsity** Select a part of the rating data at random, e.g., 20%, 40%, 60% of the data set.

For consistency with experiments reported in the literature, e.g., [9, 15, 19]), we report the mean absolute error (MAE) evaluation metric. MAE corresponds to the average absolute deviation of predictions to the ground truth data, for all test item ratings and test users:

$$MAE = \frac{\sum_{k,m} |x_{k,m} - \widehat{x}_{k,m}|}{L}, \tag{18}$$

where L denotes the number of tested ratings. A smaller value indicates a better performance.

[1] `http://www.grouplens.org/`
[2] `http://research.compaq.com/SRC/eachmovie/`
[3] `http://www.informatik.uni-freiburg.de/~cziegler/BX/`



(a) Lambda (b) Delta

Figure 2: Impact of the two parameters.

Figure 3: Size of neighborhood.

5.2 Individual Predictors

We first report some properties of the three types of individual predictions used in our approach. Table 1 illustrates the availability of the top-4 neighborhood ratings in the MovieLens data set. The first column contains the top-4 *SUR* ratings, the first row the top-4 *SIR* ratings; the remaining cells correspond to the top-4x4 *SUIR* ratings. We observe that only about half of these ratings are given. Table 2 summarizes recommendation MAE of individual predictors (applying Eq. 5) using leave-one-out cross-validation. Clearly, more similar ratings provide more accurate predictions. While *SUIR*s ratings are in general less accurate than *SUR*s and *SIR*s, these may indeed complement missing or unreliable *SIR* and *SUR* ratings.

5.3 Impact of Parameters

Recall the two parameters in Eq. 17: λ balances the predictions between *SUR* and *SIR*, and δ smoothes the fused results by interpolation with a pool of *SUIR* ratings.

We first test the sensitivity of λ, setting δ to zero. This scheme, called *SF1*, combines user-based and item-based approaches, but does not use additional background information. Fig. 2(a) shows recommendation MAE against varying λ from zero (a pure item-based approach) to one (a pure user-based approach). The graph plots test user sparsity 5 and 20, and test item sparsity settings '< 5' and unconstrained. The value of the optimal λ demonstrates that interpolation between user-based and item-based approaches (*SF1*) improves the recommendation performance. More specifically, the best results are obtained with λ between 0.6 and 0.9. This optimal value emphasizing the *SUR* ratings may be somewhat surprising, as Table 2 indicated that the *SIR* ratings should be more reliable for prediction. However, in the data sets considered, the number of users is smaller than the number of items, causing the user weights $s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)$ to be generally smaller than the item weights $s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)$. When removing the constraint on test item sparsity, the optimal λ shifts down from about 0.9 for the two upper curves ('< 5') to 0.6 for the two lower curves (unconstrained). A lower λ confirms the expectation that *SIR* ratings gain value when more items have been rated.

Fig. 2 (b) shows the sensitivity of δ after fixing λ to 0.7. The graph plots the MAE for the same four configurations when parameter δ is varied from zero (without smoothing) to one (rely solely on the background model: *SUIR* ratings). When δ is non-zero, the *SF1* results are smoothed by a pool of *SUIR* ratings, which we called fusion scheme *SF2*. We observe that δ reaches its optimal in 0.8 when the rating data is sparse in the neighborhood ratings from the item and user aspects (upper two curves). In other words, smoothing from a pool of *SUIR* ratings improves the performance for sparse data. However, when the test item sparsity is not constrained, its optimum spreads a wide range of values, and the improvement over MAE without smoothing ($\delta = 0$) is not clear.

Additional experiments (not reported here) verified that there is little dependency between the choice of λ and the optimal value of δ. The optimal parameters can be identified by using the cross validation from the training data.

Like pure user-based and item-based approaches, the size of neighborhood N also influences the performance of our fusion methods. Fig. 3 shows MAE of *SF2* when the number of neighborhood ratings is varied. The optimal results are obtained with the neighborhood size between 50 and 100. We select 50 as our optimal choice.

5.4 Data Sparsity

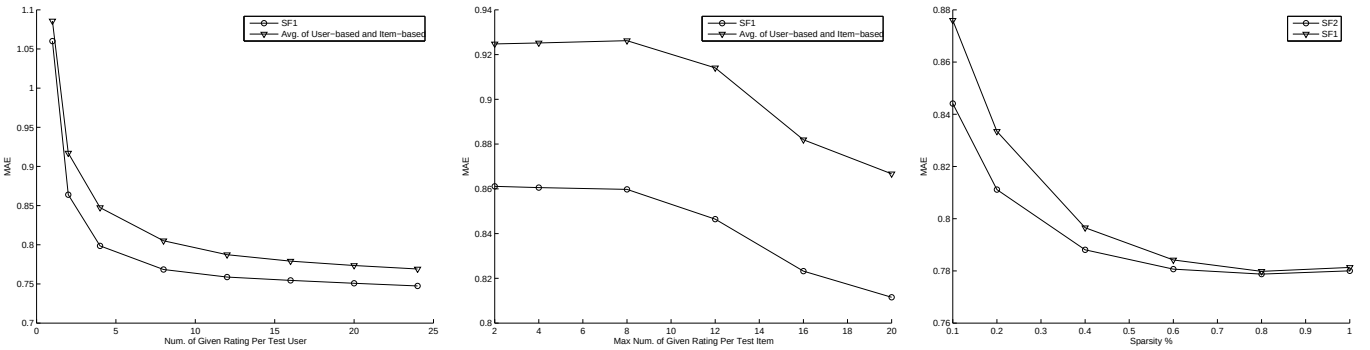
The next experiments investigate the effect of data sparsity on the performance of collaborative filtering in more detail. Fig. 4(a) and (b) compare the behavior of scheme *SF1* to that obtained by simply averaging user-based and item-based approaches, when varying test user sparsity (Fig. 4(a)) and test item sparsity (4(b)). The results indicate that combining user-based and item-based approaches (*SF1*) consistently improves the recommendation performance regardless neighborhood sparsity of test users or items.

Next, Fig. 4(c) plots the gain of *SF2* over *SF1* when varying overall training user sparsity. The figure shows that *SF2* improves *SF1* more and more when the rating data becomes more sparse. This can be explained as follows. When the user-item matrix is less dense, it contains insufficient test item ratings by similar users (for user-based recommendation), and insufficient similar item ratings by the test user (for item-based recommendation) as well. Therefore, smoothing using ratings by similar items made by similar users improves predictions.

We conclude from these experiments that the proposed fusion framework is effective at improving the quality of recommendations, even when only sparse data are available.

5.5 Comparison to Other Methods

We continue with a comparison to results obtained with other methods, setting λ to 0.7 and δ to 0 for *SF1* and



(a) Test User Sparsity (b) Test Item Sparsity (c) Overall Training User Sparsity

Figure 4: Performance under different sparsity.

Table 3: Comparison with other memory-based approaches. A smaller value means a better performance.

Ratings Given (Test Item):	**< 5**			**< 10**			**< 20**			**No constrain**		
Ratings Given (Test User):	**5**	**10**	**20**	**5**	**10**	**20**	**5**	**10**	**20**	**5**	**10**	**20**
SF2	**1.054**	**0.966**	**1.070**	**0.995**	**0.917**	**0.997**	**0.945**	**0.879**	**0.923**	**0.825**	**0.794**	**0.805**
SF1	1.086	1.007	1.097	1.035	0.942	1.024	0.976	0.898	0.936	0.836	0.796	0.809
UBVS	1.129	1.034	1.117	1.052	0.972	1.054	0.996	0.913	0.969	0.891	0.809	0.836
IBVS	1.190	1.055	1.131	1.108	0.992	1.068	1.066	0.954	0.977	0.938	0.842	0.842

(a) Number of Training Users: 100

Ratings Given (Test Item):	**< 5**			**< 10**			**< 20**			**No constrain**		
Ratings Given (Test User):	**5**	**10**	**20**	**5**	**10**	**20**	**5**	**10**	**20**	**5**	**10**	**20**
SF2	**0.960**	**0.945**	**0.948**	**0.915**	**0.875**	**0.885**	**0.826**	**0.802**	**0.828**	**0.806**	**0.786**	**0.803**
SF1	0.976	0.960	0.963	0.927	0.883	0.895	0.832	0.804	0.831	0.808	0.786	0.804
UBVS	1.108	1.028	1.024	1.070	0.962	0.972	0.914	0.842	0.885	0.879	0.811	0.848
IBVS	1.187	1.071	1.034	1.122	1.006	0.976	0.974	0.875	0.886	0.921	0.840	0.847

(b) Number of Training Users: 200

Ratings Given (Test Item):	**< 5**			**< 10**			**< 20**			**No constrain**		
Ratings Given (Test User):	**5**	**10**	**20**	**5**	**10**	**20**	**5**	**10**	**20**	**5**	**10**	**20**
SF2	**0.956**	**0.908**	**0.941**	**0.911**	**0.885**	**0.912**	**0.842**	**0.828**	**0.859**	**0.798**	**0.782**	**0.805**
SF1	1.013	0.968	0.977	0.928	0.908	0.938	0.847	0.834	0.867	0.802	0.783	0.807
UBVS	1.024	0.971	1.044	0.966	0.919	0.980	0.921	0.877	0.936	0.886	0.808	0.852
IBVS	1.117	1.043	1.024	1.044	0.990	1.004	0.962	0.910	0.932	0.914	0.837	0.850

(c) Number of Training Users: 300

using $\lambda = 0.7$ and $\delta = 0.7$ for *SF2*. We first compare our results to the standard user-based vector similarity (UBVS) approach of [1] and the item-based adjusted cosine similarity (IBVS) of [15]. We report results for test user sparsity 5, 10, or 20, and test item sparsity '< 5', '< 10', '< 20' or 'No constrain'. Table 3 summarizes the results, showing how *SF1* and *SF2* outperform the other methods in all twelve resulting configurations.

Next, we adopt the subset of MovieLens (see [9, 19]), which consists of 500 users and 1000 items. We followed the exact evaluation procedure described in [19] to compare the performance of our *SF2* scheme with the state-of-art results listed in [19]. Table 4 presents our experimental results, as well as the four best methods according to their experiments, i.e., cluster-based Pearson Correlation Coefficient (SCBPCC) [19], the Aspect Model (AM) ([7]), 'Personality Diagnosis' (PD) ([12]) and the user-based Pearson Correlation Coefficient (PCC) ([1]). Our method outperforms these methods in all configurations.

6. CONCLUSIONS

We proposed a novel algorithm to unify the user-based and item-based collaborative filtering approaches to overcome limitations specific to either of them. We showed that user-based and item-based approaches are only two special cases in our probabilistic fusion framework. Furthermore, by using a linear interpolation smoothing, other ratings by similar users towards similar items can be treated as a background model to smooth the rating predictions. The experiments showed that our new fusion framework is effective in improving the prediction accuracy of collaborative filtering and dealing with the data sparsity problem. In the future, we plan to conduct better formal analyses of the fusion model and more complete comparisons with previous methods.

7. REFERENCES

[1] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of UAI*, 1998.

[2] J. Canny. Collaborative filtering with privacy via factor analysis. In *Proc. of SIGIR*, 1999.

[3] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[4] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval Journal*, 4(2):133–151, July 2001.

[5] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proc. of SIGIR*, 1999.

[6] D. Hiemstra. Term-specific smoothing for the language modeling approach to information retrieval: the importance of a query term. In *Proc. of SIGIR*, pages 35–41, 2002.

Table 4: Comparison with the result reported in [19]. A smaller value means a better performance.

Num. of Training Users:	100			200			300		
Ratings Given (Test User):	5	10	20	5	10	20	5	10	20
SF2	**0.847**	**0.774**	**0.792**	**0.827**	**0.773**	**0.783**	**0.804**	**0.761**	**0.769**
SCBPCC	0.848	0.819	0.789	0.831	0.813	0.784	0.822	0.810	0.778
AM	0.963	0.922	0.887	0.849	0.837	0.815	0.820	0.822	0.796
PD	0.849	0.817	0.808	0.836	0.815	0.792	0.827	0.815	0.789
PCC	0.874	0.836	0.818	0.859	0.829	0.813	0.849	0.841	0.820

[7] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Info. Syst.*, Vol 22(1):89–115, 2004.

[8] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):116–142, 2004.

[9] R. Jin, J. Y. Chai, and L. Si. An automatic weighting scheme for collaborative filtering. In *Proc. of SIGIR*, 2004.

[10] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(3):226–239, 1998.

[11] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, Jan/Feb.:76–80, 2003.

[12] D. M. Pennock, E. Horvitz, S. Lawrence, and C. Giles. Collaborative filtering by personality diagnosis: a hybrid memory and model based approach. In *Proc. of UAI*, 2000.

[13] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proc. of ICML*, 2005.

[14] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proc. of ACM CSCW*, 1994.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. of the WWW Conference*, 2001.

[16] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *Proc. of ACM WebKDD Workshop*, 2000.

[17] L. Si and R. Jin. Flexible mixture model for collaborative filtering. In *ICML*, 2003.

[18] J. Wang, A. P. de Vries, and M. J. Reinders. A user-item relevance model for log-based collaborative filtering. In *Proc. of ECIR06, London, UK*, 2006.

[19] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proc. of SIGIR*, 2005.

APPENDIX

A. NORMALIZATION

We first normalize the matrix by subtracting the average item ratings:

$$n(x_{a,b})_I = x_{a,b} - \frac{1}{K}\sum_i x_{i,b} = x_{a,b} - \bar{x}_b$$

where $n(x_{a,b})_I$ normalizes ratings by subtracting the mean item rating. $\bar{x}_b$ is the average rating of item b.

We normalize again by the average user rating:

$$\begin{aligned} & n(x_{a,b})_{I,U} \\ &= n(x_{a,b})_I - \frac{1}{M}\sum_j n(x_{a,j})_I \\ &= x_{a,b} - \frac{1}{K}\sum_i x_{i,b} - \frac{1}{M}\sum_j \big(x_{a,j} - \frac{1}{K}\sum_i x_{i,j}\big) \\ &= x_{a,b} - \frac{1}{K}\sum_i x_{i,b} - \frac{1}{M}\sum_j x_{a,j} + \frac{1}{MK}\sum_{i,j} x_{i,j} \\ &= x_{a,b} - \bar{x}_b - \bar{x}_a + \bar{x} \end{aligned}$$

where $n(x_{a,b})_{I,U}$ is the normalization of both item and user aspects. $\bar{x}_a$ is the average rating from user a. $\bar{x}$ is the average of all the ratings. From here, we see that the result does not depend on the order of normalization (whether to normalize first by user or by item).

Treating each normalized individual rating as individual predictor results in:

$$\begin{aligned} & \hat{x}_{k,m} - \bar{x}_m - \bar{x}_k + \bar{x} = x_{a,b} - \bar{x}_b - \bar{x}_a + \bar{x} \\ \therefore \quad & p_{k,m}(x_{a,b}) = \hat{x}_{k,m} = x_{a,b} - (\bar{x}_a - \bar{x}_k) - (\bar{x}_b - \bar{x}_m) \end{aligned}$$

B. A UNIFIED WEIGHTING FUNCTION

More specifically, replacing three conditional probabilities with Eq. 15, the following can be derived from Eq. 13:

$$\begin{aligned} & \widehat{x}_{k,m} \\ &= \sum_{r=1}^{|r|} r\Big(\sum_{p_{k,m}(x_{a,b})=r} \big(\sum_{x_{a,b}\in SUR} A + \sum_{x_{a,b}\in SIR} B + \sum_{x_{a,b}\in SUIR} C\big)\Big) \\ &= \sum_{\forall x_{a,b}: x_{a,b}\in SUR} p_{k,m}(x_{a,b})A + \sum_{\forall x_{a,b}: x_{a,b}\in SIR} p_{k,m}(x_{a,b})B + \\ & \qquad \sum_{\forall x_{a,b}: x_{a,b}\in SUIR} p_{k,m}(x_{a,b})C \end{aligned}$$

where

$$A = \frac{s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)}{\sum_{\forall x_{a,b}: x_{a,b}\in SUR} s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)}\lambda(1-\delta)$$

$$B = \frac{s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)}{\sum_{\forall x_{a,b}: x_{a,b}\in SIR} s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)}(1-\lambda)(1-\delta)$$

$$C = \frac{s_{\mathbf{ui}}(x_{k,m}, x_{a,b})}{\sum_{\forall x_{a,b}: x_{a,b}\in SUIR} s_{\mathbf{ui}}(x_{k,m}, x_{a,b})}\delta$$

where A,B and C act as the weights to combine the predictors from three different sources. Unifying them we can obtain Eq. 16.