Novel Significance Weighting Schemes for Collaborative Filtering: Generating Improved Recommendations in Sparse Environments

Mustansar Ali Ghazanfar and Adam Prugel-Bennett
School of Electronics and Computer Science, University of Southampton,
Highfield Campus, SO17 1BJ, United Kingdom
email: {mag208r, adp}@ecs.soton.ac.uk

Abstract— *Collaborative filtering is the most famous and adopted recommendation algorithm, which recommends items by identifying other similar users, in case of user-based collaborative filtering, or similar items, in case of item-based collaborative filtering. Significance weighting schemes assign different weights to neighboring users/items found against an active user/item. In this paper, we claim that the significance weighting schemes proposed in the literature, are flawed by the fact that they can not be applied to general recommender system datasets. We provide the correct significance weighting schemes using different novel heuristics, and by extensive experimental results on two different datasets, show how significance weighting schemes affect the performance of a recommender system. Furthermore, we claim that the conventional weighted sum prediction formula used in item-based collaborative filtering is not correct for very sparse datasets. We provide the correct prediction formula and empirically evaluate it.*

Keywords: Recommender Systems; Collaborative Filtering; Significance Weighting Schemes; Heuristics

1. Introduction

There has been an exponential increase in the volume of available digital information, electronic sources, and online services in recent years. This information overload has created a potential problem—how to filter and efficiently deliver relevant information to a user—which highlights a need for information extraction systems that can filter unseen information and can predict whether a user would like a given source. Such systems are called *recommender systems*, and they mitigate the aforementioned problems to a great extent. Given a new item, recommender systems can predict whether a user would like this item or not, based on user preferences (likes—positive examples, and dislikes—negative examples), observed behavior (explicit or implicit feedback), and information (demographic or content information) about items.

Collaborative filtering (CF) recommender systems are the most widely used recommender systems, which recommend items by taking into account the taste (in terms of preferences of items) of users, under the assumption that users will be interested in items that users similar to them have rated highly. Examples of these systems include Amazon's recommender system [1], Ringo[1], etc. Collaborative filtering can be classified into two sub-categories: memory-based CF and model-based CF. Memory-based approaches make a prediction by taking into account the entire collection of previous rated items by a user, for example the GroupLens recommender system [2]. Model-based approaches use rating patterns of users in the training set, group users into different classes, and use ratings of predefined classes to generate recommendation for an *active user*[2] on a *target item*[3]. Some examples of these approaches include item-based CF [3], Singular Value Decomposition (SVD) based models [4], factorization methods [5], and clustering methods [6].

Recommendations can be presented to an active user in the followings two different ways: by predicting ratings of items a user has not seen before and by constructing a list of items ordered by his preferences. In the former case, an active user provides the prediction engine with the list of items to be predicted, prediction engine uses other users (or items) ratings or content information, and then predict how much the user would like the given item in some numeric or binary scale. In the latter case, different heuristics are used for producing an ordered list of items, sometimes termed as *top-N recommendations* [7], [8]. For example, in collaborative filtering recommender system this list is produced by making the rating predictions of all items an active user has not yet rated, sorting the list, and then keeping the top-N items the active user would like the most. In this paper, we focused on the former case—predicting the ratings of items—however we can easily construct a list of top-N items for each user by selecting highly predicted items.

1.1 Problem Statement

CF works by finding the similar users (in case of user-based CF) or similar items (in case of item-based CF). In case of user-based CF, significance weighting schemes give more weight (similarity) to a user who has co-rated a number of items with the active user. Similarly, in case of item-based CF, significance weighting schemes give more weight

[1] www.ringo.com
[2] The user for whom the recommendations are computed.
[3] The item a system wants to recommend.

(similarity) to an item which shares a number of common users with the target item.

We claim that the schemes presented in [9], [10], [11], [12] can not be generalized to real world recommender systems. We provide the correct significance weighting scheme and propose different novel heuristics for it. We empirically show that the proposed schemes outperform others in terms of accuracy metrics and coverage, especially in case of a very sparse dataset. We focused on item-based CF, and evaluate our algorithm on MovieLens[4] and FilmTrust[5] datasets.

The rest of the paper has been organized as follows. Section 2 discusses the background concepts. Section 3 presents a detailed overview of the significant weighting schemes proposed in the literature, under different conditions. Section 4 presents our proposed significance weighting schemes. Section 5 describes the dataset and metrics used in this work. Section 6 compares the performance of the proposed schemes with the existing ones followed by the discussion of results in section 7. Finally, section 8 concludes the work.

2. Background

Let $M = \{ m_1, m_2, \cdots, m_x \}$ be the set of all users, $N = \{ n_1, n_2, \cdots, n_y \}$ be the set of all possible items that can be recommended, r_{m_i,n_j} be the rating of user m_i on item n_j, and $|R_a \cap R_b|$ be the common ratings between two profiles.

2.1 Item-Based Collaborative Filtering

Item-based CF [3] builds a model of item similarities using an off-line stage. Suppose we want to make prediction for an item n_t for an active user m_a. Let Mn_in_j be the set of all users, who have co-rated item n_i and n_j. There are three main steps in this approach as follows:

- In the first step, all items rated by an active user are retrieved.
- In the second step, target item's similarity is computed with the set of retrieved items. A set of K most similar items $n_1, n_2 \cdots n_K$ with their similarities (or weights) $w_{n_1}, w_{n_2} \cdots w_{n_K}$ are selected. Similarity w_{n_i,n_j}, between two items n_i and n_j, is computed by first isolating the users who have rated these items (i.e. Mn_in_j), and then applying the adjusted cosine similarity [3].
- In the last step, prediction for the target item is made by computing the weighted average of the active user's rating on the K most similar items. Using weighted sum, the prediction p_{m_a,n_t} on item n_t for user m_a is computed as follows [3]:

$$P_{m_a,n_t} = \frac{\sum_{i=1}^{K}(w_{n_t,i} \times r_{m_a,i})}{\sum_{i=1}^{K}(|w_{n_t,i}|)} . \quad (1)$$

2.2 Significance Weighting Schemes

The similarity between two users can be misleading if they have rated very few items in common, and the same is true for similarity between items, which have been rated by very few users. For example, two items can have similarity of 1, if they have been identically rated by only two users, which is not true. Significance weighting schemes overcome this problems by, decreasing the similarity if the rating profiles of two user/items have very few ratings in common (denoted by $|R_a \cap R_b| < \alpha$ in this work), and enhancing the similarity between two rating profiles if they have a sufficient number of ratings in common (denoted by $|R_a \cap R_b| \geq \alpha$ in this work).

3. Item-Based CF: Significance Weighting Schemes

Several significance weighting schemes have been proposed[6], such as [9], [10], [11], [12]. We claim they can not be generalized to all similarities weights and datasets.

3.1 Case Amplification (CA) Proposed in [9]

Case amplification refers to the transformation that emphasizes weights that are closer to one, and punishes the lower ones.

$$w'_{a,b} = \begin{cases} w^p_{a,b} & if\ w_{a,b} \geq 0\,, \\ -(-w^p_{a,b}) & if\ w_{a,b} < 0. \end{cases} \quad (2)$$

Where $w_{a,b}$ is the similarity between user a and b, and p is a constant whose value is application dependent[7]. We notice that, the authors did not take the number of items in common between two users while calculating the similarity, hence similarity weights may be misleading.

3.2 Significance Weighting Scheme Proposed in [12]

In [12], the authors proposed a significance weighting scheme, claiming it devalues similarity weights that are based on a small number of co-rated items between two

[4] www.grouplens.org/node/73

[5] www.filmtrust.com

[6] These schemes focused on user-based CF using Pearson correlation [9], however we can apply them to item-based CF using adjusted cosine similarity [3] as well. The reason is these schemes emphasizes or punish similarities weights, which lies in range of $\{+1,-1\}$ for Pearson correlation and adjusted cosine similarity.

[7] It must be noted that the given formula is incorrect for negative weights, and the correct formula is: $w'_{a,b} = -(-w_{a,b})^p$. We reckon this is a typo and not an error.

users. The proposed similarity weighting scheme can be represented as follows:

$$w'_{a,b} = \frac{\max(|R_a \cap R_b|, \alpha)}{\alpha}.w_{a,b}. \quad (3)$$

It was corrected in [11], by replacing the Max by Min. We claim that both approaches are flawed by the fact that they can not be generalized. Let us take $\alpha = 50$[8], and validate our hypothesis.

3.2.1 For $w_{a,b} > 0$

Let us assume we have $w_{a,b} = 0.5$, $|R_a \cap R_b| = 60, 40$.

$$w'_{a,b} = \begin{cases} \frac{\max(60,50)}{50}.(0.5) = \frac{60}{50}.(0.5) > 0.5 \\ \qquad for |R_a \cap R_b| = 60\,, \\ \frac{\max(40,50)}{50}.(0.5) = \frac{50}{50}.(0.5) = 0.5 \\ \qquad for |R_a \cap R_b| = 40\,. \end{cases}$$

We can see that it is not decreasing cases, where the common items between two users are less than a threshold (α). This was pointed out in [11]. We show this formula is also not correct for negative correlations as shown below.

3.2.2 For $w_{a,b} < 0$

Let us assume we have $w_{a,b} = -0.5$, $|R_a \cap R_b| = 60, 40$.

$$w'_{a,b} = \begin{cases} \frac{\max(60,50)}{50}.(-0.5) = \frac{60}{50}.(-0.5) < -0.5 \\ \qquad for |R_a \cap R_b| = 60\,, \\ \frac{\max(40,50)}{50}.(-0.5) = \frac{50}{50}.(-0.5) = -0.5 \\ \qquad for |R_a \cap R_b| = 40\,. \end{cases}$$

The results shows that, this formula is decreasing weights in cases where the common items between two users is greater than a threshold and leaving it as it is otherwise, which is not correct.

3.3 Significance Weighting Scheme Proposed in [11]

In [11], the authors proposed a significance weighting scheme, claiming it corrects the flaw in significance weighting scheme presented in [12], which can be represented as follows:

$$w'_{a,b} = \frac{\min(|R_a \cap R_b|, \alpha)}{\alpha}.w_{a,b}. \quad (4)$$

Let us evaluate it for negative weights.

[8] Any other value can be assumed, 50 is a reasonable choice in MovieLens dataset.

3.3.1 For $w_{a,b} < 0$

Let us assume we have $w_{a,b} = -0.5$, $|R_a \cap R_b| = 60, 40$.

$$w'_{a,b} = \begin{cases} \frac{\min(60,50)}{50}.(-0.5) = \frac{50}{50}.(-0.5) = -0.5 \\ \qquad for |R_a \cap R_b| = 60\,, \\ \frac{\min(40,50)}{50}.(-0.5) = \frac{40}{50}.(-0.5) > -0.5 \\ \qquad for |R_a \cap R_b| = 40. \end{cases}$$

The results shows that, this formula is increasing weights in cases where the common items between two users is less than a threshold and leaving it as it is otherwise, which is not correct.

3.4 Significance Weighting Scheme Proposed in [10]

In [10], the author proposed a significance weighting scheme, which can be represented as follows:

$$w'_{a,b} = \frac{(|R_a \cap R_b|)}{\alpha}.w_{a,b}. \quad (5)$$

We found it reasonably good for $w_{a,b} > 0$, but it is wrong for $w_{a,b} < 0$.

3.4.1 For $w_{a,b} < 0$

Let us assume we have $w_{a,b} = -0.5$, $|R_a \cap R_b| = 60, 40$.

$$w'_{a,b} = \begin{cases} \frac{60}{50}.(-0.5) = \frac{6}{5}.(-0.5) < -0.5 \\ \qquad for |R_a \cap R_b| = 60\,, \\ \frac{40}{50}.(-0.5) = \frac{4}{5}.(-0.5) > -0.5 \\ \qquad for |R_a \cap R_b| = 40. \end{cases}$$

We observe that if two user have less items in common, it is increasing the weight, and decreasing the weights otherwise, which is obviously not correct.

4. Proposed Significance Weighting Scheme and Heuristics

4.1 Correct General Significance Weighting Scheme

We claim a general significance weighting scheme should have the following properties:

$$w'_{a,b} \begin{cases} > w_{a,b} & : \forall_{w_{a,b} \geq 0 \wedge |R_a \cap R_b| \geq \alpha}\,, \\ < w_{a,b} & : \forall_{w_{a,b} \geq 0 \wedge |R_a \cap R_b| < \alpha}\,, \\ > w_{a,b} & : \forall_{w_{a,b} < 0 \wedge |R_a \cap R_b| \geq \alpha}\,, \\ \gg w_{a,b} & : \forall_{w_{a,b} < 0 \wedge |R_a \cap R_b| < \alpha}. \end{cases} \quad (6)$$

Intuitively, we emphasize positive weight which are close to one and punish the remaining ones. For negative weights, we take opposite of what is communicated by the weight. A weight is punished more (shown by $\gg$ in equation 6) when common ratings in two profiles are less than α, than the case where, common ratings in two profiles are greater than

α. Correct, generalized form (denoted by $Proposed_G$ in the results, where *G* is for General) is given by:

$$w'_{a,b} = \begin{cases} \frac{(|R_a \cap R_b|)}{\alpha}.w_{a,b} & : \forall_{w_{a,b} \geq 0}\,, \\ \frac{|R_a \cap R_b|}{\alpha + \max(|R_a \cap R_b|, \alpha)}.w_{a,b} & : \forall_{w_{a,b} < 0}. \end{cases} \quad (7)$$

4.2 Novel Heuristics for Significance Weighting Scheme

4.2.1 Adding 1 to all similarity weights

$$w'_{a,b} = (w_{a,b} + 1) \ : \forall_{w_{a,b} \in \{-1,+1\}}. \quad (8)$$

This heuristic (denoted by $Sim + 1$ in the results) will take advantage of negatively correlated weights as well, however, the positively correlated neighbors would heavily affect the prediction rather than negatively correlated ones[9].

4.2.2 Adding 1 to all similarity weights and decreasing based on common ratings

$$w'_{a,b} = \frac{|R_a \cap R_b|}{\alpha + |R_a \cap R_b|}(w_{a,b} + 1) \ : \forall_{w_{a,b} \in \{-1,+1\}}. \quad (9)$$

This heuristic (denoted by Dev_{Sim+1}) decreases weights computed by equation 8.

4.2.3 Considering only positive similarity weights and decreasing based on common ratings

$$w'_{a,b} = \frac{|R_a \cap R_b|}{\alpha}(w_{a,b}) \ : \forall_{w_{a,b} \geq 0}. \quad (10)$$

This heuristic (denoted by Dev_{PosSim}) only takes into account the positive weights and decreases them based on the common ratings.

4.2.4 Adding constant to negative similarity weights and decreasing weights

$$w'_{a,b} = \frac{(w_{a,b} + 1)}{10} \ : \forall_{w_{a,b} < 0}. \quad (11)$$

This heuristic (denoted by Div_{10} in the results) will map negative similarities[10] into the range of 0 (in case $w_{a,b} = -1$ to approximately 0.1 (in case $w_{a,b} = -0.01$), which sound reasonable. In this way, we can meaningfully take into consideration the information encoded in the negatively correlated weights. We further, decrease this heuristic's similarities using common ratings (not shown), however, it does not result in significant improvement.

[9] $\forall_{w_{a,b} \in \{-1,+1\}}$ denotes the range a weight can lie in.

[10] Unless specified, for the positive similarities, equation 7 was used.

4.2.5 Taking log of negative similarity weights

We take the log (base 10) of negative similarities as follows:

$$w'_{a,b} = \log(w_{a,b} + 2) \ : \forall_{w_{a,b} < 0}. \quad (12)$$

Let us evaluate this heuristic (denoted by log in the results) by assuming $w_{a,b} = -0.01, -0.5, -0.99, -1$.

$$w'_{a,b} = \begin{cases} \log(w_{a,b} + 2) = \log(-0.01 + 2) = 0.299\,, \\ \log(w_{a,b} + 2) = \log(-0.5 + 2) = 0.176\,, \\ \log(w_{a,b} + 2) = \log(-0.99 + 2) = 0.004\,, \\ \log(w_{a,b} + 2) = \log(-1 + 2) = 0. \end{cases}$$

We observe that it is reasonably mapping negative weights to positive ones, i.e. a highly negative weight is punished more (shown by 0) than others.

4.2.6 Taking log of negative similarity weights and decreasing based on common ratings

We take log of negative similarities, and further decrease them using a heuristic (denoted by Log_{Dev} in the results) as follows:

$$w'_{a,b} = \frac{|R_a \cap R_b|}{|R_a \cap R_b| + \alpha}(\log(w_{a,b} + 2)) \ : \forall_{w_{a,b} < 0}. \quad (13)$$

Let us take the case from equation 12 where $w_{a,b} = -0.5$. Assume we have $\alpha = 50$, and $|R_a \cap R_b| = 10, 60$.

$$w'_{a,b} = \begin{cases} \frac{10}{10+50}.(0.176) = 0.029 \ for |R_a \cap R_b| = 10, \\ \frac{60}{60+50}.(0.176) = 0.096 \ for |R_a \cap R_b| = 60. \end{cases}$$

We find, it is further punishing weights as described in equation 6.

Another heuristic (denoted by Log_{CA}) is to apply CA (Case Amplification) over log similarities, as follows:

$$w'_{a,b} = \begin{cases} (\log(w_{a,b} + 2))^3 & : \forall_{w_{a,b} < 0 \wedge |R_a \cap R_b| < \alpha}, \\ (\log(w_{a,b} + 2))^2 & : \forall_{w_{a,b} < 0 \wedge |R_a \cap R_b| \geq \alpha}, \end{cases} \quad (14)$$

where, a similarity is punished more when $|R_a \cap R_b| < \alpha$ as compared to when $|R_a \cap R_b| >= \alpha$.

Finally, a heuristic (denoted by Log_{Max} in the results), which gave us reasonable results is given below:

$$w'_{a,b} = \frac{|R_a \cap R_b|}{\max(|R_a \cap R_b|, \alpha) + |R_a \cap R_b|}(\log(w_{a,b} + 2)) \ : \forall_{w_{a,b} < 0}. \quad (15)$$

4.3 Shortcomings in the Prediction Formula of Item-based CF Proposed in [3]

We claim that the the weighted sum prediction formula (see equation 1) proposed in [3] and used in [13], [14], [15], [16], [17] can not be generalized to very sparse datasets. If most of the item-item similarities are negative, then it would result in negative prediction, which is not correct. This formula can be corrected, by using the *adjusted weighted sum* that considers the deviation of ratings from the average rating of the active user.

$$P_{m_a,n_t} = \bar{r}_{m_a} + \frac{\sum_{i=1}^{K}(w_{n_t,i} \times \hat{r}_{m_a,i})}{\sum_{i=1}^{K}(|w_{n_t,i}|)}, \qquad (16)$$

where, $\hat{r}_{m_a,i} = r_{m_a,i} - \overline{r}_{m_a}$.

5. Experimental Evaluation

5.1 Dataset

We used MovieLens (SML) and FilmTrust (FT) datasets for evaluating our algorithm. SML dataset contains 943 users, 1682 movies, and $100\,000$ ratings on an integer scale 1 (bad) to 5 (excellent). It has been used in many research projects [3], [4], [18], [14]. The sparsity of this dataset is $93.7\%\Big(1 - \frac{\textit{non zero entries}}{\textit{all possible entries}} = 1 - \frac{100000}{943\times1682} = 0.937\Big)$.

We created the second dataset by crawling the FilmTrust website. The dataset retrieved (on 10th of March 2009) contains 1592 users, 1930 movies, and $28\,645$ ratings on a floating point scale of 1 (bad) to 10 (excellent). The sparsity of this dataset is 99.06%[11].

5.2 Metrics

Several metrics have been used for evaluating recommender systems which can broadly be categorized into *Predictive Accuracy Metrics*, *Classification Accuracy Metrics*, and *Rank Accuracy Metrics* [19]. The Predictive Accuracy Metrics measure how close is the recommender system's predicted value of a rating, with the true value of that rating assigned by the user. These metrics include mean absolute error, mean square error, and normalized mean absolute error, and have been used in research projects such as [9], [20], [7], [3]. The Classification Accuracy Metrics determine the frequency of decisions made by a recommender system, for finding and recommending a good item to a user. These metrics include precision, recall, $F1$ measure, and Receiver Operating Characteristic curve, and have been used in [7], [21]. The last category of metrics, Rank Accuracy Metrics measure the proximity between the ordering predicted by a recommender system to the ordering given by the actual user, for the same set of items. These metrics include half-life utility metric proposed by Brease [9]. Our specific task in this paper is to predict scores for items that already have been rated by actual users, and to check how well this prediction helps users in selecting high quality items. Keeping this into account, we use *Mean Absolute Error (MAE)* and *Receiver Operating Characteristic (ROC) sensitivity*.

[11] All dataset can be downloaded from: https://sourceforge.net/projects/hybridrecommend.

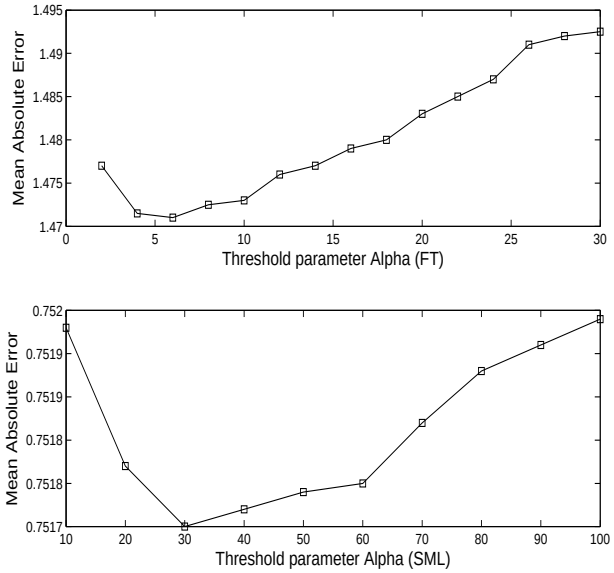

Fig. 1: Determining the Optimal Value of Threshold Parameter α.

MAE measures the average absolute deviation between a recommender system's predicted rating and a true rating assigned by the user. It is computed as follows:

$$|\overline{E}| = \frac{\sum_{i=1}^{T}|p_i - a_i|}{T},$$

where p_i and a_i are the predicted and actual values of a rating respectively, and T is the total number of test samples[12] in the test set. It has been used in [9], [3], [14], [18], [9], [7], [13], [4], [22].

ROC is the extent to which an information filtering system can distinguish between good and bad items. *ROC sensitivity* measures the probability with which a system accept a good item. The ROC sensitivity ranges from 1 (perfect) to 0 (imperfect) with 0.5 for random. To use this metric for recommender systems, we must first determine which items are good (*signal*) and which are bad (*noise*). We consider an item good if a user rated it with a score higher than his average (in the training set) and bad otherwise as used in [18].

[12] A test sample consists of a tuple, $< uid, mid, rating >$, and the objective is to predict the ratings assigned by the actual users.

Table 1: A comparison of proposed algorithm with existing in terms of accuracy metrics, and coverage. Average and standard deviation of the results, over 5 folds, is shown. The best results are shown in bold. We observe that, the proposed scheme ($Proposed_G$, given in equation 7) outperform others significantly over FilmTrust dataset.

Scheme	Best MAE		ROC-Sensitivity		Coverage	
	(SML)	(FT)	(SML)	(FT)	(SML)	(FT)
No	0.808 ± 0.004	1.639 ± 0.017	0.534 ± 0.003	0.563 ± 0.005	99.725 ± 0.030	91.873 ± 0.300
$\frac{\lvert R_a \cap R_b \rvert}{\alpha}$	$\mathbf{0.747 \pm 0.003}$	1.602 ± 0.019	0.784 ± 0.006	0.579 ± 0.005	$\mathbf{99.744 \pm 0.020}$	92.040 ± 0.241
CA	0.844 ± 0.004	1.642 ± 0.016	0.506 ± 0.004	0.589 ± 0.005	99.724 ± 0.031	91.201 ± 0.412
Max	0.749 ± 0.003	1.605 ± 0.022	0.735 ± 0.003	0.577 ± 0.036	$\mathbf{99.744 \pm 0.026}$	92.008 ± 0.283
Min	$\mathbf{0.747 \pm 0.003}$	1.610 ± 0.018	0.753 ± 0.003	0.575 ± 0.007	99.730 ± 0.021	91.855 ± 0.226
$Proposed_G$	$\mathbf{0.747 \pm 0.003}$	$\mathbf{1.485 \pm 0.005}$	$\mathbf{0.787 \pm 0.004}$	$\mathbf{0.608 \pm 0.005}$	$\mathbf{99.744 \pm 0.042}$	$\mathbf{94.110 \pm 0.312}$

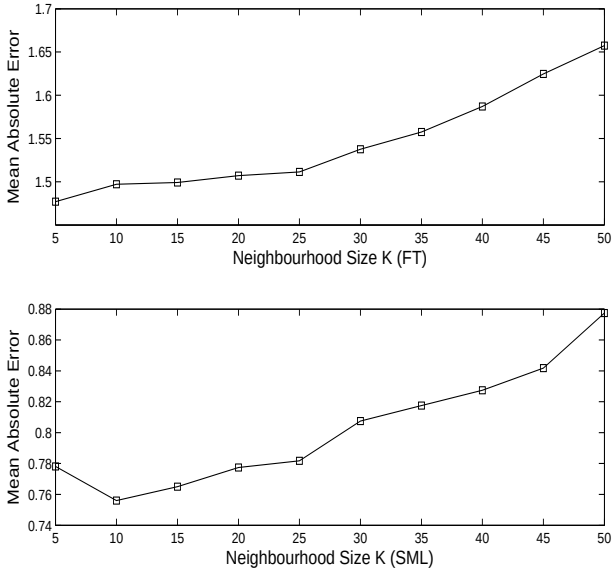

Fig. 2: Determining the Optimal Value of Neighborhood Size (K).

Furthermore, we used *coverage* that measures how many items a recommender system can make recommendation for. It has been used in [19], [14], [18].

6. Results

We performed the striated 5 fold cross validation [23] and reported the average results with standard deviation. Each distinct fold contains 20% ratings of each user as a test set and the remaining 80% ratings as a training set. We further subdivided our training set into a test set and training set for measuring the parameters sensitivity. For learning the parameters, we conducted 5-fold cross validation on the 80% training set, by selecting the different test and training set each time, and taking the average of the results.

6.1 Learning The Optimal Values of Parameters

6.1.1 Learning the optimal values of parameter α

We kept the neighborhood size fixed to 5 and 10 for FT and SML dataset respectively. We varied the value of α from 2 to 30 with a difference of 2 for FilmTrust, and from 10 to 100 with a difference of 10 for SML dataset and observed the corresponding change in MAE. Figure 1 shows how MAE changes with α for the proposed scheme. Similarly, we tuned all schemes for the best values of α[13].

6.1.2 Learning the optimal values of neighborhood size (K)

We varied the neighborhood size from 5 to 50 with a difference of 5, and observe the corresponding MAE, while keeping the optimal values of α. Figure 2 shows that 5, and 10 was found to be the best neighbor sizes for FT, and SML dataset respectively. Similarly, we tuned all algorithms for the best values of neighborhood size[14]. For the subsequent experiments, we used optimal values of α and K, for all schemes.

Table 2: Distribution of Positive and Negative Neighbors (FT dataset)

No. Of Neighbors	Similarities > 0 (%)	Similarities < 0 (%)
5	84.873	15.127
10	80.951	19.049
15	71.868	28.132
20	60.951	39.049
25	52.210	47.790
30	46.291	53.709
35	41.984	58.016
40	38.766	61.234
45	38.010	61.989
50	37.410	62.590

[13]They found to be in the range of $2-8$ for FT, and $10-30$ for SML dataset respectively.

[14]They found to be in the range of $5-10$ for FT, and $10-20$ for SML dataset respectively.

6.2 Results in Terms of MAE, ROC-Sensitivity, and Coverage

Table 1 shows the comparison of different schemes (CA was Proposed in [9], Max was Proposed in [12], Min was Proposed in [11], and $\frac{(|R_a \cap R_b|)}{\alpha}$ was Proposed in [10]) with proposed one (given in equation 7) in terms of MAE, ROC-sensitivity, and coverage. The results are statistically significant for FilmTrust dataset—10% improvement in terms of MAE, and 4% improvement in terms of ROC. Furthermore, its coverage is higher than any other scheme. The reason for good results is that, FT dataset is very sparse and most of the correlations among items are negative. Our scheme carefully assigns different significance weights to positive and negative similarities, which results in increased accuracy and coverage. To check the proportion of positive and negative neighbours against an active item, we show the distribution of similarities (in terms of positives and negatives) between a target item and other top K most similar neighbouring items. We compute similarity weights by adjusted cosine similarity and report the average results using 5-fold cross validation. We observe in table 2 that even in case of 5 neighbours against a target item, 15% of the similarities are negative.

Table 3: A Comparison of the proposed scheme with others, under varying neighborhood size (FT dataset). We observe that the performance of conventional significance weighting schemes degrades rapidly with the increase in the number of neighbors.

Scheme	Neigh. (K)	MAE	ROC-Sensitivity	Coverage
No		1.639	0.563	91.873
$\frac{\lvert R_a \cap R_b \rvert}{\alpha}$		1.602	0.579	92.040
CA	5	1.642	0.589	91.201
Max		1.605	0.577	92.008
Min		1.610	0.575	91.855
$Proposed_G$		1.485	0.608	94.110
No		2.033	0.470	85.896
$\frac{\lvert R_a \cap R_b \rvert}{\alpha}$		2.006	0.476	86.367
CA	10	1.871	0.535	83.906
Max		2.020	0.466	86.385
Min		2.024	0.471	85.634
$Proposed_G$		1.560	0.566	93.768
No		2.427	0.365	79.298
$\frac{\lvert R_a \cap R_b \rvert}{\alpha}$		2.436	0.372	80.118
CA	15	2.106	0.483	76.610
Max		2.431	0.368	80.013
Min		2.427	0.368	79.176
$Proposed_G$		1.623	0.516	93.436
No		2.971	0.192	73.363
$\frac{\lvert R_a \cap R_b \rvert}{\alpha}$		2.977	0.205	74.567
CA	20	2.977	0.205	74.567
Max		2.938	0.183	69.052
Min		3.022	0.192	72.595
$Proposed_G$		1.635	0.501	93.262

6.3 Comparison of Different Schemes Over FilmTrust Dataset

Table 3 shows the performance of item-based CF over FilmTrust dataset. We observe that the proposed scheme (given in equation 7) performs better than others, and that its performance does not degrade with the increase in the number of neighbors.

6.4 Comparison of Different Proposed Heuristics Over FilmTrust Dataset

Table 4: A comparison of different Heuristics (FT dataset)

Heuristic	MAE	ROC-Sensitivity	Coverage
NO	1.639	0.563	91.873
$Proposed_G$	1.485	0.608	94.110
$Sim+1$	1.44	0.609	95.100
Dev_{Sim+1}	1.430	0.612	95.100
Dev_{PosSim}	1.441	0.621	94.610
Div_{10}	1.422	**0.630**	**95.100**
Log	1.421	0.624	95.100
Log_{Dev}	**1.415**	0.628	**95.100**
Log_{CA}	1.417	0.627	95.100
Log_{Max}	1.432	0.614	95.100

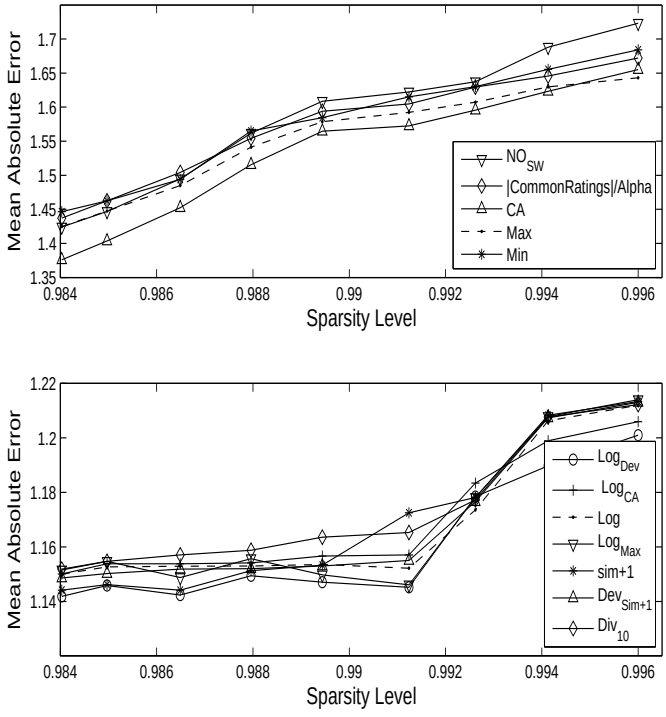

Fig. 3: The effect of Sparsity on MAE (SML dataset)

Table 4 presents a comparison of different proposed heuristics ($Proposed_G$ is given in equation 7, $Sim+1$ is given in equation 8, Dev_{Sim+1} is given in equation 9, Dev_{PosSim} is given in 10, Div_{10} is given in equation 11, $\log$ is given in equation 12, Log_{Dev} is given in equation

Table 5: The effect of sparsity on coverage (FT dataset)

Scheme	Sparsity Level	Coverage
No		57.598
$\frac{\lvert R_a \cap R_b \rvert}{\alpha}$		57.352
CA		57.725
Max		57.642
Min	0.986	57.333
All Heuristics		> 75.0
No		51.057
$\frac{\lvert R_a \cap R_b \rvert}{\alpha}$		50.875
CA		51.155
Max		51.076
Min	0.988	50.870
All Heuristics		> 70.0
No		42.009
$\frac{\lvert R_a \cap R_b \rvert}{\alpha}$		42.206
CA		41.744
Max		42.034
Min	0.990	41.740
All Heuristics		> 59.0
No		24.042
$\frac{\lvert R_a \cap R_b \rvert}{\alpha}$		24.086
CA		24.027
Max		24.061
Min	0.992	24.081
All Heuristics		> 35.0

13, Log_{CA} is given in equation 14, and Log_{Max} is given in equation 15) in terms of MAE, ROC-Sensitivity, and coverage. We performed pair-t test on predictions generated by item-based CF proposed in [3] and those of generated by applying different heuristics. The results (in terms of MAE) of pair-t test showed that the proposed heuristics are significantly better—at $> 99\%$ confidence interval—than the simple approach.

6.5 Performance Evaluation Under Different Sparsity Levels

We already showed the result in the case of FilmTrust dataset which is a good example for sparse datasets. To check the effect of sparsity on the MovieLens dataset, we increased the sparsity level of the SML training set by dropping some randomly selected entries, whereas, we kept the test set same for each sparse training set. Figure 3 shows that the performance of the conventional item-based CF and other schemes in literature degrades with the increase in the sparsity, however, the performance does suffer less in the case of proposed heuristics.

Table 5 shows how different approaches suffer in terms of coverage with the increase in sparsity. We observe that, the coverage of the conventional approaches degrade rapidly with the increase in sparsity, whereas, in the proposed heuristics, it drop gracefully with the increase in sparsity[15].

[15]We did not show results for the heuristic given in equation 10, because with the increase in sparisty, fewer positive neighbors are available, resulting in decreased MAE and coverage.

6.6 Comparison of Results of Weighted Sum and Adjusted Weighted Sum Prediction Formula

Table 6 shows the performance of item-based CF over FilmTrust dataset, using weighted sum and adjusted weighted sum formula for prediction. We observe that the adjusted cosine formula performs better than the weighted one and that with the increase in the number of neighbors, the performance of weighted one degrades. Hence, adjusted weighted sum formula should be used in item-based CF rather than the weights sum that has extensively been used in literature for item-based CF [13], [14], [15], [16], [17].

7. Discussion

We observe from the result section that significance weighting schemes taking negatively correlated neighbors into account gave better results from the rest. They give good results under sparse dataset.

It is worth noting that, if we use cosine similarity [9] as a measure of similarity between two item then similarity weights would lie in the range of $\{0, 1\}$, hence our heuristics can not give any advantage. To check the performance of item-based CF, we applied cosine similarity measure (with significance weighting schemes proposed in the literature), and the results, in terms of MAE, were not very promising in the case of FilmTrust dataset. The lowest MAE observed was 1.471 (with standard deviation of 0.022). Surprisingly, ROC was observed to be 0.68 (with standard deviation of 0.008), which is higher than any of the heuristics and schemes in case of adjusted cosine similarity. Further experiments are needed to analyze these results.

To check the performance of our heuristics when only neighbors having no or negative correlation with the target item are available, which can be the case, for example in *cold-start* scenarios [24]; we conducted experiment in the case of FilmTrust dataset, by applying adjusted cosine similarity with adjusted weighted sum formula. The results in general were insignificant as compared to the conventional approaches. We believe performance can be increased by taking data distribution into account, which is a subject of future work.

Based on the experimental results, we may conclude that the proposed heuristics are useful to make recommendations on highly sparse datasets.

8. Conclusion

We show that the significance weighting schemes for collaborative filtering presented in [9], [10], [11], [12] are flawed by the fact that they can not be generalized to all datasets. We propose various novel heuristics for significance weighting scheme and empirically evaluate them. Furthermore, we show that the conventional weighted sum

Table 6: A Comparison of Weighted Sum and Adjusted Weighted sum prediction formula (FT dataset)

No. Of Neighbors	Weighted Sum			Adjusted Weighted Sum		
	MAE	ROC-Sensitivity	Coverage	MAE	ROC-Sensitivity	Coverage
5	1.639	0.563	91.873	1.486	0.619	95.377
10	2.033	0.470	85.896	1.453	0.622	95.377
15	2.427	0.365	79.298	1.444	0.623	95.377
20	2.971	0.192	73.363	1.442	0.626	95.377
25	3.480	0.154	67.725	1.443	0.629	95.376
30	3.934	0.102	58.973	1.449	0.625	95.376

prediction formula [3] used in item-based CF is not correct, and provide the correct one.

Acknowledgment

The work reported in this paper has formed part of the Instant Knowledge Research Programme of Mobile VCE, (the Virtual Centre of Excellence in Mobile & Personal Communications), www.mobilevce.com. The programme is co-funded by the UK Technology Strategy Board's Collaborative Research and Development programme. Detailed technical reports on this research are available to all Industrial Members of Mobile VCE.

References

[1] J. Y. G Linden, B Smith, "Amazon.com recommendations: item-to-item collaborative filtering," in *IEEE, Internet Computing*, vol. 7, 2003, pp. 76–80.

[2] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "Grouplens: applying collaborative filtering to usenet news," *Commun. ACM*, vol. 40, no. 3, pp. 77–87, March 1997.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM New York, NY, USA, 2001, pp. 285–295.

[4] M. Vozalis and K. Margaritis, "Using SVD and demographic data for the enhancement of generalized collaborative filtering," *Information Sciences*, vol. 177, no. 15, pp. 3017–3037, 2007.

[5] R. Bell, Y. Koren, and C. Volinsky, "The BellKor solution to the Netflix prize," *AT&T Labs–Research: Technical report November*, 2007.

[6] Y. Park and A. Tuzhilin, "The long tail of recommender systems and how to leverage it," in *Proceedings of the 2008 ACM conference on Recommender systems*. ACM New York, NY, USA, 2008, pp. 11–18.

[7] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system–a case study," in *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*. Citeseer, 2000.

[8] S. Al Mamunur Rashid, G. Karypis, and J. Riedl, "ClustKNN: a highly scalable hybrid model-& memory-based CF algorithm," in *Proc. of WebKDD 2006: KDD Workshop on Web Mining and Web Usage Analysis, August 20-23 2006, Philadelphia, PA*. Citeseer.

[9] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering." Morgan Kaufmann, 1998, pp. 43–52.

[10] J. Herlocker, J. Konstan, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM New York, NY, USA, 1999, pp. 230–237.

[11] H. Ma, I. King, and M. Lyu, "Effective missing data prediction for collaborative filtering," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, p. 46.

[12] M. McLaughlin and J. Herlocker, "A collaborative filtering algorithm and evaluation metric that accurately model the user experience," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM New York, NY, USA, 2004, pp. 329–336.

[13] M. Vozalis and K. Margaritis, "Applying SVD on generalized item-based filtering," *International Journal of Computer Science and Applications*, vol. 3, no. 3, pp. 27–51, 2006.

[14] M. Ghazanfar and A. Prugel-Bennett, "A Scalable, Accurate Hybrid Recommender System," in *2010 Third International Conference on Knowledge Discovery and Data Mining*. IEEE, 2010, pp. 94–98.

[15] M. Vozalis and K. Margaritis, "On the combination of user-based and item-based collaborative filtering," *International Journal of Computer Mathematics*, vol. 81, no. 9, pp. 1077–1096, 2004.

[16] L. Candillier, F. Meyer, and M. Boullé, "Comparing state-of-the-art collaborative filtering systems," *Machine Learning and Data Mining in Pattern Recognition*, pp. 548–562, 2007.

[17] J. Wang, A. De Vries, and M. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2006, p. 508.

[18] M. Ghazanfar and A. Prugel-Bennett, "An improved switching hybrid recommender system using naive bayes classifier and collaborative filtering," in *The 2010 IAENG International Conference on Data Mining and Applications*, April 2010. [Online]. Available: http://eprints.ecs.soton.ac.uk/18483/

[19] L. G. T. Jonathan L. Herlocker, Joseph A. Konstan and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems (TOIS) archive*, vol. 22, pp. 734–749, 2004.

[20] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering," 2002.

[21] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of recommendation algorithms for e-commerce." ACM Press, 2000, pp. 158–167.

[22] D. Kim and B. Yum, "Collaborative filtering based on iterative principal component analysis," *Expert Systems with Applications*, vol. 28, no. 4, pp. 823–830, 2005.

[23] I. H. W. Witten and F. Eibe, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, October 1999.

[24] A. T. Gediminas Adomavicius, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 734–749, 2005.