Weighted Slope One Predictors Revisited

Danilo Menezes[1], Anisio Lacerda[2,3], Leila Silva[1], Adriano Veloso[2], Nivio Ziviani[2,3]

[1] Universidade Federal do Sergipe, Brazil
danilohfm@dcomp.ufs.br, leila@ufs.br

[2] Universidade Federal de Minas Gerais, Brazil
{anisio,adrianov,nivio}@dcc.ufmg.br

[3] Zunnit Technologies, Brazil
{anisio,nivio}@zunnit.com

ABSTRACT

Recommender systems are used to help people in specific life choices, like what items to buy, what news to read or what movies to watch. A relevant work in this context is the Slope One algorithm, which is based on the concept of differential popularity between items (i.e., how much better one item is liked than another). This paper proposes new approaches to extend Slope One based predictors for collaborative filtering, in which the predictions are weighted based on the number of users that co-rated items. We propose to improve collaborative filtering by exploiting the web of trust concept, as well as an item utility measure based on the error of predictions based on specific items to specific users. We performed experiments using three application scenarios, namely Movielens, Epinions, and Flixter. Our results demonstrate that, in most cases, exploiting the web of trust is benefitial to prediction performance, and improvements are reported when comparing the proposed approaches against the original Weighted Slope One algorithm.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Recommender systems, Slope One, Trust-aware, Collaborative filtering

1. INTRODUCTION

Recommender systems identify interesting items in situations where the number and complexity of possibilities outstrip the user's capability to survey them in order to reach a proper decision. Such complex situations are commonly observed currently, leveraging the success of large recommender systems, such as Amazon[1], Movielens[2], Flixster[3] and Netflix[4]. A dominant approach for recommender systems is collaborative filtering − a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption behind this method is that if a user A has the same opinion or taste as a user B on an issue, then A is more likely to have B's opinion on a different issue.

Lemire and Maclachlan [8] present an item-based collaborative algorithm called Slope One. This algorithm introduces the principle of differential popularity between items, and exploits this principle instead of using typical item similarity measures. The differential popularity measures how much an item is more popular than another. Also in [8], the authors present an extension of the Slope One algorithm, called Weighted Slope One, in which the predictions are weighted based on the number of users that co-rated items. Slope-One algorithms are widely recognized as being an extremely simple approach to item-based collaborative filtering based on ratings. This simplicity makes the algorithms especially easy to implement. Further, accuracy numbers provided by Slope-One algorithms is often on par with more complicated and computationally expensive algorithms. As a result, Slope-One algorithms have quickly gained the attention of both pratictioners and the research community.

In this work we propose five variants for the Weighted Slope One Algorithm. The Weighted Slope One algorithm performs predictions based on items already rated by the user and these items are always equally considered. Nevertheless, for a given user the predictions based on a specific item may be, on average, more accurate than the ones based on other items. Here we propose the use of a measure of personalized item prediction efficiency for users, which is named as *item usefulness*, as a component of the weighting system of the Weighted Slope One algorithm. Moreover, as in [4], we consider weights for users. However, the difference between our work and [4] resides in the way the weights are computed. We propose to exploit the concept of web of trust [9] to compute these weights. More specifically, we exploit the fact that users can also express their opinions concerning the reviews and ratings performed by other users. As a result, a user may be considered more reliable than others, and her predictions more valuable. The proposed algorithms showed to be, in most of the cases, at least as accurate as known collaborative filtering algorithms, with the advatages of being dynamically updatable and efficient at query time.

The remaining of this paper is organized as follows. In Section 2 we discuss related work. In Section 3 we introduce the basic concepts and the background necessary for the other parts of the paper. In Section 4 we present variants of

[1]www.amazon.com
[2]www.movielens.org
[3]www.flixter.com
[4]www.netflix.com

the Slope-One algorithm, which exploit the concept of item-usefullness and per-user trustiness. In Section 5 we present our experiments, which includes the description of datasets, evaluation metrics, and results. Finally, in Section 6, we present our conclusions and future research directions.

2. RELATED WORK

Collaborative recommendation is the most popular approach and traditional collaborative recommendation algorithms are based on user similarity to perform predictions [2, 6]. However, user-based collaborative algorithms face scalability problems when the number of users and items increase. Sarwar *et al.* [13] propose a collaborative approach based on item similarity instead of the traditional user similarity based algorithms. In this approach the authors propose that items are recommended to an user when other similar items are already rated by the user as good for him. The proposed approach showed to be more scalable and at the same time produces as good recommendations as user-based collaborative algorithms.

Gao *et. al* [4] claim that some user's recommendations are more important than others. Thus, for item-based collaborative filtering recommendations, including Slope One, the users should be weighted to produce better recommendations. In order to solve this problem they propose an user-rank approach to compute relative weights for users based on their ratings.

There are also other variants of the Slope One algorithm in the literature. For example, Wang and Ye [14] propose an algorithm based on Slope One and user-based collaborative filtering to improve the recommendation performance of the second one. This approach addresses the issue by using the Slope One to fill the missing ratings and then using the user-based collaborative filtering to produce recommendations. Zhang [15] applies the Slope One algorithm in the same way as [14]. The difference between both approaches is that [15] uses an item-based collaborative algorithm to perform the recommendations. Gao and Wu [3] incorporate personalized contextual information into Slope One to improve recommendation results.

3. BACKGROUND

The main concept introduced by Lemire and Maclachlan [8] is the principle of differential popularity to predict item ratings. Consider two users A and B and two items I and J as presented in Figure 1. User A gave item I rating 1, whereas user B gave rating 2. For item J, user A gave rating 1.5. The idea of the differential popularity is to compute the difference between ratings of item I and J for user A and to apply this difference to predict item J for user B. Thus, as the difference between the items is 0.5 for user A, it is inferred that the rating of item J for user B is 2.5 to preserve the same difference.

Generalizing, consider U and S the set of all users and items in the system, respectively. The differential popularity, also called *deviation*, can be calculated as follows. Consider $r_{u,i}$ and $r_{u,j}$ the ratings given by user $u \in U$ to the items $i, j \in S$. Clearly, $u \in S_{i,j}$, where $S_{i,j}$ is the set of all users who co-rated both items i and j. By considering all items of S, the deviation matrix dev is:

$$dev_{i,j} = \frac{\sum_{u \in S_{i,j}} (r_{u,i} - r_{u,j})}{|S_{i,j}|} \quad (1)$$

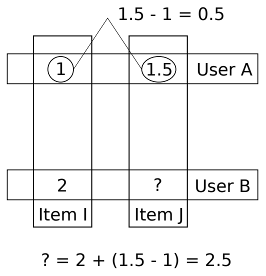

Figure 1: Example of differential popularity to predict item ratings.

The computation of the Weighted Slope One deviation matrix s illustrated in Algorithm 1.

Algorithm 1 Computation of Weighted Slope One deviation matrix.

Input: U: set of all users, S: set of all items, R: set of all rated items
Output: dev: deviation matrix, $Freq$: co-rating frequency matrix

```
1: for all $u \in U$ do
2:     for all $s \in R_u$ do
3:         for all $s' \in R_u$ do
4:             $dev_{s,s'} \leftarrow dev_{s,s'} + (r_{u,s} - r_{u,s'})$
5:             $Freq_{s,s'} \leftarrow Freq_{s,s'} + 1$
6: for all $s \in S$ do
7:     for all $s' \in S$ do
8:         $dev_{s,s'} \leftarrow dev_{s,s'} / Freq_{s,s'}$
```

Using this matrix and the users' set of ratings, the prediction for item i of user u, $p_{u,i}$, is calculated as:

$$p_{u,i} = \frac{\sum_{j \in C_i} (dev_{i,j} + r_{u,j})}{|C_i|} \quad (2)$$

where C_i is the set of all items rated by u and that have already been co-rated with i by at least another user in U. Formally, $C_i = \{j/j \in R_u, j \neq i, |S_{i,j}| > 0\}$, where R_u is the set of items rated by user u.

Equation 2 does not take into account the number of times items i and j have been co-rated. To incorporate this feature in the algorithm the predictions are perfomed as follows:

$$p_{u,i} = \frac{\sum_{j \in R_u} (dev_{i,j} + r_{u,j}) \times |S_{i,j}|}{\sum_{j \in R_u} |S_{i,j}|} \quad (3)$$

The algorithm that uses Equation 3 instead of Equation 2 to perform the predictions is called Weighted Slope One [8]. This algorithm has better performance when compared with the Slope One algorithm and is used as the baseline algorithm in our experiments. The computation of the Weighted Slope One predictions is illustrated in Algorithm 2.

Algorithm 2 Weighted Slope One predictions.

Input: u: target user of the prediction, s: item to be predicted, R_u: set of all items rated by user u, dev: deviation matrix, $Freq$: co-rating frequency matrix
Output: $p_{u,s}$: prediction of item s for user u

```
1: for all $s' \in R_u$ do
2:     $p_{u,s} \leftarrow p_{u,s} + (r_{u,s'} + dev_{s',s}) \times Freq_{s',s}$
3:     $TotalWeights \leftarrow TotalWeights + Freq_{s',s}$
4: $p_{u,s} \leftarrow p_{u,s} / TotalWeights$
```

In the following sections we present five extensions to the Weighted Slope One algorithm. In Section 4.1, we discuss two extensions based on the concept of web of trust. In Section 4.2, we present two extensions that use the measure of item-usefulness values based on the mean absolute error produced by the predictions for each user. In Section 4.3, we present a combination of the mentioned extensions.

4. WEIGHTED SLOPE-ONE REVISITED

In this section we discuss variations of the Weighted Slope-One algorithm. Such variations incorporate concepts such as item usefulness and per-user trustiness.

4.1 Per-User Trustiness Approaches

Next we present two extensions for the Weighted Slope One algorithm based on the web of trust concept. The difference between the two extensions is the way the user weight (called here *Userweight*) is computed. Both approaches are based on the concept of web of trust, which has already been used to improve collaborative filtering algorithms [9].

The web of trust is the set of all users and their trust statements, where a trust statement given by an user u to another user v means that user u trusts user v. The trust statements can be represented as edges in a digraph connecting the users (vertices); this digraph represents the web of trust. Figure 2 shows a simple web of trust consisting of four users. In the example, user A trusts users B and C and these two trust D.

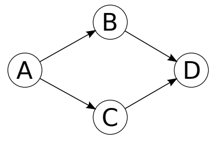

Figure 2: Example of a simple web of trust graph.

Equation 1 does not take into account who is the user that produced the deviation and so the deviations of all users are equally considered. We propose that the deviations are weighted according to the user that produces it, using as hypotesis the fact that some users represent better the community deviation than others. Therefore, we introduce the following equation to compute the deviation matrix, rather than Equation 1:

$$dev_{i,j} = \frac{\sum_{u \in S_{i,j}} (r_{u,i} - r_{u,j}) \times Userweight_u}{\sum_{u \in S_{i,j}} Userweight_u} \qquad (4)$$

where $Userweight_u$ is a weight given to each user $u \in U$, which measures how well its behavior represents the entire community of users. In Algorithm 1, lines 4, 5 and 8 are replaced by the following ones, respectively:

$$dev_{s,s'} = dev_{s,s'} + (r_{u,s} - r_{u,s'}) \times Userweight_u;$$
$$TotalWeight_{s,s'} = TotalWeight_{s,s'} + Userweight_u;$$
$$dev_{s,s'} = dev_{s,s'} / TotalWeight_{s,s'};$$

As the web of trust can be modelled as a digraph, graph algorithms can be used to compute the relevance of each user (vertex) for the community, as we explain in Sections 4.1.1 and 4.1.2.

4.1.1 Slope One Algorithm Weighted by Input Degree (IDBSO)

The first proposed variant is called Input Degree Based Slope One (IDBSO). This approach is based on the input degree of the vertex to compute *Userweight*. Our hypotesis relies on the principle that if an user u receives more trust statements than another user u', u's deviations should represent better the community deviations than u''s. Therefore, the *Userweight* for a given user u, denoted by $Userweight_u$, is computed in this approach by the following equation:

$$Userweight_u = |B_u| + 1 \qquad (5)$$

where B_u is the set of all users that have given a trust statement towards u. In the case u user has never received a trust statement, $|B_u| = 0$ and $Userweight_u = 1$, avoiding a null value for the weight of user u.

4.1.2 Slope One Algorithm Weighted by PageRank (PRBSO)

The second proposed variant is called PageRank Based Slope One (PRBSO). The computation of *Userweight* In this approach is based on the PageRank algorithm introduced in [7], which aims to compute the importance of web pages, based on the digraph where the web pages are vertices and the connections made by the hyperlinks are edges.

The algorithm is based on the principle that a vertex has a high rank if the sum of the ranks of vertices that point to it, called by the authors as *backlinks*, is high. This means that a rank can be high by either having many low ranked backlinks or few high ranked ones. Thus, this algorithm is a bit more sofisticated than the previous one, based on the input degree, because it also considers the rank of the backlinks and not only its amount.

The value of *Userweight* for a given user u, denoted by $Userweight_u$, is computed in this approach by the following equation:

$$UserWeight_u = (1 - d) + d \times \left(\sum_{v \in B_u} \frac{rank_v}{|C_v|} \right) \qquad (6)$$

where C_v is the set of all vertices that v points to and d is called the *damping factor*, which guarantees that no user will remain without rank, what would make the user's deviations to be totally disconsidered. The value of d used in the experiments is 0.15, the same value used in [7].

Equation 6 is recursive and is used until a steady state is reached. A steady state is a configuration where, even after performing Equation 6 again, the values of *Userweight* will not change. In the experiments the initial values of *Userweight* for all users is 1.

4.2 Item-Usefulness Based Approaches

In this section we present two variants of the Weighted Slope One algorithm, based on two different ways of generating the *ItemUsefulness* matrix.

Although Equation 3 takes into account the number of users who co-rated both items, from the user point of view all items are equally considered to perform the prediction. We propose an aditional criteria to weight the predictions for a given user u, using as hypotesis the fact that predictions based on an item s may be more precise than the ones based on another item s'. Therefore, the prediction equation is modified to:

$$p_{u,i} = \frac{\sum_{j \in R_u} (dev_{i,j} + r_{u,j}) \times |S_{i,j}| \times \mathit{ItemUsefulness}_{u,j}}{\sum_{j \in R_u} (|S_{i,j}| \times \mathit{ItemUsefulness}_{u,j})} \quad (7)$$

where $\mathit{ItemUsefulness}_{u,s}$ is a matrix that measures, for a given user u, how precise are the predictions based on item s. This extension introduces changes to Algorithm 2, where lines 3 and 4 must be replaced by the following ones, respectively:

$$p_{u,s} = (r_{u,s'} + dev_{s',s}) \times Freq_{s',s} \times \mathit{ItemUsefulness}_{u,s'};$$
$$TotalWeights = TotalWeights + Freq_{s',s} \times \mathit{ItemUsefulness}_{u,s'};$$

We propose an approach to measure the $\mathit{ItemUsefulness}_{u,s}$ based on how accurate the predictions based on item s are for user u. In order to do it, we use the Mean Absolute Error (MAE) accuracy metric [1], which is calculated as follows:

$$MAE = \frac{\sum_{(u,s) \in T} (|p_{u,s} - r_{u,s}|)}{|T|} \quad (8)$$

where T is a test set.

However, to train the algorithm we can not use the test set. In order to solve this problem, the ratings of the training set are used to compute the mean absolute error generated from the predictions of item s for user u, denoted by $MAE_{u,s}$, is computed as follows:

$$MAE_{u,s} = \frac{\sum_{s' \in R_u} (|(r_{u,s} + dev_{s',s}) - r_{u,s'}|)}{|R_u|} \quad (9)$$

Therefore, $\mathit{ItemUsefulness}_{u,s}$ is computed as a function of $MAE_{u,s}$. Considering $MaxMAE$ the maximum reachable MAE we propose the following two functions:

- Linear Weighted Slope One Function (LIUSO):

  $$\mathit{ItemUsefulness}_{u,s} = MaxMAE - MAE_{u,s} \quad (10)$$

  where the value of $MaxMAE$ is 5 in the experiments.

- Exponential Weighted Slope One Function (EIUSO):

  $$\mathit{ItemUsefulness}_{u,s} = a^{(MaxMAE - MAE_{u,s})} \quad (11)$$

  where the value of $MaxMAE$ is also 5 and the value of a is set empirically to 30 in the experiments. The difference between Equations (10) and (11) is that the second one punishes accuracy errors more intensively than the first one.

4.3 Mixed Slope One (MSO)

This approach is a combination of the previous ones, using as hypotesis the fact that if each one of the previous approaches, individually, have better performance than the baseline algorithm, they can be combined in order to have an even better performance. Thefore, this combined approach apply both extension points presented in Section 4.1, using either Equation 5 or 6, and the extensions presented in Section 4.2 to compute the *Userweight* values and either Equation 10 or 11 to compute the *ItemUsefulness* matrix values.

5. EXPERIMENTAL RESULTS

Before presenting in Section 5.3 the experimental results of the five proposed variants for the Weighted Slope One algorithm, in Section 5.1 we describe the datasets and in Section 5.2 the metrics applied to evaluate the results.

5.1 Datasets

Three datasets are used to perform the experiments: Movielens[5], Epinions[6] and Flixster[7]. Table 1 shows the characteristics of each dataset.

Table 1: Dataset charactecristics

	Movielens	Epinions	Flixster
Ratings	100,000	664,824	8,200,000
Users	9,43	49,290	1,000,000
Items	1,682	139,738	49,000
Trust Statements	-	487,181	26,700,000
Rating Range	1 - 5	1 - 5	0.5 - 5.0
Evaluation	5-*fold CV*	*LOO*	*LOO*

As seen from Table 1, the Movielens dataset does not have information about trust statements. This is why it was used only to perform experiments with the *LIUSO* and *EIUSO* algorithms. It is relevant to mention that the trust statements in Epinions dataset are unidirectional and the user who receives the trust statement not necessarely knows the user that votes on him. On the other hand, the trust statements in Flixster dataset are originated from friendship relationships, thus are bidirectional.

We now describe the method used to evaluate each dataset. Both, 5-fold cross-validation and leave one out cross-validation, are variants of cross-validation [11]. The k-fold cross validation works as follows: the dataset is split up into k partitions and k experiments are performed. During each experiment, $k-1$ partitions are used as trainning sets and the remaing one, distinct for each experiment, is used as test set. The leave one out cross-validation is the extreme case in which k is equal to the number of instances in the dataset.

In fact, we use a sample of Flixster dataset by choosing randomly 10,000 items and their ratings, in order to decrease the computational time to perform the experiments.

5.2 Metrics

As accuracy metrics we used the Mean Absolute Error (MAE) (as defined in Equation 8) and the Root Mean Square Error (RMSE). The RMSE metric penalizes more intensively higher errors and is calculated as follows:

$$RMSE = \sqrt{\frac{\sum_{(u,s) \in T} (p_{u,s} - r_{u,s})^2}{|T|}} \quad (12)$$

Other three metrics are used: precision, recall and F1 [1]. Precision measures the rate of the retrieved items that is relevant to the user, whereas recall measures the rate of the relevant items that are retrieved, as in:

$$P = \frac{\text{Number of Relevant Items Retrieved}}{\text{Total Number of Retrieved Items}} \quad (13)$$

[5]www.movielens.org
[6]www.epinions.com
[7]www.flixster.com

$$R = \frac{\text{Number of Relevant Items Retrieved}}{\text{Total Number of Relevant Items}} \quad (14)$$

Both precision and recall are important measures, hence is important to have both as greater as possible and not just one of them. The F1 measure is an harmonic mean that takes together precision and recall in only one measure. This measure is biased by the lowest of the two measures and is calculated as follows:

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision+Recall}} \quad (15)$$

In the experiments two relevance thresholds are considered. In the first threshold, an item is considered relevant for a user if its rating is above or equal to 4, a high rating for all datasets. The second threshold is the user mean. This threshold is also considered because some users are biased to rate the items with low ratings and an average item prediction is likely to be a relevant item for this kind of user.

5.3 Results and Discussion

In this section we compare the five proposed variants of the Weighted Slope One algorithm with the originl Weighted Slope One (WSO) algorithm. Paired t-tests [10] were performed with 95% of confidence. Except for the LIUSO algorithm on the Epinions dataset and using 4 as threshold, all other results showed to be statistically significant.

Tables 2, 3 and 4 show the results for Movielens, Flixster and Epinions datasets, respectively. It is important to notice that the experiments of the MSO algorithm are performed only on Epinions dataset due to the fact that it is the only dataset where the two proposed extension points, individually, have a better performance than the baseline algorithm.

Table 2: Results for Movielens dataset

Threshold	Metric	WSO	LIUSO	EIUSO
4	MAE	0.743	0.740	0.737
	RMSE	0.946	0.985	0.957
	Precision	83.8%	83.2%	81.2%
	Recall	38.2%	40.4%	47.1%
	F1-measure	52.5%	54.4%	59.6%
	Gain (F1)	-	1.9%	7.1%
Mean	MAE	0.743	0.740	0.737
	RMSE	0.946	0.985	0.957
	Precision	68.1%	67.8%	66.5%
	Recall	67.4%	69.5%	74.9%
	F1-measure	67.7%	68.6%	70.4%
	Gain (F1)	-	0.9%	2.7%

As can be seen in the tables, the PRBSO algorithm performs worse than the baseline algorithm. The IDBSO performs better on the Epinions dataset, however it showed to be even worse than the PRBSO for the Flixster dataset. We suppose that the difference on the results between the datasets is originated from the different meaning of trust statements in both datasets. While the Epinions dataset applies real trust relations, the Flixster dataset uses implicity trust relations originated from friendship.

The results for the LIUSO and EIUSO algorithms show that, in most cases, both algorithms perform better than the Weighted Slope One. In Movielens results there is a decrease in precision, however there is a greater increase in recall, for this reason the $F1$-measuse also increases. For Epinions and Flixster dataset in almost all cases both, precision and recall, increase and therefore the $F1$-measure also increases. The tables show that in general EIUSO algorithm performs better than LIUSO algorithm suggesting that accuracy errors should be taken into consideration more intensively.

The results for the MSO algorithm on Epinions dataset are presented on Table 4. The applied equations to compute *Userweight* and *ItemUsefulness* values for the MSO algorithm are equations 5 and 11, respectively, due to their individual performance. As can be seen, the MSO perfoms better than the original Weighted Slope One, IDBSO and EIUSO. It shows that the two proposed extensions points can work together to provide better recommendations than when applied individually.

6. CONCLUSIONS AND FUTURE WORK

This paper presents five ivariants of the Weighted Slope One scheme algorithms [8]. Two variants are based on the concept of web of trust. The difference between them relies on the way the user weight are computed. While the first variant uses a simple input degree computation algorithm, the second one applied the PageRank algorithm. Two other variants are based on a factor called item usefulness, which is used to weight the predictions according to how precise the predictions based on the specific items are, on average, for a specific user. Both variants consider the item usefulness values by measuring the Mean Absolute Error produced by the predicions of the items for each user. The difference between them is that the second variant gives more emphaphis to higher errors. Finally, the fifth variant combines both extensions, claiming that if both, individually, outperforms the baseline algorithm, a combination of them would do even better.

We performed experiments using Movielens, Flixster and Epinions datasets in order to compare the proposed algorithms with the original Weighted Slope One algorithms. These experiments showed that the input degree based algorithm had different behaviors when applied on Flixster and Epinions datasets. While on Flixster it presented negative results, on Epinions it showed to be better than the baseline algorithm. The PageRank based algorithm performed worse than the baseline for both datasets. The item usefulness based algorithms performed better than the baseline algorithm in most cases. Moreover, it could be seen that the algorithm that gives more emphasis to bigger errors has a better performance. The last algorithm, which combines both extensions points, performed better than the applied individual versions of it and consequently outperformed the baseline algorithm.

For future works, analysis of non accuracy metrics such as novelty and serendipity [5, 12], which measure the power of non obvioius recommendations, can be made. Moreover, other item usefulness metrics based on accuracy metrics such as Root Mean Square Error or even based on metrics beyond acurracy such as novelty and serendipity [5] can be developed in order to improve the algorithms results.

Acknowledgements

This work was partially sponsored by the Brazilian National Institute of Science and Technology for the Web (grant MCT/CNPq 573871/2008-6) and authors individual grants and scholarships from CNPq.

Table 3: Results for Flixster dataset

Threshold	Metric	WSO	IDBSO	PRBSO	LIUSO	EIUSO
4	MAE	0.741	1.954	0.747	0.734	0.714
	RMSE	1.059	2.499	1.068	1.056	1.058
	Precision	81.2%	59.8%	81.2%	80.7%	79.2%
	Recall	42.0%	33.3%	41.3%	44.2%	50.2%
	F1-measure	55.4%	42.7%	54.8%	57.1%	61.5%
	Gain (F1)	-	−12.7%	−0.6%	1.7%	6.1%
Mean	MAE	0.741	1.954	0.747	0.734	0.714
	RMSE	1.059	2.499	1.068	1.056	1.058
	Precision	60.4%	52.1%	59.2%	60.7%	61.2%
	Recall	53.8%	37.6%	52.5%	58.3%	69.2%
	F1-measure	56.9%	43.7%	55.6%	59.4%	64.9%
	Gain (F1)	-	−13.2%	−1.3%	2.5%	8.0%

Table 4: Results for Epinions dataset

Threshold	Metric	WSO	IDBSO	PRBSO	LIUSO	EIUSO	MSO
4	MAE	1.667	1.636	1.731	1.663	1.627	1.593
	RMSE	2.353	2.350	2.421	2.351	2.335	2.329
	Precision	86.9%	90.6%	85.3%	87.2%	87.5%	91.4%
	Recall	48.5%	50.0%	45.6%	48.2%	51.3%	52.5%
	F1-measure	62.2%	64.4%	59.4%	62.1%	64.7%	66.7%
	Gain (F1)	-	2.2%	−2.8%	−0.1%	2.5%	4.5%
Mean	MAE	1.667	1.636	1.731	1.663	1.627	1.593
	RMSE	2.353	2.350	2.421	2.351	2.335	2.329
	Precision	71.8%	76.4%	70.4%	71.2%	72.7%	77.9%
	Recall	49.2%	53.0%	47.7%	49.8%	53.6%	56.3%
	F1-measure	58.4%	62.6%	56.9%	58.8%	61.7%	65.3%
	Gain (F1)	-	4.2%	−1.5%	0.4%	3.3%	6.9%

7. REFERENCES

[1] F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web*, 5(1):2:1–2:33, 2011.

[2] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proc. WWW 2007*, pages 271–280.

[3] M. Gao and Z. Wu. Incorporating personalized contextual information in item-based collaborative filtering recommendation. *Journal of Software*, 5(7):729–736, 2010.

[4] M. Gao, Z. Wu, and F. Jiang. Userrank for item-based collaborative filtering recommendation. *Information Processing Letters*, 111(9):440–446, 2011.

[5] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[6] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[7] P. Lawrence, B. Sergey, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[8] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *SIAM Data Mining*, pages 471–480, 2005.

[9] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *CoopIS/DOA/ODBASE (1)*, pages 492–508, 2004.

[10] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1st edition, 1997.

[11] P. Refaeilzadeh, L. Tang, and H. Liu. Cross-Validation. In *Encyclopedia of Database Systems*, pages 532–538. Springer US, 2009.

[12] M. T. Ribeiro, A. Lacerda, A. Veloso, and N. Ziviani. Pareto-efficient hybridization for multi-objective recommender systems. In *Proc. RecSys 2012*, pages 19–26.

[13] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *10th proc. WWW 2001*, pages 285–295.

[14] P. Wang and H. W. Ye. A personalized recommendation algorithm combining slope one scheme and user based collaborative filtering. In *Proc. IIS 2009*, pages 152–154.

[15] D. Zhang. An item-based collaborative filtering recommendation algorithm using slope one scheme smoothing. In *ISECS 2009 - Vol. 02*, pages 215–217.