FAKULTÄT FÜR !NFORMATIK

Evaluation of Collaborative Filtering Algorithms

Bakkalaureatsarbeit

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Software & Information Engineering (033 534)

ausgeführt von

Patrick Marschik
Matrikelnummer 0625039
patrick.marschik@student.tuwien.ac.at

am:
Institut für Softwaretechnik und Interaktive Systeme

Betreuer: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber

Contents

Abstract

Collaborative filtering recommender algorithms generate personalized recommendations for users based on a set of ratings on items. In this work we will present three different algorithms for collaborative filtering based recommender systems: the Pearson Correlation Coefficient, Item-based Collaborative Filtering and Slope One. We will also introduce metrics to measure accuracy of these algorithms. Finally we will compare the accuracy and performance result of an implementation of these three algorithms.

1 Introduction

Since the amount of information is ever growing it becomes harder and harder to find information of relevance. Therefore techniques to filter the information for pieces relevant to the user need to be devised. One domain of algorithms that tries to solve this problem is Collaborative Filtering (CF) recommender systems. CF algorithms store preferences of users for items in a database. This data is then used to generate a neighborhood of similar users or items with the assumption that a user likes the same items as the users in a close neighborhood to him.

This work will compare the performance – both accuracy and runtime – of several CF algorithms. These algorithms are Slope One as described by Lemire and Maclachlan [2005], Item-based collaborative filtering by Sarwar et al. [2001] and, as a baseline reference scheme, the Pearson Correlation Coefficient as presented by Breese et al. [1998]. Although the original works did some comparison of the algorithms they introduced with other algorithms the published accuracy data cannot be directly compared since the test-sets used for evaluation were different. Also authors often chose random splits of datasets which makes reproduction of the tests impossible. This work will use the Movielens-100k[1] dataset with the original five split datasets to ease reproduction of the tests.

This paper is structured as follows:

Section 2 will briefly introduce the framework in which the algorithms were implemented.

Section 3 will introduce a common notation used in later sections of the work.

[1] http://www.movielens.org/ as of February 2010

Section 4 will cover in detail the three compared algorithms and metrics used to analyze them.

Section 5 will explain the experimental setup and procedure and will cover the results of the experiment.

Section 6 final conclusions and future work.

Appendix A contains detail of the implementations.

Appendix B contains all generated experiment data.

1.1 Motivation

Smart Agent Technologies Studio (SAT)[2] developed the recommender system easyRec. In february 2010 the source code for easyRec was released under an open source license. The source code for version 0.9 can be downloaded from Source Forge[3]. Our task was to implement three algorithms for use in easyRec. SAT would be happy for any contributions made to easyRec and plan to continue supporting it as an open source project.

2 Context

The algorithms implemented were integrated into the easyRec framework outlined in [Cerny, 2008; Gstrein, 2009].

Figure 6 on page 20 shows the software architecture of easyRec. easyRec makes its recommendation capabilities available as web services (SOAP and REST). The action service stores any action happening on a client application – such as rating or viewing an item – in a database. On request generators then store predictions in the database. Generators are implementations of recommender algorithms. The algorithms implemented in this work are implemented as easyRec generators.

Figure 7 on page 21 depicts the data model of easyRec's database. The key tables are `Action` and `ItemAssociation`. `Action` stores all actions performed on a client site. As we can see the action table stores different types of actions (`actionTypeId` column).

[2] `http://sat.researchstudio.at` as of February 2010

[3] `http://sourceforge.net/projects/easyrec/files/easyrec_0.9.zip/download` as of February 2010

Of main interest for the algorithms implemented in this work are actions of the type rating. `ItemAssociation` contains the predictions created by the generators. Since the table is designated only for item to item associations we added several other tables to the database. The tables added are listed in A.2 on page 22.

3 Notation

The following notation (adapted from the notation used by Lemire and Maclachlan [2005]) is used throughout the paper for describing the different CF recommender algorithms.

To express the cardinality of a set X we write $\text{card}(X)$. Conversely $|x|$ denotes the absolute value of a variable x.

An evaluation (ratings given by a user) is represented as vector u. Then u_i denotes the components of this vector u. These components describe that user u rated item i. χ is the set of all evaluations u in the training-set. Moreover $\overline{u}$ is the average of an evaluation u.

$S(u)$ is the subset of u consisting of the items that were rated. Furthermore $S_i(\chi)$ denotes a subset of χ containing all $u \in \chi$ where item i was rated ($S_i(\chi) = \{u \,|\, u \in \chi, i \in S(u)\}$). Likewise $S_{i,j}(\chi)$ is the subset of χ where both the item i and j were rated ($S_{i,j}(\chi) = \{u \,|\, u \in \chi,\, i, j \in S(u)\}$). Predictions for user u and item i are written as $P(u)_i$. Also given an item i the average rating of that item is denoted as $\overline{i}$.

4 Related Work

According to Burke [2002] recommender systems can be classified in five groups. Collaborative, content-based, demographic, utility-based and knowledge-based techniques. This work will focus on collaborative techniques. Collaborative techniques use the similarity relation between users to generate recommendations for users [see Gstrein, 2009]. The user is asked for his opinion of an item (a rating of that item). Based on these ratings a similarity between users can be calculated. Predictions are then made by using these similarities of a user to other users. Collaborative techniques can further be divided to memory-based and model-based techniques [see Breese et al., 1998]. Again, this paper will focus on the former ones. First the Pearson Correlation Coefficient algorithm

is introduced as a basic implementation of CF recommender algorithms. Next Item-based Collaborative Filtering is shown because the algorithm differs from the standard CF techniques – not similarities between users but between items are calculated. At last the Slope One algorithm is presented which aims to be a fast but also reliable[4] CF algorithm.

4.1 Pearson Correlation Coefficient

As a baseline comparison the Pearson Correlation Coefficient was implemented. The Pearson Correlation Coefficient is used to calculate the similarity between two users. This similarity value is thereafter utilized to calculate the prediction for a user and an item. This is accomplished by using a weighted sum. According to Breese et al. [1998] there are also other ways of using the similarities to generate predictions. In the course of this work we will be limiting us to the weighted sum.

$$P^{\mathrm{PCC}}(a)_i = \overline{a} + \kappa \sum_{o \in S_i(\chi)} w(a,o) \cdot (o_i - \overline{o}) \tag{1}$$

Let a be the active user, the one we are currently creating a prediction for. Then other users – who rated the item in question – denoted as o will be inspected for their similarity to the active user. The similarity is captured in the function $w(a,o)$ which will be described in more detail below. The weight is then multiplied with the relative voting value of the other user. κ is a factor for normalization and is usually defined as $\kappa = (\sum_{o \in \chi} |w(a,o)|)^{-1}$.

Viertl [2003, pg. 80] defined the Pearson Correlation Coefficient– which will be used as weight $w(a,o)$ – as

$$r = \frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{Var}X}\sqrt{\mathrm{Var}Y}} = \frac{\mathbb{E}(XY) - \mathbb{E}X\,\mathbb{E}Y}{\sqrt{\mathrm{Var}X}\sqrt{\mathrm{Var}Y}} \tag{2}$$

Where $\mathbb{E}X$ is the expected value of X. $\sqrt{\mathrm{Var}X}$ is the standard deviation of X. For use with CF equation 2 is as follows

$$w(a,o) = \frac{\sum_{o \in I}(a_i - \overline{a})(o_i - \overline{o})}{\sqrt{\sum_{o \in I}(a_i - \overline{a})^2 \sum_{o \in I}(o_i - \overline{o})^2}} \tag{3}$$

[4] reliable means generating reliably accurate predictions

Where I is the set of items which both – the active user a and the other user o – have rated: $I = S(a) \cap S(o)$.

To improve the prediction accuracy Breese et al. [1998] proposed a number of modifications to the Pearson Correlation Coefficient. The modifications proposed are (a) Default Voting, (b) Inverse User Frequency and (c) Case Amplification.

Default Voting tries to tackle the sparsity problem – when there are few votes – of recommender algorithms. When one of two users didn't rate for an item the other user rated, a default vote is applied. Default Voting won't be further covered in this paper.

4.1.1 Inverse User Frequency

The idea of Inverse User Frequency is to penalize often occurring items. The gist of this method is that seldom occurring items capture the similarity between two users, who have both rated that item, better. An analogy in the film environment would be that Hollywood blockbusters are penalized in weight. Conversely seldom watched art movies are rewarded, since people watching them might be more interested in another art film than yet another Hollywood blockbuster. The origin of the Inverse User Frequency can be traced back to information retrieval where a similar method – the inverse document frequency – is used [see Jones, 1972].

$$f_i = \log \frac{\text{card}(\chi)}{\text{card}(S_i(\chi))} \tag{4}$$

Where $\text{card}(\chi)$ is the total number of items in the dataset. Moreover $\text{card}(S_i(\chi))$ is the number of times item i has been rated. Note that f_i can become zero and therefore remove an entire weight from the calculation.

This value has then to be multiplied to the ratings, i.e. each component of the vectors a and o has to be weighed accordingly. Therefore equation 2 has to be transformed using a weighted expected value $\mathbb{E}_W X = \frac{\sum_i w_i x_i}{\sum_i w_i}$ and a weighted standard deviation $\sqrt{Var_W X} = \sqrt{\mathbb{E}_W X^2 - \mathbb{E}_W^2 X}$. Using the Inverse User Frequency f_i as weight w_i we

get:

$$w(a,o) = \frac{\sum_{i \in I} f_i \sum_{i \in I} f_i \, a_i \, o_i - \sum_{i \in I} f_i \, a_i \sum_i f_i \, o_i}{\sqrt{(\sum_{i \in I} f_i \sum_{i \in I} f_i \, a_i^2 - (\sum_{i \in I} f_i \, a_i)^2)(\sum_{i \in I} f_i \sum_{i \in I} f_i \, o_i^2 - (\sum_{i \in I} f_i \, o_i)^2)}} \tag{5}$$

Where I is the set of items which both – the active user a and the user o – rated: $I = S(a) \cap S(o)$.

4.1.2 Case Amplification

Case Amplification is another simple means of improving (not only) Pearson Correlation Coefficient predictions. The idea here is to mute low valued weights more than high valued ones. The adaption to include this in the prediction process is simply made in equation 1 by replacing the original weights value $w(a,o)$ with

$$w(a,o)' = \begin{cases} w^\rho(a,o) & \text{for } w(a,o) \geq 0 \\ -(-w^\rho(a,o)) & \text{for } w(a,o) < 0 \end{cases} \tag{6}$$

Where $\rho \in \mathbb{R}$. A typical value for ρ used by Breese et al. [1998] is 2.5. I.e. a high weight value of 0.9 will become 0.81 and is still high but a lower weight value of 0.6 will become much a less significant 0.36.

4.2 Item-based Collaborative Filtering

Item-based Collaborative Filtering was presented by Sarwar et al. [2001]. Usually CF recommender algorithms calculate similarities between users. Item-based Collaborative Filtering however computes it's similarity values between items. After computing the similarity between items the Item-based Collaborative Filtering algorithm uses them to calculate the predictions for the users. In a typical application of CF there are more users than items [see Sarwar et al., 2001]. Calculating the similarity between items should therfore perform better than calculating the similarity between users.

4.2.1 Similarity Calculation

Several methods for computing the similarity of items were suggested by Sarwar et al. [2001]. The basic scheme for all of them is to select the ratings of the users that rated both items. This is depicted in figure 1. The rows 1...m represent the users and the columns 1...n the items. Items i and j are fixed and only those rows that contain ratings for both are taken into consideration

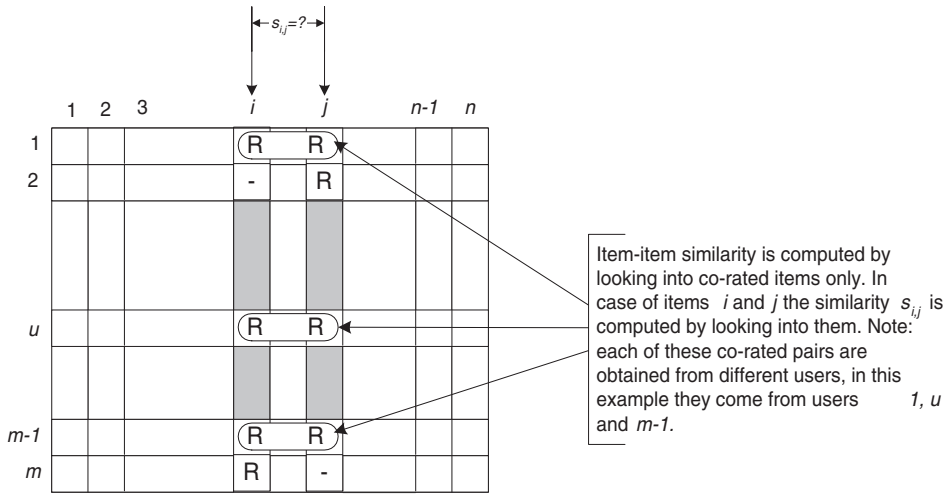

Figure 1: Isolation of the co-rated items and similarity computation[5]

Correlation-based Correlation-based similarity is – just like the plain Pearson Correlation Coefficient CF algorithm – based on the Pearson Correlation Coefficient but adapted to work on items instead of user vectors:

$$sim(i,j) = \frac{\sum_{u \in S_{i,j}(\chi)}(u_i - \bar{i})(u_j - \bar{j})}{\sqrt{\sum_{u \in S_{i,j}(\chi)}(u_i - \bar{i})^2 \sum_{u \in S_{i,j}(\chi)}(u_j - \bar{j})^2}} \tag{7}$$

Note that compared to equation 3 items and users were swapped.

Cosine-based For cosine-based similarity the items i and j are treated as vectors. The similarity between these two vectors is the cosine of the angle between them as

[5] [Sarwar et al., 2001, pg. 289 figure 2]

described in the following equation.

$$sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\| \times \|\vec{j}\|} \tag{8}$$

Where $\|\vec{x}\|$ is the euclidean length of $\vec{x}$. In addition $\vec{x}\cdot\vec{y}$ is the dot-product of the vectors $\vec{x}$ and $\vec{y}$.

Adjusted Cosine-based Sarwar et al. [2001] found that basic cosine-based similarity computation in Item-based Collaborative Filtering had a drawback: the rating scale – which differs between users – is not taken into account. In user-based CF this problem is addressed in the prediction calculation. In Item-based Collaborative Filtering however the information about the other users involved isn't present in the prediction calculation anymore. Therefore Sarwar et al. [2001] proposed a change to calculating the cosine. They subtract the average rating for the user addressed in the component of the item vector. With this adaption they bring the information about the users rating scale back to the computation.

$$sim(i,j) = \frac{\sum_{u\in S_{i,j}(\chi)}(u_i-\overline{u})(u_j-\overline{u})}{\sqrt{\sum_{u\in S_{i,j}(\chi)}(u_i-\overline{u})^2 \sum_{u\in S_{i,j}(\chi)}(u_j-\overline{u})^2}} \tag{9}$$

Using this adjusted cosine measure for similarity had a significant impact on the quality of predictions. It outperformed both the correlation-based and the unmodified cosine-based similarity computations [see Sarwar et al., 2001, pg. 291 figure 4].

4.2.2 Prediction Calculation

After the similarities are calculated they are used to compute the prediction for a specific user. Sarwar et al. [2001] proposed two different prediction calculation algorithms: a weighted sum and regression.

The idea for the prediction generation – using the similarity pairs of items the user rated and the item in question – is shown in figure 2. User u rated the items 1, 3, $i-1$, $i+1$ and $n-1$. Suppose the prediction for user u and item i is calculated. The prediction for i is based on the similarities of the item i to the other items rated by user u.

[6] Sarwar et al. [2001, pg. 290 figure 3]

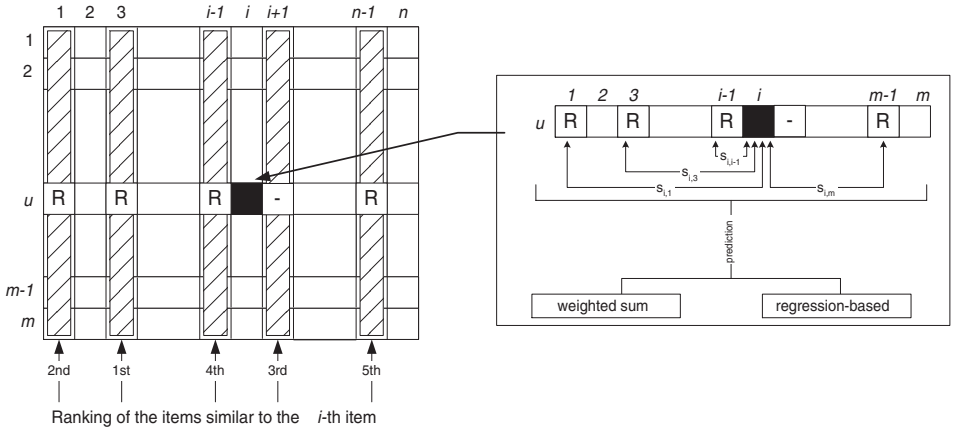

Figure 2: The prediction generation process is illustrated for 5 neighbors[6]

Weighted Sum The weighted sum for Item-based Collaborative Filtering is:

$$P^{\text{ICF}}(a)_i = \frac{\sum_{j \in S(a)} (sim(i,j) + 1) \cdot a_j}{\sum_{j \in S(a)} |sim(i,j)|} \tag{10}$$

The similarity values are in the range of $[-1, 1]$. Since the similarities are used as weight they might cancel out a rating $a_j$when they become 0. Therefore they are shifted to $[0, 2]$ by adding 1. Another way of removing the canceling effect would be to center a_j's rating on 0 and add that offset to the value generated by the basic prediction.

Regression Another method for prediction calculation Sarwar et al. [2001] presented is based on linear regression. The notion is that computed similarity values might be distant even though they have high similarity. Therefore instead of using the raw rating value a_j in equation 10 they used approximated values $j^{'}$:

$$\overline{j'} = \alpha\, \overline{i} + \beta + \epsilon \tag{11}$$

Sarwar et al. [2001] observed that regression-based predictions outperform the simple weighted sum for sparse data. But the more data is added the quality of regression-based predictions significantly declines. They ascribe this to over-fitting the regression model at high data density levels.

4.3 Slope One

Slope One is a CF algorithm proposed by Lemire and Maclachlan [2005] with the main goal to be easily implementable. Nonetheless it should deliver reasonably accurate predictions, fast on-line query processing and dynamic updates of the generated predictions when new ratings occur. Lemire and Maclachlan state that the strong points of their algorithm are that it (a) is easy to implement, (b) is updatable on the fly, (c) is efficient at query time, (d) doesn't need many ratings to generate predictions and (e) is reasonably accurate.

Similar to Item-based Collaborative Filtering, Slope One uses an item similarity measure. This metric however – and also the prediction computation – are different from either variation of Item-based Collaborative Filtering. The idea is to store the differential rating values of item pairs – called "popularity differential" by Lemire and Maclachlan. Figure 3 displays this idea. To predict the rating of user B for item J the differential between the ratings of user A is used. User A already rated $I = 1$ and $J = 1.5$ therefore the differential is 0.5. Since user B already rated item $I = 2$ the prediction for item J is $2 + 0.5 = 2.5$.

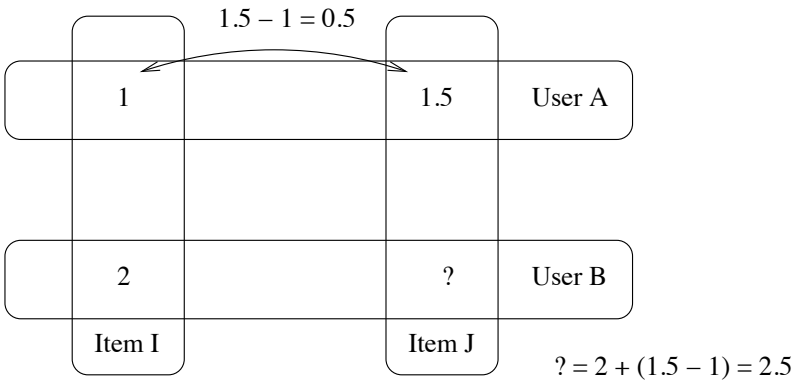

Figure 3: The rating of two items user A rated in common with user B is used to predict user B's item J[8]

Expanding this scheme to several items the "similarity" in Slope One is defined as the average deviation of the items i and j given by:

$$dev(i,j) = \sum_{u \in S_{i,j}(\chi)} \frac{u_i - u_j}{\text{card}(S_{i,j}(\chi))} \tag{12}$$

[8] Lemire and Maclachlan [2005, pg. 1]

If the numerator and the denominator of the average deviation are stored separately $dev(i,j)$ can easily be upgraded when new ratings are entered. Also since $dev(i,j)$ is symmetric, computation time is halved.

As seen in figure 3 $dev(i,j) + u_j$ can be used as prediction for u_i. Therefore a usable predictor is the average of all such predictions:

$$P^{\mathrm{S1}_{\mathrm{scheme}}}(a)_i = \frac{1}{\mathrm{card}(R_i)} \sum_{j \in R_i} (dev(i,j) + u_j) \tag{13}$$

Where $R_i = \{j \mid j \in S(u),\, i \neq j,\, \mathrm{card}(S_{i,j}(\chi)) > 0\}$ are all items to be taken into consideration. Lemire and Maclachlan [2005] present an approximation based on the fact that in a dense dataset – where almost all pairs of items i and j were rated – $R_i = S(u) \setminus \{i\}$. Also in a dense dataset $\sum_{j \in R_i} \frac{u_j}{\mathrm{card}(R_j)}$ is approximately $\overline{u}$. The approximated prediction is

$$P^{\mathrm{S1}}(a)_i = \overline{u} + \frac{1}{\mathrm{card}(R_i)} \sum_{i \in R_i} dev_{i,j} \tag{14}$$

As a further improvement, a weighted variant of equation 13 was suggested. For weights the number of users who rated both items i and j are used. The more users rated an item pair, the more valuable it is considered. Note that this is contradictory to the idea of Inverse User Frequency (section 4.1.1) where less user ratings mean a greater weight for the prediction. This is applicable because average deviations are used and not correlation values.

$$P^{\mathrm{wS1}}(a)_i = \frac{\sum_{j \in S(u) - \{i\}} (dev(i,j) + u_j)\, \mathrm{card}(S_{i,j}(\chi))}{\sum_{j \in S(u) - \{i\}} \mathrm{card}(S_{i,j}(\chi))} \tag{15}$$

Lemire and Maclachlan [2005] described a third variant of Slope One, the bi-polar Slope One. The idea of bi-polar Slope One is that average deviations are split in like and dislike sets. These sets are based on the users average rating and Liked/disliked items are then only weighted among themselves. Other than that the scheme is identical to the weighted Slope One.

4.4 Evaluation of Recommendations

Herlocker et al. [2004] reviewed methods for evaluating CF recommender systems. They split the evaluation into several steps. Some of them are the identification of the user tasks to be evaluated and accuracy metrics.

Among the user tasks "Find Good Items" is of most interest to us since it is used by most commercial systems. "Find Good Items" is the task of showing the user a list of (ranked) items that he might like.

Several different measurements for accuracy of a CF recommender system were presented. Herlocker et al. separated them into three groups: (a) predictive accuracy metrics, (b) classification accuracy metrics and (c) rank accuracy metrics. Predictive accuracy metrics – such as MAE and RSME – try to capture how close the predictions were to the actual rating of a user. Classification accuracy metrics measure how often a system makes incorrect decisions about wether a recommended item is good. Finally rank accuracy metrics try to appraise the order the recommender system generates versus the order the user would have made.

4.4.1 Mean Absolute Error

The Mean Absolute Error (MAE) metric has been used several times to analyze CF recommender systems [see Breese et al., 1998; Gstrein, 2009; Lemire and Maclachlan, 2005; Sarwar et al., 2001]. Although predictive accuracy metrics can be used for "Find Good Items", rank accuracy metrics are better fitted for the task. However the Mean Absolute Error has the advantages that it is easy to calculate and also statistically well studied. Therefore it is good for comparison of different recommender systems.

$$\text{MAE} = \frac{\sum_{(u,i)\in\psi} |P(u)_i - u_i|}{\text{card}(\psi)} \tag{16}$$

Where $\psi = \{(u,i) \,|\, \forall u \in S(\chi'), \forall i \in S(u)\}$ is the set of all user-item pairs in the test-set χ' and $P(u)_i$ is the prediction generated from the training-set χ.

4.4.2 Root Mean Squared Error

The Root Mean Squared Error (RMSE) is related to the Mean Absolute Error but stresses large errors more seriously.

$$\text{RMSE} = \sqrt{\frac{\sum_{(u,i)\in\psi} |P(u)_i - u_i|^2}{\text{card}(\psi)}} \tag{17}$$

Where – like before – $\psi = \{(u,i) \,|\, \forall u \in S(\chi'), \forall i \in S(u)\}$ is the set of all user-item pairs in the test-set χ' and $P(u)_i$ is the prediction generated from the training-set χ.

5 Contribution

In the course of this work Slope One, Pearson Correlation Coefficient and Item-based Collaborative Filtering algorithms were implemented for the easyRec[9] recommender system framework outlined in [Cerny, 2008; Gstrein, 2009].

5.1 Experiment Setup

The dataset used for comparison was Movielens-100k[10] with the default five-split subsets. Movielens-100k contains 100,000 ratings in the range from 1 to 5 given by 943 users on 1682 movies. Each of the users rated at least 20 movies. The default five-split subsets are splits of 80% training data and 20% test data and are disjoint.

The data for the test-sets was imported to a MySQL 5.1 database. Indices were added to the tables/columns most beneficial to time consuming queries. The algorithms were implemented in Java version 1.6 and the operating system used was Mac OS X 10.6. This software setup is summarized in table 1.

Software	Version
Operating System	Mac OS X 10.6.2
Programming Language	Java SE 1.6.0_17
Database	MySQL 5.1.41

Table 1: Software Setup

[9] http://easyrec.org/ as of February 2010
[10] http://www.movielens.org/ as of February 2010

Accuracy as well as performance evaluation were performed on a MacBook with a processor speed of 2.4 GHz and 2 GB of RAM. The hardware setup is summarized in table 2.

Part	Specifications
CPU	Intel Core 2 Duo
CPU frequency	2.4 GHz
Main Memory Size	2 GB
Main Memory Speed	1.07 GHz

Table 2: Hardware Setup

5.2 Experiment Procedure

When running the tests for a single algorithm the Java Virtual Machine was not restarted until the algorithm ran for all five test sets. After an algorithm finished running, the database tables were cleared of the generated data. Then for the next test, the Java Virtual Machine was restarted.

As a metric both MAE and RSME described in section 4.4 were used. The computed recommendations were truncated to fit in the rating scale [1...5]. All recommendations the algorithms could produce were generated – not just the ones in the test-set. For performance the time needed by the algorithms was taken and converted into a recommendations/second measure.

5.3 Pearson Correlation Coefficient

For our experiment the Pearson Correlation Coefficient algorithm was implemented with both Inverse User Frequency and Case Amplification. For Case Amplification a parameter of $\rho = 2.5$ was assumed like Breese et al. [1998] proposed. The weights generated were stored in the database. We ran several variations of the algorithm with Inverse User Frequency and Case Amplification turned either completely off, one of the two off or both on.

5.4 Item-based Collaborative Filtering

Similarity calculation for Item-based Collaborative Filtering was implemented in all three variants introduced in [Sarwar et al., 2001]. The similarities calculated were stored in a database table for querying in the prediction phase. For prediction calculation only the basic weighted sum (see equation 10) was implemented. Since the exact method used in [Sarwar et al., 2001] was not given, we implemented both variations described in section 4.2.2. We have found that adding 1 to the similarity $sim(i, j)$ delivers a much better MAE than the method of centering the rating a_j. Our implementation of Item-based Collaborative Filtering also differs in the size of neighboring item similarities used. Sarwar et al. used only a limited set of neighbors whereas we used the complete neighborhood available.

5.5 Slope One

Slope One was implemented in both, the simple and the weighted variant described in section 4.3. The average deviations of the items were stored in a database table. The columns for the deviations were split for the numerator and denominator. This allows to make differential updates – adding new ratings after the average deviations have been calculated.

5.6 Results

In figure 4 we can see that the Pearson Correlation Coefficient CF algorithm has the best accuracy of the algorithms compared. Second best is Slope One followed by Item-based Collaborative Filtering. This confirms that Slope One is an algorithm with reasonably accurate prediction quality. On the contrary the Item-based Collaborative Filtering algorithm didn't do better than user-based CF recommenders.

In figure 5 the results of the performance measures are shown. Since all three algorithms produced a different amount of predictions, recommendations per second were chosen as performance measure. Some Pearson Correlation Coefficient variations were run with only 20,000 predictions generated.

Slope One clearly dominates the performance test. It outperformed the second placed Item-based Collaborative Filtering by a factor of roughly 4 and the last placed Pearson Correlation Coefficient by a factor of 15. This confirms that Slope One is fast for

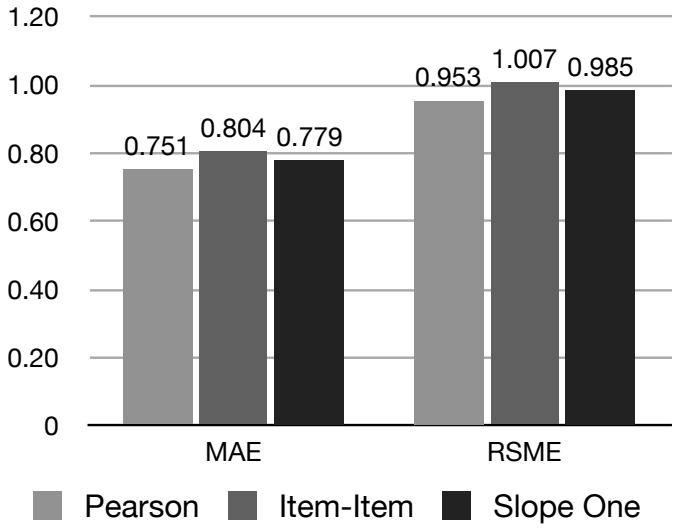

Figure 4: Accuracy of the algorithms

online queries and – as we have seen before – reasonably accurate. The second placed Item-based Collaborative Filtering is still about 3 times faster than Pearson Correlation Coefficient, therefore confirming the results of Sarwar et al. [2001].

Table 7 on page 30 displays the detailed accuracy results for the Pearson Correlation Coefficient. We can see that the unmodified Pearson Correlation Coefficient variant is the most accurate of the variants tested. The Inverse User Frequency is the second most accurate followed by Case Amplification. The combination of Case Amplification and Inverse User Frequency performs worst of all variants.

On table 8 on page 31 we can see the accuracy results for Item-based Collaborative Filtering. The calculations were performed with the weighted sum adapted to add 1 to $sim(i, j)$. As expected adjusted-cosine similarity calculation performed best followed by cosine and Pearson similarity calculation. When using centered ratings a_j the accuracy drops significantly as shown in table 9 on page 32.

Slope One accuracy data is shown in table 10 on page 33. As anticipated the weighted variant is better than the simple variant.

Detailed performance results for all algorithms are on the tables 11, 12, 13 and 14 on pages 34, 35, 36 and 37 respectively.

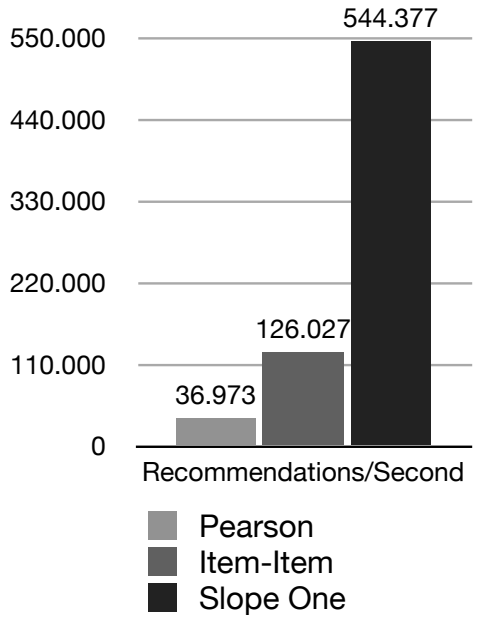

Figure 5: Performance of the algorithms

6 Conclusion

In this work we introduced three Collaborative Filtering recommender algorithms. Firstly the Pearson Correlation Coefficient, which is based on the relation between pairs of users. Secondly the Item-based Collaborative Filtering algorithm that creates its predictions on the relation between pairs of items. And finally Slope One which tries to be a fast but sufficiently accurate CF recommender algorithm. We compared these three algorithms using MAE and RSME as accuracy metric and recommendations per second as performance metric.

The results showed that, although Pearson Correlation Coefficient was the most accurate, it was also several times slower than the fastest algorithm Slope One. Despite the fact that Item-based Collaborative Filtering was faster than Pearson Correlation Coefficient it was also the most inaccurate of the three compared algorithms.

Further work should include the bi-polar variant of Slope One. Moreover experiments with different neighborhood sizes and an implementation of regression based predictions for Item-based Collaborative Filtering could be implemented. Furthermore the comparison could be extended by implementing default voting for the Pearson Correlation Coefficient.

List of Figures

List of Tables

List of Algorithms

References

John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 43–52. Morgan Kaufmann, 1998.

Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002. ISSN 0924-1868. doi: http://dx.doi.org/10.1023/A:1021240730564.

Roman Cerny. Design and implementation of a generic recommender and its application to the music domain. Master's thesis, Vienna University of Technology, Vienna, Austria, October 2008.

Erich Gstrein. *Adaptive Personalization: A multi view personalization approach incorporating contextual information.* PhD thesis, Vienna University of Technology, 2009.

Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, John, and T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22:5–53, 2004.

Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining (SDM'05)*, pages 471–475, 2005.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0. doi: http://doi.acm.org/10.1145/371920.372071.

Reinhard Viertl. *Einführung in die Stochastik - Mit Elementen der Bayes-Statistik und der Analyse unscharfer Information.* Springers Lehrbücher der Informatik. Springer-Verlag, Vienna, Austria, 3rd edition, 2003. ISBN 3-211-00837-3.

A Implementation

A.1 easyRec

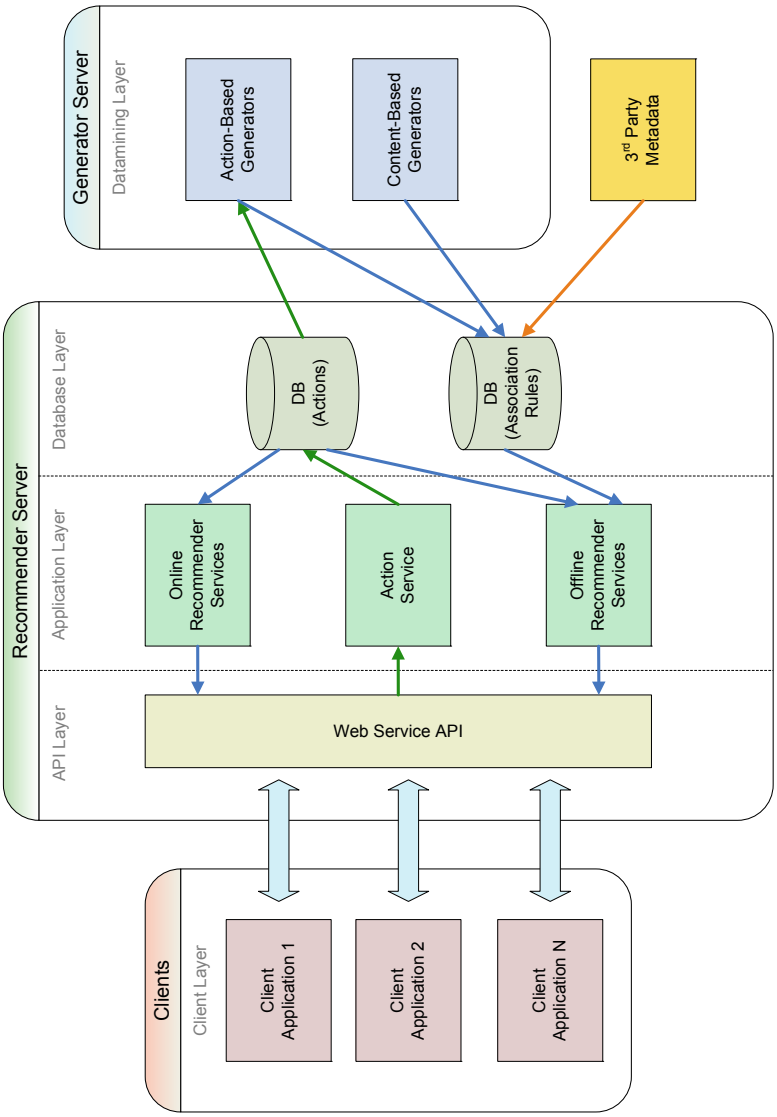

Figure 6: Overview of easyRec Software Architecture[11]

[11] [Cerny, 2008, pg. 34]

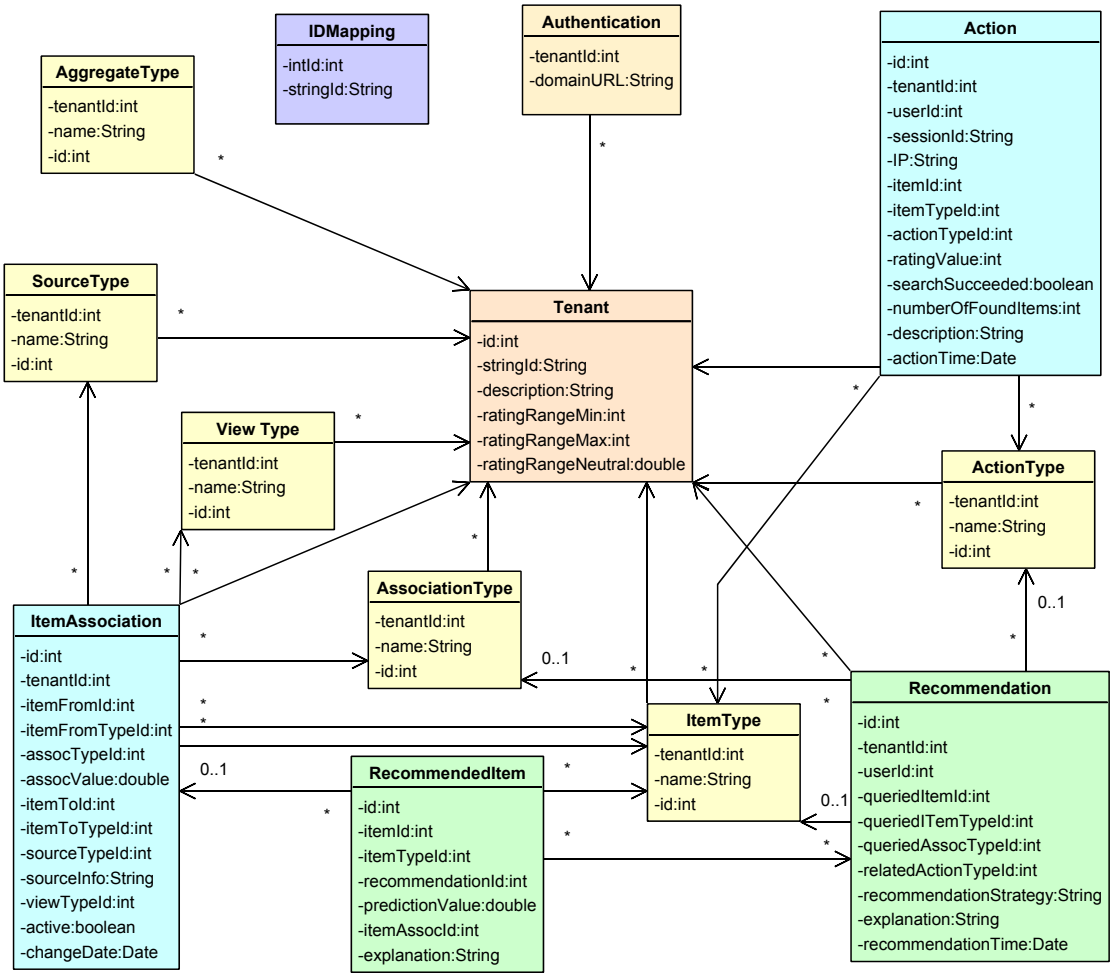

Figure 7: easyRec's data model[12]

[12] [Cerny, 2008, pg. 47]

A.2 Extension of Data Model

A.2.1 cont_lastaction

Field	Type	Description
id	int(11)	Identifier
tenantId	int(11)	Identifier for the tenant the action belongs to
userId	int(11)	Identifier for the user that did the action
itemId	int(11)	Identifier for the item on which the action applies
itemTypeId	int(11)	Identifier for the type of the involved item
actionTypeId	int(11)	Identifier for the action that was performed
ratingValue	int(11)	Value of the rating
actionTime	datetime	Time when the action occured
previousRatingValue	int(11)	The last rating value of the action before an update
previousActionTime	datetime	The last action time of the action before an update

Table 3: Data Model for cont_lastaction

A.2.2 cont_userassoc

Field	Type	Description
id	int(11)	Identifier
tenantId	int(11)	Identifier for the tenant the association belongs to
userFromId	int(11)	Identifier for the user that is associated
assocValue	double	Value of the association
itemToId	int(11)	Identifier for the item that is associated
itemToTypeId	int(11)	Identifier for the type of the item that is associated
sourceTypeId	int(11)	Identifier for the source that generated the association
changeDate	datetime	Date when the association was last changed

Table 4: Data Model for cont_userassoc

A.2.3 so_lastupdate

Field	Type	Description
tenantId	int(11)	Identifier for the tenant that was updated
lastUpdate	datetime	Date when the last update of the tenant occured

Table 5: Data Model for so_lastupdate

A.2.4 so_deviation

Field	Type	Description
id	int(11)	Identifier
tenantId	int(11)	Identifier for the tenant the deviation belongs to
item1Id	int(11)	Identifier for the first item involved in the deviation
item1TypeId	int(11)	Identifier for the type of the first item involved in the deviation
item2Id	int(11)	Identifier for the second item involved in the deviation
item2TypeId	int(11)	Identifier for the type of the second item involved in the deviation
difference	double	Sum of differences for the deviations
count	int(11)	Number of actions involved in the deviation

Table 6: Data Model for so_deviation

A.3 Pearson Correlation Coefficient

Algorithm 1 Pearson Correlation Coefficient weight calculation

Require: U set of users

Require: A map of average user ratings

Require: iuf a boolean describing if Inverse User Frequency is enabled

function CALCULATEWEIGHTS(U,A,iuf)
  for $i \leftarrow 1, \text{card}(U)$ **do**
    $u_a \leftarrow U_i$
    for $j \leftarrow i+1, \text{card}(U)$ **do**
      $u_o \leftarrow U_j$
      $R \leftarrow$ getItemsRatedTogether(u_a, u_o)
      if $\text{card}(R) = 0$ **then**
        continue
      end if
      $f \leftarrow 1$
      if $iuf = true$ **then**
        $f \leftarrow \log(\text{card}(U)/\text{card}(R))$
        if $f = 0$ **then**
          continue
        end if
      end if
      $f_{sum} \leftarrow 0$
      $e_{both} \leftarrow 0$; $e_{active} \leftarrow 0$; $e_{other} \leftarrow 0$
      $e_{active\,square} \leftarrow 0$; $e_{other\,square} \leftarrow 0$
      for all $r \in R$ **do**
        $f_{sum} \leftarrow f_{sum} + f$
        $e_{both} \leftarrow e_{both} + (f \cdot r_{active} \cdot r_{other})$
        $e_{active} \leftarrow e_{active} + (f \cdot r_{active})$
        $e_{other} \leftarrow e_{other} + (f \cdot r_{other})$
        $e_{active\,square} \leftarrow e_{active} + (f \cdot r_{active}^2)$
        $e_{other\,square} \leftarrow e_{other} + (f \cdot r_{other}^2)$
      end for

```
28:             var_active ← f_sum · e_active square − e²_active
29:             var_other ← f_sum · e_other square − e²_other
30:             n_1 ← f_sum · e_both; n_2 ← e_active · e_other
31:             d ← √(var_active · var_other)
32:             n_1 ← n_1/d; n_2 ← n_2/d
33:             w ← n_1 − n_2
34:             if w = NaN ∨ w = ∞ then
35:                 continue
36:             end if
37:             storeWeight(u_a,u_o,w,now())
38:         end for
39:     end for
40: end function
```

Algorithm 2 Pearson Correlation Coefficient prediction calculation

Require: U set of users

Require: A map of average user ratings

Require: ca a boolean giving the weight for Case Amplification or *null*

```
1: function PREDICT(U,A,ca)
2:     for all u_a ← U do
3:         I ← getItemsNotRatedByUser(u_a)
4:         for all i ∈ I do
5:             κ ← 0; n ← 0
6:             W ← getWeights(u_a,i)
7:             if card(W) = 0 then
8:                 continue
9:             end if
10:            for all w, u_o ∈ W do
11:                r ← getRating(u_o, i)
12:                if ca ≠ null then
13:                    if w ≥ 0 then
14:                        r ← r^ca
```

```
                    else
                        r ← −(−r^ca)
                    end if
                end if
                κ ← κ + |w|
                n ← n + (w · (r − A_{u_o}))
            end for
            p ← A_{u_a} + n/κ
            storeUserAssoc(u_a,i,p,now(),"pearson")
        end for
    end for
end function
```

A.4 Item-based Collaborative Filtering

Algorithm 3 Item-based Collaborative Filtering similarity calculation

Require: I set of items

Require: A map of average ratings. Depending if Pearson, cosine or adjusted cosine similarity is chosen this contains either average ratings between items, all 0s or users

```
function PREDICT(I,A)
    for i ← 1, card(I) do
        for j ← i + 1, card(I) do
            n ← 0; d_1 ← 0; d_2 ← 0
            R ← getItemsRatedTogether(I_i, I_j)      ▷ ratings stored in a tuple (r_i, r_j)
            for all r ∈ R do
                r_{i,difference} ← r_i − A_i; r_{j,difference} ← r_j − A_j
                r_{i,difference squared} ← r^2_{i,difference}; r_{j,difference squared} ← r^2_{j,difference}
                n ← n + r_{i,difference} · r_{j,difference}
                d_1 ← d_1 + r_{i,difference squared}
                d_2 ← d_2 + r_{j,difference squared}
            end for
            d_1 ← √d_1; d_2 ← √d_2
            s ← n/(d_1 · d_2)
            storeItemAssoc(I_i,I_j,s,now(),"itembased")
        end for
    end for
end function
```

Algorithm 4 Item-based Collaborative Filtering prediction calculation

Require: U set of users

Require: I set of items

function CALCULATESIMILARITIES(U,I)
    for all $i \in I$ **do**
        $IA \leftarrow$ getItemAssocsFrom(i) ▷ get all associations this item has, these are the similarities
        for all $u \in U$ **do**
            if didUserRateItem(u,i) **then**
                continue
            end if
            $R \leftarrow$ getRatingMapOfUser(u) ▷ all ratings by the user stored in a map
            $n \leftarrow 0$; $d \leftarrow 0$
            for all $a \in IA$ **do**
                $r \leftarrow R_a$ ▷ get the rating the user has for the associated item
                if $r = null$ **then** ▷ user did not rate the other item
                    continue
                end if
                $s \leftarrow a_{value}$ ▷ the association value is the calculated similarity
                $n \leftarrow n + s \cdot r$
                $d \leftarrow d + |s|$
            end for
            $p \leftarrow n/d$
            storeUserAssoc(u,i,p,now(),"itembased")
        end for
    end for
end function

A.5 Slope One

Algorithm 5 Slope One calculation of average deviations

Require: U set of users

Require: R set of ratings

```
1: function CALCULATEAVERAGEDEVIATIONS(U,R)
2:     for all u ∈ U do
3:         for i ← 1, card(R) do
4:             for j ← i, card(R) do
5:                 d ← R_j − R_i
6:                 c ← 1
7:                 if existsRating(j,i) then
8:                     r ← getRating(j.i)
9:                     c ← r_count + 1
10:                    d ← r_difference + d
11:                end if
12:                storeDifference(i,j,d,c,now())
13:            end for
14:        end for
15:    end for
16: end function
```

Algorithm 6 Slope One calculation of predictions

Require: U set of users

Require: I set of items

```
1: function PREDICT(U,I)
2:     for all u ∈ U do
3:         P ← queryDatabase(query_1)          ▷ See below for query_1
4:         for all p ∈ P do
5:             storeUserAssoc(u,p_item,p_value,now(),"slopeone")
6:         end for
7:     end for
8: end function
```

$query_1$ referenced in algorithm 6

```
SELECT
        a.tenantId,
        a.userId,
        d.item1id AS itemId,
        d.item1typeId AS itemTypeId,
        max(a.actionTime) AS actionTime,
        sum(d.difference + d.count * a.ratingValue) / sum(d.count)
                AS predictedRatingValue
FROM
        cont_latestaction AS a,
        so_differences AS d
WHERE
        a.userId = 1 AND
        a.tenantId = 0 AND
        a.actionTypeId = 3 AND
        a.itemTypeId = 1 AND
        d.tenant = a.tenantId AND
        d.item1typeid = a.itemTypeId AND
        d.item2typeId = a.itemTypeId AND
        d.item1id <> a.itemId AND
        d.item2id = a.itemId
GROUP BY
        d.item1id, d.item1typeid
ORDER BY
        predictedRatingValue DESC
```

B Detailed Results

B.1 Detailed Accuracy Results

Dataset	MAE	RSME
	Pearson Correlation Coefficient	
1	0.7635520	0.9702232
2	0.7505339	0.9562967
3	0.7437165	0.9451121
4	0.7452924	0.9454570
5	0.7526264	0.9495611
Average	0.7511443	0.9533300
	Pearson Correlation Coefficient with Inverse User Frequency	
1	0.7639219	0.9705908
2	0.7506847	0.9563386
3	0.7441498	0.9455021
4	0.7452835	0.9453658
5	0.7531001	0.9501610
Average	0.7514280	0.9535917
	Pearson Correlation Coefficient with Case Amplification	
1	0.7705180	0.9818705
2	0.7577260	0.9680626
3	0.7516665	0.9565630
4	0.7528841	0.9571213
5	0.7617860	0.9630655
Average	0.7589161	0.9653366
	Pearson Correlation Coefficient w. Inverse User Frequency & Case Amplification	
1	0.7712756	0.9826301
2	0.7580820	0.9683381
3	0.7524764	0.9572186
4	0.7528907	0.9570222
5	0.7627600	0.9643410
Average	0.7594970	0.9659100

Table 7: Accuracy of Pearson Correlation Coefficient and variants

Dataset	MAE	RSME
	Item-based CF with Pearson	
1	0.8451958	1.0560540
2	0.8326298	1.0403383
3	0.8187247	1.0239130
4	0.8243788	1.0293141
5	0.8288750	1.0322253
Average	0.8299608	1.0363689
	Item-based CF with Cosine	
1	0.8526758	1.0655582
2	0.8399762	1.0487119
3	0.8259486	1.0324829
4	0.8312965	1.0373955
5	0.8355467	1.0399038
Average	0.8370888	1.0448105
	Item-based CF with Adjusted Cosine	
1	0.8184876	1.0260597
2	0.8064411	1.0105663
3	0.7932508	0.9956074
4	0.7985840	1.0004563
5	0.8032043	1.0027744
Average	0.8039935	1.0070928

Table 8: Accuracy of Item-based CF and variants with $sim(i, j) + 1$

Dataset	MAE	RSME
	Item-based CF with Pearson	
1	0.8665182	1.0569228
2	0.8587559	1.0484764
3	0.8448666	1.0342531
4	0.8448015	1.0334911
5	0.8492508	1.0371189
Average	0.8528386	1.0420525
	Item-based CF with Cosine	
1	0.8517630	1.0646594
2	0.8391258	1.0478695
3	0.8250294	1.0316673
4	0.8304550	1.0365816
5	0.8347454	1.0391338
Average	0.8362237	1.0439823
	Item-based CF with Adjusted Cosine	
1	0.9255928	1.1145369
2	0.9089847	1.0961304
3	0.8859999	1.0736999
4	0.8881302	1.0764718
5	0.8935878	1.0830849
Average	0.9004591	1.0887848

Table 9: Accuracy of Item-based CF and variants with centered a_j

Dataset	MAE	RSME
	Slope One	
1	0.7906500	1.0148195
2	0.7815405	1.0037249
3	0.7764282	0.9944784
4	0.7746065	0.9891329
5	0.7709014	0.9827666
Average	0.7788253	0.9969845
	Weighted Slope One	
1	0.7748267	0.9990537
2	0.7653394	0.9824848
3	0.7668576	0.9829939
4	0.7643000	0.9776460
5	0.7696126	0.9824855
Average	0.7681873	0.9849328

Table 10: Accuracy of Slope One and variants

B.2 Detailed Performance Results

Dataset	Similarity Calculation	Recommendation Calculation	# of Recommendations	Recommendations / second
	Pearson Correlation Coefficient			
1	0h 10m 29s	0h 18m 55s	20000	11.338
2	0h 11m 33s	0h 19m 7s	20000	10.870
3	0h 10m 35s	0h 19m 48s	20000	10.971
4	0h 9m 33s	0h 18m 39s	20000	11.820
5	0h 9m 38s	0h 18m 3s	20000	12.041
Average	0h 10m 22s	0h 18m 54s	20000.000	11.408
	Pearson Correlation Coefficient with Inverse User Frequency			
1	0h 15m 40s	0h 27m 50s	20000	7.663
2	0h 14m 18s	0h 21m 33s	20000	9.298
3	0h 14m 53s	0h 21m 49s	20000	9.083
4	0h 15m 58s	0h 26m 11s	20000	7.908
5	0h 15m 42s	0h 26m 0s	20000	7.994
Average	0h 15m 18s	0h 24m 41s	20000.000	8.389
	Pearson Correlation Coefficient with Case Amplification			
1	0h 24m 42s	0h 33m 30s	20000	5.727
2	0h 32m 43s	0h 40m 32s	20000	4.551
3	0h 33m 53s	0h 42m 15s	20000	4.378
4	0h 30m 23s	0h 46m 38s	20000	4.328
5	0h 27m 16s	0h 34m 38s	20000	5.385
Average	0h 29m 47s	0h 39m 31s	20000.000	4.874
	Pearson Correlation Coefficient w. Inverse User Frequency & Case Amplification			
1	0h 12m 34s	10h 24m 3s	1457039	38.145
2	0h 11m 0s	10h 28m 40s	1456296	37.944
3	0h 10m 57s	11h 5m 35s	1456474	35.881
4	0h 11m 4s	11h 0m 49s	1465695	36.358
5	0h 10m 41s	10h 53m 19s	1455596	36.536
Average	0h 11m 15s	10h 46m 29s	1458220.000	36.973
Average	0h 16m 41s	3h 2m 24s	379555.000	15.411

Table 11: Performance of Pearson Correlation Coefficient and variants

Dataset	Similarity Calculation	Recommendation Calculation	# of Recommendations	Recommendations / second
	Item-based CF with Pearson			
1	3h 22m 27s	1h 19m 31s	1472447	87.034
2	2h 58m 51s	0h 42m 40s	1470947	110.672
3	2h 11m 3s	0h 47m 8s	1471837	137.671
4	3h 8m 11s	0h 43m 23s	1480727	106.573
5	3h 7m 55s	1h 23m 37s	1471531	90.322
Average	2h 57m 41s	0h 59m 16s	1473497.800	106.455
	Item-based CF with Cosine			
1	1h 45m 2s	1h 5m 43s	1472447	143.723
2	1h 17m 51s	0h 50m 21s	1470947	191.231
3	1h 8m 55s	0h 50m 24s	1471837	205.593
4	1h 26m 49s	0h 52m 14s	1480727	177.481
5	2h 59m 13s	1h 16m 7s	1471531	96.053
Average	1h 43m 34s	0h 58m 58s	1473497.800	162.816
	Item-based CF with Adjusted Cosine			
1	2h 42m 45s	1h 0m 11s	1472447	110.081
2	2h 39m 10s	1h 23m 33s	1470947	101.006
3	1h 56m 56s	1h 24m 2s	1471837	122.063
4	2h 24m 33s	0h 49m 44s	1480727	127.025
5	1h 35m 58s	0h 48m 20s	1471531	169.962
Average	2h 15m 52s	1h 5m 10s	1473497.800	126.027
Average	2h 19m 3s	1h 1m 8s	1473497.800	131.766

Table 12: Performance of Item-based CF and variants with $sim(i, j) + 1$

Dataset	Similarity Calculation	Recommendation Calculation	# of Recommendations	Recommendations / second
	Item-based CF with Pearson			
1	3h 22m 27s	0h 51m 53s	1472447	96.491
2	2h 58m 51s	1h 4m 35s	1470947	100.708
3	2h 11m 3s	0h 56m 17s	1471837	130.946
4	3h 8m 11s	1h 28m 7s	1480727	89.319
5	3h 7m 55s	1h 4m 8s	1471531	97.304
Average	2h 57m 41s	1h 5m 0s	1473497.800	102.954
	Item-based CF with Cosine			
1	1h 45m 2s	1h 4m 49s	1472447	144.485
2	1h 17m 51s	0h 55m 33s	1470947	183.776
3	1h 8m 55s	0h 56m 7s	1471837	196.193
4	1h 26m 49s	1h 10m 3s	1480727	157.323
5	2h 59m 13s	0h 40m 6s	1471531	111.827
Average	1h 43m 34s	0h 57m 20s	1473497.800	158.721
	Item-based CF with Adjusted Cosine			
1	2h 42m 45s	1h 25m 2s	1472447	99.041
2	2h 39m 10s	1h 26m 21s	1470947	99.854
3	1h 56m 56s	1h 31m 37s	1471837	117.625
4	2h 24m 33s	1h 16m 13s	1480727	111.787
5	1h 35m 58s	1h 3m 11s	1471531	154.103
Average	2h 15m 52s	1h 20m 29s	1473497.800	116.482
Average	2h 19m 3s	1h 7m 36s	1473497.800	126.052

Table 13: Performance of Item-based CF and variants with centered a_j

Dataset	Similarity Calculation	Recommendation Calculation	# of Recommendations	Recommendations / second
	Slope One			
1	0h 31m 54s	0h 19m 19s	1472447	479.156
2	0h 36m 9s	0h 14m 55s	1346515	439.463
3	0h 30m 52s	0h 14m 42s	1267205	463.499
4	0h 31m 21s	0h 14m 53s	1276178	460.050
5	0h 32m 30s	0h 14m 13s	1271996	453.798
Average	0h 33m 42s	0h 15m 36s	1326868.200	459.193
	Weighted Slope One			
1	0h 25m 5s	0h 14m 10s	1472447	625.243
2	0h 25m 19s	0h 13m 44s	1346515	574.697
3	0h 26m 51s	0h 17m 22s	1267205	477.650
4	0h 27m 51s	0h 13m 23s	1276178	515.836
5	0h 27m 8s	0h 12m 59s	1271996	528.457
Average	0h 26m 27s	0h 14m 20s	1326868.200	544.377
Average	0h 29m 30s	0h 14m 58s	1326868.200	501.785

Table 14: Performance of Slope One and variants