OpenCF Documentation

Collaborative Filtering Implementation and Evaluation Toolkit

Jun Yang
School of Electronic and Computer Engineering
Peking University

June 24, 2014

Contents

1 Introduction

The explosive growth of the world-wide-web and the emergence of e-commerce has led to the development of *recommender systems* - a personalized information filtering technology to identify a set of items that will be of interest to a certain user.[2]

Collaborative filtering is the most successful technology for building recommender systems, in which similarity between items/users will be analyzed using historical data. Other technics also exist, such as clusters, classifiers, LSM, tf-idf.

User-based Collaborative Filtering is extensively used in many commercial recommender systems:

$$P_{u,i} = \sum_{Rank(s_{u,v}) \geq k} s_{u,v} \cdot R_{v,i} \tag{1}$$

While User-based Collaborative Filtering suffers problems such as sparcity scalability and new-user problem, *Item-based Collaborative Filtering* identifies a neighborhood of items, then analyze this neighborhood to find out recommends:

$$P_{u,i} = \sum_{Rank(s_{i,j}) \geq k} s_{i,j} \cdot R_{u,j} \tag{2}$$

In OpenCF, both user-based and item-based collaborative filtering are implemented. Moreover, there are also various normalization and summing appoarches. Source code and manual page can be accessed here.

2 Item-based CF

2.1 Terminology

- Users U
  The collection of users.
- Items I
  The collection of items.
- Rating R
  $R_{i,j}$ represents user i's rating for item j.
- Similarity S
  $S_{i,j}$ represents the similarity between user/item i and j.
- Prediction P
  $P_{i,j}$ represents user i's predicted rating for item j.

2.2 Procedure

The procedure of *Item-based Collaborative Filtering* can be listed as as follows:

1. Get input ratings: R
2. Compute similarities: S

$$S_{i,j} = sim(\vec{R_i}, \vec{R_j}) \tag{3}$$

   where:
   $\vec{R_i} = (R_{1,i}, R_{2,i}, \cdots, R_{N,i})$
   $\vec{R_j} = (R_{1,j}, R_{2,j}, \cdots, R_{N,j})$

3. Identify k neighbors $\{j_1, j_2, \cdots, j_k\}$ for each item.
4. For each user who bought a set of items C:
   (a) Neighbors of $c \in C$ forms candidates: N, remove $n \in N$ that is already in C.
   (b) For each $n \in N$, compute its similarity with C as the sum of similarities with $c \in C$.
   (c) Sort N respect to that similarities.
5. The sorted N for each user forms recommendation set.

3 Implementation

3.1 Rating

Rating is used to generate *rating matrix* from *sequential transaction dataset*. If you already have a rating matrix, skip this.

The compatible data format of sequential transaction:

```
#user_id     item_id type     month     day
847750       2235     0        4         15
847750       2215     1        4         16
6694750      14020    0        6         16
...
```

The information of the example dataset is in Table 1.

No. Users	860
No. Items	7842
No. Lines	42531
Begin Date	15th, April
End Date	15th, August

Table 1: Alibaba Dataset 2013 4-8

Ratings matrix R is required in order to apply collaborative filtering. It's generated by analyzing sequential behaviors of users. If user u bought item i, $R_{u,i}$ is set to 1.

If a certain user clicked an item, it's likely that he'll buy it. But the probability decreases as time goes. Make conditional probability statistics about probability with respective to time:

$$p = f(t) \tag{4}$$

Then derive rating value by integration of $f(t)$ during the prediction timespan:

$$R_{u,i} = \int_{prediction\ timespan} f(t)dt \tag{5}$$

Figure 1 shows the integrated probability curve for click behavior.

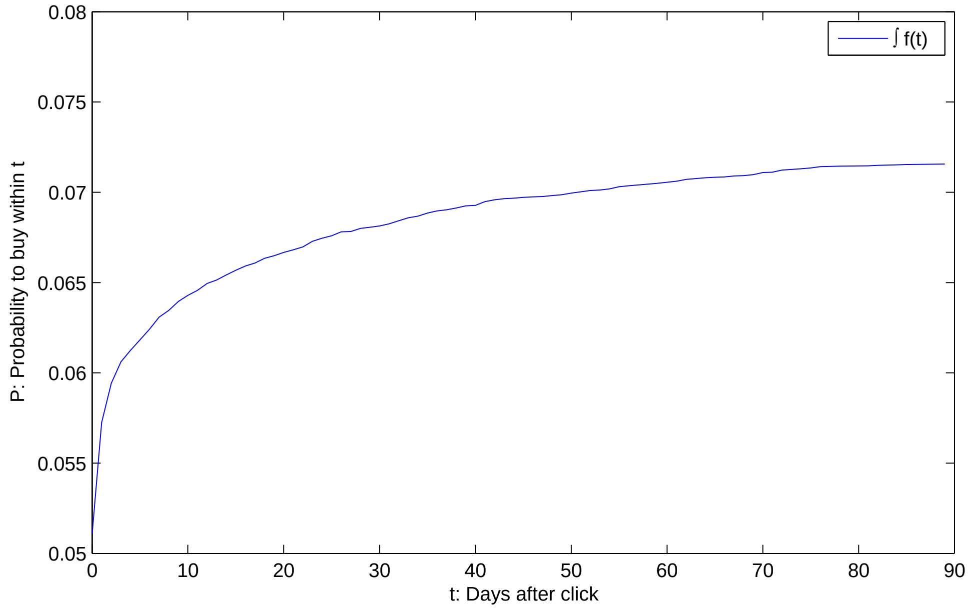


Figure 1: Integrated Transaction Probability for Click Behavior

3.2 Item Similarity

Similarity between items are usually computed in the space of *users*, treating each item as a vector. Classes of similarity methods includes *cosine similarity*, *adjusted cosine similarity* and *Pearson correlation coefficient*, all of which are implemented.

Cosine Similarity

$$sim(\vec{u}, \vec{v}) = \cos(\vec{u}, \vec{v}) = \frac{\vec{u} \cdot \vec{v}}{||\vec{u}|| \cdot ||\vec{v}||} \tag{6}$$

tends to be high when if each user who purchases u also purchases v as well. At the same time, frequently purchased items are de-emphasized by the denominator.

Adjusted Similarity[3]

$$sim(i, j) = \frac{\sum_{u \in U}(R_{u,i} - \overline{R_u})(R_{u,j} - \overline{R_u})}{\sqrt{\sum_{u \in U}(R_{u,i} - \overline{R_u})^2}\sqrt{\sum_{u \in U}(R_{u,j} - \overline{R_u})^2}} \tag{7}$$

takes different rating scales between users are taken into account.

Pearson Correlation Coefficient[3]

$$sim(i, j) = \frac{\sum_{u \in U}(R_{u,i} - \overline{R_i})(R_{u,j} - \overline{R_j})}{\sqrt{\sum_{u \in U}(R_{u,i} - \overline{R_i})^2}\sqrt{\sum_{u \in U}(R_{u,j} - \overline{R_j})^2}} \tag{8}$$

is a measure of the linear dependence between two variables. Here we need isolate co-rated cases in advance.

3.3 Prediction

$P_{u,i}$ is computed by summing the similarities between item i and items in user u's basket:

$$P_{u,i} = \sum_{Rank(s_{i,j}) \geq k} s_{i,j} \cdot R_{u,j} \tag{9}$$

This summing approach often results in high predictions when infrequently purchased items have a moderate overlap. One solution is to treat recommendations from each item j as independent events. Thus:

$$P_{u,i} = 1 - \prod_{Rank(s_{i,j}) \geq k} (1 - s_{i,j} \cdot R_{u,j}) \tag{10}$$

Only if all recommendation fails, $R_{u,i} = 0$.

Another solution is similarity normalization. We can normalize the k similarities so that they add-up to 1.

$$s'_{i,j} = \frac{s_{i,j}}{\sum_{Rank(s_{i,l}) \geq k} s_{i,l}} \tag{11}$$

At the same time, for users who purchases a lot, each item reflects his appetite less. We can normalize R before doing prediction, which is called *Row Normalization*.

3.4 Post Procession

In the case of ecommerce transaction, if user u once purchased item i, whether he'll purchase another depends on the lifetime of item i:

$$\tau_i \approx \frac{\sum_{u \in U, R_{u,i}=1} R_{u,i}}{\sum_{u \in U, R_{u,i}=1} 1 \cdot TimeSpan} \tag{12}$$

Thus the modified $P_{u,i}$ should be:

$$P'_{u,i} = \begin{cases} P_{u,i} & \text{if } R_{u,i} \neq 1 \\ \frac{Timespan_{prediction}}{\tau_i} \cdot P_{u,i} & \text{otherwise} \end{cases} \tag{13}$$

4 Evaluation

The quality of recommender system is measured by *recall* and *precision*. *recall* is the percentage of hits in the test set. *precision* is the percentage of hits in the prediction set. $F1$ is used to combine these two parameters:

$$F1 = \frac{2}{1/recall + 1/precision} \tag{14}$$

4.1 DataSet

In order to measure recommendation quality, we split the dataset into train set and test set, as in Table 2.

	Train Set	Test Set
Begin Date	15th April	15th July
End Date	14th July	14th August
No. Transaction	4971	2013

Table 2: Train and Test Dataset

4.2 Pre and Post Processing

The effect of *behavior statistic* is shown in Figure 2, compared to neglecting behaviors other than purchasement.

The effect of *post-processing* is shown in Figure 3. The precision is better between 0.01 and 0.03. The smaller minimum F1 implies that post processing needs to be conducted judiciously.

4.3 Similarity Method Comparison

Figure 4 shows the results of *correlation* and *cosine* are approximate, while *adjusted cosine* is bad. It is because adjusted cosine removes the difference in rating scales, which means positive rating could contain negative information. While for ecommerce data, positive rating is always positive.

4.4 Summing and Normalization

Apparently, summing appoach using conditional probability is better, as Figure 5 shows.

Figure 6 shows *similarity normalization* does no good in prediction, the primary reason is the dataset is so small that many items do not have k neighbors. The sparse item will divide a smaller denominator, rather than larger(expected).

Figure 7 shows *row normalization* does no good. The side-effect of row normalization is normalized purchasing power for each user, while users who have purchased alot certainly will purchase a lot. Only in the case for Top-N Recommendation, in which we only recommend N item for each user. Normalizing purchasing power has no side-effect then.[1]

4.5 Comparison with User-based CF

Figure 8 compares item-based CF with user-based CF. Almost no difference appears, while item-based CF is faster in realtime recommendation.

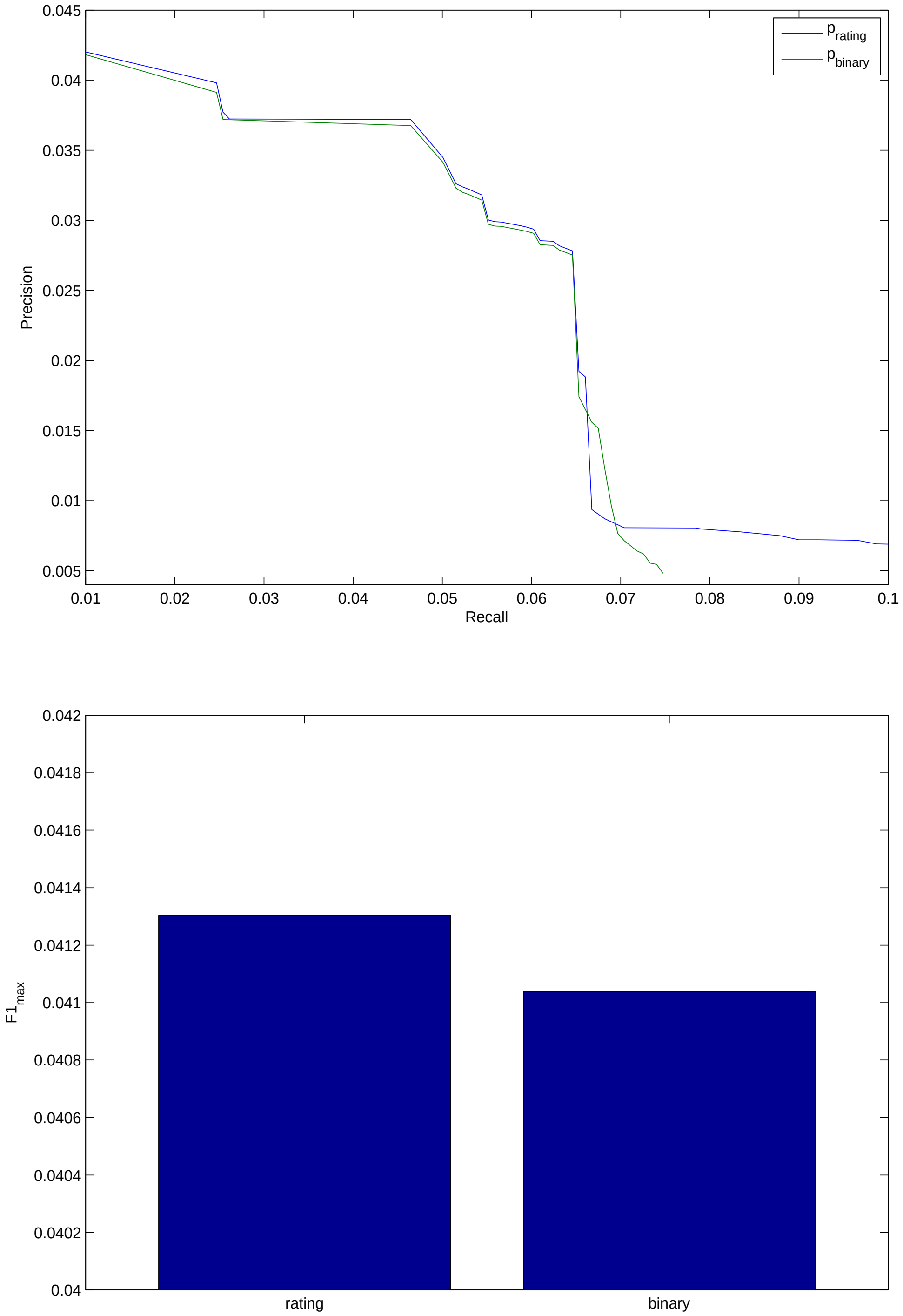


Figure 2: The Effect of Behavior Statistic

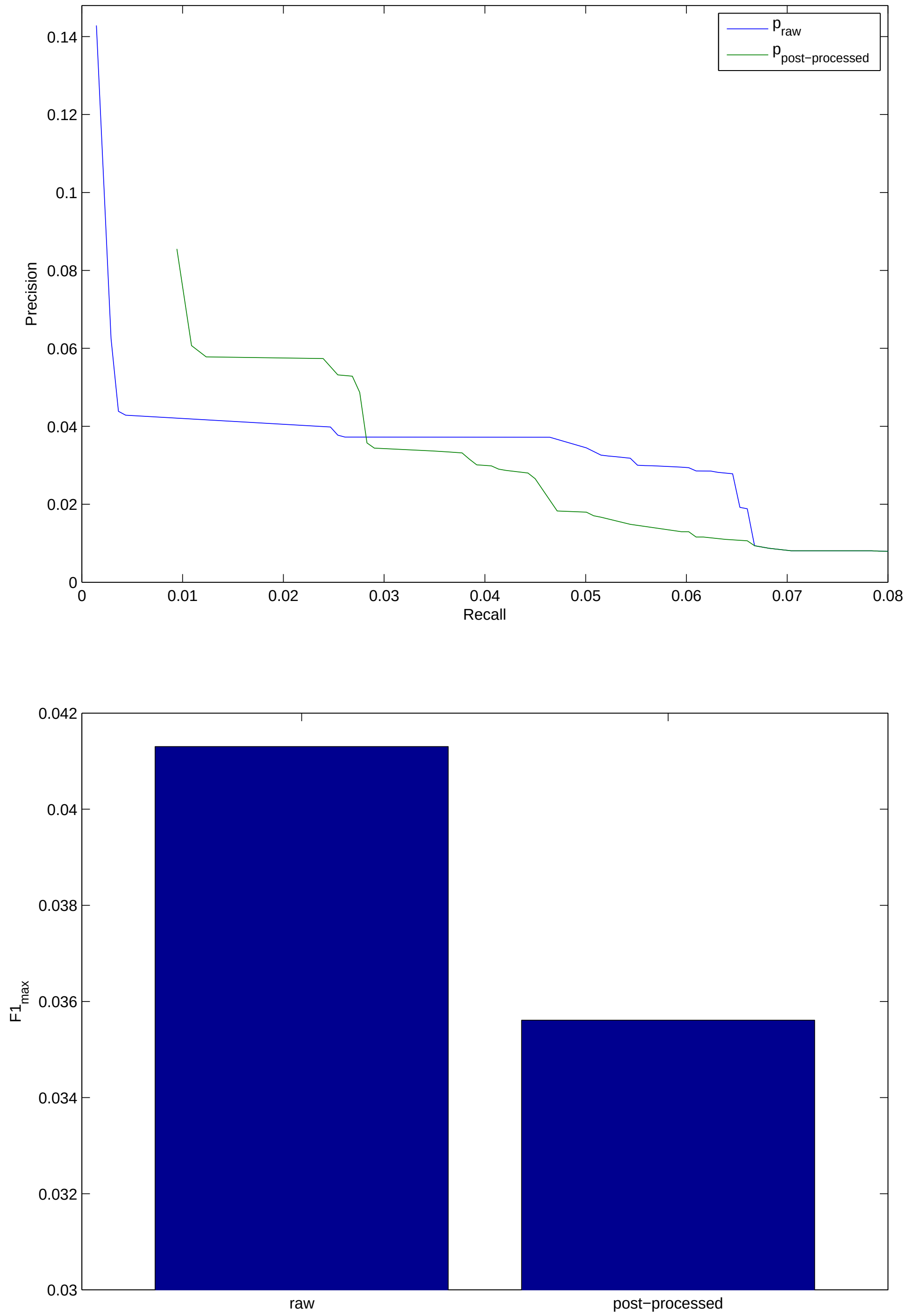


Figure 3: The Effect of Post Processing

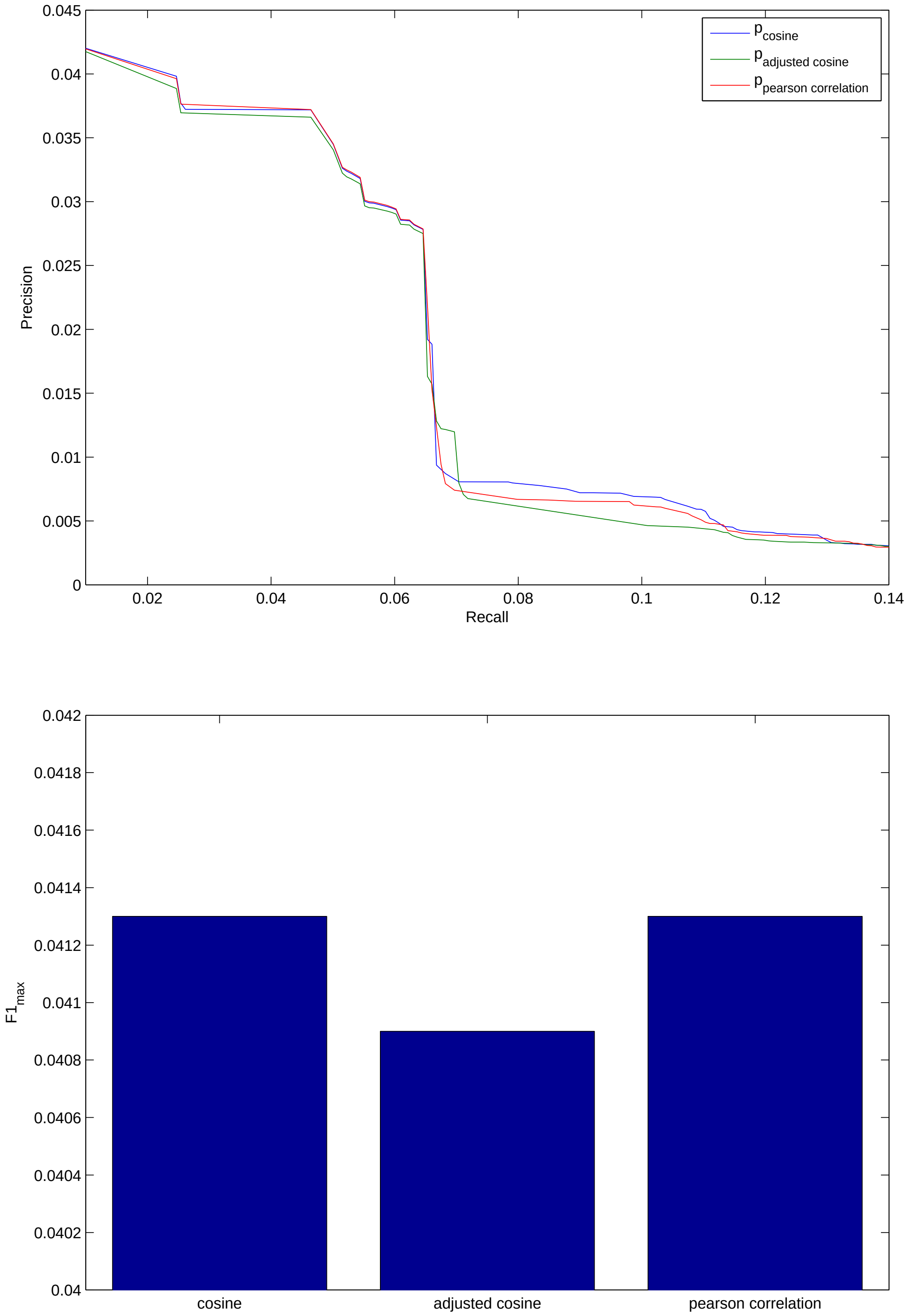


Figure 4: Comparison Between Different Similarity Methods

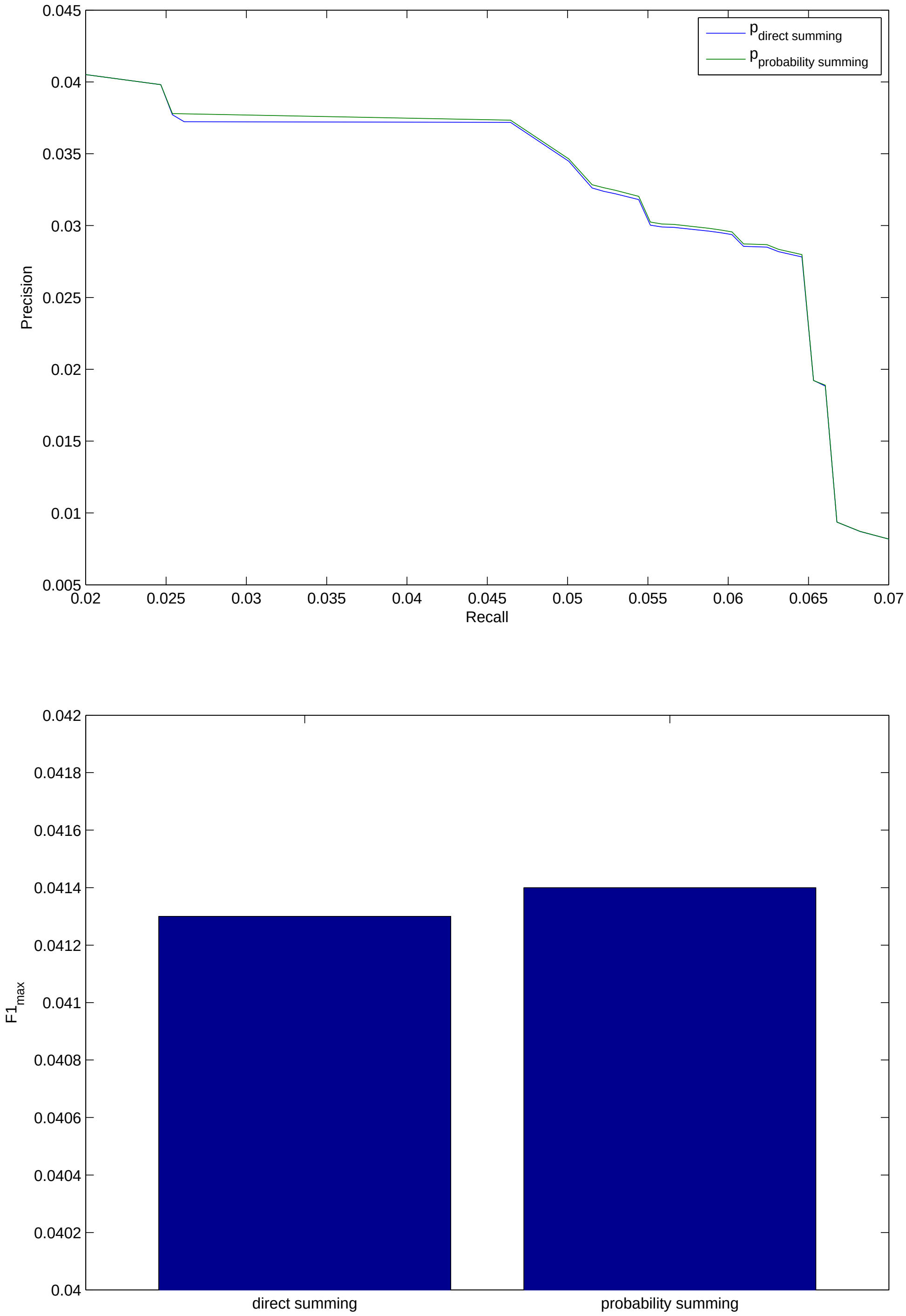
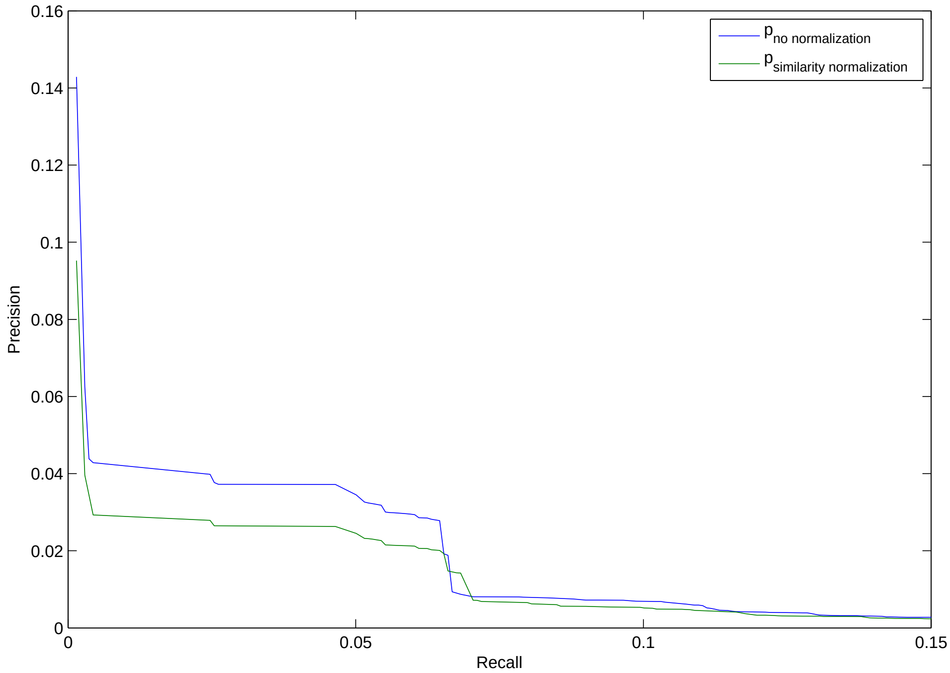


Figure 5: Comparison Between Direct Summing and Probability Summing



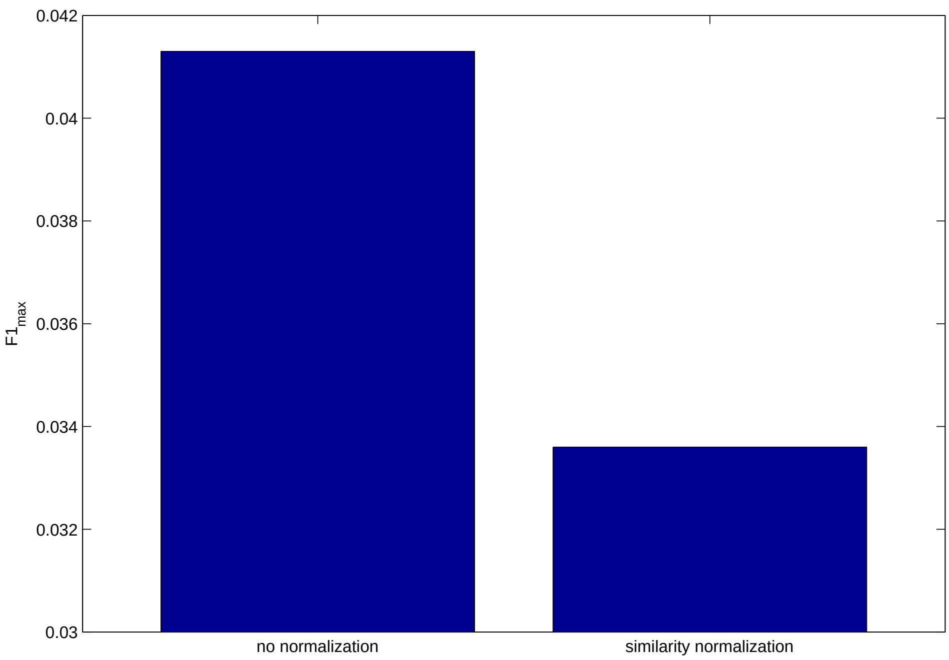
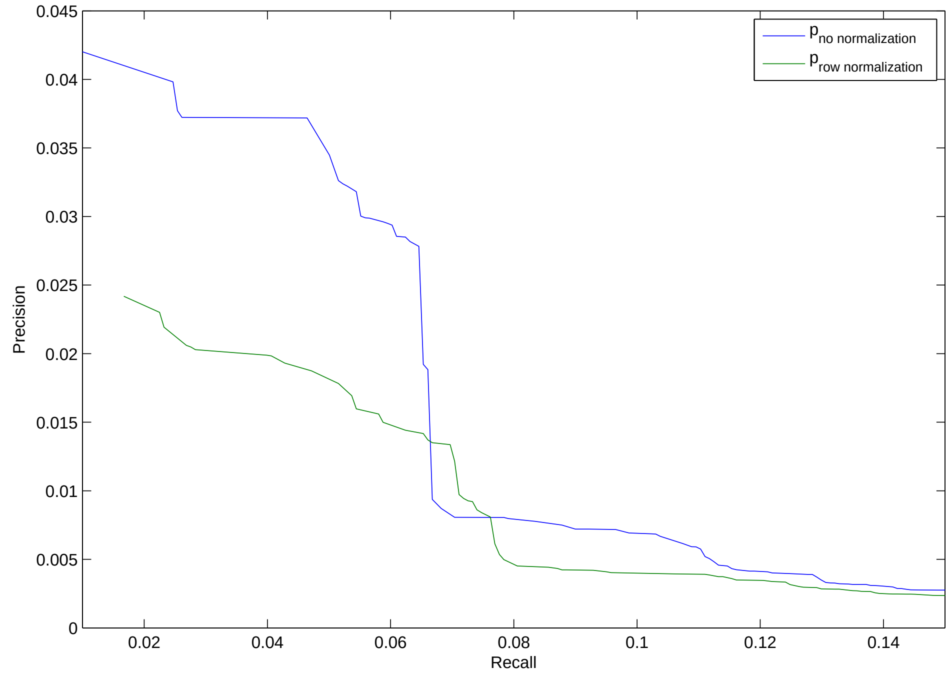


Figure 6: Effect of Similarity Normalization



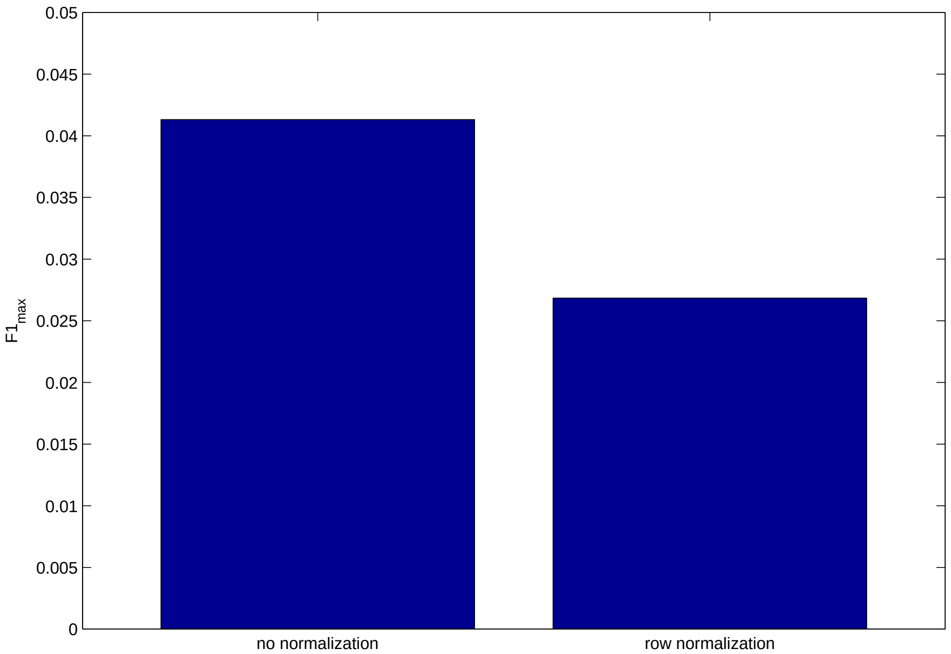


Figure 7: Effect of Row Normalization

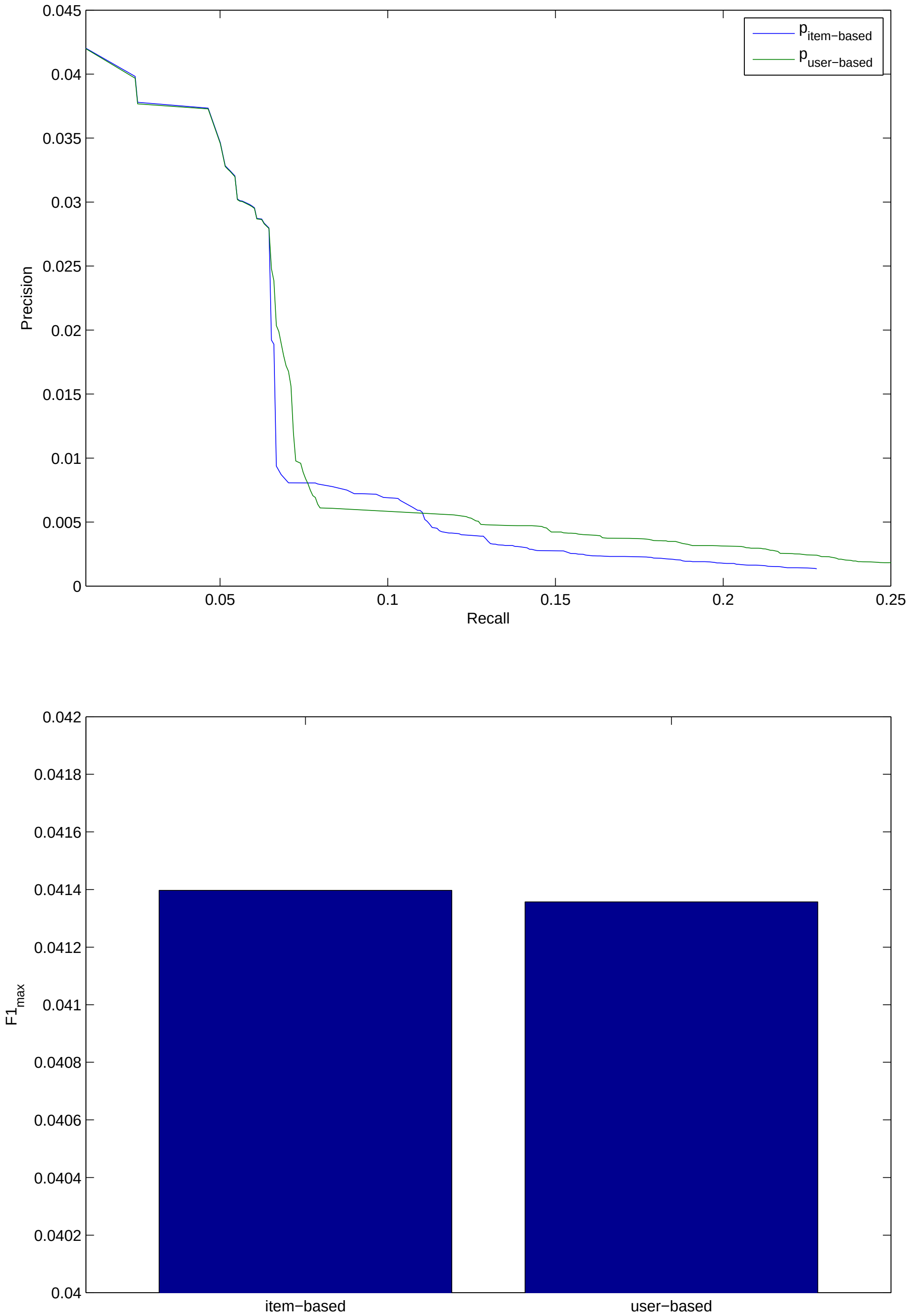


Figure 8: Comparison Between Different Similarity Methods

5 References

[1] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[2] G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 247–254. ACM, 2001.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.