

A User-Item Relevance Model for Log-based Collaborative Filtering

Jun Wang¹, Arjen P. de Vries^{1,2}, and Marcel J.T. Reinders¹

¹ Information and Communication Theory Group,
Faculty of Electrical Engineering, Mathematics and Computer Science,
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands
j.wang@ewi.tudelft.nl,
² CWI
Amsterdam, The Netherlands

Abstract. Implicit acquisition of user preferences makes *log-based* collaborative filtering favorable in practice to accomplish recommendations. In this paper, we follow a formal approach in text retrieval to re-formulate the problem. Based on the classic probability ranking principle, we propose a probabilistic user-item relevance model. Under this formal model, we show that user-based and item-based approaches are only two different factorizations with different independence assumptions. Moreover, we show that smoothing is an important aspect to estimate the parameters of the models due to data sparsity. By adding linear interpolation smoothing, the proposed model gives a probabilistic justification of using TF×IDF-like item ranking in collaborative filtering. Besides giving the insight understanding of the problem of collaborative filtering, we also show experiments in which the proposed method provides a better recommendation performance on a music play-list data set.

1 Introduction

Generally, a collaborative filtering algorithm uses a collection of user profiles to identify interesting “information” for these users. A particular user gets a recommendation based on the user profiles of other, similar users. User profiles are commonly obtained by explicitly asking users to rate the items. Collaborative filtering has often been formulated as a self-contained problem, apart from the classic information retrieval problem (i.e. ad hoc text retrieval). Research started with heuristic implementations of “Word of Mouth” (e.g. user-based approaches [1]), and moved to item-based approaches [17], and, more recently, various model-based approaches have been introduced ([8, 13]).

Previous research ([3]) has shown that users are very unlikely to provide an explicit rating. Asking the user to rate items is annoying and should be avoided when possible. Alternatively, user profiles can also be obtained by implicitly observing user interactions with the system. For instance, music play-list indicates the music taste of a user, and web query logs could indicate the interest of a user for certain web sites. The implicit acquisition of user preferences makes

the so-called “log-based” collaborative filtering more favorable in practice (see Section 2).

Therefore, this paper focuses on log-based collaborative filtering. We identify a close relationship between log-based collaborative filtering and text information retrieval. We build a user-item relevance model to re-formulate collaborative filtering under the classic probability ranking principle. Given our user-item relevance models, we also introduce a linear interpolation *smoothing* into collaborative filtering. We show that the smoothing is important to estimate the model parameters correctly due to the data sparsity. Similar to the situation in text retrieval, the user-item relevance model provides a probabilistic justification of using $TF \times IDF$ like item weighting in collaborative filtering.

2 Background

2.1 Rating-based Collaborative filtering

The preference information about items can either be based on ratings (explicit interest functions) or log-archives (implicit interest functions). Their differences lead, to our view, to two different ways to approach collaborative filtering: *rating-based* and *log-based*. Rating-based collaborative filtering is based on user profiles that contain rated items. The majority of the literature addresses rating-based collaborative filtering, which has been studied in depth ([13]). Different rating-based approaches are often classified as memory-based or model-based. In the memory-based approach, all rating examples are stored *as-is* into memory (in contrast to learning an abstraction). In the prediction phase, similar users or items are sorted based on the memorized ratings. Based on the ratings of these similar users or items, a recommendation for the query user can be generated. Examples of memory-based collaborative filtering include item correlation-based methods ([17]), user clustering ([20]) and locally weighted regression ([1]). The advantage of the memory-based methods over their model-based alternatives is that they have less parameters to be tuned, while the disadvantage is that the approach cannot deal with data sparsity in a principled manner.

In the model-based approach, training examples are used to generate a model that is able to predict the ratings for items that a query user has not rated before. Examples include decision trees ([1]), latent class models ([8]), and factor models ([2]). The ‘compact’ models in these methods could solve the data sparsity problem to a certain extent. However, the requirement of tuning an often significant number of parameters or hidden variables has prevented these methods from practical usage.

Recently, to overcome the drawbacks of these approaches to collaborative filtering, researchers have started to combine both memory-based and model-based approaches ([14, 19]).

2.2 Log-based Collaborative filtering

Implicit interest functions usually generate binary-valued preferences. That is a one indicates as a “file is downloaded”, or a “web-site is visited”. Few log-based

collaborative filtering approaches that deal with such data have been developed thus far. Two examples are the item-based Top-N collaborative filtering approach ([4, 10]) and Amazon’s item-based collaborative filtering ([12]).

The following characteristics make log-based collaborative filtering more similar to the problem of text retrieval than the rating-based approaches:

- Log-based user profiles, e.g., play-lists, are usually binary-valued. Usually, one means ‘relevance’ or ‘likeness’, and zero indicates ‘non-relevance’ or ‘non-likeness’. Moreover, in most of the situations, non-relevance and non-likeness are hardly observed. This is similar to the concept of ‘relevance’ in text retrieval.
- The goal for rating-based collaborative filtering is to predict the rating of users, while the goal for the log-based algorithms is to rank the items to the user in order of decreasing relevance. As a result, evaluation is different. In rating-based collaborative filtering, the mean square error (MSE) of the predicted rating is used, while in log-based collaborative filtering, recall and precision are employed.

Therefore, this paper proposes to apply the probabilistic framework developed for text retrieval to log-based collaborative filtering. We consider the following formal setting. The information that has to be filtered, e.g., images, movies or audio files, is represented as a set of *items*. We introduce discrete random variables $U \in \{u_1, \dots, u_K\}$ and $I \in \{i_1, \dots, i_M\}$ to represent a user and an item in the collection, respectively. K is the number of users while M is the number of items in the collection. Let L_{u_k} denote a user profile list for user $u_k \in U$. L_{u_k} is a set of items that user u_k has previously shown interest in. $L_{u_k}(i_m) = 1$ (or $i_m \in L_{u_k}$) indicates that item $i_m \in I$, is in the list while $L_{u_k}(i_m) = 0$ (or $i_m \notin L_{u_k}$) otherwise. The number of items in the list is denoted as $|L_{u_k}|$.

The purpose of log-based collaborative filtering is to rank the relevance of a target item to a user. This could be represented by the retrieval status value (RSV) of a target item towards a user, denoted as: $RSV_{u_k}(i_m)$. Heuristic implementations of ‘Word of Mouth’ introduced in [4, 6] give the following basic item-based and user-based approaches to calculate the RSV when we consider the binary case:

$$\begin{aligned}
 \text{User-based : } RSV_{u_k}(i_m) &= \sum_{\text{Top-N similar } u_b} s_U(u_k, u_b) L_{u_b}(i_m) \\
 \text{Item-based : } RSV_{u_k}(i_m) &= \sum_{\forall i_b: i_b \in L_{u_k}} s_I(i_b, i_m)
 \end{aligned} \tag{1}$$

where s_I and s_U are the two similarity measures between two items and two users, respectively. The two commonly used similarity measures are the Pearson Correlation and the Cosine similarity ([1]). Alternatively, frequency counting has been used as a basis for similarity measures in ([4, 10]). To suppress the influence of items that are being purchased frequently, they have introduced a

TF×IDF-like weighting (similarity) function:

$$s_I(i_b, i_m) = \frac{c(i_b, i_m)/c(i_m)}{c(i_b)^\alpha} \quad (2)$$

where $c(i_b, i_m) = \sum_{k=1}^K L_{u_k}(i_m) \cap L_{u_k}(i_b)$ is the number of user profiles in which both items i_b and i_m exist (i.e., items they *co-occur*); and $c(i)$ is the number of user profiles containing item i . α is a tuning parameter.

3 A User-Item Relevance Model

In log-based collaborative filtering, users want to know which items fit their interests best. This section adopts the probabilistic relevance model proposed in text retrieval domain ([11, 15]) to measure the relevance between user interests and items. We intend to answer the following basic question:

- What is the probability that *this* item is relevant to *this* user, given his or her profile.

To answer this question, we first define the sample space of relevance: Φ_R . It has two values: ‘relevant’ r and ‘non-relevant’ \bar{r} . Let R be a random variable over the sample space Φ_R . Likewise, let U be a discrete random variable over the sample space of *user id*’s: $\Phi_U = \{u_1, \dots, u_K\}$ and let I be a random variable over the sample space of *item id*’s: $\Phi_I = \{i_1, \dots, i_M\}$, where K is the number of users and M the number of items in the collection. In other words, U refers to the user identifiers and I refers to the item identifiers.

We then denote P as a probability function on the joint sample space $\Phi_U \times \Phi_I \times \Phi_R$. In a probability framework, we can answer the above basic question by estimating the probability of relevance $P(R = r|U, I)$. The relevance rank of items in the collection Φ_I for a given user $U = u_k$ (i.e. retrieval status value (RSV) of a given target item toward a user) can be formulated as the odds of the relevance:

$$RSV_{u_k}(i_m) = \frac{\log P(r|u_k, i_m)}{\log P(\bar{r}|u_k, i_m)} \quad (3)$$

For simplicity, $R = r$, $R = \bar{r}$, $U = u_k$, and $I = i_m$ are denoted as r , \bar{r} , u_k , and i_m , respectively.

Hence, the evidence for the relevance of an item towards a user is based on both the positive evidence (indicating the relevance) as well as the negative evidence (indicating the non-relevance). Once we know, for a given user, the RSV of each item I in the collection (excluding the items that the user has already expressed interest in), we sort these items in decreasing order. The highest ranked items are recommended to the user.

In order to estimate the conditional probabilities in Eq. 3, i.e. the relevance and non-relevance between the user and the item, we need to factorize the equation along the item or the user dimension. We propose to consider both *item-based generation* (i.e., using items as features to represent the user) and *user-based generation* (i.e., treating users as features to represent an item). Since the

two generative models are very similar to each other, for readability, the remainder of the paper presents our results using the item-based generation model; the analogous model based on user-based generation is given in Appendix A.

3.1 Item-Based Generation

By factorizing $P(\bullet|u_k, i_m)$ with $\frac{P(u_k|i_m, \bullet)P(\bullet|i_m)}{P(u_k|i_m)}$, the following log-odds ratio can be obtained from Eq. (3):

$$RSV_{u_k}(i_m) = \log \frac{P(r|i_m, u_k)}{P(\bar{r}|i_m, u_k)} = \log \frac{P(u_k|i_m, r)}{P(u_k|i_m, \bar{r})} + \log \frac{P(i_m|r)P(r)}{P(i_m|\bar{r})P(\bar{r})} \quad (4)$$

Without explicit evidence for non-relevance, and following the language modelling approach to information retrieval ([11]), we now assume that: 1) independence between u_k and i_k in the non-relevance case (\bar{r}), i.e., $P(u_k, |i_m, \bar{r}) = P(u_k|\bar{r})$; and, 2) equal priors for both u_k and i_m , given that the item is non-relevant. Then the two non-relevance terms can be removed and the RSV becomes:

$$RSV_{u_k}(i_m) = \log P(u_k|i_m, r) + \log P(i_m|r) \quad (5)$$

Note that the two negative terms in Eq. (4) can always be added to the model, when the negative evidences are captured.

To estimate the conditional probability $P(u_k|i_m, r)$ in Eq. (5), consider the following: Instead of placing users in the sample space of user id's, we can also use the set of items that the user likes (L_{u_k}) to represent the user (u_k). This step is similar to using a 'bag-of-words' representation of queries or documents in the text retrieval domain ([16]). This implies: $P(u_k|i_m, r) = P(L_{u_k}|i_m, r)$. We call these representing items as *query items*. Note that, different with the target item i_m , the query items do not need to be ranked since the user has already expressed interest in them.

Further, we assume that the items in the user profile list L_{u_k} (query items) are conditionally independent from each other. Although this naive Bayes assumption does not hold in many real situations, it has been empirically shown to be a competitive approach (e.g., in text classification ([5, 18])). Under this assumption, Eq. (5) becomes:

$$\begin{aligned} RSV_{u_k}(i_m) &= \log P(L_{u_k}|i_m, r) + \log P(i_m|r) \\ &= \sum_{\forall i_b: i_b \in L_{u_k}} \log P(i_b|i_m, r) + \log P(i_m|r) \end{aligned} \quad (6)$$

The conditional probability $P(i_b|i_m, r)$ corresponds to the relevance of an item i_b , given that another item i_m is relevant. This probability can be estimated by counting the number of user profiles that contain both items i_b and i_m , divided by the total number of user profiles in which i_m exists (see also, [10]):

$$P_{ml}(i_b|i_m, r) = \frac{P(i_b, i_m|r)}{P(i_m|r)} = \frac{c(i_b, i_m)}{c(i_m)} \quad (7)$$

3.2 Probability Estimation and Smoothing

Using the frequency count in Eq. (7) to estimate the above probability corresponds to using its maximum likelihood estimator. However, many item-to-item co-occurrence counts will be zero, due to the sparseness of the user-item matrix. Therefore, we apply a smoothing technique to adjust the maximum likelihood estimation ([18]).

In information retrieval ([21]), most smoothing methods apply two different distributions: one for the words that occur in the document, and one for the words that do not. Here, we also adopt this formulation. To estimate $P(i_b|i_m, r)$, we use $P_s(i_b|i_m, r)$, when $c(i_b, i_m) > 0$, while when $c(i_b, i_m) = 0$ (i.e., i_b and i_m do not co-occur in any of the user profiles), we assume the probability is proportional to the general frequency of i_b for the whole user profile set. That is $P(i_b|i_m, r) = \alpha_{i_m} P(i_b|r)$, where α_{i_m} depends on item i_m . Then, the conditional probability between a user and an item can be formulated as follows:

$$\begin{aligned}
& \log P(u_k|i_m, r) = \log P(L_{u_k}|i_m, r) \\
& = \sum_{\forall i_b: i_b \in L_{u_k} \cap c(i_b, i_m) > 0} \log P_s(i_b|i_m, r) + \sum_{\forall i_b: i_b \in L_{u_k} \cap c(i_b, i_m) = 0} \log \alpha_{i_m} P(i_b|r) \\
& = \sum_{\forall i_b: i_b \in L_{u_k} \cap c(i_b, i_m) > 0} \log \frac{P_s(i_b|i_m, r)}{\alpha_{i_m} P(i_b|r)} + \sum_{\forall i_b: i_b \in L_{u_k}} \log \alpha_{i_m} P(i_b|r) \quad (8) \\
& = \sum_{\forall i_b: i_b \in L_{u_k} \cap c(i_b, i_m) > 0} \log \frac{P_s(i_b|i_m, r)}{\alpha_{i_m} P(i_b|r)} + |L_{u_k}| \log \alpha_{i_m} + \sum_{\forall i_b: i_b \in L_{u_k}} \log P(i_b|r)
\end{aligned}$$

Since the last term is independent from the target item i_m , it can be dropped when we calculate the RSV of item i_m . Combining Eq. (6) and Eq. (8), we obtain the following:

$$\begin{aligned}
& RSV_{u_k}(i_m) \\
& = \sum_{\forall i_b: i_b \in L_{u_k} \cap c(i_b, i_m) > 0} \log \frac{P_s(i_b|i_m, r)}{\alpha_{i_m} P(i_b|r)} + |L_{u_k}| \log \alpha_{i_m} + \log P(i_m|r) \quad (9)
\end{aligned}$$

Eq. (9) gives a generative ranking formula. Next, we consider a special case: a linear interpolation smoothing.

The linear interpolation smoothing A linear interpolation smoothing can be defined as a linear interpolation between the maximum likelihood estimation and background model. To use it, if we define:

$$\begin{aligned}
& P_s(i_b|i_m, r) = (1 - \lambda)P_{ml}(i_b|i_m, r) + \lambda P(i_b|r) \\
& \alpha_{i_m} = \lambda
\end{aligned} \quad (10)$$

where $P_{ml}(i_b|i_m, r)$ is the maximum likelihood estimation as given in Eq. (7). The item prior probability $P(i_b|r)$ is used as background model. Furthermore,

the parameter $\lambda \in [0, 1]$ is a parameter that balances the maximum likelihood estimation and background model (a larger λ means more smoothing). Usually, the best value for λ is found from a training data. The linear interpolation smoothing leads to the following RSV:

$$RSV_{u_k}(i_m) = \sum_{\forall i_b: i_b \in L_{u_k} \cap c(i_b, i_m) > 0} \log\left(1 + \frac{(1 - \lambda)P_{ml}(i_b|i_m, r)}{\lambda P(i_b|r)}\right) + \log P(i_m|r) \quad (11)$$

3.3 Discussion

IDF The usage of TF×IDF-like ranking shown in Eq. 2 was studied in [4] and has been shown to have the best performance. However, [4] does not provide the justification about the usage of the *inverse item frequency* ($1/P(i_b|r)$) by probability theory. By considering the log-based collaborative filtering probabilistically and proposing the linear interpolation smoothing, our user-item relevance model in Eq. 11 provides a probabilistic justification. Our ranking formula can directly be interpreted as TF×IDF-like ranking, since:

$$P_{ml}(i_b|i_m, r) \propto c(i_b, i_m)/c(i_m) \text{ and } P(i_b|r) \propto c(i_b) \quad (12)$$

Also, Eq. 11 allows a very intuitive understanding of the statistical ranking mechanisms that play a role in log-based collaborative filtering:

- The relevance rank of a target item i_m is the sum of both its popularity (prior probability $P(i_m|r)$) and its co-occurrence (first term in Eq. 11) with the items (i_b) in the profile list of the target user. The co-occurrence is higher if more user expresses interest in target item (i_m) as well as item i_b . However, the co-occurrence should be suppressed more when the popularity of the item in the profile of the target user ($P(i_b|r)$) is higher.
- When λ approaches 0, smoothing from the background model is minimal. It emphasizes the co-occurrence count, and the ranking becomes equivalent to *coordination level matching* ([7]), which is simply counting the number of times for which $c(i_b, i_m) > 0$. When the λ is equal to 0, the model reduces to the traditional item-based approach ([12]). When the λ approaches 1, the model is more smooth, emphasizing the background model.

Two representations Traditionally, collaborative filtering makes a distinction between user-based and item-based approaches. Our probabilistic user-item relevance model, derived with an information retrieval view on collaborative filtering, demonstrates that the user-based (Eq. 16) and item-based (Eq. 11) models are equivalent from the probabilistic point of view, since they have actually been derived from the same generative relevance model (Eq. 3). The only difference in derivation corresponds to the choice of independence assumptions, leading to the two different factorizations.

Consequently, this formula gives a much better understanding of the underlying statistical assumptions that are made in these two approaches. In the user-based approach, a target item is assumed to be judged or rated independently (see the Appendix A) while in the item-based approach, a target user is assumed to independently judge or rate each query item. Besides the differences in the number of users (K) and the number of items (M), we believe that these underlying assumptions are the major factors to influence the performances of these two approaches in practice.

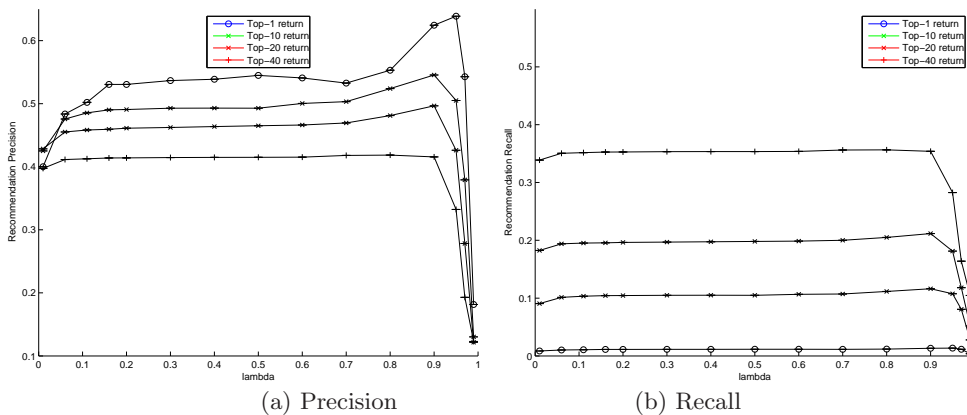


Fig. 1. Performance of the linear interpolation smoothing.

4 Experiments

The standard data set used in the evaluation of collaborative filtering algorithms (MovieLens) is rating-based, which is not suitable for testing our method using log-based user profiles. The user logs we used were collected from the *Audioscrobbler*³ community. The audioscrobbler data set is collected from the play-lists of the users in the community by using a plug-in in the users' media players (for instance, Winamp, iTunes, XMMS etc). Plug-ins send the title (song name and artist name) of every song users play to the Audioscrobbler server, which updates the user's musical profile with the new song. That is, when a user plays a song in a certain time, this transaction is recorded as a form of $\{\text{userID}, \text{itemID}, t\}$ tuple in the database.

For computational reasons, we randomly sampled the data set to limit the number of users to 428 users and the number of items to 516. The sparsity (percentage of zero values in the user-item matrix) is 96.86% .

For cross-validation, we randomly divided this data set into a training set (80% of the users) and a test set (20% of the users). Results are obtained by averaging 5 different runs (sampling of training/test set). The training set was

³ Audioscrobbler is found at <http://www.audioscrobber.com/>.

used to estimate the model. The test set was used for evaluating the accuracy of the recommendations on the new users, whose user profiles are not in the training set. For each test user, 50% of the items of a test user were put into the user profile list. The other 50% of the items were used to test the recommendations. By doing so, the number of items in the user profiles reflects the distribution in the overall data set.

The effectiveness of the log-based collaborative filtering experiments can be measured using the *precision* and *recall* of the recommendations. Precision measures the proportion of recommended items that are ground truth items (only partially known, by the half of the user profiles). The recall measures the proportion of the ground truth items that are recommended. Note that the items in the profiles of the test user represent only a fraction of the items that the user *truly* liked. Therefore, the measured precision *underestimates* the true precision ([9]). In the case of making recommendations, precision seems more important than recall. However, to analyze the behavior of our method, we report both metrics on our experimental results.

We first studied the behavior of the linear interpolation smoothing. For this, we plotted the average precision and recall rate for the different values of the smoothing parameter λ . This is shown in Fig. 1.

Fig. 1 (a) and (b) show that both precision and recall drop when λ reaches its extreme values zero and one. The precision is sensitive to λ , especially the early precision (when only a small number of items are recommended). Recall is less sensitive to the actual value of this parameter, having its optimum at a wide range of values. Effectiveness tends to be higher on both metrics when λ is large; when λ is approximately 0.9, the precision seems optimal. An optimal range of λ near one can be explained by the sparsity of user profiles, causing the prior probability $P(i_b|r)$ to be much smaller than the conditional probability $P_{mi}(i_b|i_m, r)$. The background model is therefore only emphasized for values of λ closer to one. In combination with the experimental results that we obtained, this suggests that smoothing the co-occurrence probabilities with the background model (prior probability $P(i_b|r)$) improves recommendation performance.

Next, we compared our user-item relevance model to other log-based collaborative filtering approaches. Our goal here is to see, using our user-item relevance model, whether the smoothing and inverse item frequency should improve recommendation performance with respect to the other methods. For this, we focused on the item-based generation (denoted as UIR-Item). We set λ to the optimal value 0.9. We compared our results to those obtained with the *Top-N-suggest* recommendation engine, a well-known log-based collaborative filtering implementation ([10]).⁴ This engine implements a variety of log-based recommendation algorithms. We compared our own results to both the item-based TF \times IDF-like version (denoted as ITEM-TFIDF) as well the user-based cosine similarity method (denoted as User-CosSim), setting the parameters to the optimal ones according to the user manual. Additionally, for item-based approaches, we also used other similarity measures: the commonly used cosine similarity

⁴ <http://www-users.cs.umn.edu/~karypis/suggest/>

(denoted as Item-CosSim) and Pearson correlation (denoted as Item-CorSim). Results are shown in Table 1. For the precision, our user-item relevance model

	Top-1 Item	Top-10 Item	Top-20 Item	Top-40 Item
UIR-Item	0.62	0.52	0.44	0.35
Item-TFIDF	0.55	0.47	0.40	0.31
Item-CosSim	0.56	0.46	0.38	0.31
Item-CorSim	0.50	0.38	0.33	0.27
User-CosSim	0.55	0.42	0.34	0.27

(a) Precision

	Top-1 Item	Top-10 Item	Top-20 Item	Top-40 Item
UIR-Item	0.02	0.15	0.25	0.40
Item-TFIDF	0.02	0.15	0.26	0.41
Item-CosSim	0.02	0.13	0.22	0.35
Item-CorSim	0.01	0.11	0.19	0.31
User-CosSim	0.02	0.15	0.25	0.39

(b) Recall

Table 1. Comparison of Recommendation Performance.

with the item-based generation (UIR-Item) outperforms other log-based collaborative filtering approaches for all four different number of returned items. Overall, TF×IDF-like ranking ranks second. The obtained experimental results demonstrate that smoothing contributes to a better recommendation precision in the two ways also found by [21]. On the one hand, smoothing compensates for missing data in the user-item matrix, and on the other hand, it plays the role of inverse item frequency to emphasize the weight of the items with the best discriminative power. With respect to recall, all four algorithms perform almost identically. This is consistent to our first experiment that recommendation precision is sensitive to the smoothing parameters while the recommendation recall is not.

5 Conclusions

This paper identified a close relationship between log-based collaborative filtering and the methods developed for text information retrieval. We have built a user-item relevance model to re-formulate the collaborative filtering problem under the classic probability ranking principle. Using this probabilistic framework of user-item relevance models, we introduced a linear interpolation *smoothing* in collaborative filtering. We showed that smoothing is an important aspect to estimate models due to the data sparsity. Similar to the situation in text retrieval, the user-item relevance model provides a probabilistic justification of using TF×IDF-like item weighting in collaborative filtering.

Our further research aims to introduce relevance feedback into collaborative filtering. One of the powerful characteristics of linear interpolation smoothing is

that we can vary smoothing parameter: $\lambda \rightarrow \lambda(i_b)$ for the different items i_b in the user profile. It can then be treated as the importance of the query item. In the beginning, all the items in the user profile are treated equally. From relevance feedback, the importance value for different query items can be updated by using EM algorithm ([7]).

References

1. J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of UAI*, 1998.
2. J. Canny. Collaborative filtering with privacy via factor analysis. In *Proc. of SIGIR*, 1999.
3. M. Claypool, M. W. P. Le, and D. C. Brown. Implicit interest indicators. In *Proc. of UI'01*, pages 31–36, 2001.
4. M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.
5. S. Eyheramendy, D. Lewis, and D. Madigan. On the naive bayes model for text categorization. In *Proc. of Artificial Intelligence and Statistics*, 2003.
6. J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proc. of SIGIR*, 1999.
7. D. Hiemstra. Term-specific smoothing for the language modeling approach to information retrieval: the importance of a query term. In *Proc. of SIGIR*, pages 35–41, 2002.
8. T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proc. of IJCAI*, 1999.
9. D. Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proc. of SIGIR*, 1993.
10. G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proc. of the tenth international conference on Information and knowledge management*, 2001.
11. J. Lafferty and C. Zhai. Probabilistic relevance models based on document and query generation. *Language Modeling and Information Retrieval, Kluwer International Series on Information Retrieval*, V.13, 2003.
12. G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, Jan/Feb.:76–80, 2003.
13. B. Marlin. Collaborative filtering: a machine learning perspective. Master's thesis, Department of Computer Science, University of Toronto, 2004.
14. D. M. Pennock, E. Horvitz, S. Lawrence, and C. Giles. Collaborative filtering by personality diagnosis: a hybrid memory and model based approach. In *Proc. of UAI*, 2000.
15. J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proc. of SIGIR*, 1998.
16. G. Salton and M. J. McGill. *Introduction to modern information retrieval*. New York : McGraw-Hill, 1983.
17. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. of the WWW Conference*, 2001.
18. C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.
19. J. Wang, J. Pouwelse, R. Legendijk, and M. R. J. Reinders. Distributed collaborative filtering for peer-to-peer file sharing systems. In *Proc. of ACM SAC*, 2006.

20. G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proc. of SIGIR*, 2005.
21. C. Zhai and J. D. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR*, pages 334–342, 2001.

A User-Based Generation

By factorizing $P(\bullet|u_k, i_m)$ with $P(i_m|u_k, \bullet)P(\bullet|u_k)/P(i_m|u_k)$, the following log-odds ratio can be obtained from Eq. 3 :

$$\begin{aligned}
 RSV_{u_k}(i_m) &= \log \frac{P(r|i_m, u_k)}{P(\bar{r}|i_m, u_k)} \\
 &= \log \frac{P(i_m|u_k, r)}{P(i_m|u_k, \bar{r})} + \log \frac{P(u_k|r)P(r)}{P(u_k|\bar{r})P(\bar{r})} \\
 &\propto \log \frac{P(i_m|u_k, r)}{P(i_m|u_k, \bar{r})}
 \end{aligned} \tag{13}$$

When the non-relevance evidence is absent, and following the language model ([11]), we now assume equal priors for i_m in the non-relevant case. Then, the non-relevance term can be removed and the RSV becomes:

$$RSV_{u_k}(i_m) = \log P(i_m|u_k, r) \tag{14}$$

Instead of using the item list to represent the user, we use each user’s judgment as a feature to represent an item. For this, we introduce a list L_{i_m} for each item i_m , where $m = \{1, \dots, M\}$. This list enumerates the users who have expressed interest in the item i_m . $L_{i_m}(u_k) = 1$ (or $u_k \in L_{i_m}$) denotes that user u_k is in the list, while $L_{i_m}(u_k) = 0$ (or $u_k \notin L_{i_m}$) otherwise. The number of users in the list corresponds to $|L_{i_m}|$.

Replacing i_m with L_{i_m} , after we assume each user’s judgment to a particular item is independent, we have:

$$RSV_{u_k}(i_m) = \log P(i_m|u_k, r) = \sum_{\forall u_b: u_b \in L_{i_m}} \log P(u_b|u_k, r) \tag{15}$$

Similar to the item-based generation, when we use linear interpolation smoothing to estimate $P(u_b|u_k, r)$, we obtain the final ranking formula:

$$\begin{aligned}
 RSV_{u_k}(i_m) &= \sum_{\forall u_b: u_b \in L_{i_m}} \log P(u_b|u_k, r) \\
 &\propto \sum_{\forall u_b: u_b \in L_{i_m} \cap c(u_b, u_k) > 0} \log \left(1 + \frac{(1 - \lambda)P_{ml}(u_b|u_k, r)}{\lambda P(u_b|r)} \right) + |L_{i_m}| \log \lambda
 \end{aligned} \tag{16}$$

where $\lambda \in [0, 1]$ is the smoothing parameter.