Relevance Feedback Models for Recommendation

Masao Utiyama
National Institute of Information and Communications Technology
3-5 Hikari-dai, Soraku-gun, Kyoto 619-0289 Japan
mutiyama@nict.go.jp

Mikio Yamamoto
University of Tsukuba, 1-1-1 Tennodai, Tsukuba, 305-8573 Japan
myama@cs.tsukuba.ac.jp

Abstract

We extended language modeling approaches in information retrieval (IR) to combine collaborative filtering (CF) and content-based filtering (CBF). Our approach is based on the analogy between IR and CF, especially between CF and relevance feedback (RF). Both CF and RF exploit users' preference/relevance judgments to recommend items. We first introduce a multinomial model that combines CF and CBF in a language modeling framework. We then generalize the model to another multinomial model that approximates the Polya distribution. This generalized model outperforms the multinomial model by 3.4% for CBF and 17.4% for CF in recommending English Wikipedia articles. The performance of the generalized model for three different datasets was comparable to that of a state-of-the-art item-based CF method.

1 Introduction

Recommender systems (Resnick and Varian, 1997) help users select particular items (e.g, movies, books, music, and TV programs) that match their taste from a large number of choices by providing recommendations. The systems either recommend a set of *N* items that will be of interest to users (*top-N recommendation problem*) or predict the degree of users' preference for items (*prediction problem*).

For those systems to work, they first have to aggregate users' evaluations of items explicitly or implicitly. Users may explicitly evaluate certain movies as rating five stars to express their preference. These evaluations are used by the systems as *explicit ratings (votes)* of items or the systems infer the evaluations of items from the behavior of users and use these inferred evaluations as *implicit ratings*. For example, systems can infer that users may like certain items if the systems learn which books they buy, which articles they read, or which TV programs they watch.

Collaborative filtering (CF) (Resnick et al., 1994; Breese et al., 1998) and *content-based (or adaptive) filtering* (CBF) (Allan, 1996; Schapire et al., 1998) are two of the most popular types of algorithms used in recommender systems. A CF system makes recommendations to current (*active*) users by exploiting their ratings in the database. *User-based CF* (Resnick et al., 1994; Herlocker et al., 1999) and *item-based CF* (Sarwar et al., 2001; Karypis, 2001), among other CF algorithms, have been studied extensively. User-based CF first identifies a set of users (*neighbors*) that are similar to the active user in terms of their rating patterns in the database. It then uses the neighbors' rating patterns to produce recommendations for the active user. On the other hand, item-based CF calculates the similarity between items beforehand and then recommends items that are similar to those preferred by the active user. The performance of item-based CF has been shown to be comparable to or better than that of user-based CF (Sarwar et al., 2001; Karypis, 2001). In contrast to CF, CBF uses the contents (e.g., texts, genres, authors, images, and audio) of items to make recommendations for the active user. Because CF and CBF are complementary, much work has been done to combine them (Basu et al., 1998; Yu et al., 2003; Si and Jin, 2004; Basilico and Hofmann, 2004).

The approach we took in this study is designed to solve top-N recommendation problems with im-

Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006), pages 449–456,
Sydney, July 2006.

plicit ratings by using an item-based combination of CF and CBF. The methods described in this paper will be applied to recommending English Wikipedia[1] articles based on those articles edited by active users. (This is discussed in Section 3.) We use their editing histories and the contents of their articles to make top-N recommendations. We regard users' editing histories as implicit ratings. That is, if users have edited articles, we consider that they have positive attitudes toward the articles. Those implicit ratings are regarded as positive examples. We do not have negative examples for learning their negative attitudes toward articles. Consequently, handling our application with standard machine learning algorithms that require both positive and negative examples for classification (e.g., support vector machines) is awkward.

Our approach is based on the advancement in language modeling approaches to information retrieval (IR) (Croft and Lafferty, 2003) and extends these to incorporate CF. The motivation behind our approach is the analogy between CF and IR, especially between CF and relevance feedback (RF). Both CF and RF recommend items based on user preference/relevance judgments. Indeed, RF techniques have been applied to CBF, or adaptive filtering, successfully (Allan, 1996; Schapire et al., 1998). Thus, it is likely that RF can also be applied to CF.

To apply RF, we first extend the representation of items to combine CF and CBF under the models developed in Section 2. In Section 3, we report our experiments with the models. Future work and conclusion are in Sections 4 and 5.

2 Relevance feedback models

The analogy between IR and CF that will be exploited in this paper is as follows.[2] First, a document in IR corresponds to an item in CF. Both are represented as vectors. A document is represented as a vector of words (bag-of-words) and an item is represented as a vector of user ratings (bag-of-user ratings). In RF, a user specifies documents that are relevant to his information need. These documents are used by the system to retrieve new relevant documents. In CF, an active user (implicitly) specifies items that he likes. These items are used to search new items that will be preferred by the active user.

We use *relevance models* (Lavrenko and Croft, 2001; Lavrenko, 2004) as the basic framework of our relevance feedback models because (1) they perform relevance feedback well (Lavrenko, 2004) and (2) they can simultaneously handle different kinds of features (e.g., different language texts (Lavrenko et al., 2002), such as texts and images (Leon et al., 2003). These two points are essential in our application.

We first introduce a multinomial model following the work of Lavrenko (2004). This model is a novel one that extends relevance feedback approaches to incorporate CF. It is like a combination of relevance feedback (Lavrenko, 2004) and cross-language information retrieval (Lavrenko et al., 2002). We then generalize that model to an approximated Polya distribution model that is better suited to CF and CBF. This generalized model is the main technical contribution of this work.

2.1 Preparation

Lavrenko (2004) adopts the method of kernels to estimate probabilities: Let $\mathbf{d}$ be an item in the database or training data, the probability of item $\mathbf{x}$ is estimated as $p(\mathbf{x}) = \frac{1}{M}\sum_{\mathbf{d}} p(\mathbf{x}|\theta_{\mathbf{d}})$, where M is the number of items in the training data, $\theta_{\mathbf{d}}$ is the parameter vector estimated from $\mathbf{d}$, and $p(\mathbf{x}|\theta_{\mathbf{d}})$ is the conditional probability of $\mathbf{x}$ given $\theta_{\mathbf{d}}$.[3] This means that once we have defined a probability distribution $p(\mathbf{x}|\theta)$ and the method of estimating $\theta_{\mathbf{d}}$ from $\mathbf{d}$, then we can assign probability $p(\mathbf{x})$ to $\mathbf{x}$ and apply language modeling approaches to CF and CBF.

To begin with, we define the representation of item $\mathbf{x}$ as the concatenation of two vectors $\{\mathbf{w_x}, \mathbf{u_x}\}$, where $\mathbf{w_x} = w_{\mathbf{x}1}w_{\mathbf{x}2}\ldots$ is the sequence of words (contents) contained in $\mathbf{x}$ and $\mathbf{u_x} = u_{\mathbf{x}1}u_{\mathbf{x}2}\ldots$ is the sequence of users who have rated $\mathbf{x}$ implicitly. We use V_w and V_u to denote the set of words and users in the database. The parameter vector θ is also the concatenation of two vectors $\{\omega, \mu\}$, where ω and μ are the parameter vectors for V_w and V_u, respectively. The probability of $\mathbf{x}$ given θ is defined as $p(\mathbf{x}|\theta) = p_\omega(\mathbf{w_x}|\omega)p_\mu(\mathbf{u_x}|\mu)$.

[1] http://en.wikipedia.org/wiki/Main_Page

[2] The analogy between IR and CF has been recognized. For example, Breese et al. (1998) used the vector space model to measure the similarity between users in a user-based CF framework. Wang et al. (2005) used a language modeling approach different from ours. These works, however, treated only CF. In contrast with these, our model extends language modeling approaches to incorporate both CF and CBF.

[3] Item $\mathbf{d}$ in summation $\sum_{\mathbf{d}}$ and word w in $\sum_w$ and $\prod_w$ go over every distinct item d and word w in the training data, unless otherwise stated.

2.2 Multinomial model

Our first model regards that both p_ω and p_μ follow multinomial distributions. In this case, $\omega(w)$ and $\mu(u)$ are the probabilities of word w and user u. Then, $p_\omega(\mathbf{w_x}|\omega)$ is defined as

$$p_\omega(\mathbf{w_x}|\omega) = \prod_{i=1}^{|\mathbf{w_x}|} \omega(w_{\mathbf{x}i}) = \prod_{w \in V_w} \omega(w)^{n(w,\mathbf{w_x})} \quad (1)$$

where $n(w, \mathbf{w_x})$ is the number of occurrences of w in $\mathbf{w_x}$. In this model, we use a linear interpolation method to estimate probability $\omega_\mathbf{d}(w)$.

$$\omega_\mathbf{d}(w) = \lambda_\omega P_l(w|\mathbf{w_d}) + (1-\lambda_\omega) P_g(w) \quad (2)$$

where $P_l(w|\mathbf{w_d}) = \frac{n(w,\mathbf{w_d})}{\sum_{w'} n(w',\mathbf{w_d})}$, $P_g(w) = \frac{\sum_\mathbf{d} n(w,\mathbf{w_d})}{\sum_\mathbf{d}\sum_{w'} n(w',\mathbf{w_d})}$ and λ_ω $(0 \leq \lambda_\omega \leq 1)$ is a smoothing parameter. The estimation of user probabilities goes similarly: Let $n(u, \mathbf{u_x})$ be the number of times user u implicitly rated item $\mathbf{x}$, we define or estimate p_μ, λ_μ and $\mu_\mathbf{d}$ in the same way. In summary, we have defined a probability distribution $p(\mathbf{x}|\theta)$ and the method of estimating $\theta_\mathbf{d} = \{\omega_\mathbf{d}, \mu_\mathbf{d}\}$ from $\mathbf{d}$.

To recommend top-N items, we have to rank items in the database in response to the implicit ratings of active users. We call those implicit ratings *query* $\mathbf{q}$. It is a set of items and is represented as $\mathbf{q} = \{\mathbf{q}_1 \ldots \mathbf{q}_k\}$, where $\mathbf{q}_i$ is an item implicitly rated by an active user and k is the size of $\mathbf{q}$. We next estimate $\theta_\mathbf{q} = \{\omega_\mathbf{q}, \mu_\mathbf{q}\}$. Then, we compare $\theta_\mathbf{q}$ and $\theta_\mathbf{d}$ to rank items by using Kullback-Leibler divergence $D(\theta_\mathbf{q}||\theta_\mathbf{d})$ (Lafferty and Zhai, 2001; Lavrenko, 2004).

$\omega_\mathbf{q}(w)$ can be approximated as

$$\omega_\mathbf{q}(w) = \frac{1}{k}\sum_{i=1}^{k} \omega_{\mathbf{q}_i}(w) \quad (3)$$

where $\omega_{\mathbf{q}_i}(w)$ is obtained by Eq. 2 (Lavrenko, 2004). However, we found in preliminary experiments that smoothing query probabilities hurt performance in our application. Thus, we use

$$\omega_{\mathbf{q}_i}(w) = P_l(w|\mathbf{w}_{\mathbf{q}_i}) = \frac{n(w,\mathbf{w}_{\mathbf{q}_i})}{\sum_{w'} n(w',\mathbf{w}_{\mathbf{q}_i})} \quad (4)$$

instead of Eq. 2 when $\mathbf{q}_i$ is in a query.

Because KL-divergence is a distance measure, we use a score function derived from $-D(\theta_\mathbf{q}||\theta_\mathbf{d})$ to rank items. We use $S_\mathbf{q}(\mathbf{d})$ to denote the score of $\mathbf{d}$ given $\mathbf{q}$. $S_\mathbf{q}(\mathbf{d})$ is derived as follows. (We ignore terms that are irrelevant to ranking items.)

$$\begin{aligned} -D(\theta_\mathbf{q}||\theta_\mathbf{d}) &= -D(\omega_\mathbf{q}||\omega_\mathbf{d}) - D(\mu_\mathbf{q}||\mu_\mathbf{d}) \\ -D(\omega_\mathbf{q}||\omega_\mathbf{d}) &\overset{rank}{=} \frac{1}{k}\sum_{i=1}^{k} S(\omega_{\mathbf{q}_i}||\omega_\mathbf{d}) \end{aligned} \quad (5)$$

where

$$S(\omega_{\mathbf{q}_i}||\omega_\mathbf{d}) = \sum_w P_l(w|\mathbf{w}_{\mathbf{q}_i}) \times \log\left(\frac{\lambda_\omega P_l(w|\mathbf{w_d})}{(1-\lambda_\omega)P_g(w)} + 1\right). \quad (6)$$

The summation goes over every word w that is shared by both $\mathbf{w}_{\mathbf{q}_i}$ and $\mathbf{w_d}$. We define $S(\mu_{\mathbf{q}_i}||\mu_\mathbf{d})$ similarly.[4] Then, the score of $\mathbf{d}$ given $\mathbf{q}_i$, $S_{\mathbf{q}_i}(\mathbf{d})$ is defined as

$$S_{\mathbf{q}_i}(\mathbf{d}) = \lambda_s S(\mu_{\mathbf{q}_i}||\mu_\mathbf{d}) + (1-\lambda_s) S(\omega_{\mathbf{q}_i}||\omega_\mathbf{d}) \quad (7)$$

where λ_s $(0 \leq \lambda_s \leq 1)$ is a free parameter. Finally, the score of $\mathbf{d}$ given $\mathbf{q}$ is

$$S_\mathbf{q}(\mathbf{d}) = \frac{1}{k}\sum_{i=1}^{k} S_{\mathbf{q}_i}(\mathbf{d}). \quad (8)$$

The calculation of $S_\mathbf{q}(\mathbf{d})$ can be very efficient because once we cache $S_{\mathbf{q}_i}(\mathbf{d})$ for each item pair of $\mathbf{q}_i$ and $\mathbf{d}$ in the database, we can reuse it to calculate $S_\mathbf{q}(\mathbf{d})$ for any query $\mathbf{q}$. We further optimize the calculation of top-N recommendations by storing only the top 100 items (*neighbors*) in decreasing order of $S_{\mathbf{q}_i}(\cdot)$ for each item $\mathbf{q}_i$ and setting the scores of lower ranked items as 0. (Note that $S_{\mathbf{q}_i}(\mathbf{d}) >= 0$ holds.) Consequently, we only have to search small part of the search space without affecting the performance very much. These two types of optimization are common in item-based CF (Sarwar et al., 2001; Karypis, 2001).

2.3 Polya model

Our second model is based on the Polya distribution. We first introduce (hyper) parameter $\Theta = \{\alpha^\omega, \alpha^\mu\}$ and denote the probability of $\mathbf{x}$ given Θ as $p(\mathbf{x}|\Theta) = p_\omega(\mathbf{w_x}|\alpha^\omega)p_\mu(\mathbf{u_x}|\alpha^\mu)$. α^ω and α^μ are the parameter vectors for words and users. $p_\omega(\mathbf{w_x}|\alpha^\omega)$ is defined as follows.

$$p_\omega(\mathbf{w_x}|\alpha^\omega) = \frac{\Gamma(\sum_w \alpha_w^\omega)}{\Gamma(\sum_w n_w^\mathbf{x} + \alpha_w^\omega)} \prod_w \frac{\Gamma(n_w^\mathbf{x} + \alpha_w^\omega)}{\Gamma(\alpha_w^\omega)} \quad (9)$$

[4] $S(\mu_{\mathbf{q}_i}||\mu_\mathbf{d}) = \sum_u P_l(u|\mathbf{u}_{\mathbf{q}_i}) \times \log\left(\frac{\lambda_\mu P_l(u|\mathbf{u_d})}{(1-\lambda_\mu)P_g(u)} + 1\right)$, where $P_l(u|\mathbf{u}_{\mathbf{q}_i}) = \frac{n(u,\mu_{\mathbf{q}_i})}{\sum_{u'} n(u',\mu_{\mathbf{q}_i})}$, $P_l(u|\mathbf{u_d}) = \frac{n(u,\mathbf{u_d})}{\sum_{u'} n(u',\mathbf{u_d})}$, and $P_g(u) = \frac{\sum_\mathbf{d} n(u,\mathbf{u_d})}{\sum_\mathbf{d}\sum_{u'} n(u',\mathbf{u_d})}$.

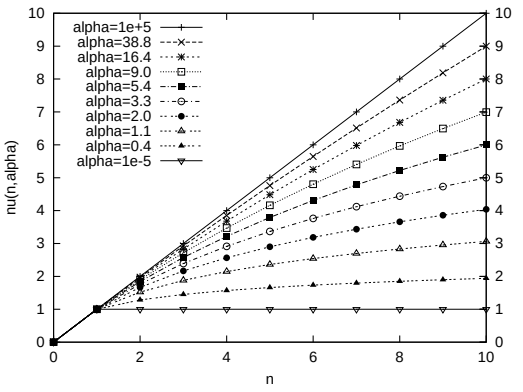

Figure 1: Relationship between original count n and dumped count $\nu(n, \alpha)$

where Γ is known as the gamma function, α_w^ω is a parameter for word w and $n_w^{\mathbf{x}} = n(w, \mathbf{w_x})$. This can be approximated as follows (Minka, 2003).

$$p_\omega(\mathbf{w_x}|\alpha^\omega) \sim \prod_w \omega(w)^{\tilde{n}(w,\mathbf{w_x})} \tag{10}$$

where

$$\begin{aligned} \tilde{n}(w, \mathbf{w_x}) &= \alpha_w^\omega(\Psi(n_w^{\mathbf{x}} + \alpha_w^\omega) - \Psi(\alpha_w^\omega)) \\ &\equiv \nu(n_w^{\mathbf{x}}, \alpha_w^\omega) \end{aligned} \tag{11}$$

Ψ is known as the digamma function and is similar to the natural logarithm. We call Eq. 10 the approximated Polya model or simply the Polya model in this paper.

Eq. 10 indicates that the Polya distribution can be interpreted as a multinomial distribution over a modified set of counts $\tilde{n}(\cdot)$ (Minka, 2003). These modified counts are *dumped* as shown in Fig. 1. When $\alpha_w^\omega \to \infty$, $\nu(n_w^{\mathbf{x}}, \alpha_w^\omega)$ approaches $n_w^{\mathbf{x}}$. When $\alpha_w^\omega \to 0$, $\nu(n_w^{\mathbf{x}}, \alpha_w^\omega) = 0$ if $n_w^{\mathbf{x}} = 0$ otherwise it is 1. For intermediate values of α_w^ω, the mapping ν dumps the original counts.

Under the approximation of Eq. 10, the estimation of parameters can be understood as the maximum-likelihood estimate of a multinomial distribution from dumped counts $\tilde{n}(\cdot)$ (Minka, 2003). Indeed, all we have to do to estimate the parameters for ranking items is replace P_l and P_g from Section 2.2 with $P_l(w|\mathbf{w_d}) = \frac{\tilde{n}(w,\mathbf{w_d})}{\sum_{w'} \tilde{n}(w',\mathbf{w_d})}$, $P_g(w) = \frac{\sum_{\mathbf{d}} \tilde{n}(w,\mathbf{w_d})}{\sum_{\mathbf{d}} \sum_{w'} \tilde{n}(w',\mathbf{w_d})}$, and $P_l(w|\mathbf{w_{q_i}}) = \frac{\tilde{n}(w,\mathbf{w_{q_i}})}{\sum_{w'} \tilde{n}(w',\mathbf{w_{q_i}})}$. Then, as in the multinomial model, we can define $S(\omega_{\mathbf{q_i}}||\omega_{\mathbf{d}})$ with these probabilities. This argument also applies to $S(\mu_{\mathbf{q_i}}||\mu_{\mathbf{d}})$.

The approximated Polya model is a generalization of the multinomial model described in Section 2.2. If we set α_w^ω and α_u^μ very large then the Polya model is identical to the multinomial model. By comparing Eqs. 1 and 10, we can see why the Polya model is superior to the multinomial model for modeling the occurrences of words (and users). In the multinomial model, if a word with probability p occurs twice, its probability becomes p^2. In the Polya model, the word's probability becomes $p^{1.5}$, for example, if we set $\alpha_w^\omega = 1$. Clearly, $p^2 < p^{1.5}$; therefore, the Polya model assigns higher probability. In this example, the Polya model assigns probability p to the first occurrence and $p^{0.5}(> p)$ to the second. Since words that occur once are likely to occur again (Church, 2000), the Polya model is better suited to model the occurrences of words and users. See Yamamoto and Sadamitsu (2005) for further discussion on applying the Polya distribution to text modeling.

Zaragoza et al.(2003) applied the Polya distribution to ad hoc IR. They introduced the exact Polya distribution (see Eq. 9) as an extension to the Dirichlet prior method (Zhai and Lafferty, 2001). However, we have introduced a multinomial approximation of the Polya distribution. This approximation allows us to use the linear interpolation method to mix the approximated Polya distributions. Thus, our model is similar to two-stage language models (Zhai and Lafferty, 2002) that combine the Dirichlet prior method and the linear interpolation method. In contrast to our model, Zaragoza et al.(2003) had difficulty in mixing the Polya distributions and did not treat that in their paper.

3 Experiments

We first examined the behavior of the Polya model by varying the parameters. We tied α_w^ω for every w and α_u^μ for every u; for any w and u, $\alpha_w^\omega = \alpha_\omega$ and $\alpha_u^\mu = \alpha_\mu$. We then compared the Polya model to an item-based CF method.

3.1 Behavior of Polya model

3.1.1 Dataset

We made a dataset of articles from English Wikipedia[5] to evaluate the Polya model. English Wikipedia is an online encyclopedia that anyone

[5]We downloaded `20050713_pages_full.xml.gz` and `20050713_pages_current.xml.gz` from `http://download.wikimedia.org/wikipedia/en/`.

can edit, and it has many registered users. Our aim is to recommend a set of articles to each user that is likely to be of interest to that user. If we can successfully recommend interesting articles, this could be very useful to a wide audience because Wikipedia is very popular. In addition, because wikis are popular media for sharing knowledge, developing effective recommender systems for wikis is important.

In our Wikipedia dataset, each item (article) $\mathbf{x}$ consisted of $\mathbf{w_x}$ and $\mathbf{u_x}$. $\mathbf{u_x}$ was the sequence of users who had edited $\mathbf{x}$. If users had edited $\mathbf{x}$ multiple times, then those users occurred in $\mathbf{u_x}$ multiple times. $\mathbf{w_x}$ was the sequence of words that were typical in $\mathbf{x}$. To make $\mathbf{w_x}$, we removed stop words and stemmed the remaining words with a Porter stemmer. Next, we identified 100 typical words in each article and extracted only those words ($|\mathbf{w_x}| \geq 100$ because some of them occurred multiple times). Typicality was measured using the log-likelihood ratio test (Dunning, 1993). We needed to reduce the number of words to speed up our recommender system.

To make our dataset, we first extracted 302,606 articles, which had more than 100 tokens after the stop words were removed. We then selected typical words in each article. The implicit rating data were obtained from the histories of users editing these articles. Each rating consisted of {*user*, *article*, *number of edits*}. The size of this original rating data was 3,325,746. From this data, we extracted a dense subset that consisted of users and articles included in at least 25 units of the original data. We discarded the users who had edited more than 999 articles because they were often software robots or system operators, not casual users. The resulting 430,096 ratings consisted of 4,193 users and 9,726 articles. Each user rated (edited) 103 articles on average (the median was 57). The average number of ratings per item was 44 and the median was 36.

3.1.2 Evaluation of Polya model

We conducted a four-fold cross validation of this rating dataset to evaluate the Polya model. We used three-fourth of the dataset to train the model and one-fourth to test it.[6] All users who existed in both training and test data were used for evaluation. For each user, we regarded the articles in the training data that had been edited by the user as a query and ranked articles in response to it. These ranked top-N articles were then compared to the articles in the test data that were edited by the same user to measure the precisions for the user. We used P@N (precision at rank N = the ratio of the articles edited by the user in the top-N articles), S@N (success at rank $N = 1$ if some top-N articles were edited by the user, else 0), and R-precision (= P@N, where N is the number of articles edited by the user in the test data). These measures for each user were averaged over all users to get the mean precision of each measure. Then, these mean precisions were averaged over the cross validation repeats.

Here, we report the averaged mean precisions with standard deviations. We first report how R-precision varied depending on α (α_ω or α_μ). α was varied over $10^{-5}, 0.4, 1.1, 2, 3.3, 5.4, 9, 16.4, 38.8,$ and 10^5. The values of $\nu(10, \alpha)$ were approximately 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10, respectively, as shown in Fig. 1. When $\alpha = 10^5$, the Polya model represents the multinomial model as discussed in Section 2.3. For each value of α, we varied λ (λ_ω or λ_μ) over 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, and 0.99 to obtain the optimum R-precision. These optimum R-precisions are shown in Fig. 2. In this figure, CBF and CF represent the R-precisions for the content-based and collaborative filtering part of the Polya model. The values of CBF and CF were obtained by setting $\lambda_s = 0$ and $\lambda_s = 1$ in Eq. 7 (which is applied to the Polya model instead of the multinomial model), respectively. The error bars represent standard deviations.

At once, we noticed that CBF outperformed CF. This is reasonable because the contents of Wikipedia articles should strongly reflect the users (authors) interest. In addition, each article had about 100 typical words, and this was richer than the average number of users per article (44). This observation contrasts with other work where CBF performed poorly compared with CF, e.g., (Ali and van Stam, 2004).

Another important observation is that both curves in Fig. 2 are concave. The best R-precisions were obtained at intermediate values of α for both CF and CBF as shown in Table 1.

[6] We needed to estimate probabilities of users and words. We used only training data to estimate the probabilities of users. However, we used all 9,726 articles to estimate the probabilities of words because the articles are usually available even when editing histories of users are not.

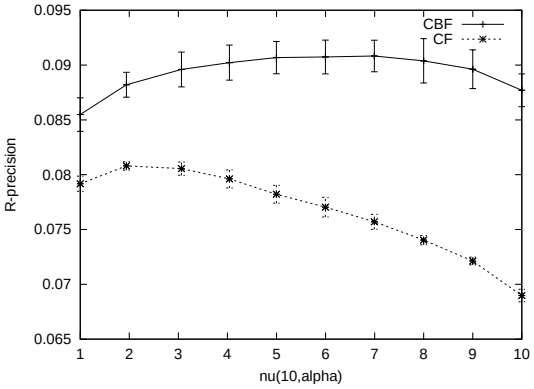

Figure 2: R-precision for Polya model

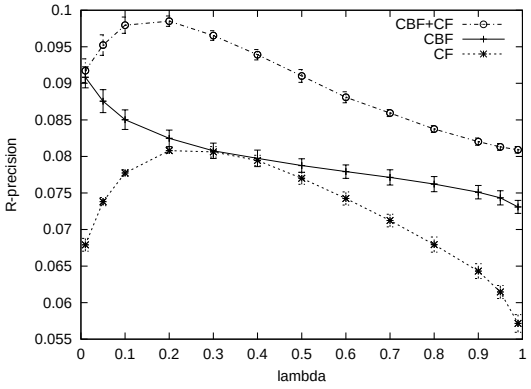

Figure 3: Combination of CBF and CF.

Table 1: Improvement in R-precision (RP)

	best RP ($\nu(\cdot)/\alpha$)	RP ($\nu(\cdot)/\alpha$)	%change
CBF	0.091 (7/9.0)	0.088 ($10/10^5$)	$+3.4\%$
CF	0.081 (2/0.4)	0.069 ($10/10^5$)	$+17.4\%$

When $\alpha = 10^5$ or $\nu(10, \alpha) \sim 10$, the Polya model represents the multinomial model as discussed in Section 2.3. Thus, Fig. 2 and Table 1 show that the best R-precisions achieved by the Polya model were better than those obtained by the multinomial model. The improvement was 3.4% for CBF and 17.4% for CF as shown in Table 1. The improvement of CF was larger than that of CBF. This implies that the occurrences of users are more clustered than those of words. In other words, the degree of repetition in the editing histories of users is greater than that in word sequences. A user who edits an article are likely to edit the article again.

From Fig. 2 and Table 1, we concluded that the generalization of a multinomial model achieved by the Polya model is effective in improving recommendation performance.

3.1.3 Combination of CBF and CF

Next, we show how the combination of CBF and CF improves recommendation performance. We set α (α_ω and α_μ) to the optimum values in Table 1 and varied λ (λ_s, λ_ω and λ_μ) to obtain the R-precisions for CBF+CF, CBF and CF in Fig. 3. The values of CBF were obtained as follows. We first set $\lambda_s = 0$ in Eq. 7 to use only CBF scores and then varied λ_ω, which is the smoothing parameter for word probabilities, in Eq. 2. To get the values of CF, we set $\lambda_s = 1$ in Eq. 7 and then varied λ_μ, which is the smoothing parameter for user probabilities. The values of CBF+CF were obtained by varying λ_s in Eq. 7 while setting λ_ω and λ_μ to the optimum values obtained from CBF and CF (see Table 2). These parameters (λ_s, λ_ω and λ_μ) were defined in the context of the multinomial model in Section 2.2 and used similarly in the Polya model in this experiment.

Table 2: Precision and Success at top-N

	CBF+CF		CBF		CF	
N	P@N	S@N	P@N	S@N	P@N	S@N
5	0.166	0.470	0.149	0.444	0.137	0.408
10	0.135	0.585	0.123	0.562	0.112	0.516
15	0.117	0.650	0.107	0.628	0.098	0.582
20	0.105	0.694	0.096	0.671	0.089	0.627
R-precision	0.099		0.091		0.081	
optimum λ	$\lambda_s = 0.2$		$\lambda_\omega = 0.01$		$\lambda_\mu = 0.2$	

We can see that the combination was quite effective as CBF+CF outperformed both CBF and CF. Table 2 shows R-precision, P@N and S@N for $N = 5, 10, 15, 20$. These values were obtained by using the optimum values of λ in Fig. 3.

Table 2 shows the same tendency as Fig. 3. For all values of N, CBF+CF outperformed both CBF and CF. We attribute this effectiveness of the combination to the feature independence of CBF and CF. CBF used words as features and CF used user ratings as features. They are very different kinds of features and thus can provide complementary information. Consequently, CBF+CF can exploit the benefits of both methods. We need to do further work to confirm this conjecture.

3.2 Comparison with a baseline method

We compared the Polya model to an implementation of a state-of-the-art item-based CF method, *CProb* (Karypis, 2001). CProb has been tested with various datasets and found to be effective in top-N recommendation problems. CProb has also been used in recent work as a baseline method (Ziegler et al., 2005; Wang et al., 2005).

In addition to the Wikipedia dataset, we used two other datasets for comparison. The first was

	R-precision			P@10		
	WP	ML	BX	WP	ML	BX
Polya-CF	0.081	0.272	0.066	0.112	0.384	0.054
CProb	0.082	0.258	0.071	0.113	0.373	0.057
%change	-1.2%	+5.4%	-7.0%	-0.9%	+2.9%	-5.3%

Table 3: Comparison of Polya-CF and CProb

the 1 million MovieLens dataset.[7] This data consists of 1,000,209 ratings of 3,706 movies by 6,040 users. Each user rated an average of 166 movies (the median was 96). The average number of ratings per movie was 270 and the median was 124. The second was the BookCrossing dataset (Ziegler et al., 2005). This data consists of 1,149,780 ratings of 340,532 books by 105,283 users. From this data, we removed books rated by less than 20 users. We also removed users who rated less than 5 books. The resulting 296,471 ratings consisted of 10,345 users and 5,943 books. Each user rated 29 books on average (the median was 10). The average number of ratings per book was 50 and the median was 33. Note that in our experiments, we regarded the ratings of these two datasets as implicit ratings. We regarded the number of occurrence of each rating as one.

We conducted a four-fold cross validation for each dataset to compare CProb and Polya-CF, which is the collaborative filtering part of the Polya model as described in the previous section. For each cross validation repeat, we tuned the parameters of CProb and Polya-CF on the test data to get the optimum R-precisions, in order to compare best results for these models.[8] P@N and S@N were calculated with the same parameters. These measures were averaged as described above. R-precision and P@10 are in Table 3. The maximum standard deviation of these measures was 0.001. We omitted reporting other measures because they had similar tendencies. In Table 3, WP, ML and BX represent the Wikipedia, MovieLens, and BookCrossing datasets.

In Table 3, we can see that the variation of performance among datasets was greater than that between Polya-CF and CProb. Both methods performed best against ML. We think that this is because ML had the densest ratings. The average number of ratings per item was 270 for ML while that for WP was 44 and that for BX was 50.

Table 3 also shows that Polya-CF outperformed CProb when the dataset was ML and CProb was better than Polya-CF in the other cases. However, the differences in precision were small. Overall, we can say that the performance of Polya-CF is comparable to that of CProb.

An important advantage of the Polya model over CProb is that the Polya model can unify CBF and CF in a single language modeling framework while CProb handles only CF. Another advantage of the Polya model is that we can expect to improve its performance by incorporating techniques developed in IR because the Polya model is based on language modeling approaches in IR.

4 Future work

We want to investigate two areas in our future work. One is the parameter estimation and the other is the refinement of the query model.

We tuned the parameters of the Polya model by exhaustively searching the parameter space guided by R-precision. We actually tried to learn α_ω and α_μ from the training data by using an EM method (Minka, 2003; Yamamoto and Sadamitsu, 2005). However, the estimated parameters were about 0.05, too small for better recommendations. We need further study to understand the relation between the probabilistic quality (*perplexity*) of the Polya model and its recommendation quality.

We approximate the query model as Eq. 3. This allows us to optimize score calculation considerably. However, this does not consider the interaction among items, which may deteriorate the quality of probability estimation. We want to investigate more efficient query models in our future work.

5 Conclusion

Recommender systems help users select particular items from a large number of choices by providing recommendations. Much work has been done to combine content-based filtering (CBF) and collaborative filtering (CF) to provide better recommendations. The contributions reported in this paper are twofold: (1) we extended relevance feedback approaches to incorporate CF and (2) we introduced the approximated Polya model as a gen-

[7] http://www.grouplens.org/

[8] CProb has two free parameters. Polya-CF also has two free parameters (α_μ and λ_μ). However, for MovieLens and BookCrossing datasets, Polya-CF has only one free parameter λ_μ, because we regarded the number of occurrence of each rating as one, which means $\nu(1, \alpha_\mu) = 1$ for all $\alpha_\mu > 0$ (See Fig. 1). Consequently, we don't have to tune α_μ. Since the number of free parameters is small, the comparison of performance shown in Table 3 is likely to be reproduced when we tune the parameters on separate development data instead of test data.

eralization of the multinomial model and showed that it is better suited to CF and CBF. The performance of the Polya model is comparable to that of a state-of-the-art item-based CF method.

Our work shows that language modeling approaches in information retrieval can be extended to CF. This implies that a large amount of work in the field of IR could be imported into CF. This would be interesting to investigate in future work.

References

Kamal Ali and Wijnand van Stam. 2004. TiVo: Making show recommendations using a distributed collaborative filtering architecture. In *KDD'04*.

James Allan. 1996. Incremental relevance feedback for information filtering. In *SIGIR'96*.

Justin Basilico and Thomas Hofmann. 2004. Unifying collaborative and content-based filtering. In *ICML'04*.

Chumki Basu, Haym Hirsh, and William Cohen. 1998. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI-98*.

John S. Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. Technical report, MSR-TR-98-12.

Kenneth W. Church. 2000. Empirical estimates of adaptation: The chance of two Noriegas is closer to $p/2$ than p^2. In *COLING-2000*, pages 180–186.

W. Bruce Croft and John Lafferty, editors. 2003. *Language Modeling for Information Retrieval*. Kluwer Academic Publishers.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. 1999. An algorithmic framework for performing collaborative filtering. In *SIGIR'99*, pages 230–237.

George Karypis. 2001. Evaluation of item-based top-N recommendation algorithms. In *CIKM'01*.

John Lafferty and ChengXiang Zhai. 2001. Document language models, query models and risk minimization for information retrieval. In *SIGIR'01*.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance-based language models. In *SIGIR'01*.

Victor Lavrenko, Martin Choquette, and W. Bruce Croft. 2002. Cross-lingual relevance models. In *SIGIR'02*, pages 175–182.

Victor Lavrenko. 2004. *A Generative Theory of Relevance*. Ph.D. thesis, University of Massachusetts.

J. Leon, V. Lavrenko, and R. Manmatha. 2003. Automatic image annotation and retrieval using cross-media relevance models. In *SIGIR'03*.

Thomas P. Minka. 2003. Estimating a Dirichlet distribution. http://research.microsoft.com/~minka/papers/dirichlet/.

Paul Resnick and Hal R. Varian. 1997. Recommender systems. *Communications of the ACM*, 40(3):56–58.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An open architecture for collaborative filtering of netnews. In *CSCW'94*, pages 175–186.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW10*.

Robert E. Schapire, Yoram Singer, and Amit Singhal. 1998. Boosting and Rocchio applied to text filtering. In *SIGIR'98*, pages 215–223.

Luo Si and Rong Jin. 2004. Unified filtering by combining collaborative filtering and content-based filtering via mixture model and exponential model. In *CIKM-04*, pages 156–157.

Jun Wang, Marcel J.T. Reinders, Reginald L. Lagendijk, and Johan Pouwelse. 2005. Self-organizing distributed collaborative filtering. In *SIGIR'05*, pages 659–660.

Mikio Yamamoto and Kugatsu Sadamitsu. 2005. Dirichlet mixtures in text modeling. Technical report, University of Tsukuba, CS-TR-05-1.

Kai Yu, Anton Schwaighofer, Volker Tresp, Wei-Ying Ma, and HongJiang Zhang. 2003. Collaborative ensemble learning: Combining collaborative and content-based information filtering via hierarchical Bayes. In *UAI-2003*.

Hugo Zaragoza, Djoerd Hiemstra, and Michael Tipping. 2003. Bayesian extension to the language model for ad hoc information retrieval. In *SIGIR'03*.

ChengXiang Zhai and John Lafferty. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR'01*.

ChengXiang Zhai and John Lafferty. 2002. Two-stage language models for information retrieval. In *SIGIR'02*, pages 49–56.

Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *WWW'05*, pages 22–32.