Pseudo-Relevance Feedback Based on Matrix Factorization

Hamed Zamani[†*] Javid Dadashkarimi[‡] Azadeh Shakery[‡] W. Bruce Croft[†]
[†]Center for Intelligent Information Retrieval, College of Information and Computer Sciences, University of Massachusetts Amherst, MA 01003
[‡]School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Iran
{zamani,croft}@cs.umass.edu {dadashkarimi,shakery}@ut.ac.ir

ABSTRACT

In information retrieval, pseudo-relevance feedback (PRF) refers to a strategy for updating the query model using the top retrieved documents. PRF has been proven to be highly effective in improving the retrieval performance. In this paper, we look at the PRF task as a recommendation problem: the goal is to recommend a number of terms for a given query along with weights, such that the final weights of terms in the updated query model better reflect the terms' contributions in the query. To do so, we propose *RFMF*, a PRF framework based on matrix factorization which is a state-of-the-art technique in collaborative recommender systems. Our purpose is to predict the weight of terms that have not appeared in the query and matrix factorization techniques are used to predict these weights. In RFMF, we first create a matrix whose elements are computed using a *weight* function that shows how much a term discriminates the query or the top retrieved documents from the collection. Then, we re-estimate the created matrix using a matrix factorization technique. Finally, the query model is updated using the re-estimated matrix. RFMF is a general framework that can be employed with any retrieval model. In this paper, we implement this framework for two widely used document retrieval frameworks: language modeling and the vector space model. Extensive experiments over several TREC collections demonstrate that the RFMF framework significantly outperforms competitive baselines. These results indicate the potential of using other recommendation techniques in this task.

Keywords

Query expansion; pseudo-relevance feedback; matrix factorization; term recommendation; language model

1. INTRODUCTION

Users often issue very short queries when using search engines. Therefore, queries usually miss several important terms, which leads to poor retrieval performance. One of the main techniques proposed for this problem is pseudo-relevance feedback (PRF), also known as blind feedback. PRF is described as a strategy to reformulate the query model based on pseudo-relevant (top retrieved) documents. The main goal of PRF is to improve the retrieval performance, and PRF methods have been shown to be highly effective in many retrieval models [14, 20, 22, 31, 43].

The main idea behind this paper is to recast the PRF task as a collaborative recommender system problem: the query and its pseudo-relevant documents play the role of users and terms play the role of items. Each term has a *weight* in each query/document and this *weight* can be considered as the rating that the query/document (user) gives to that term (item). Hence, the query model can be updated by recommending a number of terms to the query in a weighted manner. The intuition of the proposed framework is that the query and the pseudo-relevant documents have several commonalities, which are similar to a group of users that have similar preferences, and thus collaborative recommendation ideas are strongly related to what we need in the pseudo-relevance feedback problem.

Matrix factorization techniques have been found to be highly effective in collaborative recommender systems, especially in those suffering from data sparsity [29]. As is widely known, most information retrieval (IR) tasks also suffer from the data sparsity problem, since compared to the vocabulary size, only a few words appear in the queries and documents. As a result, we consider matrix factorization techniques for solving our collaborative recommendation problem.

Based on the aforementioned idea, in this paper we propose *RFMF*[1], a PRF framework based on matrix factorization techniques. RFMF tries to estimate the weights of all terms for the query. Indeed, RFMF first creates a matrix in which each row corresponds to the query or one of the pseudo-relevant documents. In this matrix, each column is associated with each unique term in the feedback document set. Each element of this matrix represents the weight of a given term in a given query/document. RFMF employs matrix factorization to predict the weights of unseen words for the query. We select non-negative matrix factorization for implementing our framework, which is a suitable matrix decomposition method for our problem. To implement the RFMF framework, we also need to define a proper *weight* function, i.e., the weight of each term given by each query/document. The *weight* function must show how much a term discriminates a given query/document from

*A part of this work was done while the author was with the University of Tehran.

CIKM'16 , October 24-28, 2016, Indianapolis, IN, USA

DOI: http://dx.doi.org/10.1145/2983323.2983844

[1]Relevance Feedback based on Matrix Factorization

the collection because our purpose is to update the query model in such a way that the final term weights better reflect the terms' contributions in the query. From the recommender systems perspective, the weight of each term in each query/document should be similar to the rating that each user gives to a specific item. To implement the *weight* function, in this paper, we only consider the query and the top retrieved documents without any external information, e.g., term dependency, linguistic resources, etc. The *weight* function can potentially be enhanced using these kinds of information that have been shown to be useful in the PRF task [3, 21, 23, 40].

RFMF is a general framework that is independent of the employed retrieval model. In this paper, we first implement the *weight* function for the language modeling framework [28], a well-structured and state-of-the-art retrieval framework. To demonstrate the generality of RFMF, we also study a number of heuristic *weight* functions for the well-known vector space model.

We evaluate the proposed framework using five TREC newswire and web collections. The results show that in most cases the proposed PRF method for language modeling framework significantly outperforms competitive baselines in terms of mean average precision (MAP). Moreover, RFMF always outperforms all the baselines in terms of precision at top-ranked documents (P@10 and P@20). The experiments also show that the proposed method is more robust than the baselines, in all the collections. The excellent performance of the proposed PRF method for the vector space retrieval model indicates the generality of the RFMF framework.

It should be noted that in addition to achieving good results, the proposed framework opens up a new research direction for the pseudo-relevance feedback problem and similar tasks (e.g., user profile updating), which is relating the task to a recommendation problem. This work also indicates the potential for other collaborative recommendation techniques in this task.

2. RELATED WORK

In this section, we first present a number of existing PRF methods and discuss how they differ from the proposed framework. We further briefly review the applications of matrix factorization techniques in related tasks.

2.1 Pseudo-Relevance Feedback

Query expansion via pseudo-relevance feedback is a common technique used to improve the retrieval effectiveness in many retrieval models [14, 20, 21, 22, 31, 35, 42, 43]. In this subsection, we review the PRF methods which are the most related to our research.

Rocchio algorithm [31] is one of the earliest relevance feedback methods, which was developed for the vector space retrieval model. Rocchio algorithm combines the original query vector with positive and negative feedback vectors which are created using the relevant and non-relevant documents, respectively. Croft and Harper [5] proposed to improve the retrieval effectiveness without relevance information using pseudo-relevance feedback. The negative feedback vector is not often used for PRF since there are too many non-relevant documents which are spread out in the space. Later on, with the development of classical probabilistic models, a number of PRF algorithms based on the Robertson/Sparck-Jones weight [30] have been proposed.

Because of the well-defined structure of the language modeling framework, several PRF methods have been proposed for this framework, such as relevance models [14] and model-based feedback methods [43]. The idea behind the language model-based PRF methods is to use top retrieved documents to provide more accurate query language models. Model-based feedback methods, including the mixture feedback model and the divergence minimization model (DMM), try to separate the topical model of top retrieved documents from the background model. Recently, Lv and Zhai [22] showed that DMM generates a skewed feedback model. They proposed the maximum-entropy divergence minimization model (MEDMM) by adding an entropy term to regularize DMM, which leads to significant improvements. The regularization strategy for PRF models was previously used by Tao and Zhai [35] for the mixture model. Their main focus was to generate a feedback model that does not need to be interpolated with the original query model. The relevance models [14] are the other early PRF methods for the language modeling framework that are still among the state-of-the-art methods. Unlike the model-based feedback methods [43] that explicitly model the pseudo-relevant documents, Lavrenko and Croft [14] modeled a more generalized notion of relevance in the relevance models. The comparative analysis of PRF methods done by Lv and Zhai [20] showed that the mixture model and a variant of relevance models (i.e., RM3 [1]) outperform other PRF methods, including the regularized mixture model [35] and the divergence minimization model [43]. In their experiments, the RM3 method was shown to be more robust than the other methods. Parapar et al. [26] employed relevance models for collaborative recommendation. Recently, Dehghani et al. [8] proposed a PRF model which penalizes both general and rare terms.

The aforementioned methods that are the most similar to the proposed RFMF model, are based on unigram language models without having access to additional sources of information. Using other information, such as term proximity [21, 24], term topics [38], term dependency [23], and semantic similarity [25, 40, 41] has been shown to be effective in PRF and query expansion. There are also a number of learning-based query expansion methods that use thesauruses and external resources [32, 33]. In addition, there has been research to determine which documents can be useful in generating feedback models [7, 10]. Although some of these methods perform well, they have different constraints than our work and are not used in this paper. Note that the RFMF framework can potentially use these sources of information, and this will be investigated in future work.

In addition to the main idea behind the RFMF framework, there is a fundamental difference between the proposed PRF method and the existing ones (more details in Section 4): in many existing methods, the feedback model is only constructed using the pseudo-relevant documents (the feedback model is usually interpolated with the query model); however, in our proposed framework, we consider the original query in addition to the feedback documents. Indeed, the core idea of the proposed framework is to expand the original query using the words that not only discriminate feedback documents from other documents, but are also related to the original query.

2.2 Matrix Factorization Techniques

In general, matrix factorization techniques learn low-ranked representations (also called latent factors) of an input matrix. These latent factors have been widely used in a number of IR tasks, such as document clustering [2]. In recommender systems, the latent factors are used to predict unseen values in the initial matrix. In fact, decomposing a given matrix and multiplying the latent factors will give us a re-estimated version of the initial matrix, and thus it will help to predict unknown values in the initial matrix.

Matrix factorization techniques have been widely studied in various tasks. They have attracted considerable attentions in collaborative recommender systems due to their efficiency and effectiveness [34]. Latent semantic indexing (LSI) [6] is an application of matrix factorization in information retrieval that uses the singular value decomposition method. Unlike in recommender systems, LSI directly uses the latent factors. Matrix factorization has also been studied in other IR-related tasks, such as topic modeling [36], word embedding [27], and document summarization [17].

In the context of pseudo-relevance feedback, Wu et al. [37] proposed to use matrix factorization to cluster terms and expand the query using these clusters. He et al. [11] also integrated the results of text and image contents to improve the image retrieval performance using local LSI. The main idea in these methods are fundamentally different from ours.

3. RFMF: A PRF FRAMEWORK BASED ON MATRIX FACTORIZATION

The purpose of the RFMF framework is to recommend a number of words for a given query in a weighted manner, such that the final weights of terms in the updated query model better reflect the terms' contributions in the query. This recommendation should be done using feedback documents. To this aim, we use the following mappings to cast the PRF task to a simple collaborative recommendation problem:[2] the query and the feedback documents play the role of users, words play the role of items, and the rates are computed using the *weights* of seen terms in the query and the feedback documents.[3]

Each query/document contains a limited number of words and thus is a small sample for the model that represents it. Therefore, the weights of a few terms are available for each query/document and we should estimate the weights of all terms even though most of them have not appeared in that query/document. This shows that predicting term weights for the PRF task suffers from the sparsity issue. Similarly, collaborative recommender systems typically have the same issue and a number of matrix factorization techniques have been shown to be effective in such situations [29, 34].

An overview of the RFMF framework is presented in Figure 1. As shown in this figure, RFMF takes as input the query and the feedback documents. Feedback documents could be selected from relevant documents in case that relevance judgements are available; otherwise, it can be assumed that the top-k retrieved documents for a given query are relevant to the query, and thus they could be considered as pseudo-relevant documents.[4] To update the query model using the feedback documents, RFMF creates a matrix from the inputs, in which rows and columns correspond to query/documents and terms, respectively. Then, it employs a matrix factorization algorithm to decompose the created matrix. Multiplying the resulted matrices will result a re-estimated version of the input matrix. Indeed, matrix factorization techniques predict the weights of terms that have not appeared in the query and these predictions are done based on the available term weights. Since the query weights are put in the first row of the input matrix (see Figure 1), RFMF also uses the first row of the output matrix to re-estimate the query model. As described, the RFMF framework is independent of the retrieval model that is used. Therefore, RFMF is a *general* framework and can be deployed on top of any retrieval model. In more detail, RFMF includes three main components: the model-specific *weight* function, the model-specific query updating method, and the matrix factorization component. The first two components depend on the retrieval model, and thus they should be implemented for each retrieval model.

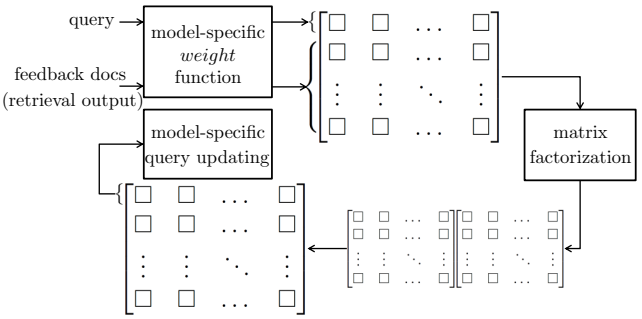

Figure 1: Overview of the RFMF framework.

In the proposed feedback procedure, RFMF first creates a matrix $R = [r_{ij}]$ with $k+1$ rows where the first row corresponds to the input query and each of the remaining k rows corresponds to each of the top-k documents retrieved in response to the query. The matrix R has $|W|$ columns where W is the set of unique words appeared in the top-k documents or the query. Each element of the matrix R is calculated as:

$$r_{ij} = \begin{cases} weight(w_j, q) & \text{if } i = 1 \\ weight(w_j, d_{i-1}) & o.w. \end{cases} \quad (1)$$

where q, d_{i-1}, and w_j respectively denote the given query, the $(i-1)^{th}$ retrieved document, and the j^{th} element of the set W. The function *weight* shows how much the given word that appeared in the given query/document distinguishes that query/document from the other documents (or collection).

To implement the RFMF framework, we should answer three questions: *i)* how to decompose the created matrix?, *ii)* how to define the *weight* function?, and *iii)* how to update the query model? In the rest of this section, we first

[2]Collaborative recommender systems seek to predict the rating that user would give to a given item. The predictions are usually done based on the similarity between the users' behavior [29, 34].

[3]Documents are often much longer than queries. Hence, from the recommender systems perspective, the purpose is to make recommendations for cold-start users (queries). Although matrix factorizations are not particularly designed for cold-start recommendation problems, in this paper, we consider these techniques which are highly successful in various recommendation scenarios. We leave the study of cold-start recommendation solutions for PRF as a future work.

[4]In this paper, we focus on pseudo-relevance feedback which is a more realistic case.

introduce a non-negative matrix factorization technique for providing an answer to the first question. Then, we answer the next two questions for two widely used retrieval models: language modeling and vector space model. Finally, we discuss the computational complexity of the implemented framework to assess the possibility of employing RFMF in real-world applications.

3.1 Non-Negative Matrix Factorization

Non-negative matrix factorization (NMF) is a matrix decomposition method that has been extensively exploited in various tasks, e.g., document clustering [2], document summarization [17], and probabilistic latent semantic indexing [9]. NMF has been shown to be highly effective in collaborative recommender systems [18, 44].

NMF is a matrix factorization algorithm which finds non-negative factorization of a given non-negative matrix. Formally writing, the purpose of NMF is to decompose a non-negative matrix $A = [a_{ij}] \in \mathbb{R}_+^{m \times n}$ to two matrices $U = [u_{ik}] \in \mathbb{R}_+^{m \times r}$ and $V = [v_{kj}] \in \mathbb{R}_+^{r \times n}$ where $A \approx UV$. It should be noted that the parameter r is a positive integer (usually $r < m, n$) which should be given as the input of the NMF algorithm. To find the matrices U and V, an objective function which computes the difference between A and UV should be minimized. We consider the most popular objective function for the NMF algorithm, i.e., the Frobenius norm, also known as the Euclidean distance, which can be calculated as:

$$\{U, V\} = \arg\min_{U,V} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} (a_{ij} - [UV]_{ij})^2$$

Note that the objective function is only computed for the non-zero elements of the matrix A. It can be shown that although the above objective function is convex when one of the matrices U and V is fixed, it is not convex in the case that both matrices should be predicted. Hence, it is not expected to find the global optimal solution for an NMF problem [12]. Gradient descent might be the simplest way to find local solutions, but its convergence is very slow. Conjugate gradient is an alternative for solving this problem, which has been shown to be faster. The gradient-based methods suffer from being very sensitive to the step size [16]. We consider multiplicative update rules to find the solution of NMF problems as follows:

$$u_{ik} \leftarrow u_{ik} \frac{[AV^T]_{ik}}{[UVV^T]_{ik}}, \qquad v_{kj} \leftarrow v_{kj} \frac{[U^T A]_{kj}}{[U^T UV]_{kj}} \tag{2}$$

where $1 \leq i \leq m$, $1 \leq k \leq r$, and $1 \leq j \leq n$. It has been proven by Lee and Seung [16] that the defined objective function is non-increasing under the above update rules, and thus its convergence is guaranteed.

After performing the mentioned iterative algorithm, we can compute the matrix $\widehat{A}$ by multiplying the matrices U and V. Although the matrices U and V are used in many tasks as latent features, in this paper we just use the matrix $\widehat{A}$ as the output of the NMF algorithm. Indeed, the matrix $\widehat{A}$ is an approximate estimation for matrix A.

NMF could be a useful method for information retrieval tasks where there is no negative element in the matrix. In addition, the non-negativity constraint makes NMF a suitable method for sparse representations [15], and thus NMF can cope with the sparsity characteristic of IR tasks. That is why we choose NMF among the matrix factorization techniques for implementing the RFMF framework.

3.2 RFMF for Language Models

In this subsection, we explain how to choose a proper *weight* function and how to update the query model for the language modeling framework [28]. Our goal is to define the weight of each word w for each document d, such that it reflects how much this word discriminates the document d from the other documents. In other words, we assign high probabilities to the words with high frequencies in the document d, but with low frequencies in the collection. Indeed, similar to [43], the words of each document d are categorized into two types: background words and topical words. To this end, we consider a boolean hidden variable t where the values 0 and 1 denote the background words and topical words, respectively.

According to the purpose of pseudo-relevance feedback, we define the function $weight(w, d)$ (used in Equation (1)) as $p(t = 1|w, d)$. Considering the Bayes rule, we can compute this probability as:

$$\begin{aligned} weight(w, d) &= p(t = 1|w, d) = 1 - p(t = 0|w, d) \\ &= 1 - \frac{p(t = 0|d)p(w|t = 0, d)}{p(w|d)} \end{aligned} \tag{3}$$

Under the assumption that conditioned on the event $t = 0$, the word w is independent of the document d,[5] i.e., $p(w|t = 0, d) = p(w|t = 0)$, Equation (3) becomes:

$$p(t = 1|w, d) = 1 - \frac{\lambda_d p(w|\mathcal{C})}{p(w|d)} \tag{4}$$

where $\mathcal{C}$ denotes the background collection. In the calculation of Equation (4), we estimate $p(w|t = 0)$ as $p(w|\mathcal{C})$. The parameter λ_d is also equal to $p(t = 0|d)$ which shows how much the representation of document d comes from the background words. To calculate $p(w|d)$, if we use the maximum likelihood estimation and one of the simple smoothing methods such as, Absolute discounting, Jelinek-Mercer, and Dirichlet prior, with the unseen coefficient of α_d, Equation (4) can be rewritten as:

$$\begin{aligned} p(t = 1|w, d) &= 1 - \frac{\lambda_d p(w|\mathcal{C})}{\alpha_d p(w|\mathcal{C}) + (1 - \alpha_d) p_{ML}(w|d)} \\ &= \frac{(\alpha_d - \lambda_d) p(w|\mathcal{C}) + (1 - \alpha_d) p_{ML}(w|d)}{\alpha_d p(w|\mathcal{C}) + (1 - \alpha_d) p_{ML}(w|d)} \end{aligned} \tag{5}$$

where $p_{ML}(w|d)$ is the maximum likelihood estimation of the probability of word w in the document d.[6]

Since $p(t = 1|w, d)$ is a probability and should be in the $[0, 1]$ interval, we can conclude that $\lambda_d \leq \alpha_d$. On the other hand, by decreasing the value of the parameter λ_d, the influence of $p(w|\mathcal{C})$ will be increased and in the extreme case, i.e., $\lambda_d = 0$, $p(t = 1|w, d)$ is equal to 1 for all w in d. To put the issue into perspective, the parameter λ_d should be less than or equal to α_d and also very close to it.

[5] It is not an unusual assumption since $t = 0$ shows the background words. In the probabilistic retrieval models, it is assumed that in the non-relevancy conditions, document and query are independent [42], which is similar to our assumption.

[6] $weight(w, q)$ can be computed similarly. The unseen coefficient for computing $p(t = 1|w, q)$ should be selected carefully to give a high weight to query terms.

It should be noted that since the function *weight* is equal to a probability, it is positive and there is no problem for the NMF computations. In the next step, we perform matrix factorization and then compute the feedback language model $\theta_{\mathcal{F}}$ using the following equation:

$$p(w_j|\theta_{\mathcal{F}}) = \frac{\widehat{r}_{1j}}{\sum_{q=1}^{|W|} \widehat{r}_{1q}}$$

where $\widehat{R} = [\widehat{r}_{ij}]$ is the estimation of the matrix R after performing the NMF algorithm. Since we first put the weight of each term in the query in the first row of matrix R, the first row of matrix $\widehat{R}$ shows the re-estimated version of the query model with the help of feedback documents. Therefore, we can use these weights for estimating the feedback language model. Because there are too many words in the feedback language model and adding all of them to the query may decrease both the retrieval efficiency and effectiveness, similar to previous work [20, 21, 22, 43] we add the top m terms of the feedback language model to the query. Hence, the language model $\theta_{\mathcal{F}'}$ is computed using the normalized values of the top m terms in $\theta_{\mathcal{F}}$. We linearly interpolate the maximum likelihood estimation of the original query model θ_Q with the feedback language model $\theta_{\mathcal{F}'}$ as follows:

$$\theta_{Q'} = (1-\alpha)\theta_Q + \alpha\theta_{\mathcal{F}'} \quad (6)$$

where the parameter $\alpha \in [0, 1]$ controls the weight of θ_Q and $\theta_{\mathcal{F}'}$ in the computation of the modified query language model $\theta_{Q'}$.

3.3 RFMF for Vector Space Model

Based on the aforementioned intuition of the function *weight*, this function should represent how much each word w makes the document d specific. Because of the heuristic nature of the vector space model, we can consider several heuristic *weight* functions. One of these functions is defined as below:[7]

$$weight(w, d) = TF(w, d) * IDF(w)$$

where TF and IDF are term frequency and inverse document frequency, respectively. The multiplication of TF and IDF shows how much the word w is specific for document d, and thus it would be a proper definition for the function *weight*. We can also consider other heuristic *weight* functions, such as $TF(w, d)$. But according to the intuition behind the function *weight* and our results[8], we propose to use TF-IDF function which has been widely used in the literature and makes more sense compared to the others. Note that TF can be computed as raw TF, logarithmic TF, or even length normalized TF which are extensively exploited in the literature. In these formulations, it is clear that *weight* is a non-negative function and satisfies the non-negativity constraint of the NMF algorithm.

After performing the matrix factorization, we define the feedback vector $\vec{F}$ as $\vec{F}_j = \widehat{r}_{1j}$ where $\widehat{R} = [\widehat{r}_{ij}]$ is the estimation of the matrix R after performing the NMF algorithm. Similar to the proposed feedback method for the language modeling framework (see Section 3.2), we consider the first row of matrix $\widehat{R}$ to estimate the feedback vector $\vec{F}$, since we put the query vector in the first row of matrix R. Thus, $\vec{F}$ is the re-estimated version of the query vector $\vec{Q}$. Again, we only consider the top m terms with the highest values to create the feedback vector $\vec{F}'$.

Since the goal of most retrieval tasks, including ad-hoc retrieval, is to rank the documents based on their similarities to the query, the norm of the query vector is not important; it can be easily proved that ranking based on the dot product or the cosine similarity does not depend on the norm of the query vector. Therefore, the important issue in combining the query vector and the feedback vector is to be able to produce all possible angles between the angles of these two vectors. According to the aforementioned arguments, we linearly interpolate the unit query vector with the unit feedback vector. As a result, the modified query vector $\vec{Q}'$ is calculated as below:

$$\vec{Q}' = (1-\alpha)\hat{Q} + \alpha\hat{F}'$$

where $\hat{Q}$ and $\hat{F}'$ respectively denote the unit original query and feedback vectors. The parameter α controls the influence of each vector and it should be in the $[0, 1]$ interval.

4. DISCUSSION

In this section, we first discuss the time complexity of the proposed framework to show that it is fast enough to be deployed in real-world applications. We further explain why the proposed PRF framework should perform well.

4.1 Time Complexity

The computational complexity of each NMF step (see Equation (2)) in the proposed framework is $O(k.|W|.r)$ [19], where k, $|W|$, and r denote the number of feedback documents, the number of unique terms in the feedback documents and the query, and the rank of matrix factorization, respectively. Furthermore, as mentioned in Section 3.1, r is less than or equal to the matrix dimensions. In addition, since NMF is an iterative algorithm, $O(k^2.|W|.t)$ is an upper-bound for the total time complexity of the RFMF framework where t is the number of iterations.

As widely discovered in previous work [43, 20, 22, 25, 40], k is usually set to a small number, e.g., 10. Hence, by considering a reasonable value for t (e.g., 1000), the computations of the proposed method will be completed in a very short time. Moreover, there are a number of research studies that try to reduce the running time of NMF methods. For instance, Yu et al. [39] proposed a parallel algorithm to improve the efficiency of the NMF algorithm. All these show that the proposed framework can be used in real-world applications.

4.2 Why should RFMF work well?

According to the intuition behind the proposed framework which is mentioned in Section 3, we believe it is clear why the proposed framework selects good expansion terms for a given query. In this subsection, we explain why the proposed framework can perform better than other methods?

In the following, we list the special characteristics of the proposed framework:

• RFMF uses matrix factorization techniques that are well-known to be able to capture latent features. In fact, matrix factorization techniques find low-ranked representations of the input matrix. These matrices with lower dimensions can model latent factors. In our particular problem, we expect that matrix factorization techniques capture latent fea-

[7] $weight(w, q)$ can be computed similarly.

[8] We study a few other heuristic weightings, such as TF and term occurrence, in our experiments.

Table 1: Collections statistics.

ID	collection	queries (title only)	#docs	avg doc length	#qrels
AP	Associated Press 88-89	TREC 1-3 Ad-Hoc Track, topics 51-200	165k	287	15,838
Robust	TREC Disks 4 & 5 minus Congressional Record	TREC 2004 Robust Track, topics 301-450 & 601-700	528k	254	17,412
WT2g	general web crawl	TREC 8 Web Track, topics 401-450	247k	645	2279
WT10g	general web crawl	TREC 9-10 Web Track, topics 451-550	1692k	399	5931
GOV2	2004 crawl of .gov domains	TREC 2004-2006 Terabyte Track, topics 701-850	25,205k	648	26,917

tures between the important terms in pseudo-relevant documents. These latent features can help to avoid adding non-relevant terms to the query. This property of RFMF can increase its robustness.

• RFMF considers both the original query and the pseudo-relevant documents for query expansion; while many existing PRF methods, e.g., mixture model, divergence minimization model, etc., only construct the feedback model from the pseudo-relevant documents. In other words, RFMF tries to expand the query using the terms that not only discriminate the feedback documents from a collection, but also those that are relevant to the original query terms. We believe that this can be successfully done by RFMF, since the goal of collaborative recommender systems is to recommend items that are similar to those that the user likes based on the preferences of other users with similar preferences. A few PRF methods, such as RM3 [14] which is known as one of the best and the most robust PRF methods [20] also consider the query terms with independence or conditionally independence assumptions. RFMF can relax these independence assumptions by capturing latent features.

• Unlike a number of the existing PRF methods, such as the mixture model [43] and its variations, which put all the feedback documents together as a unit, RFMF considers each document separately. In other words, in these models, each *word* contributes equally, while in the proposed method each *document* contributes equally. The most important advantage of our assumption is to penalize long feedback documents. Note that a number of PRF methods, such as relevance models [14], also have the same assumption as ours.

5. EXPERIMENTS

5.1 Experimental Setup

We use five standard test collections in our experiments: AP (Associated Press 1988-89), Robust (TREC Robust Track 2004 collection), WT2g (TREC Web Track 2000 collection), WT10g (TREC Web Track 2001-2002 collection), and GOV2 (TREC Terabyte Track 2004-2006 collection). The first two collections are homogeneous datasets containing news articles. WT2g, WT10g, and GOV2 are small, medium, and large web collections, respectively. The statistics of these datasets are reported in Table 1. All documents were stemmed using the Porter stemmer. Stopwords were removed in all the experiments using the standard INQUERY stopword list. In all experiments, only the title field of the TREC topics were used as queries.

For the language modeling framework, we employed the KL-divergence retrieval model [13] with the Dirichlet prior smoothing method. We used the dot product similarity in the experiments related to the vector space model. Experiments were carried out using the Lemur and the Galago toolkits.[9] We also employed the non-negative matrix factorization method implemented in the NIMFA library[10] [45].

5.1.1 Parameter Setting

In all experiments, the Dirichlet prior smoothing parameter μ is set to the typical value of 1000. The number of feedback documents, the feedback term count, and the feedback coefficient are set using 2-fold cross validation over each collection. We sweep the number of feedback documents between $\{10, 25, 50, 75, 100\}$, the feedback term count between $\{10, 25, 50, 75, 100\}$, and the feedback coefficient between $\{0, 0.1, \cdots, 1\}$. The other hyper-parameters of the baselines, if any, are also set using the same procedure.

As suggested in the literature [15, 16], the rank of the NMF algorithm should not exceed the initial matrix dimensions. Therefore, since there are a few number of rows in all the matrices created in the proposed method, we set the rank of NMF algorithm to the number of rows. We use the KL-divergence as the objective function for NMF. The maximum number of NMF iterations is also set to 1000. Note that although solving NMF is not a convex optimization problem, the results obtained by different NMF runs in our task are very close to each other.

5.1.2 Evaluation Metrics

To evaluate retrieval effectiveness, we use mean average precision (MAP) of the top-ranked 1000 documents as the main evaluation metric. In addition, we also report the precision of the top 10 and 20 retrieved documents (P@10 and P@20). Although P@10 is a popular metric to show the effectiveness of methods in top retrieved documents, it would not be enough for our task, since the query is updated using the top 10 documents[11]. Therefore, we also consider P@20 which offers us an insight into the effectiveness of methods in the next top retrieved documents. Statistically significant differences of MAP, P@10, and P@20 values are determined using the two-tailed paired t-test computed at a 95% confidence level ($p\_value < 0.05$).

To evaluate the robustness of methods, we use the robustness index (RI) [4] which is defined as $\frac{N_+ - N_-}{|Q|}$, where $|Q|$ denotes the number of queries. N_+/N_- shows the number of queries improved/decreased by the feedback method. The RI values are always in the $[-1, 1]$ interval and the methods with higher values are more robust.[12]

[9] http://www.lemurproject.org/

[10] http://nimfa.biolab.si/

[11] Interestingly, although we do cross-validation to set the number of feedback documents, all the feedback methods that we study in this paper always select 10 as the number of feedback documents.

[12] To avoid the influence of very small performance changes in the RI values, we only consider the improvements/losses higher than 10% (relatively).

Table 2: Comparison of different pseudo-relevance feedback methods in the language modeling framework. Superscripts 0/1/2/3/4 indicate that the improvements over MLE/MIX/RM3/RM4/MEDMM are significant. The highest value in each row is marked in bold.

Dataset	Metric	MLE	MIX	RM3	RM4	MEDMM	RFMF
AP	MAP	0.2644	0.3106	0.3187	0.2875	0.3269	**0.3296**^{0123}
	P@10	0.4462	0.4450	0.4470	0.4208	0.4551	**0.4577**^{03}
	P@20	0.3792	0.4232	0.4294	0.3876	0.4289	**0.4356**^{013}
	RI	–	0.43	0.42	0.14	0.20	**0.54**
Robust	MAP	0.2490	0.2721	0.2820	0.2656	0.2793	**0.2899**^{01234}
	P@10	0.4237	0.4177	0.4356	0.4253	0.4406	**0.4442**^{013}
	P@20	0.3580	0.3677	0.3733	0.3610	0.3679	**0.3823**^{01234}
	RI	–	0.13	0.26	0.11	0.27	**0.29**
WT2g	MAP	0.3034	0.3299	0.3238	0.3092	**0.3300**	0.3290^{03}
	P@10	0.4480	0.4660	0.4680	0.4520	0.4680	**0.4700**^{03}
	P@20	0.3770	0.3980	0.3950	0.3990	0.4010	**0.4020**^{0}
	RI	–	0.22	0.18	0.10	0.26	**0.30**
WT10g	MAP	0.2080	0.2060	0.2197	0.2092	0.2226	**0.2266**^{0123}
	P@10	0.3030	0.3040	0.3141	0.3030	0.3111	**0.3221**^{013}
	P@20	0.2626	0.2687	0.2687	0.2566	0.2808	**0.2868**^{0123}
	RI	–	0.05	0.14	-0.04	0.23	**0.25**
GOV2	MAP	0.2965	0.3099	0.3135	0.2938	0.3116	**0.3221**^{01234}
	P@10	0.5372	0.5345	0.5358	0.5182	0.5500	**0.5736**^{01234}
	P@20	0.5122	0.5209	0.5274	0.5068	0.5314	**0.5416**^{0123}
	RI	–	0.05	0.15	-0.07	0.05	**0.22**

5.2 Results and Discussion

In this section, we evaluate RFMF for both language modeling and the vector space model frameworks.

5.2.1 RFMF for Language Models

Our baseline methods include (1) the standard maximum likelihood estimation (MLE) of the query model without feedback, (2) the mixture model (MIX) [43], (3,4) the relevance models (RM3 and RM4) [1, 14], and (5) the maximum-entropy divergence minimization model (MEDMM) [22] which was recently proposed to improve the performance of the original divergence minimization model [43]. We do not consider other methods, such as the regularized mixture model and the divergence minimization model, since the mentioned baselines have shown to outperform these methods in different test collections [20, 22].

As described in Section 2, there are a number of PRF methods that have extended PRF by incorporating additional evidences, such as term proximity [21, 24], semantic similarities [25, 40], and term dependencies [23], or by employing feature-based methods (e.g., [4, 23]) based on the standard PRF methods. They mainly depend on the standard PRF methods [22], and focus on various feedback evidences that are orthogonal to our research. Therefore, we do not consider these methods in our experiments to avoid unnecessary apples-to-oranges comparisons.

The results obtained by the proposed method and the baselines are reported in Table 2. According to this table, all the baseline feedback methods outperform MLE (without feedback) which shows the effectiveness of the PRF methods in the language modeling framework. In Table 2, the proposed method outperforms all the baselines in terms of MAP, P@10, P@20, and RI in all the collections, except in one case (MAP in WT2g collection) where MEDMM and MIX perform better than RFMF, but these improvements are not statistically significant. The statistical t-test shows that the MAP, P@10, and P@20 improvements over MLE are always significant. These improvements over the state-of-the-art baselines are also usually significant, especially in Robust and GOV2. These results show the effectiveness of the proposed method compared to strong PRF methods.

Furthermore, compared to all the baselines, the proposed method is shown to be more robust in all the collections, in terms of the robustness index. Interestingly, even in the WT2g collection where MEDMM and MIX perform better than RFMF in terms of MAP, the proposed method is shown to be more robust than the baselines.

By computing the absolute and the relative improvements of P@10 and P@20 obtained by the proposed method compared to MLE, we can observe that P@20 improvements are often higher than P@10 improvements. This shows that since the top 10 documents are used as the pseudo-relevant documents, precision of the top 10 documents does not increase as much as precision of the top 20 documents. In other words, the PRF method extracts the feedback terms from the top 10 documents, and thus it is likely that these documents will be retrieved again after the query expansion.

To see the sensitivity of the proposed method to the hyper-parameters, in the next set of experiments, we evaluate RFMF with different parameter values. In these experiments, we sweep one of the parameters and fix the other ones to their default values: 50 for the feedback term count, 0.5 for the feedback interpolation coefficient (see Equation (6)), and 1 for λ_d/α_d (see Equation (5)). Since all the feedback methods always select 10 as the number of feedback documents during the cross-validation (see Section 5.1.1), we fix this parameter and set it to 10. The results of these experiments in terms of MAP are plotted in Figure 2.

According to Figure 2, the proposed method is stable with respect to changes in the number of feedback terms when this parameter is sufficiently high. In other words, there is no statistically significant differences between the MAP

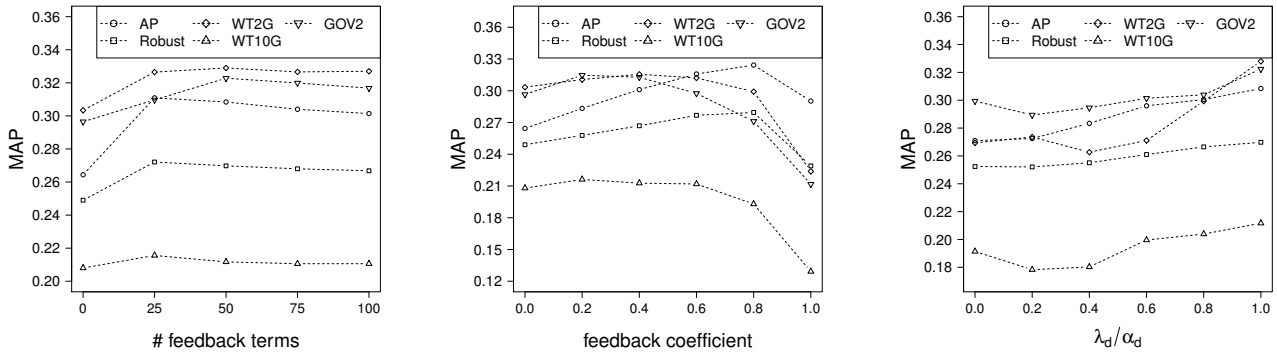

Figure 2: Sensitivity of RFMF in the language modeling framework to the number of feedback terms, the feedback interpolation coefficient, and the parameter λ_d.

values achieved by the proposed method for various number of feedback terms in $\{50, 75, 100\}$. Usually the best result is achieved when 25 terms are added to the query (this value is 50 for GOV2). This can be an advantage of the proposed method compared to a number of existing methods, such as the relevance models, which need many words (usually 100) to show their best performance; since adding many terms to the query will make the second phase of retrieval process (after query expansion) much slower. In addition, the methods that need too many terms to perform well are trying to increase the recall value by adding more terms to the query, while the proposed method increases the precision value as previously shown in Table 2.

The changes of MAP values with respect to sweeping the feedback coefficient value in Figure 2 demonstrate that when this parameter is set to 1 (when the feedback model is not interpolated with the query model), the performance dramatically drops in all collections. The reason is related to the *weight* function that we use for creating the feedback matrix. Regarding the *weight* function introduced in Section 3.2, we give a lower weight to the query/document terms that are more common in the collection and if a query contains general terms, a low weight will be assigned to them. Thus, combining the feedback language model with the query language model can give a more reasonable weight to general query terms. Therefore, to have an efficient estimation of the query model and to improve the performance, the feedback language model should be interpolated with the original query language model. An interesting observation here (in sweeping the feedback coefficient) is that the curve patterns of web collections (WT2g, WT10g, and GOV2) are very similar to each other and the curves of newswire collections also follow a similar pattern. According to these observations, the weight of generated feedback model should be near to 1 for the newswire collections, but much lower in the web collections. The reason is that web collections contain noisier and longer documents compared to the news articles, and thus the generated feedback model in these collections might be less accurate.

In the last plot in Figure 2, we change the value of λ_d. According to this plot, the best value for this parameter is equal to α_d. Note that when $\lambda_d = 0$, the matrix R becomes a binary matrix which shows the occurrence of terms in the documents. In addition, the MAP changes in response to changing the value of λ_d are more stable in newswire collections compared to the web collections. The reason is that λ_d controls the weight of each term that comes from a noisy collection and the web collections contain noisier documents.

5.2.2 RFMF for Vector Space Model

To show the generality of the RFMF framework, we also implement it for the vector space model framework. To do so, we compare the proposed method with two baselines: (1) document retrieval without feedback (NoPRF) and (2) the modified Rocchio algorithm [31, 32] which has been extensively used as a state-of-the-art PRF method in the vector space model framework for many years. We consider the following heuristics as the *weighting* function in our proposed method: (1) term occurrence (TO) which is a binary function which specifies the occurrence of terms in the documents, (2) term frequency (TF) which is a well-known heuristic in the vector space model framework, (3,4) TO-IDF and TF-IDF which are respectively computed as the product of the first two heuristics and inverse document frequency which shows how much a term is general based on its occurrences in the collection.[13]

The results obtained by the proposed method and the baselines over all the collections are reported in Table 3. According to this table, RFMF with different heuristics, as well as the Rocchio algorithm, always outperforms NoPRF. The MAP, P@10, and P@20 improvements of RFMF (with TF-IDF weighting) over the NoPRF method are always statistically significant. Among the heuristics that are considered as the weighting function for RFMF, TF-IDF performs better than the other ones in all the collections, in terms of MAP. This result was expected since our *weighting* function should show how much a term discriminates a document/query from the collection, and thus both TF and IDF are essential components in the *weighting* function. In addition, RFMF outperforms the Rocchio algorithm in terms of MAP, P@10, and P@20 in all the collections except in WT2g (in terms of MAP) and in AP (in terms of P@20).

Considering the robustness index reported in Table 3, the proposed method is always more robust than the Rocchio algorithm, which means that in general, the number of queries improved/decreased by the proposed method are higher/lower than those by the Rocchio algorithm.

To capture the sensitivity of the proposed framework to the input parameters, i.e., the number of feedback terms and the feedback interpolation coefficient, we plot the MAP values achieved by sweeping these parameters in Figure 3. In these experiments, we consider TF-IDF as the *weighting* function. According to this figure, by increasing the number of feedback terms, the performance of RFMF also increases.

[13]For all methods, we use the well-known logarithmic TF and IDF formulas used in the Lemur toolkit.

Table 3: Comparison of PRF baselines and RFMF for the vector space model framework. Superscripts 0/1 indicate that the improvements over NoPRF/Rocchio are statistically significant. The highest value in each row is marked in bold.

Dataset	Metric	NoPRF	Rocchio	RFMF			
				TO	TF	TO-IDF	TF-IDF
AP	MAP	0.2609	0.3198	0.3099^{0}	0.3201^{0}	0.3065^{0}	$\mathbf{0.3263}^{01}$
	P@10	0.3913	0.4523	0.4470^{0}	0.4356^{0}	0.4289^{0}	$\mathbf{0.4550}^{0}$
	P@20	0.3715	**0.4258**	0.4228^{0}	0.4178^{0}	0.4060^{0}	0.4205^{0}
	RI	–	0.16	**0.42**	0.22	0.22	0.38
Robust	MAP	0.2294	0.2657	0.2553^{0}	0.2646^{0}	0.2589^{0}	$\mathbf{0.2729}^{01}$
	P@10	0.4024	0.4233	0.4309^{0}	0.4229^{0}	0.4092	$\mathbf{0.4353}^{01}$
	P@20	0.3301	0.3590	0.3635^{0}	0.3612^{0}	0.3502^{0}	$\mathbf{0.3697}^{01}$
	RI	–	0.23	0.26	**0.32**	0.21	**0.32**
WT2G	MAP	0.2456	**0.2948**	0.2740^{0}	0.2783^{0}	0.2738^{0}	0.2826^{0}
	P@10	0.3980	0.4520	0.4240^{0}	$\mathbf{0.4580}^{0}$	0.4260^{0}	$\mathbf{0.4580}^{0}$
	P@20	0.3270	0.3580	0.3530^{0}	0.3610^{0}	0.3450^{0}	$\mathbf{0.3620}^{0}$
	RI	–	0.5	0.24	0.26	0.32	**0.54**
WT10G	MAP	0.1836	0.2063	0.1951	0.2064^{0}	0.1920	$\mathbf{0.2101}^{0}$
	P@10	0.2879	0.2970	0.3030^{0}	0.3061^{0}	0.2939	$\mathbf{0.3131}^{01}$
	P@20	0.2505	0.2702	0.2788^{0}	0.2763^{0}	0.2717^{0}	$\mathbf{0.2863}^{01}$
	RI	–	0.11	0.09	0.27	0.06	**0.33**
GOV2	MAP	0.2726	0.2996	0.2834^{0}	0.2972^{0}	0.2914^{0}	$\mathbf{0.3088}^{01}$
	P@10	0.4892	0.5264	0.5237^{0}	0.5325^{0}	0.5277^{0}	$\mathbf{0.5439}^{01}$
	P@20	0.4702	0.5141	0.5117^{0}	0.5223^{0}	0.5158^{0}	$\mathbf{0.5357}^{01}$
	RI	–	0.09	0.04	0.13	0.09	**0.19**

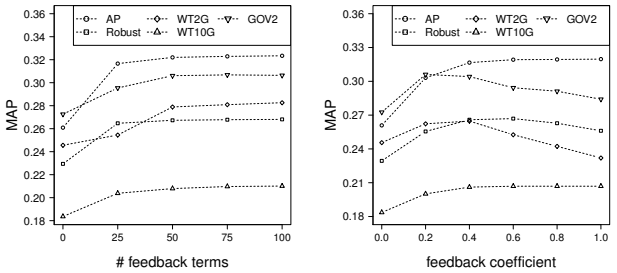

Figure 3: Sensitivity of the proposed framework in the vector space model framework to the number of feedback terms and feedback interpolation coefficient in terms of MAP.

This behavior is the same for all the collections. According to Figures 2 and 3, RFMF in the language modeling and the vector space model frameworks show different behaviors when we sweep the number of feedback terms. The reason is related to the *weighting* function, which plays a key role in the RFMF effectiveness. The results of RFMF with different heuristic functions as the *weighting* function can confirm this claim (see Table 3).

When the value of feedback interpolation coefficient changes, the performances of RFMF in different collections do not follow similar patterns. In other words, the feedback coefficient is a collection-dependant parameter and should be set in a proper way, such as cross-validation that was done for generating the results in Table 3. In AP and WT10g, our method successfully generates good expansion vectors that do not need to be interpolated with the original query vector. For the other collections, in particular for WT2g and GOV2, interpolating the expansion vector with the original query vector is vital.

By looking at Tables 2 and 3 to compare the results achieved by the RFMF framework in both language modeling and the vector space model frameworks, we can figure out that RFMF in the language modeling framework performs better than in the vector space model framework, especially in the Robust, WT2g, WT10g, and GOV2 collections. A reason is related to the *weight* function that is defined for each of these retrieval frameworks. By considering other heuristic TF weighting formulations or better *weight* functions in the RFMF framework, the results of RFMF in the vector space model framework may be improved.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed RFMF, a general pseudo-relevance feedback framework based on matrix factorization, which can be deployed on top of any retrieval model. RFMF recasts the relevance feedback task as a collaborative recommendation framework. We implemented this framework using the non-negative matrix factorization technique for two widely used retrieval frameworks: language modeling and the vector space model.

We evaluated our implemented framework on five standard TREC newswire and web collections. The results indicate that the proposed method for the language modeling framework significantly outperforms competitive PRF baselines in nearly all cases. The vector space model experiments demonstrate the generality of the RFMF framework. Furthermore, the proposed method was shown to be more robust than the baselines in both retrieval models.

The successful development of the RFMF framework for two retrieval models indicates the potential of this PRF framework to be further developed for other retrieval models (e.g., Okapi BM25 and the divergence from randomness model) in the future. In addition, enriching the defined weighting functions using additional resources and evidences (e.g., term dependencies) can be a focus of future work. The proposed framework opens up a new research direction to study other recommendation approaches, especially those

designed for cold-start recommendation problems, for PRF and related tasks (e.g., user profile updating).

7. ACKNOWLEDGMENTS

The authors gratefully thank the anonymous reviewers for their insightful comments. This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #CNS-0934322. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

8. REFERENCES

[1] N. Abdul-jaleel, J. Allan, W. B. Croft, F. Diaz, L. Larkey, X. Li, D. Metzler, M. D. Smucker, T. Strohman, H. Turtle, and C. Wade. UMass at TREC 2004: Novelty and HARD. In *TREC '04*, 2004.

[2] C. Aggarwal and C. Zhai. *Mining Text Data*. 2012.

[3] C. Carpineto and G. Romano. A Survey of Automatic Query Expansion in Information Retrieval. *ACM Comput. Surv.*, 44(1):1–50, 2012.

[4] K. Collins-Thompson. Reducing the Risk of Query Expansion via Robust Constrained Optimization. In *CIKM '09*, pages 837–846, 2009.

[5] W. B. Croft and D. J. Harper. Using Probabilistic Models of Document Retrieval Without Relevance Information. *J. of Documentation*, 35(4):285–295, 1979.

[6] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by Latent Semantic Analysis. *J. Assoc. Inf. Sci.*, 41(6):391–407, 1990.

[7] M. Dehghani, S. Abnar, and J. Kamps. The Healing Power of Poison: Helpful Non-relevant Documents in Feedback. In *CIKM '16*, 2016.

[8] M. Dehghani, H. Azarbonyad, J. Kamps, D. Hiemstra, and M. Marx. Luhn Revisited: Significant Words Language Models. In *CIKM '16*, 2016.

[9] E. Gaussier and C. Goutte. Relation Between PLSA and NMF and Implications. In *SIGIR '05*, pages 601–602, 2005.

[10] B. He and I. Ounis. Finding Good Feedback Documents. In *CIKM '09*, pages 2011–2014, 2009.

[11] R. He, Y. Zhu, and W. Zhan. Using Local Latent Semantic Indexing with Pseudo Relevance Feedback in Web Image Retrieval. In *NCM '09*, pages 1354–1357, 2009.

[12] N.-D. Ho. *Nonnegative Matrix Factorization Algorithms and Applications*. PhD thesis, Universite Catholique de Louvain, 2008.

[13] J. Lafferty and C. Zhai. Document Language Models, Query Models, and Risk Minimization for Information Retrieval. In *SIGIR '01*, pages 111–119, 2001.

[14] V. Lavrenko and W. B. Croft. Relevance Based Language Models. In *SIGIR '01*, pages 120–127, 2001.

[15] D. D. Lee and H. S. Seung. Learning the Parts of Objects by Non-negative Matrix Factorization. *Nature*, 401:788–791, 1999.

[16] D. D. Lee and H. S. Seung. Algorithms for Non-negative Matrix Factorization. In *NIPS '01*, pages 556–562. 2001.

[17] J.-H. Lee, S. Park, C.-M. Ahn, and D. Kim. Automatic Generic Document Summarization Based on Non-negative Matrix Factorization. *Inf. Process. Manage.*, 45(1):20–34, 2009.

[18] Y. Li, J. Hu, C. Zhai, and Y. Chen. Improving One-class Collaborative Filtering by Incorporating Rich User Information. In *CIKM '10*, pages 959–968, 2010.

[19] C.-J. Lin. Projected Gradient Methods for Nonnegative Matrix Factorization. *Neural Comput.*, 19(10):2756–2779, 2007.

[20] Y. Lv and C. Zhai. A Comparative Study of Methods for Estimating Query Language Models with Pseudo Feedback. In *CIKM '09*, pages 1895–1898, 2009.

[21] Y. Lv and C. Zhai. Positional Relevance Model for Pseudo-relevance Feedback. In *SIGIR '10*, pages 579–586, 2010.

[22] Y. Lv and C. Zhai. Revisiting the Divergence Minimization Feedback Model. In *CIKM '14*, pages 1863–1866, 2014.

[23] D. Metzler and W. B. Croft. Latent Concept Expansion Using Markov Random Fields. In *SIGIR '07*, pages 311–318, 2007.

[24] J. Miao, J. X. Huang, and Z. Ye. Proximity-based Rocchio's Model for Pseudo Relevance. In *SIGIR '12*, pages 535–544, 2012.

[25] A. Montazeralghaem, H. Zamani, and A. Shakery. Axiomatic Analysis for Improving the Log-Logistic Feedback Model. In *SIGIR '16*, pages 765–768, 2016.

[26] J. Parapar, A. Bellogín, P. Castells, and A. Barreiro. Relevance-based Language Modelling for Recommender Systems. *Inf. Process. Manag.*, 49(4):966–980, 2013.

[27] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global Vectors for Word Representation. In *EMNLP '14*, pages 1532–1543, 2014.

[28] J. M. Ponte and W. B. Croft. A Language Modeling Approach to Information Retrieval. In *SIGIR '98*, pages 275–281, 1998.

[29] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. 2011.

[30] S. E. Robertson and K. S. Jones. Relevance Weighting of Search Terms. *J. Assoc. Inf. Sci.*, 27(3):129–146, 1976.

[31] J. J. Rocchio. Relevance Feedback in Information Retrieval. In *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice Hall, 1971.

[32] I. Ruthven and M. Lalmas. A Survey on the Use of Relevance Feedback for Information Access Systems. *Knowl. Eng. Rev.*, 18(2):95–145, 2003.

[33] X. Shen and C. Zhai. Active Feedback in Ad Hoc Information Retrieval. In *SIGIR '05*, pages 59–66, 2005.

[34] Y. Shi, M. Larson, and A. Hanjalic. Collaborative Filtering Beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges. *ACM Comput. Surv.*, 47(1):1–45, 2014.

[35] T. Tao and C. Zhai. Regularized Estimation of Mixture Models for Robust Pseudo-relevance Feedback. In *SIGIR '06*, pages 162–169, 2006.

[36] Q. Wang, Z. Cao, J. Xu, and H. Li. Group Matrix Factorization for Scalable Topic Modeling. In *SIGIR '12*, pages 375–384, 2012.

[37] Y. Wu, Q. Zhang, Y. Zhou, and X. Huang. Pseudo-Relevance Feedback Based on mRMR Criteria. In *AIRS '10*, pages 211–220, 2010.

[38] Z. Ye, J. X. Huang, and H. Lin. Finding a Good Query-Related Topic for Boosting Pseudo-Relevance Feedback. *J. Assoc. Inf. Sci. Technol.*, 62(4):748–760, 2011.

[39] H.-F. Yu, C.-J. Hsieh, S. Si, and I. Dhillon. Scalable Coordinate Descent Approaches to Parallel Matrix Factorization for Recommender Systems. In *ICDM '12*, pages 765–774, 2012.

[40] H. Zamani and W. B. Croft. Embedding-based Query Language Models. In *ICTIR '16*, 2016.

[41] H. Zamani and W. B. Croft. Estimating Embedding Vectors for Queries. In *ICTIR '16*, 2016.

[42] C. Zhai. *Statistical Language Models for Information Retrieval*. 2008.

[43] C. Zhai and J. Lafferty. Model-based Feedback in the Language Modeling Approach to Information Retrieval. In *CIKM '01*, pages 403–410, 2001.

[44] S. Zhang, W. Wang, J. Ford, and F. Makedon. Learning from Incomplete Ratings Using Non-negative Matrix Factorization. In *SDM '06*, pages 549–553, 2006.

[45] M. Zitnik and B. Zupan. NIMFA: A Python Library for Nonnegative Matrix Factorization. *J. Mach. Learn. Res.*, 13:849–853, 2012.