Learning to Rank Documents for Ad-Hoc Retrieval with Regularized Models

Guihong Cao, Jian-Yun Nie
Département d'Informatique et de Recherche Opérationnelle, Université de Montréal
{caogui,nie}@iro.umontreal

Luo Si
Department of Computer Science
Purdue University
lsi@cs.purdue.edu

Jing Bai
Département d'Informatique et de Recherche Opérationnelle, Université de Montréal
baijing@iro.umontreal.ca

ABSTRACT

In language modeling (LM) approaches for information retrieval (IR), the estimation of document model is critical for retrieval effectiveness. Recent studies have proven that mixture models combining multiple resources can improve the accuracy of the estimation. There arises the problem of how to estimate the mixture weights in the model. In most previous studies, the mixture weights are assigned manually. In some other studies, the mixture weights are assigned using supervised or unsupervised learning. However, we observe that the mixture weights are the same for all the queries. In addition, they can be very unbalanced. In this paper, we proposed two regularized models to estimate the query-dependent weights, one is a variant of EM algorithm - Deterministic Annealing EM (DAEM), and another is a L_2-regularized log-linear model (RLM). Both of them rely on some regularization methods to avoid unbalanced mixture weight and to make them better fit the test data. We evaluate the two models on one TREC collection. Experimental results show that the two models perform very well. Especially, the RLM even outperforms the model using optimal mixture weights obtained by exhaustive search in the parameter space.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval Models – regularized mixture model, parameter setting

General Terms

Algorithms, Experimentation, Performance

Keywords

Information Retrieval, Language Modeling, Regularized Mixture Model, unsupervised learning

1. INTRODUCTION

In recent years, the utilization of language models in information retrieval (IR) has increased in popularity due to their simplicity, clear probabilistic meaning, as well as efficiency and excellent performance [1, 2, 3, 4]. The basic idea behind is to compute the conditional probability *P(q|d)*, i.e., the likelihood of the query *q* given the observation of a document *d*, and the documents are ranked in decreasing order of the likelihood. A number of methods have been proposed to compute this conditional probability. In most approaches, the computation is conceptually decomposed into two distinct steps: (1) Estimating the document model; (2) Computing the query likelihood using the estimated document model.

In all approaches, smoothing of document models has been proven to be very critical [4, 5]. Smoothing is originally proposed to avoid zero probability: the maximum likelihood estimator assigns zero probability to terms which do not occur in the document. This is unreasonable because the zero count is usually caused by the small sample set of the data, and a larger data set would probably contain the term. Therefore, smoothing has been used to correct this problem.

Most smoothing methods leverage the occurrence of the term in the whole collection with a simple interpolation [2, 3, 6]. However, these methods are too coarse to estimate an accurate document model. For example, a document about "algorithm" may not contain the term "computer". With the simple smoothing methods, the smoothed document model will assign the term "computer" with a non-zero probability according to its occurrence in the global collection. However, at the same time, the term "year", which does not appear in the document, will also be assigned a non-zero probability. Its probability is even larger than that of "computer" because of its higher frequency in the document collection. This is obviously contradictory to our intuition. This problem is caused because the smoothing methods only consider the term occurrence within the document and the global collection without taking into account the relationships between terms.

However, recent studies have shown that if the document model is enriched with other information resources, such as local corpus structure [5, 22] as well as word relationships [8]. Whatever resources are used, the above models are formulated as an n-component mixture model, with each component corresponding to one resource. To be specific, most approaches formulate each resource as a probabilistic model and combine all resources via a linear interpolation. Therefore, there arises a problem of determining the mixture weight of each component. The weight measures the importance of the corresponding component. It is very important to assign the mixture weights in an appropriate way; otherwise, the potential of the mixture model cannot be fully exploited, and the retrieval effectiveness can even be degraded. Instead of exploiting other useful resources, we focus in this paper

SIGIR'07, July 23–27, 2007, Amsterdam, Netherland.

on the estimation of the mixture weights, while assuming that the resources have been given.

Several methods have been used in previous studies to determine the mixture weights. We classify the methods into three categories: manually setting, supervised learning and unsupervised learning. [5, 23] set the parameters manually. This method requires the user to have good prior knowledge about the resources used; otherwise the inappropriate weights, that do not fit the data, may be used. [7] followed the supervised learning principle. In their work, the quality of mixture weights is measured by a cost function (or loss function). The weights were then learned so as to minimize the cost function over the training data. This method does not require that the user has prior knowledge about the resources and it also makes the mixture model extendable. However, it also runs the risk of determining inappropriate weights if the training data does not fit the test data very well. [4, 8] adopted the unsupervised learning methods. In their work, they try to determine mixture weights in order to maximize the likelihood of the query given the document models, using EM algorithm [9]. Although the unsupervised learning method can avoid some of the problems occurring in other two methods (such as the necessity to have relevance judgments), the EM algorithm used is not regularized. The weights estimated may overstress the importance of one component, while making the weights of other components close to zero. Therefore, early stop was often used in their work to stop the EM algorithm after a predefined number of iterations [4, 8] before convergence. Such an approach is not motivated by a strong theoretical justification, but reflects the inability for the EM algorithm to capture the element to be learnt.

Another strong characteristic of the previous approaches is that the mixture weights are query-independent, or the same mixture weights are used for all the queries. Indeed, some component model can have a larger impact on some queries than other component models. Therefore, it is more reasonable to determine query-dependent mixture weights.

In this paper, we proposed two methods to train regularized mixture models, one is a variant of EM algorithm and another is based on the log-linear model that determines query-dependent mixture weights. Both methods follow the unsupervised learning principle so that no prior knowledge and training data is required. With regularization, the mixture weights can be assigned in a more appropriate manner. It is thus a more principled way than the EM algorithms with early stop used in [4, 8]. We evaluate the two methods with a TREC collection. The experiment results show that one method can produce comparable results as the model with the optimal query-independent mixture weights obtained by exhaustive search (OptM) and another one with query-dependent weights even outperforms the OptM substantially. This shows that it is important to assign mixture weights to the component models according to the query at hand.

The remaining of the paper is organized as follows. We discuss the mixture modeling approaches for IR and some previous studies on the estimation of mixture weights in section 2. Two regularized models to estimate the mixture weights are described in section 3. Section 4 describes the mixture model we used and the way to estimate the parameters in each component. We then evaluate the two regularized models with a set of experiments in section 5, followed by some discussions about the experimental results. Section 6 concludes the paper and suggests some avenues for future work.

2. Related Work

In the classical LM approaches [2, 3, 6] to IR, a multinomial model $P(w|d)$ over terms is estimated for each document d in the collection C to be indexed and searched. This model is used to assign a likelihood to a user's query $q = q_1 q_2 \dots q_n$. In most cases, each query term is assumed to be independent of the others, so that the query likelihood is estimated by $P(q|d) = \prod_{i=1}^{n} P(q_i|d)$, which is used to rank the documents.

As in speech recognition, the document model for IR must be smoothed to adjust zero probability and small probabilities. Several smoothing strategies, such as Jelinek-Mercer smoothing, Absolute discounting smoothing and Dirichlet smoothing, are discussed in [6]. All the three smoothing methods utilize the global collection information with a simple interpolation. The Jelinek-Mercer method, for instance, smoothes the document model with the following formula:

$$P(w|d) = \lambda P_{ml}(w|d) + (1-\lambda)P(w|C) \quad (1)$$

where $P_{ml}(w|d)$ is the probability of the term estimated by maximum likelihood estimator, $P(w|C)$ is the term probability within the whole collection, which is called the collection model, and λ is the interpolation factor. Clearly, these smoothing methods can overestimate the probabilities of terms which occur in the collection frequently, such as "year", "month" and so on.

Recent studies have proven that other resources, such as the local corpus structure and external thesaurus, are helpful in the estimation of the document model. [5, 22, 24] exploited the local corpus structures to improve the document model estimation. In these studies, similar documents in the collection are clustered. The document model is then smoothed by both the cluster model and collection model. We thus have the following formula:

$$P(w|d) = \lambda_1 P_{ml}(w|d) + \lambda_2 P(w|cls) + (1-\lambda_1-\lambda_2)P(w|C) \quad (2)$$

where $P(w|cls)$ is the cluster model. Other studies utilize the word relationship for model smoothing. The basic idea is that even if a term does not occur in a document, it should also be assigned a high probability if it is strongly related to words that occur frequently in the document. Cao et al. [8] used two types of term relationship, one is based on co-occurrences of terms and another is derived from WordNet. In their work, the document model is formulated as follows:

$$P(w|d) = (1-\lambda_1-\lambda_2)P_u(w|d) + \lambda_1 P_{coc}(w|d) + \lambda_2 P_{wn}(w|d) \quad (3)$$

where $P_u(w|d)$, $P_{coc}(w|d)$ and $P_{wn}(w|d)$ denotes the unigram model, co-occurrence model and WordNet model respectively, λ_1 and λ_2 are the mixture weights for P_{coc} and P_{wn}.

Although various resources are used in the studies mentioned above, equation (2) and (3) show that the models share two characteristics: (1) each resource used is formulated as a probabilistic model; (2) all the resources are combined via model smoothing, i.e., linear interpolation. In other words, the document model is estimated with an n-component mixture model, with each component corresponding to one resource, regardless to the query at hand.

Although the mixture models improved the retrieval effectiveness significantly, one intrinsic problem has not been resolved: how to estimate the mixture weights? The mixture weights play an

important role in the mixture model because each weight represents the importance of the corresponding resource.

A lot of methods have been proposed to adjust the mixture weights in the previous studies. Wei and Croft [23] as well as Tao et al. [5] set the mixture weights manually. This method requires the user to have good prior knowledge about the resources used so that the user can tell which resource is more important, otherwise, it may hurt the performance. Gao et al. [7] followed the supervised learning principle. Two linear discriminant models were proposed to learn the mixture weights. Given a set of training data, one used the averaged perceptron [10] to maximize the score of relevant documents and minimize the score of irrelevant documents; another method used a global optimization approach, line search, to maximize the Mean Average Precision (MAP) of training data. Liu and Croft [22] used the exhaustive search method to maximize the MAP of the training data directly. The supervised learning method does not need any prior knowledge about the resources and it can integrate any additional resource. However, it requires a set of judged queries, i.e. we know what documents are relevant to them. Relevance judgments are also difficult to obtain in practice. In addition, as the mixture weights are trained on a training data, it is possible that the weights obtained do not fit the test data. To overcome these problems, other studies adopted the unsupervised learning principle. Zhai and Lafferty [4] and Cao et al. [8] used EM algorithm to maximize the likelihood of the query with respect to all the documents in the whole collection. In their models, each query is generated with three steps (1) a document d_i is chosen according to a weight π_i, and this document model is used to generate the query; (2) for each query term, one component of the mixture model is chosen according to the mixture weights; (3) the query term is generated according to the generation probability of chosen component (resource). [8] represented the document model with a three-component mixture model which is shown in equation (3), and then the likelihood of one query is calculated with the following formula:

$$P(q|C) = \sum_{i=1}^{N} \pi_i \prod_{j=1}^{n} \begin{bmatrix} (1-\lambda_1-\lambda_2)P_u(q_j|d_i) \\ +\lambda_1 P_{coc}(q_j|d_i) + \lambda_2 P_{wn}(q_j|d_i) \end{bmatrix} \quad (4)$$

where C is the document collection, N is the number of documents in C, n is the number of query terms and π_i the importance of document d_i in the likelihood estimation. π_i and λ_i are estimated with EM algorithm in order to maximize $P(q|C)$.

Both co-occurrence (P_{coc}) and Wordnet (P_{wn}) models integrate term relationships that allow a document to be able to generate related terms. We will provide more details on them in Section 4.

Actually equation (4) describes a two-layer mixture model. The first layer is to choose a document model and the second layer to choose a component (resource) to generate one query term. Once the document model is chosen, the whole query will be generated with it. We therefore call it query-wise EM algorithm. Although excellent performance was obtained with this model, it also has several drawbacks. Firstly, the query-wise EM algorithm relies on one document model to generate all query terms, which may make π_i very unbalanced, i.e., we may assign the whole probability to one document which is mostly relevant to the query. Clearly, it is not desirable and leads to non-optimal performance. To avoid unbalanced mixture weights, early stop, i.e., stopping EM algorithm after pre-defined iterations instead of convergence, is usually adopted [4, 8]. However, this method is obviously not a principled way to deal with the problem. Secondly, this model is time-consuming since it goes through all the documents in the whole collection in each iteration. Thirdly, it is not reasonable to use the same component model to generate the whole query once a component model is selected. In fact, for a query, some of its terms can be better generated by a component model while the other terms by another model. For example, suppose a query "computer algorithm", and a document about "algorithm". The term "algorithm" may appear frequently in a document, thus it can be generated directly from the document's unigram model. However, the term "computer" can be absent from the document, so it can only be generated from some other model (e.g. by applying co-occurrence or Wordnet relations). Despite the fact that no one single component model can generate all the query terms, this document should be considered to have a high generation capacity of the query. A more reasonable method is to allow query terms to be generated from different component models.

In this paper, we propose two models to estimate the mixture weights, which also belong to the unsupervised learning family. However, they are different from the query-wise EM algorithm in three aspects: (1) Once a document is chosen, it does not generate the whole query but one query term. Therefore, we call them term-wise models. (2) Regularization is used to avoid unbalanced mixture weights. (3) To calculate the likelihood of a query, we do not go through all documents in the collections, but only the top *n* documents returned in an initial retrieval, which are called pseudo-relevant documents (PRD).

3. Regularized Mixture Model

In this section, we describe two regularized model to estimate the mixture weights, one is a variant of EM algorithm, which is called Deterministic Annealing EM algorithm (DAEM) [11], and another is based on the L_2-regularization Log-linear model [12]. In the two models, we assume each query term to be generated independently in a three-step process: (1) A document model among a set of pseudo-relevant documents is chosen with a probability π_i; (2) One component in the mixture model is chosen according to the mixture weights λ_k; (3) the query term is generated according to the generation probability $P_k(q_j|d_i)$. Therefore, the log-likelihood of the query is calculated as:

$$L(\pi, \lambda) = \sum_{j=1}^{n} log \sum_{i=1}^{R} \pi_i \sum_k \lambda_k P_k(q_j|d_i) \quad (5)$$

where π and λ are two vectors consisting of π_i and λ_k respectively, R is the number of PRD, which is set to be top 10 documents in the first retrieval, $P_k(q_j|d_i)$ is the probability with which d_i generates q_j using the *k-th* resource. From the equation, we can also observe that π and λ are estimated by one query.

3.1 Deterministic Annealing EM Algorithm

In this section, we set the objective function to be the log-likelihood of the query as Equation (5). EM algorithm is usually used to estimate the mixture weights by maximizing the log-likelihood. It is not difficult to derive the formulas to update π_i and λ_k in EM. We have:

$$\pi_i^{(m+1)} = \frac{1}{n} \sum_{j=1}^{n} \frac{\pi_i^{(m)} \sum_k \lambda_k P_k(q_j|d_i)}{\sum_{i=1}^{R} \pi_i^{(m)} \sum_k \lambda_k P_k(q_j|d_i)} \quad (6)$$

$$\lambda_k^{(m+1)} = \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{R} \pi_i^{(m+1)} \frac{\lambda_k^{(m)} P_k(q_j|d_i)}{\sum_k \lambda_k^{(m)} P_k(q_j|d_i)} \quad (7)$$

However, preliminary experimental results show that although π_i is not as unbalanced as in the query-wise EM algorithm, λ_k is still very unbalanced. Therefore, we have to do a regularization on λ_k.

Ueda and Nakano [11] proposed the DAEM algorithm, which can achieve this goal. Suppose we have got the value of π_i with equation (6), and we consider the hidden variable K as the indicator to which component is used to generate the query term. Then with standard EM algorithm, the posterior probability $\tilde{P}(K|q_j, d_i, \lambda^{(m)}, \pi^{(m+1)})$ is calculated as:

$$\tilde{P}^{(m)}\left(K\middle|q_j, d_i, \lambda^{(m)}\right) = \frac{\lambda_k^{(m)} P_k(q_j|d_i)}{\sum_{j=1}^{n} \lambda_k^{(m)} P_k(q_j|d_i)}$$

In contrast, DAEM calculates the posterior probability as follows:

$$\tilde{P}^{(m)}\left(K\middle|q_j, d_i, \lambda^{(m)}\right) = \frac{\left[\lambda_k^{(m)} P_k(q_j|d_i)\right]^{\beta}}{\sum_{j=1}^{n}\left[\lambda_k^{(m)} P_k(q_j|d_i)\right]^{\beta}}$$

where β >0 is the temperature. Therefore, we have:

$$\lambda_k^{(m+1)} = \frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{R} \pi_i^{(m+1)} \frac{\left[\lambda_k^{(m)} P_k(q_j|d_i)\right]^{\beta}}{\sum_{j=1}^{n}\left[\lambda_k^{(m)} P_k(q_j|d_i)\right]^{\beta}} \qquad (7')$$

When $\beta = 1$, DAEM becomes the standard EM algorithm. When $\beta \approx 0$, λ_k becomes a uniform distribution. If $\beta \to +\infty$, λ tends to place all the probability on one dimension, which is the most likely component to be selected. For simplicity, we abbreviate $\tilde{P}\left(K\middle|q_j, d_i, \lambda\right)$ as $\tilde{P}$. Neal and Hinton [13] show theoretically how the EM algorithm can be viewed as optimizing a single objective function over both λ and $\tilde{P}^{(m)}$. DAEM can also be seen in this way, and its objective function at a given β is:

$$\mathcal{F}\left(\lambda^{(m+1)}, \beta, \tilde{P}^{(m)}\right) = \frac{1}{\beta} H\left(\tilde{P}^{(m+1)}\right) + \sum_{j=1}^{n}\sum_{i=1}^{R} \pi_i^{(m+1)} E_{\tilde{P}^{(m)}}\left[\log P(q_j, K|\lambda^{(m+1)})\right]$$

The induction of the above equation is not difficult but lengthy. So, we do not describe it here. The entry $\frac{1}{\beta} H\left(\tilde{P}^{(m+1)}\right)$ is for regularization. When $\beta \to 0$, $H\left(\tilde{P}^{(m+1)}\right)$ becomes more important in the objective function and makes the mixture weights more uniform; when $\beta \to +\infty$, the mixture weights becomes more unbalanced. In our experiments, we set β=0.1. DAEM was also used in other studies, for example, Smith and Eisner [14] used it for grammar induction tasks.

3.2 Regularized Log-Linear Model

When setting β to a value less than 1.0, the λ estimated by DAEM tends to be more uniform, and it should be less unbalanced than the one estimated by the standard EM. However, we find the objective function described by equation (5) does not match our goal exactly. The optimal mixture weight should imply the importance of the corresponding resource. In other words, it represents the ability of the resource to differentiate relevant documents from irrelevant documents. However, the objective function of DAEM and the models used in [4, 8] is simply set to be the log-likelihood of the query. Therefore, the resource in the pseudo-relevant documents which has larger probability to generate the query terms will be emphasized. However, the same resource may also lead the irrelevant documents to have a large generation probability of the query terms. For example, we had this problem when using Cao et al.'s [8] three-component mixture model (as equation 3) in the experiments. We found that $P_u(w|d)$ and $P_{wn}(w|d)$ usually have larger generation probabilities than $P_{coc}(w|d)$. As a consequence, the mixture weights of the former two resources are always larger than the latter. But it may not be desirable. If P_{coc} has low probability for the relevant documents, and it has even lower probability for the irrelevant documents, then it does make relevant documents different from the irrelevant ones and should be favored by the mixture model. Actually, this problem is caused by the gap between log-likelihood of the query and the MAP, i.e., higher log-likelihood does not necessarily lead to better MAP. In this section, we define a new objective function, which measure the difference made between relevant and irrelevant documents, that is:

$$\mathcal{F}(\lambda) = \alpha \sum_{j=1}^{n} log \sum_{d\in U} \pi_d^U \sum_k \lambda_k P_k\left(q_j\middle|d\right) - \sum_{j=1}^{n} log \sum_{d\in R} \sum_k \pi_d^R \lambda_k P_k\left(q_j\middle|d\right) \qquad (8)$$

where α is a scale factor, which is set to be 1.8 in our experiments, R is PRD and U is the set of pseudo-irrelevant documents (PIRD), π_d^U and π_d^R are the probability of the documents to be chosen in R and U respectively. For simplicity, we assume the documents in both R and U have equal probabilities to be chosen, i.e., $1/|R|$ and $1/|U|$ respectively. Then the first term on the right side of the above equation is the log-likelihood of the query with respect to PIRD, while the second term is the log-likelihood of the query with respect to PRD. We estimate λ_k by minimizing $\mathcal{F}(\lambda)$. However, it is a constrained optimization problem as follows:

$$\lambda^* = \min_\lambda \mathcal{F}(\lambda)$$
$$\text{Subject to:}$$
$$\sum_k \lambda_k = 1$$
$$\lambda_k > 0$$

We convert the constrained optimization problem to an unconstrained one with the following transformation:

$$\lambda_k = \frac{\exp(\gamma_k)}{\sum_k \exp(\gamma_k)}$$

Then equation (8) becomes a log-linear model with only one fixed value feature. To avoid unbalanced mixture weights, we use L_2-regularization [12]. We call this Regularized Log-Linear Model (RLM). Putting them together, we get the following formula:

$$\mathcal{F}(\lambda) = \mathcal{L}(\gamma) = \alpha \sum_{j=1}^{n} log \sum_{d\in R} \sum_k \lambda_k P_k\left(q_j\middle|d\right) - \sum_{j=1}^{n} log \sum_{d\in U} \sum_k \lambda_k P_k\left(q_j\middle|d\right) + \delta \sum_k \gamma_k^2 + \text{const} \qquad (9)$$

where δ is the regularization factor, and const is a function independent of γ. This regularization method is equivalent to adopting a Gaussisn prior of the mixture weights. To be specific, it is a zero-mean isotopic Gaussian governed by a single precision parameter δ [18], which is set to be 0.05 empirically. In section 5.3, we will investigate the impact of the value of δ empirically. Then the estimation of the λ is equivalent to estimate γ, which is formulized as:

$$\gamma^* = \min_\gamma \mathcal{L}(\gamma)$$

We used Quasi-Newton method [14] to search the optimal γ.

In this model, we try to maximize the difference between the log-likelihood of pseudo-relevant and irrelevant documents. It is similar to the maximum-margin principle [15]. We used two document sets, i.e., PRD and PIRD. The former is set to be top 10 documents in the initial retrieval. The question now is how to

Model	ABS			DIR			JM		
	MAP	Imp. Over UM	Imp. Over OptM	MAP	Imp. Over UM	Imp. Over OptM	MAP	Imp. Over UM	Imp. Over OptM
UM	0.1771	------	------	0.1913	------	------	0.1726	------	------
OptM	0.1918	8.30%**	------	0.2116	10.61%**	------	0.1914	10.89%**	------
EM	0.1837	3.72%	-4.22%	0.1938	1.31%	-8.41%	0.1831	6.08%*	-4.33%
DAEM	0.1916	8.07%*	-0.1%	0.2055	7.32%*	-2.88%	0.1900	10.08%**	-0.73%
RLM	0.1969	11.18%**	2.65%	0.2124	11.03%**	0.38%	0.1946	12.75%**	1.67%
*means p-value < 0.05; ** means the p-value< 0.01;									

Table 1: Performance of Mixture Models

create the latter. One intuitive approach is to select some documents which are in the bottom of the rank list of the initial retrieval. However, this method does not produce satisfactory empirical results. We then select the documents which are closer to PRD in the rank list, namely the documents ranked from 151 to 200 in the list. This idea is similar to the active learning [16] and Boosting algorithm [17], which favor the instances close to the decision boundary and difficult to be classified. We will conduct a series of experiments in section 5 to investigate the impact of the selection of PIRD to the retrieval effectiveness in details.

4. Estimating the Components of the Mixture Model

Since we focus on the estimation of the mixture weights in this paper, we do not investigate the problem of which resources to be used. In this paper, we adopted Cao et al.'s three-component mixture model [8]. This model addresses the "synonym" problem: a document about "Bush" may be relevant to a query about "president" even it does not contain the term "president". To achieve this goal, [8] made use of two term relations, namely co-occurrence and the lexical relation derived from the WordNet. The two relations are complementary because the former is derived from data automatically and has high coverage but low accuracy, while the latter is defined manually and has low coverage but high accuracy. Plus the traditional unigram model, [8] estimated the document model with a three-component mixture model as described in equation (3). In both $P_{coc}(w|d)$and $P_{wn}(w|d)$, terms are not assumed to be independent, we call them dependency models. Each dependency model generates a query term with a two-step process described as follows

$$P_R(w|d) = \sum_{w' \in d} P_R(w|w') P_{ml}(w'|d)$$

where $P_R(w|d)$ is the dependency model, R can be either *coc* or *wn*, $P_{ml}(w'|d)$ is the probability of *w'* within *d* estimated by MLE, and $P_R(w|w')$ models the relationship between the two terms. The estimation of $P_{coc}(w|w')$ is based on the co-occurrence of terms within a pre-defined window (15 words). The estimation of $P_{wn}(w|w')$ is similar to that of $P_{coc}(w|w')$ except that it requires the co-occurring two terms are also related in the WordNet. Interested readers can refer to [8] for more details.

5. Experiments

5.1 Experiment Setting

We evaluated the two regularized mixture models described in section 3 using one TREC collection AP90-92, which contains 242,918 document and amounts to 729MB. All documents have been processed in a standard manner: terms were stemmed using the Porter stemmer and stopwords were removed. The queries are TREC 51-100. We only used the title field of the queries.

The WordNet we use for experiments is WordNet2.0. For each word in the vocabulary of dataset, we extract its synonym, hypernym and hyponym from WordNet and build a relation pool for it. The processing is done offline. When counting the co-occurrences of terms in $P_{wn}(w|w')$, the relation pool is used to determine whether the terms are related.

One baseline in the experiments is the traditional LM approaches for IR, i.e., the unigram model. We smoothed the unigram model with three methods, and they were compared with the corresponding mixture models respectively. In the LM approach for IR, there are several free parameters to be estimated, for instance, the smoothing parameters. In our experiments, we empirically set the parameters for unigram model by trial and error, and the parameter of the mixture models are blindly set as the same as the unigram models. So our mixture models are not tuned to its best. Even though, mixture models outperform the baseline substantially.

The effectiveness of IR is mainly measured by MAP. For each query, we retrieve top 1000 documents. We also calculated the t-test for statistical significance and conducted query-by-query analysis.

5.2 Do the Regularized Mixture Models Work?

Table 1 shows the results to compare the two regularized mixture models with two baselines. One baseline is the unigram model, i.e., UM. It is smoothed with three methods. ABS (absolute discounting), DIR (Dirichlet) and JM (Jelinek-Mercer). Another baseline is the OptM, which are assigned with the optimal mixture weights. These optimal weights are obtained by exhaustive search in the parameter space by maximizing the MAP of the 50 queries. Therefore, all the queries share the same mixture weights.

In the following three models, namely EM, DAEM and RLM, each query is assigned a group of specific mixture weights. EM denotes the methods that estimates the mixture weights with term-wise EM algorithm, i.e., the special case of DAEM when $\beta = 1$. We see that in this case, the EM model outperforms UM models, which shows that the mixture model outperforms the traditional LM approaches for IR.

From table 1, we also observe that the two regularized mixture models (DAEM and RLM) perform well. Both models outperform UM and EM significantly. This shows that the regularized mixture model is better than the unregularized one. The performance of DAEM is similar to that of OptM. In particular, RLM outperforms OptM for all the three configurations. This indicates that it is a good strategy to allow each query to have its own mixture weights,

and this improves the retrieval effectiveness. It also shows that if the mixture weights are set in an appropriate way, the performance of the mixture models is higher than the UM model. Since the EM algorithm (for EM and DAEM) and Quasi-Newton method (for RLM) converges very fast, it takes only several seconds to process one query in our experiments.

5.3 The Impact of Regularization Factor

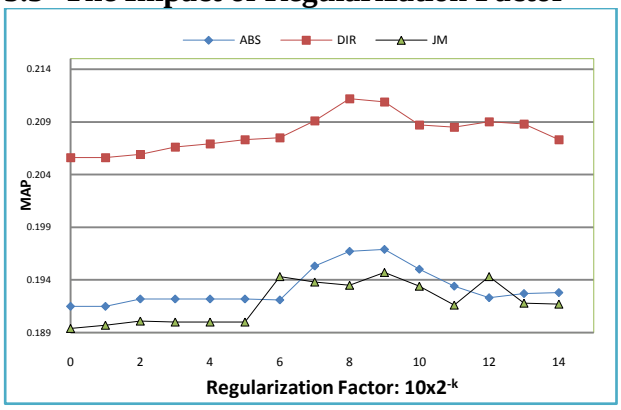

Figure 1: The Impact of the Regularization Factor in RLM

In this section, we investigate the impact of the regularization factor, which is denoted as δ in equation (9). We mentioned in section 3.2 that δ is the single precision governing the zero-mean isotropic Gaussian. Therefore, if δ is very large, the Gaussian is peaked around the mean so that the mixture weights tend to be uniform; otherwise, the Gaussian is flat and the mixture weights tends to be unbalanced.

In this section, we set $\delta = 10 \times 2^{-k}$, and k goes through 0 to 14. We compared the MAP at the 15 values for all the three configurations. Figure 1 shows the results. From this figure, we find that the optimal δ is around 10×2^{-9}. However, the MAP does not change much when δ is between 10×2^{-10} and 10×2^{-7}

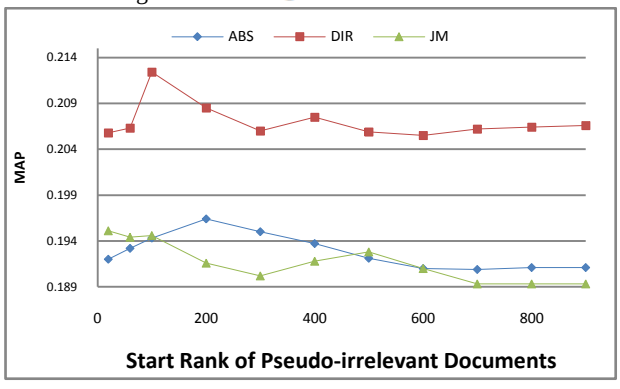

Figure 2: The Impact of the Selection of Pseudo-irrelevant Documents in RLM

for all the three configurations. Figure 1 shows that δ has an important impact on retrieval effectiveness, but it can be set at a reasonable range for different collections.

5.4 The Impact of Selection of Pseudo-irrelevant Documents

In the RLM model, we used two set of documents, namely PRD and PIRD. In the experiments, PRD is set to be top 10 documents in the initial retrieval. We also conducted a series of experiments to investigate the impact of the number PIRD and found that there was only a very small change on the MAP. However, the selection of PIRD is critical for the retrieval effectiveness. When selecting the documents, we first determine the rank of the first document in PIRD and consider the following consecutive 50 documents as irrelevant documents. In order to test the impact of the rank of the pseudo irrelevant documents, we chose 11 different ranks, i.e., 20, 40, 100, 200, … till 900, and compared the MAP. Figure 2 shows the results. It is interesting that the optimal value is less than 200, which shows that the optimal PIRD is close to the PRD. If we view the log-likelihood of the query as the discriminant function to classify relevant/irrelevant document (actually we do so), then the experimental results implies that selecting the documents close to the decision boundary is better than selecting documents at the bottom of the initial rank list to train the discriminant function. In fact, this conclusion is consistent with the active learning theory [17] and the principle of boosting algorithm [18], which prefer to use ambiguous instances (close to the decision boundary) to train the classifier.

5.5 The Impact of Scale Factor

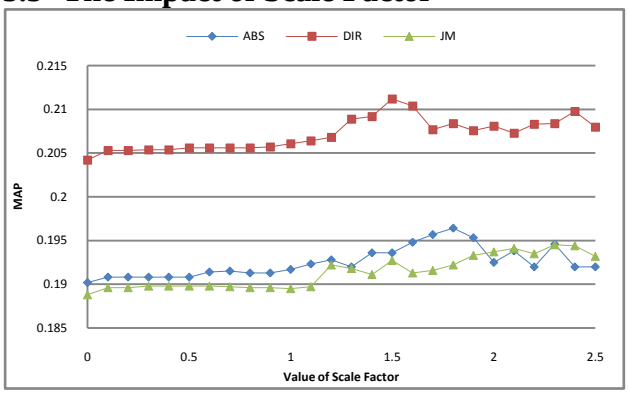

Figure 3: The Impact of the Scale Factor in RLM

The scale factor is α in equation (9). When it is set to zero, the PIRD does not affect RLM, and the mixture weights are simply estimated by maximizing the regularized log-likelihood of the query with the given PRD. When α becomes larger, the effect of PIRD is more emphasized. In this section, we conduct experiments to study the impact of the value of the scale factor. Figure 3 plots the MAP values for α varying from 0 to 2.5. We observe that the optimal value is larger than 1.5, which means that it is important and useful to incorporate PIRD. From the figure, we also observe that when the scale factor is set to a value between 1.5 and 2.0, the MAP seems to be the highest. Therefore, it is not difficult to assign a reasonable value to the scale factor. In our experiments reported in the previous tables, we set it to 1.8.

6. Conclusion and Future Work

The mixture model combining multiple resources to estimate the document model has been proven to be effective for IR. One important issue in the mixture model is how to set an appropriate weight to each component. In previous studies, the weights were usually set manually, which requires the user to have good prior knowledge about the resources. Even the weights were set automatically in other studies; there is still a problem with the unbalanced weights which over stress one of the components. In this paper, we proposed two regularized models to produce more reasonable estimation of the mixture weights: one is a variant of EM algorithm, i.e., DAEM, and another is the L_{-2} regularized log-

linear model. We conducted experiments to evaluate the two models.

In both methods, mixture weights are estimated for different terms. Therefore, it is allowed that different query terms to be generated from different component models with different weights. Experimental results show that (1) The two regularized models are effective to estimate the mixture weights. DAEM performs similarly to the OptM while RLM outperforms OptM. This shows that term-wise mixture weight estimation is better than the optimal query-wise estimation. (2) The regularized models outperform the unregularized model: Both DAEM and RLM outperform EM significantly. (3) It is better to try to maximize the difference between pseudo relevant documents and pseudo irrelevant documents, than to simply maximize the pseudo relevant documents alone. Such an estimation allows us to know which component is the most discriminant for a query term, and to assign a mixture weight accordingly. (4) When pseudo-irrelevant documents are used, it is better to use a document set which is close to the pseudo-relevant documents.

In this paper, we only use one feature and the value of the feature is also fixed. One interesting future work is to use more features to build the log-linear model. In next step, we will incorporate other features related the specific term and make the mixture weight depend on terms. Another research avenue is to assign different weights to the document models in RLM. Since we have found that emphasizing the documents close to the decision boundary improves the effectiveness, it is also reasonable to assign different weights to documents to vary their importance in the training.

7. REFERENCES

[1] Berger, A. and Lafferty, J. (1999) Information retrieval as statistical translation. *In Proceedings of the 1999 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 222-229.

[2] Miller, D. Leek, T. and Schwartz, R.M. (1999). A hidden Markov model information retrieval system. *In Proceedings of the 1999 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 214-222

[3] Ponte, J. and Croft, W.B. (1998). A language modeling approach to information retrieval. *In Proceedings of the 1998 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275-281.

[4] Zhai, CX, and Lafferty, J. (2002). Two-stage language models for information retrieval. *In Proceedings of the 2002 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 49-56

[5] Tao, T., Wang, X., Mei, Q. and Zhai, C.X. (2006). Language Model Information Retrieval with Document Expansion. In the Proceedings of HLT/NAACL2006.

[6] Zhai, CX, and Lafferty, J. (2001). A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *In Proceedings of the 2001 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334-342

[7] Gao, J. Qi, H., Xia, X. and Nie, J.Y. (2005). Linear Discriminant Model for Information Retrieval. In the Proceedings of SIGIR2005.

[8] Cao, G., Nie, J.Y. and Bai, J. (2005). Integrating Word Relationships into Language Models. In the Proceedings of SIGIR2005.

[9] Dempster, A.P, Laird, N. M., and Rubin, D. B. (1977) Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39:1-38

[10] Collin, Michael. (2002). Discriminative training methods for Hidden Markov Models: theory and experiments with the perceptron algorithm. In the Proceedings of EMNLP2002, pp1-8.

[11] Ueda, N., and Nakano, R. (1998). Deterministic annealing EM algorithm. Neural Networks 11 (1998)

[12] Chen, S.F. and Rosenfeld, R .(2000). A gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108, Carnegie Mellon University.

[13] Neal, R. and Hinton, G. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In M.I. Joardan, editor, Learning in Graphical Models. Kluwer.

[14] Smith, N., and Eisner, J. (2006). Annealing Techniques for Unsupervised Statistical Language Learning. In the Proceedings of ACL2006.

[15] Nocedal, J., and Wright, S.J. (2006). Numerical Optimization. Springer, New York

[16] Abe, S. (2005). Support Vector Machines for Pattern Classification, Springer.

[17] Tong, S., and Koller, D. (2001). Support Vector Machine Active Learning with Applications to Text Classification. Journal of Machine Learning Research (2001), pp.45-66

[18] Freund, Y. and Schapire, R. (1999). A Short Introduction to Boosting. Journal of Japanese Society of Artificial Intelligence, 14(5):771-780.

[19] Bishop, Christopher (2006). Pattern Recognition and Machine Learning. Springer.

[20] Nallapati, R. (2004). Discriminative models for information retrieval. In the Proceedings of SIGIR2004, pp64-71.

[21] Miller, G. (1990). Special Issue, WordNet: An on-line lexical database. International Journal of Lexicography, 3(4), 1990.

[22] Liu, X. and Croft, B. (2004). Cluster-Based Retrieval Using Language Models. In SIGIR2004: Proceedings of the 27th annual international conference on Research and development in information retrieval, pp.186-193.

[23] Wei, X. and Croft, B. (2006). LDA-Based Document Models for Ad-hoc Retrieval. In the Proceedings of SIGIR2006, pp178-185.

[24] Kurland, O. and Lee, L. (2004). Corpus structure, language models, and ad hoc information retrieval. In SIGIR2004: Proceedings of the 27th annual international conference on Research and development in information retrieval, pp 194-2001. ACMPress