Inf Retrieval (2006) 9: 5–31
DOI 10.1007/s10791-005-5719-z

Two-stage statistical language models for text database selection

Hui Yang · Minjie Zhang

Received: May 11, 2004 / Revised: October 15, 2004 / Accepted: October 18, 2004

Abstract As the number and diversity of distributed Web databases on the Internet exponentially increase, it is difficult for user to know which databases are appropriate to search. Given database language models that describe the content of each database, database selection services can provide assistance in locating databases relevant to the information needs of users. In this paper, we propose a database selection approach based on statistical language modeling. The basic idea behind the approach is that, for databases that are categorized into a topic hierarchy, individual language models are estimated at different search stages, and then the databases are ranked by the similarity to the query according to the estimated language model. Two-stage smoothed language models are presented to circumvent inaccuracy due to word sparseness. Experimental results demonstrate that such a language modeling approach is competitive with current state-of-the-art database selection approaches.

Keywords Database language model · Text database selection · Distributed information retrieval · Hierarchical topics · Statistical language modeling · Query expansion

1. Introduction

The rapid proliferation of online text databases on the Internet has made it difficult for a user to determine which databases to search for desired information. To reduce network traffic as well as the cost of searching irrelevant databases, the idea of sending the user query only to those potentially useful databases has become more and more attractive to both users and information science researchers.

Database classification, the technique to partition multiple, distributed Web databases into a structured hierarchy of topics, provides a useful and efficient way to organize and manage a vast number of Web databases. At the same time, it is also helpful in simplifying the task of database selection. Recently, several researchers have investigated the use of topic hierarchies

H. Yang · M. Zhang
School of Information Technology and Computer Science, University of Wollongong, Wollongong, 2500, Australia
e-mail: {hy92, minjie}@uow.edu.au

for database classification and have received encouraging achievements. Gauch et al. (1996) manually construct query probes to facilitate the classification of text databases. Ipeirotis, Gravano et al. (2001, 2003) present a method for automating the database classification based on the number of matches that each query probe generates from the databases. The formation of queries comes from document classifiers. More recently, Ipeirotis and Gravano (2004) introduce the notion of "shrinkage" which exploits database classification information by use of a topic hierarchy to compensate for incomplete content summaries. In Meng et al. (2002) and Yu et al. (1999b), a concept hierarchy is constructed for text database categorization. Each concept description is treated as a query that is submitted to the database. The documents retrieved from the database are used to calculate the similarity between the concept and the database. In our previous work (Yang and Zhang 2004), we proposed a probability framework for the hierarchical classification of distributed databases with a set of Naive Baysian classifiers.

A number of different algorithms for database selection have been proposed during the last decade, including GLOSS (Gravano et al. 1999), CORI (Callan 2000), RDD (Voorhees et al. 1995), CVV (Yuwono and Lee 1997), query probing (Hawking and Thistlewaite 1999, Meng et al. 1998), lexicon inspection (Zobel 1997), a probability model (Baumgarten 1999), and a KL-divergence algorithm (Xu and Croft 1999). Moreover, extensive comparative studies among these algorithms have also been performed (French et al. 1999, Craswell et al. 2000, D'Souza et al. 2000, Powell and French 2003). In general, database selection algorithms do not directly have access to each database. Instead, they mainly interact with a *database language model* which is constructed based on database statistical information, such as subject domains (topics) of database content, a list of index terms that occur in the database and their frequency occurrence, and the number of documents containing each index term. The statistical information indicates the approximate content of databases. With database language models, selection algorithms compute a ranking score for each database which characterizes its relative usefulness to a query. Thus database language models are perhaps the most important component of database selection.

The use of statistical language models for information retrieval has been studied extensively for decades and has generated encouraging results (David et al. 1999, Hiermstra 2001, Song and Croft 2002, Lafferty and Zhai 2002). However, most existing language models are only involved in document indexing and document retrieval. The work in statistical language models for database selection seems relatively new. However, there are several works that have looked at development of the database language models (Xu and Croft 1999, Si and Callan 2003, Si et al. 2002). Xu and Croft (1999) proposed a KL-divergence algorithm that used the Kullback-Leibler (KL) divergence between the word frequency distribution of the query and the database to measure the relevance of the database to the query. This algorithm is represented in a language modeling framework consisting of a database language model and a query language model, respectively. Si et al. (2002, 2003) proposed a language model (LM) algorithm that was the extension of the KL-based database selection method. The LM algorithm incorporated the database and query language models into an integrated language model for database selection. However, the work of these two groups did not take into account the relationships between topic classes related to the content of the database during database selection.

In this paper, we present a novel way of looking at the problem of database selection from the viewpoint of statistical language modeling. The statistical language model explored in this paper is, in practice, a two-stage database language model which is the combination of a class-based language model and a term-based language model. This work is an extension of our previous research (Yang and Zhang 2004) which introduced a clustering method for

hierarchically categorizing multiple, distributed Web databases. For text databases that have been categorized into a hierarchical structure of topics, the task of database selection is conceptually divided into two distinct steps:

(1) First, with a class-based language model, the search focuses on databases in confined domains, e.g., a certain specific subject area in which a user is interested.
(2) Second, the selection algorithm computes the likelihood of separate databases to the query using a term-based language model, and further selects the best databases for the query.

Obviously, the two-stage language model approach explicitly captures the different influences of the category-specific search stage and the term-specific search stage on the optimal settings of selection parameters. Our experimental results reported here have demonstrated that the exact performance of this two-stage language model is significantly better than other traditional selection methods, such as the state-of-the-art CORI method. Our study has three important contributions:

- First, we acquire database models independently according to different search stages, and intentionally keep these models separate.
- Second, we consider general ways of combining different database models together by introducing suitable parameters. The synthetic model can be further individualized and optimized for database selection.
- Third, we propose a query expansion method to alleviate the problem of query ambiguity using a query translation model.

It is believed that using such simple and effective language models as a solid baseline method opens up the door to the improvement in database selection performance.

The paper is organized as follows: in Section 2 we discuss the language modeling approach to IR, and briefly review previous work in this area; Section 3 lays out the basic theory of the two-stage database language model and develops the formulas for its simple realization; experimental methodology and a series of experiments to evaluate our language model are presented in Section 4 and 5; and finally, conclusions and contributions of this work are summarized in Section 6.

2. The language models in information retrieval

An information retrieval (IR) system consists of three basic components: the representation of documents, the formulation of a user query, and the construction of a ranking function which compares the likelihood between the document and the user's information need. Language models explicitly define how documents and queries should be analyzed and they reasonably model the way that the ranking is produced. So research on language models has received by far the most attention in the information retrieval community. To date different categories of language models have been developed. Among them, three commonly used traditional retrieval models include the Boolean model (Salton and McGill 1983), the vector-space model (Salton and McGill 1983) and the classic probabilistic model (Robertson and Jones 1976, Van Rijsbergen 1992).

The Boolean model is a simple retrieval model based on set theory and Boolean algebra. The relevance of documents is specified as a Boolean expression which has a strict logic implication (Salton and McGill 1983). Due to the exact matching strategy which is unable to recognize uncertain matches, the Boolean model is considered as the weakest classic method.

The vector-space model (Salton and McGill 1983) assumes that a query and a document are both represented by term vectors. The similarity between a query $Q = \{q_1, q_2, \ldots, q_n\}$ and a document $D = \{d_1, d_2, \ldots, d_n\}$ can be measured by $\sum_i q_i \cdot d_i$, which is the dot product of the two vectors. The similarity values are assumed to reflect the degree of relevance of individual document with respect to a user query. Similarity measures and term-weighting methods used in the vector-space model strongly depend on the type of query and the characteristics of the document collection. The optimal settings of retrieval parameters have been a great challenge.

The classic probabilistic modeling approaches (Robertson 1977, Robertson and Jones 1976) can be characterized as methods for estimating the conditional probability of the relevance of a document to a user query. In essence, the probabilistic model is an adaptive model based on the Bayesian decision theory. In the probabilistic framework, the term weights on both documents and queries are represented as probabilities. The relevance of a document to a query is simply the product of the individual term probabilities $P(Q \mid D) = \prod_i P(q_i \mid D)$. The most obvious problem with the probabilistic model is that it is possible to assign a zero probability to a document that is missing one or more of the query terms. In addition, the imprecise definition of the concept of relevance, and the lack of a large amount of available relevance training data, make reliable estimation of these probabilities a difficult task.

In recent years, there have been controversies about which model, the vector-space model or the classic probabilistic model, is better. In practice, each of the two models has its own advantages and disadvantages in different situations. Nevertheless, they both play an important role in the development of information retrieval.

In order to circumvent individual problems of these three traditional categories of retrieval models mentioned above, several researchers attempt to develop some hybrid models that combine the strengths of these three models within a unified framework. For example, Turtle and Croft (1990) introduced an inference network model in which the indexing and retrieval models are integrated by making inferences of concepts from features by means of a Bayesian network. The inference network can be used to simulate the Boolean, vector-space, and classic probabilistic models. Other studies on mixture retrieval models can be found in Wong et al. (1987) and Van Rijsbergen (1989).

Due to the relative simplicity and effectiveness of statistical methods that have been applied successfully in a wide variety of speech and language recognition areas, statistical language models have recently attracted significant attention as a new alternative to traditional retrieval models.

The statistical language modeling approach was first proposed in Ponte and Croft (1998). Ponte et al. presented a simple unigram document language model, which estimated the probability of producing the query for each document, and then ranked the document according to that probability. In the use of the language model by Miller et al. (1999), multiple word generation mechanisms are incorporated within the same model using Hidden Markov Models (HMM) theory. The retrieval parameters are trained and optimized through a learning procedure. In a linguistically motivated model proposed by Hiemstra (1998), documents and queries are defined by an ordered sequence of single terms rather than unordered collections of terms or phrases. The probability estimation of $df \times icf$ term weighting is based on statistical natural language processing.

A more sophisticated systematic framework for this new family of retrieval methods has been developed in a recent study by Lafferty and Zhai (2001). A probabilistic retrieval model is motivated to unify document models and query models in a framework based on Bayesian decision theory. Kullback-Leibler divergence is introduced as a loss function for

risk minimization. Markov chains that are defined based on the inverted indices of a document collection, are used for estimating expanded query models.

There are other statistical language modeling approaches to information retrieval including title language models (Jin et al. 2002), and statistical translation (Berger and Lafferty 1999). Although the details differ in these approaches, the language models are similar in principle. The statistical language models are often approximated by a N-gram model for estimating the probability of query terms.

The simplicity and robustness of the statistical language modeling approach make it a new important research direction for text retrieval.

3. Two-stage database language models for database selection

One important advantage of the two-stage language models over traditional database language models is their capability of modeling both category-specific selection and term-specific selection directly through statistical language models. Instead of combining two distinct selection steps together in an ad hoc manner, we intentionally keep them separate and consider general ways to represent them.

The basic idea of the statistical language modeling approach to information retrieval is to define a N-gram language model for each document in a collection (Ponte and Croft, 1998). For each document, the language model computes the conditional probabilities of generating a sequence of query terms, and then multiplies these probabilities in order to estimate the likelihood value of the document with respect to the query.

Our work is originally motivated by the fact that a textual database can be regarded as a vast, *virtual* document and be represented by a list of index terms since this textual database consists of a collection of text documents. Therefore, a document language model can be easily borrowed to model the likelihood measure of the database to a user query.

In this section, we extend recent advance in the use of statistical language models to information retrieval and explore two-stage database language models based on statistical language modeling for database selection. A brief overview of this language model and its application to database selection is described as follows.

As we know, database selection is difficult partly because database selection algorithms do not interact directly with the full contents of a database. Instead, they utilize a *resource description* (Callan and Connell 2001) that primarily contains the statistical information such as the words that appear in the database and their document frequency, plus simple additional information such as the size of the database. Therefore, the first problem in developing the database language model is the acquisition of resource descriptions about the database contents. Resource descriptions usually come from cooperative resource providers. When Web databases tend to be "uncooperative", the resource descriptions for the "uncooperative" Web databases have to be obtained by an alternative approach named *query-based sampling*. The basic idea behind the query-based sampling methods is that a set of simple queries are submitted to each *searchable* database, and a small document sample (e.g., 300 documents) is extracted via querying. The document sample is used to construct an *approximate* resource description for this database. Different query-based sampling strategies had been developed, such as the Query-based Sampling (QBS) method (Callan and Connell 2001) and the Focused-probing Sampling (FPS) method (Ipeirotis and Gravano 2002). Moreover, Ipeirotis and Gravano (2004) proposed a shrinkage method to create category content summaries of databases using a topic hierarchy.

However, as described earlier, the emphasis of our work is on building effective database selection models to choose the useful databases for search. Therefore, the discussion about the acquisition of resource descriptions has been beyond the scope of this paper. For more detail on the acquisition of resource descriptions, please refer to Gravano et al. (1997) and Hawking and Thistlewaite (1999) for the "cooperative" databases, and Callan and Connell (2001), Ipeirotis and Gravano (2002, 2004), Xu and Croft (1999) for the "uncooperative" databases. Here, we only focus on the construction of the database selection model, assuming that the resource descriptions for the databases have been known.

3.1. Hierarchical structure of topics for text databases

Hierarchical structure of topics have been long extensively studied for text classification. More recently, several large-scale commerce Web search services such as Yahoo and Infoseek have also adopted such hierarchies to manage the Wide World Web. By browsing these categories in the topic hierarchy, the users can conveniently find the appropriate topic that they are interested in. A simple hierarchical structure of topics is shown as Figure 1.

We can give some formal definitions of a hierarchical structure of topics as follows:

Definition 1. A structured hierarchy of topics is a rooted directed tree which contains a number of topics (classes), namely, $c_1, c_2, \ldots, c_K$, organized into multiple levels and each node corresponds to a topic.

In general, in such a topic hierarchy, topics are ordered from general topics at the higher level to specific topics at the lower level. There exist parent-child relationships between two adjacent layers. Each parent class has a set of child classes, and these child classes together cover different aspects of the parent class.

Definition 2. In each parent-child pair, the weight of the connection between parent class c_k^p and child class c_k is denoted by $w_{c_k^p c_k}$, which represents the degree of their association. For a given parent class, $c_k^p = \{c_1, \ldots, c_k, \ldots, c_T\}$, the weight $w_{c_k^p c_k}$ can be calculated as

$$w_{c_k^p c_k} = \frac{P(c_k^p)}{P(c_k)} = \frac{1 + \sum_{i=1}^{i=T} P(c_i)}{N + T} \cdot \frac{1}{P(c_k)} \tag{1}$$

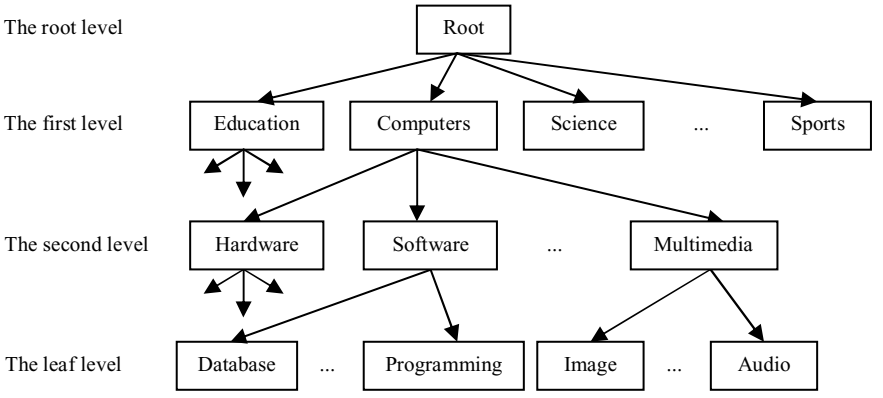

Fig. 1 A small fraction of the topic hierarchical structure

where N indicates the number of classes in the level in which the parent class c_k^p lies, and T is the number of child classes in the parent class c_k^p. The prior probability for each child class c_i, $P(c_i)$, is obtained from the training data set used for the construction of the topic hierarchy.

Definition 3. For each topic (class) c_k in the topic hierarchy, it is represented by a feature space F^{c_k}, which is denoted as $F^{c_k} = \{f_1, f_2, \ldots, f_M\}$, where f_i $(1 \leq i \leq M)$ is a distinct feature vector in the feature space F^{c_k}.

The features are the words that are strongly associated with one specific category (class). The features have enough discrimination power to distinguish this class from a set of classes in the hierarchical classification scheme. Due to computational cost which is exponential in the number of features, the feature space F^{c_k} is usually minimized into a much small set of features with minimal loss in accuracy (the issue on how to optimally select features for the classes is out of the scope of this paper). For a given class c_k,the probabilistic distribution for each feature f_i to the class c_k, $P(f_i \mid c_k)$ is known during the construction of the topic hierarchy.

Definition 4. Cluster c^p is a parent class that consists of a number of child classes, $c^p = \{c_1, c_2, \ldots, c_T\}$, where T is the number of the child classes. F^{c^p} is a feature space of the cluster c^p, which is described as $F^{c^p} = \{f_1, f_2, \ldots, f_S\}$, where $F^{c^p}$is the feature space set of all the child classes, namely, $F^{c^p} = F^{c_1} \cup F^{c_2} \cup \cdots \cup F^{c_T}$, and f_i $(1 \leq i \leq S)$ is a distinct feature vector in the feature space F^{c^p}.

Definition 5. With a hierarchical classification scheme with categories, $c_1, c_2, \ldots, c_K$, a database S can be classified into one or more classes, which is denoted by a class set C^S, $C^S = \{c_1^S, c_2^S, \ldots, c_K^S\}$. Each class in the class set C^S is a 2-dimension vector $\{c_i^S, t_i^S\}$ $(1 \leq i \leq K)$, where t_i^S is the degree of relevance of the database S to a certain class c_i. t_i^S can be represented by the posterior probability $P(c_i \mid S)$ of class c_i for the database S, whose value is normalized into the range from 0 to 1.

It is possible for a large-scale general-purpose database to be assigned to multiple topics (classes), since it usually contains the documents of different subject areas. The details of how to categorize databases in a hierarchical classification scheme can be found in our previous work (Yang and Zhang 2004).

It is easily noted that if a database is assigned to a topic (a leaf node or a node of lower level in the hierarchy tree), it is also assigned to its parent or grandparent classes located on the path from this node to the root node. For example, if class "*database*" is chosen by the classification scheme as an appropriate topic for database S, then classes "*computers*" and "*software*" must be in the class set C^S of database S (see Figure 1).

The topic hierarchy and its associated Web databases could be used in a database selection environment as follows. Once a user submits a query, the selection system selects the databases to search in two stages: in the first stage, a preliminary selection is performed through the process. More specially, the system identifies one or more topics that best match/matches the user's information need, and then the databases associated with these topics (classes) firstly are chosen. In the second stage, a term-based database selection algorithm is used to further select the more appropriate databases from those chosen databases based on the degree of relevance to the query.

3.2. The statistical language modeling approach

We now turn to the central issue of how to automatically select the appropriate databases that the user is interested in. Firstly, let us take a brief look at the standard statistical language model.

In automatic speech recognition, language models are capable of assigning a probability $P(W)$ to a word string W ($W = w_1 w_2 \cdots w_n$). The probability of the word string W occurring in English text can be characterized by a set of conditional probabilities $P(w_i \mid w_1^{i-1})$, which can be expressed as a product

$$P(W) = \prod_{i=1}^{n} P\left(w_i \mid w_1^{i-1}\right) \tag{2}$$

where $P(w_i \mid w_1^{i-1})$ is the probability that the word w_i will be occurred when all of the previous words $w_1^{i-1}(w_1^{i-1} = w_1 w_2 \cdots w_{i-1})$ occur.

The task of a statistical language model is to make it tractable to estimate the probabilities $P(w_i \mid w_1^{i-1})$. At present, the dominant technology in language modeling used in IR is the unigram model, which makes a strong assumption that each word occurs independently. Consequently, the probability of a word string becomes the product of the probabilities of the individual words. Therefore, Eq. (1) can be simplified as

$$P(W) = \prod_{i=1}^{n} P(w_i) \tag{3}$$

For a vocabulary of size V, the unigram model is a multinomial distribution, i.e., multinomial distributions over words in V, which has $V - 1$ independent parameters.

The basic idea of the statistical language modeling approach for database selection is to treat each database S as a language sample, and to estimate the conditional probability $P(Q \mid S)$, i.e., the probability of generating a query $Q = \{q_1, q_2, \ldots, q_N\}$ given an observation of a database S:

$$P(Q \mid S) = \prod_{i} P(q_i \mid S) \tag{4}$$

The databases are ranked by the probability $P(Q \mid S)$.

Applying Bayes' rule of probability, the posterior probability of the database S to the query Q can be written as

$$P(S \mid Q) = \frac{P(Q \mid S)P(S)}{P(Q)} \tag{5}$$

The $P(Q)$ term represents the probability that query Q is generated from a document-independent language model. Since $P(Q)$ is constant for all databases given a specific query Q, it does not affect the ranking of the databases and can be ignored for the purpose of ranking databases. Therefore, Eq. (4) can be rewritten as

$$P(S \mid Q) \propto P(Q \mid S)P(S) \tag{6}$$

As to the $P(S)$ term, it is a query-independent term, which measures the quality of the databases according to the user's general preference and information need. In general, $P(S)$ can be explored to capture database characteristics, e.g., the length of a database. In this work, since a user query probably involves in a variety of topics and we cannot predict its content in advance, the prior term $P(S)$ is assumed to be uniform over all databases, and so it does not affect database ranking. A similar assumption has been made in most existing work (Zhai and Lafferty 2001, Ponte and Croft 1998, Song and Croft 1998). Therefore, in practice, only the $P(Q \mid S)$ term has final influence on determining the relevance of the database to the query.

Just as in the use of language model for speech recognition, one of the most obvious practical problems in applying statistical language modeling to IR is the problem of sparse data, that is, it is possible to assign a probability of zero to a database that is missing one or more of the query terms.

The sparseness problem can be avoided by data smoothing methods. In most data smoothing methods, two distributions are used for smoothing. One is high probability for "seen" words that occur in the database, and the other is low probability such as zero probability for "unseen" words. Smoothing methods tend to make distributions more uniform by discounting high probability and adjusting low probability upward. Not only do smoothing methods prevent zero probability, but they have also the potential to improve the accuracy and reliability of the models.

3.3. The smoothed two-stage language models

The generic language model for the two-stage database selection procedure can be refined by a concrete class-based model $P(Q \mid C)$ for generating the specific subject classes that best match the query, and a concrete term-based model $P(Q \mid S)$ for generating the most likely similar databases to the query. Different specifications lead to different selection formulas. Next, we present the generic two-stage models with data smoothing methods.

3.3.1. The smoothed class-based language model.

Assume that we have successfully assigned a feature space F^{c_k} to each class c_k in the topic hierarchy. Given a user query Q, it may be possible to make reasonable predictions of determining the appropriate topic(s) that the user is interested in by using the feature vectors in the feature space. Let $F^{c_k} = \{f_1, f_2, \ldots, f_M\}$ denotes a feature space of class c_k, $Q = \{q_1, q_2, \ldots, q_N\}$ denotes a user query and $F^C = \{f_1, f_2, \ldots, f_{|v|}\}$ denotes the feature vocabulary of all the classes in the topic hierarchy. Obviously, the feature space F^{c_k} of a class c_k only contains a subset of features of the feature vocabulary F^C ($1 \leq M \leq |V|$). For a certain topic class c_k, the likelihood function of class c_k to a query is described as follows,

$$P(Q \mid c_k) = \prod_i P(q_i \mid c_k) = \prod_i P(q_i \mid F^{c_k}) \tag{7}$$

where $P(q_i \mid c_k)$ is the probability of query term q_i in the feature space F^{c_k} of class c_k.

As discussed earlier, due to computation cost, the feature space F^{c_k} is actually a much smaller feature set of fixed size and content. This makes it difficult to distinguish the effects of the missing query terms in the feature space F^{c_k}. To avoid the sparse data problem, we describe a smoothing technique called the Jelinek-Mercer (JM) method (also called linear interpolation) (Jelinek and Mercer 1980) that combines the probability estimates of several

language models within a unified model in order to alleviate the suffering from the data sparseness problem. The probability estimates of this linearly interpolated language model will become more reliable and more accurate.

In order to capture the common and non-discriminating words in a query, we assume that a query is generated by sampling words from a three-component mixture of language models with one component being $P(q \mid c_k)$, and the two others being the parent-class language model $P(q \mid c_k^p)$ and the class-corpus language model $P(q \mid C)$. That is,

$$P_{JM}(Q \mid c_k) = \prod_i (\lambda_1 P(q_i \mid c_k) + \lambda_2 P\left(q_i \mid c_k^p\right) + \lambda_3 P(q_i \mid C)) \tag{8}$$

where c_k^p is the parent class of the class c_k, and C is the class corpus of the topic hierarchy; λ_1, λ_2 and λ_3 are coefficients, which control the influence of each model, $0 \leq \lambda_1, \lambda_2, \lambda_3 \leq 1$ and $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

With reasonably smoothed class models, we can now consider the process of query generation. In this formula, the class-based language model is effectively smoothed in the following two steps:

(1) First, it is smoothed with the correlation information about parent-child pairs. Since the feature space of the parent class c_k^p is the feature space set of all child classes (recall Definition 4), the size of the feature space $F^{c_k^p}$ of the parent class c_k^p is far larger than that of the feature space F^{c_k}. Therefore, such a parent-class model $P(q \mid c_k^p)$ can help us differentiate the contribution of different missing query terms in the feature space F^{c_k}. In consideration of the associative degree between a parent class and child classes, there exists the following relationship between λ_1 and λ_2:

$$\lambda_2 = w_{c_k^p c_k} \lambda_1 \tag{9}$$

where $w_{c_k^p c_k}$ is the weight of the connection between parent class c_k^p and child class c_k (recall Definition 2). Combining this equation with the class model, we can rewrite Eq. (8) as follows,

$$P_{JM}(Q \mid c_k) = \lambda_1 P(Q \mid c_k) + \lambda_1 w_{c_k^p c_k} P\left(Q \mid c_k^p\right) + \left[1 - \left(1 + w_{c_k^p c_k}\right)\lambda_1\right] P(Q \mid C) \tag{10}$$

$$= \prod_i (\lambda_1 P(q_i \mid c_k) + \lambda_1 w_{c_k^p c_k} P\left(q_i \mid c_k^p\right) + \left[1 - \left(1 + w_{c_k^p c_k}\right)\lambda_1\right] P(q_i \mid C)) \tag{11}$$

(2) Second, in order to avoid the probability that some query terms may still be missing in the feature space of the parent class, this class-based language model is further interpolated with a class-corpus model. Note that, in the class-corpus model, the probability of term q_i, $P(q_i \mid C)$, is much smaller than the probabilities in the first two feature spaces F^{c_k} and $F^{c_k^p}$, $P(q_i \mid c_k)$ and $P(q_i \mid c_k^p)$, since the features in the feature space F^C are the sum of the features of all classes in the topic hierarchy.

In the use of the linearly interpolated language model, the coefficient λ_1 plays a crucial role in information loss and the improvement of selection performance. In our work, an automatic procedure, called EM (Expectation Maximization) (Dempster et al. 1977) is executed to learn the optimization of the value of the λ_1 given some training data set. Given some relevant class feature spaces, the probability $P_{JM}(Q \mid c_k)$ will be maximized with the optimum λ_1 value. The EM-algorithm can be defined as follows.

First, the evaluation of the expectation is called the E-step of the algorithm, which is described as

$$m = \sum_{k=1}^{R} \sum_{i=1}^{N} \frac{\lambda_1^{(p)} P(q_i \mid c_k)}{\left(\lambda_1^{(p)} P(q_i \mid c_k) + w_{c_k^p c_k} \lambda_1^{(p)} P(q_i \mid c_k^p) + [1 - (1 + w_{c_k^p c_k}) \lambda_1^{(p)} P(q_i \mid C)\right)} \tag{12}$$

Second, the second step (the M-step) of the EM algorithm is to maximize the expectation that we computed in the first step. That is,

$$\lambda_1^{(p+1)} = \frac{m}{R} \tag{13}$$

These two steps are iterated as necessary to maximize the probability of the query given R relevant classes, $c_1, c_2, \ldots, c_R$. Before the iteration process starts, the coefficient λ_1 is initiated with a default value $\lambda_1^{(0)}$. Each turn is guaranteed to increase the probability of the generation of the query. The algorithm finally converges to a local maximum of the likelihood function.

With the smoothed class-based language model, one or more classes, whose likelihood probabilities $P_{JM}(Q \mid c_k)$ are greater than the threshold τ_e, will be labeled as appropriate topic areas that the user is interested in. Obviously, choosing the subject classes that we explore at the category-specific stage depends on the threshold τ_e of choice. In this paper, τ_e is set empirically by a number of experiments.

Assuming that there are u topics $(c_1, c_2, \ldots, c_u)$ selected by the class-based language model, the likelihood of the database associated with the chosen topics to the query Q can be calculated as

$$P(Q \mid S) = \sum_{k=1}^{u} P_{JM}(Q \mid c_k) P(c_k \mid S) \tag{14}$$

where $P(c_k \mid S)$ is the posterior probability of class c_k for the database S (recall Definition 5). The databases will be ordered by the likelihood probabilities, and only the top K databases in the ranking list will be chosen as preliminary relevant databases to the query.

3.3.2. The smoothed term-based language model.

Once a number of relevant databases are chosen from the category-specific search stage, the next step is to further reduce the search range for efficiency and effectiveness. There are two basic similarity measure techniques for database selection. The first technique is a document-based method. The basic idea underlying this technique is that the degree of relevance of a database to a query is related to relevant documents in the database. Two important parameters, *document quality* and *document quantity*, are considered as selection criteria. *Document quality* refers to the similarity degree of the most likely similar documents in the database, while *document quantity* is the number of relevant documents whose similarity scores are greater than a threshold. This approach is explored in the works of (Yu et al. 2000a, Ipeirotis and Gravano 2001). The second technique is a term-based method, which uses statistical information on term distributions in the database to calculate estimates that attempt to characterize the usefulness of a database. The terms associated with the database provide approximately the content of the database. The ranking score based on term distributions reflects the relative usefulness of the database to a query. Related works are employed in Yang and Zhang (2004), Manber and Bigot (1997).

In our previous work (Yang and Zhang 2004), we learned that the term-based method has similar performance on database classification to that of the document-based method, but as for the computation time and resource consumption, the term-based method is far better than the document-based method. Hence, the language model that we propose here is based on the terms associated with the database.

3.3.2.1. A maximum likelihood estimation (MLE) language model smoothed with dirichlet prior. Given a database as a language sample, a straightforward method to estimate the probabilities of individual query terms is the Maximum Likelihood Estimation (MLE) (Moode and Graybill 1963). The Maximum Likelihood Estimation of the probability of term t_i under the term distribution for database S is

$$P_{MLE}(t_i \mid S) = \frac{C(t_i \mid S)}{\sum_{t_j} C(t_j \mid S)} \tag{15}$$

where $C(t_i \mid S)$ is the frequency occurrence of term t_i in database S, and $\sum_{t_j} C(t_j \mid S)$ is the sum of the occurrence times of all terms in database S. The MLE assigns the probability of the observed data in the database as high as possible and assigns zero probability to the missing data.

Our term-based language model is a unigram language model based on the MLE. Given a user query Q, the probability $P(Q \mid S)$ of a database S to the query can be expressed as

$$P_{MLE}(Q \mid S) = \prod_i P_{MLE}(q_i \mid S) = \prod_i \frac{C(q_i \mid S)}{\sum_{t_j} C(t_j \mid S)} \tag{16}$$

For a database with a large number of documents, there is enough data for $P_{MLE}(q_i \mid S)$ to be robustly estimated. However, the MLE may still assign zero probabilities to query terms that do not appear in the database. Since the probability of the query is computed by multiplying the probability of individual query terms, these zeros are propagate and give a bad estimate for the probability of the query. To compensate for this sparse data problem, one of the smoothing techniques for language models is the *maximum a prior* (MAP) estimator with Dirichlet Prior (DP) (Zaragoza et al. 2003). With a Dirichlet Prior with parameters αm ($m = \{m_1, m_2, \ldots, m_n\}$), the smoothed $P_{MLE}(Q \mid S)$ can be rewritten as

$$P_{DP}(Q \mid S) = \prod_i \frac{C(q_i \mid S) + \alpha m_i}{\sum_{t_j} C(t_j \mid S) + \alpha} \tag{17}$$

where α is a positive scalar; and m_i is the posterior probability of query term q_i, which is denoted as $P(q_i \mid C')$. The probability $P(q_i \mid C')$ can be estimated based on a collection language model $P(t \mid C')$. The collection language model $P(t \mid C')$ is constructed with a large amount of training data $C\prime$, which contains the average probability of the term through a geometric distribution. The model $P(t \mid C')$ is used as a reference model to smooth the probability distribution of query terms. For a term t_i, $P(t_i \mid C')$ is normalized, i.e., $\sum_{t_i} P(t_i \mid C') = 1$.

3.3.2.2. Query expansion with translation models. Users usually submit short queries that have only one or a few words. Such short queries tend to be inexact and ambiguous in identifying the users' information needs. As a result, the selection system has very few clues to work with for predicting the relevance of a database to a query. The accuracy of a likelihood function of the database to the query will suffer from the problem of query ambiguity. The

use of query expansion can alleviate this problem. The basic idea of query expansion is to discover related terms or concepts, along with their relationships with query terms in the user's query. The terms in the expanded query help disambiguate the meanings of query terms in the original query, which in turn make database selection more accurate.

In this paper, we propose a query expansion method which makes use of the relationships of the user query, and the topic classes of interests for the user. The intuition behind our approach is that once relevant topic classes are determined, the query can be mapped to a number of features associated with the identified topic classes. Assume that a set of topic classes, $c_1, c_2, \ldots, c_u$, are relevant to the query Q. For each topic class c_k $(1 \leq k \leq u)$, a query translation model $T(Q \mid F_Q^{c_k})$ is used to map a query Q to a feature set $F_Q^{c_k}(F_Q^{c_k} = \{f_1, f_2, \ldots, f_p\})$(note that the feature set $F_Q^{c_k}$ is only a subset of the feature set F^{c_k} of the class c_k). These features are more specific (or related) terms in the relevant class c_k to the query, which provide sufficient content to clear up the ambiguity. Thus, an expanded query consists of two parts: the first is for the *original query* terms, and the second is for the *expanded query* terms from the relevant topic classes. Using translation models, the term-based language model for the expanded query can be rewritten

$$P_{QE}(Q \mid S) = \underbrace{P(Q \mid S)}_{\text{originalquery}} \underbrace{\prod_{k=1}^{u} P(Q^{c_k} \mid S)}_{\text{expandedquery}} = \underbrace{P(Q \mid S)}_{\text{originalquery}} \underbrace{\prod_{k=1}^{u} T\left(Q \mid F_Q^{c_k}\right) P\left(F_Q^{c_k} \mid S\right)}_{\text{expandedquery}} \tag{18}$$

where $T(Q \mid F_Q^{c_k})$ is the query translation model which will be discussed in the remaining section; and $P(F_Q^{c_k} \mid S)$ can be approximately regarded as the posterior probability of the class c_k for the database S, $P(c_k \mid S)$(recall Definition 5).

For computative simplification and analysis convenience, the likelihood function can be performed in the form of logarithm, which is described as

$$P_{QE}(Q \mid S) \propto \log P(Q \mid S) + \sum_{k=1}^{u} \log\left(T\left(Q \mid F_Q^{c_k}\right) P\left(F_Q^{c_k} \mid S\right)\right) \tag{19}$$

Note that due to the property of monotonic transformation of the logarithm, the log-likelihood ranking remains identical to the original ranking.

As described previously, the key component in the query expansion model is the query translation model $T(Q \mid F_Q^{c_k})$. This model is related to the feature set F^{c_k} in the topic c_k, which can be denoted as

$$T(Q \mid F_Q^{c_k}) = T(Q \mid c_k) P(c_k \mid F_Q^{c_k}) = T(Q \mid c_k) P\left(F^{c_k} \mid F_Q^{c_k}\right) \tag{20}$$

where $T(Q \mid c_k)$ is the degree of relevance of the topic c_k to the query, which is related to the weight of the expanded terms. To avoid the over- influence of the expanded query terms on the result of database selection, the weight of the expanded terms are usually diminished compared with the original query terms. Here, $T(Q \mid c_k)$ is normalized, which is denoted as

$$T(Q \mid c_k) = \frac{P_{JM}(Q \mid c_k)}{\sqrt{\sum_{k=1}^{u} (P_{JM}(Q \mid c_k))^2}} \tag{21}$$

where $P_{JM}(Q \mid c_k)$ is known at the category-specific stage (recall Section 3.3.2.1).

We note that the feature spaces, $F_Q^{c_k}$ and F^{c_k}, are both feature distributions in class c_k. $P(F^{c_k} \mid F_Q^{c_k})$ is related to the Kullback-Leibler distance (also known as cross entropy) (Kullback and Leibler 1951) which is an information theoretic measure of the divergence

of two probability distributions $F_Q^{c_k}$ and F^{c_k}. Our goal is to select a set of features $F_Q^{c_k}$ from F^{c_k}. We wish that the information loss between these two distributions is minimized, i.e., $KL(F^{c_k}, F_Q^{c_k})$ is as small as possible. We define that, given a feature vector f_i, $\delta_Q(f_i) = KL(P(F^{c_k} \mid f_i), P(F_Q^{c_k} \mid f_i))$. We therefore want to find the feature set $F_Q^{c_k}$ for which information loss $\Delta_Q = \sum_{f_i \in F_Q^{c_k}} P(f_i)\delta_Q(f_i)$ is reasonably small.

Intuitively, we use a greedy algorithm to obtain the feature set $F_Q^{c_k}$, which is described as follows:

1. Initially, we begin with a current feature set $F_Q^{c_k} = F_{c_k}$.
2. At each state, we eliminate a feature f_i in a way that allows $\Delta_{(F_Q^{c_k} - f_i)}$ to remain as close as possible to $\Delta_{F_Q^{c_k}}$. On each turn the elimination of feature f_i causes a small increase in Δ_Q.
3. Repeat step 2 until Δ_Q is increased to the desired information loss.
   Once the feature set $F_Q^{c_k}$ is selected from the feature space F^{c_k}, $P(F^{c_k} \mid F_Q^{c_k})$ can be described as

$$P\left(F^{c_k} \mid F_Q^{c_k}\right) = \frac{\sum_{f_i \in F_Q^{c_k}} P(f_i \mid c_k)}{\sum_{f_j \in F_{c_k}} P(f_j \mid c_k)} \tag{22}$$

4. Experimental design

4.1. The data sets

In order to demonstrate the effectiveness of our modeling techniques, we conduct a number of experiments to evaluate the selection performance of two-stage database language models. For our experiments, we do not use the TREC data set which is commonly used in distributed IR. The reason for this is that the topics for the TREC data set cannot be constructed in the form of a hierarchical structure. Since our database language modeling approach is based on the context of a topic hierarchy, we conducted a series of experiments on two hierarchically-structured data sets: the Reuters 21578 data set and the LookSmart Web data set.

The Reuters 21578 data set (http://www.daviddlewis.com/resources/testcollections/~reuters21578. html) is used by much of previous work on hierarchy methods for text classification (D'Alessio et al. 1995, Weighend et al. 1998, Kohler and Sahami 1997). This data set consists of 21578 articles labeled with 135 topics with no hierarchy structure. To construct a hierarchy of topics, we manually construct a 3-layer hierarchy of 4 top-level, 18 second-level, and 96 leaf-level categories. 4 top-level categories come from meta-categories codes (economic indicators, currency, commodities and energy). 18 second-level categories are the categories that tend to subsume some labels (e.g., the labels '*barley*' and *corn* are subsumed by the topic '*Agriculture*'; the labels '*gold*' and '*silver*' are subsumed by the topic '*Metallurgy*'). 96 topics are extracted from 135 category labels as the leaf-level categories.

However, Reuters-21578 is a small and homogeneous collection, which makes it problematic to understand how our approach is applicable to large, complex and heterogeneous Web collections. To overcome this problem, we investigate the use of hierarchies for searching very heterogeneous Web contents. For our experiments, we use a large collection of heterogeneous Web pages from LookSmart's Web directory (www.Looksmart.com). The LookSmart Web hierarchy is a 7-level hierarchy. We focused on the top two levels of the hierarchy which include all 13 of the top-level and 150 second-level categories. The classes are general subjects such as *Sports & Recreation*, and *Society & Politics*. This Web collection consists

of 370, 597 unique pages (from a May 1999 Web snapshot) as reported in (Dumais and Chen 2000), which have been hierarchically classified by trained professional Web editors. Each Web page had been assigned to zero or more categories. Pre-processing is executed to extract plain text from each Web page by removing HTML markup tags. In addition, the title description and keyword fields from the META tag were also extracted because they provide useful descriptions of the Web page. After pre-processing, a pseudo document was generated as a description of the original Web page.

4.2. Evaluation metrics

At the category-specific search stage, we draw the 11 point average precision-recall curve to compare the selection performance between the JM model and non-interpolated model without the interpolation of the parent class and the class corpus. The 11 point average precision-recall curve plots the average precision values at each of the 11 recall points 0, 0.1, 0.2, ..., 1.0, where *precision* P_{Class} and *recall* R_{Class}, are expressed by the following equations:

$$P_{\text{Class}} = \frac{\text{the number of relevant classes}}{\text{The total number of classes selected}} \tag{23}$$

$$R_{\text{Class}} = \frac{\text{the number of relevant classes}}{\text{The total number of relevant classes in the class set}} \tag{24}$$

At the term-specific search stage, we use the $\hat{R}_k(E, B)$ metric to compare the effectiveness of variations of three types of language models, namely, the MLE model, the DP model and the DP + QE model. Note that the names of all the models are abbreviated with the two initial letters (e.g., DP for Dirichlet Prior smoothing).

To compare the selection performance of our proposed approach, we provide a widely-used keyword-based technique—the CORI database selection algorithm (Callan 2000) as the experimental baseline. The CORI algorithm uses a variant of $tf \cdot idf$ [Salton and McGill 1983) adapted for ranking databases. The relevance score of a database S to the query Q is calculated as:

$$relevance\_score(Q \mid S) = \sum_i P(q_i \mid S) \tag{25}$$

$$p(q_i \mid S) = 0.4 + 0.6 \cdot T \cdot I \tag{26}$$

$$T = \frac{df}{df + 50 + 150 \cdot cw/avg\_cw} \tag{27}$$

$$I = \frac{\log\left(\frac{C+0.5}{cf}\right)}{\log(C + 1.0)} \tag{28}$$

where df is the document frequency of the query term q_i in database S, cw is the number of indexing term occurrences in database S, avg_cw is the average of cw in database set; C is the number of databases, and cf is the collection frequency of query terms q_i in database collection. Databases are then ranked based on the relevance score.

As to the performance metrics of database selection, we choose the $\hat{R}_k(E, B)$ metric for comparison, which had been used by several researchers in database selection (Gravano et al. 1999, French et al. 1999, Callan and Connell 2001, Ipeirotis and Gravano 2004). For each

query, two database ranks are provided: one is a *baseline or desired rank B* in which databases are ranked by their $r(Q, S)$ value, where $r(Q, S)$ is the number of documents contained in database S which are relevant to the query Q; the other is a *estimated rank E* which is ranked by the relevance score calculated by the database selection algorithm. The $\hat{R}_k(E, B)$ metric measures the percentage of relevant documents contained in the k top-ranked database, which is defined as

$$\hat{R}_k(E, B) = \frac{\sum_{S_i \in E_k} r(Q, S_i)}{\sum_{S_i \in B_{k^*}} r(Q, S_i)} \tag{29}$$

where E_k is the *estimated rank* of the k top-ranked database, and B_{k^*} is the *baseline rank* of all the databases that are useful for the query. The primary objective of database selection is to select a small set of databases that cover as many relevant documents as possible. This means that the higher the $\hat{R}_k(E, B)$ value, the better the database selection algorithm.

4.3. Experimental setup

In this paper, our experiments, in practice, are made up of two phases:

(1) *Training phase*: tuning parameters in the language models
In this phase, we consider a number of variations on language models and evaluate the impact that these variations had on the database selection results. These variations are:

- the effects of a wide range of parameter values, the JM coefficient λ_1 and the DP parameter α;
- the effects of query expansion size of the DP + QE model on database selection.

(2) *Test phase*: the comparison of different language models on selection performance
In this phase, we conduct a series of experiments to compare the selection performance for different language models. These comparisons are:

- the comparison of The JM model and the non-interpolated model on the selection of relevant topics at the class-specific search stage;
- the comparison of 3 different language models: the MEL model, the DP model, the DP + QE model on database selection at the term-specific search stage;
- the comparison of the two-stage selection approach and the general one-stage selection approach—the state-of-the-art CORI model on database selection.

In order to carry out the above experiments, the datasets are divided into two parts: the first is used to build the topic hierarchy used for the class-specific search stage; the second is decomposed into 130 smaller databases for each collection. Among these databases, 30 databases are used for the tuning of the model parameters in the language models at the training phase, and the rest of the 100 databases are for performance evaluation at the test phase. Tables 1 and 2 below separately list the detailed statistics of the topic hierarchy and the *training* and *test* databases for the Reuters-21578 and LookSmart datasets. Note that for the Reuters-21578 dataset, we only use 93 leaf-level classes to build the topic hierarchy other than 96 topics. The reason is that we found there are three topics that only have fewer than 5 documents. The number of documents is not enough for the training and test phase. So we have to discard these three topics in the construction of the topic hierarchy.

As we known, the performance of a selection system may vary significantly according to the test datasets used. To fairly compare the behavior of our language modeling method

Table 1 Statistics about the construction of the topic hierarchy in the Reuters-21578 and LookSmart Datasets

Collections	Reuters - 21578	LookSmart
Size of documents	2850	18780
Leaf-level classes	93	150
Level of class hierarchy	3	3
Mean relevant documents per class	32	125

Table 2 Statistics about the training and test databases for the Reuters-21578 and LookSmart Datasets

Collections	Reuters - 21578	LookSmart
The number of databases	30 (training) + 100 (test)	30 (training) + 100 (test)
The size per database	0.01–1.2 M	7.2–36.4 M
The number of documents per databases ($\times 100$)	0.5–10.2	30.2–126.3
The classes per database	2–8	12–30

with that of other methods, we perform our experiments on databases with different database characteristics, such as the large-scale databases with a variety of topics that consist of the LookSmart documents, and the small-scale databases with a few topics that consist of the Reuters-21578 documents. Each test dataset was divided into 130 smaller databases that were of different sizes (between 0.01–1.2 megabytes per Reuters-21578 database, and between 7.2–36.4 megabytes per LookSmart database), and varied in the number of documents they contain. The documents inside a database are from the same source or Web site (for the LookSmart databases) or the same time-frame (for the Reuters-21578 databases). Table 2 depicts the summarized statistics for the test databases in these two datasets.

For each small or medium-scale dataset, we recognize that the construction of 130 databases possibly leads to the problem of *document overlap* in the database collection, especially for the small Reuters-21578 dataset. To lessen the problem of document overlap, the document number of most of the databases in the Reuters-21578 dataset, in practice, is only within the range of 50–200. This is the main reason why the smallest size of Reuters databases is only 0.01 M (see Table 2). According to the statistical information obtained from the Reuters-21578 and LookSmart database collections, the percentage of document overlap is only 11.3 and 6.5%, respectively. We think they are reasonable figures for database construction.

For our experiments, 50 trials (25 for short queries and 25 for long queries) were conducted for each group of experiments. All trials were completely automatic, starting with a different selected query and then generating a ranked list of candidate topics or test databases at different search stages. The experimental results reported here are averages of results returned from the 50 trials.

To avoid the sensitivity of selection performance to query length, in this paper, we experimented with both short and long queries. Unlike the TREC data where these two types of queries can be created based on the *title*, *description* and *narrative* fields from the TREC-style topics, the Reuters-21578 and LookSmart dataset only contain the *title* selection in the documents. In order to acquire the proper queries, in this paper, we used the title as the short query, and the title concatenated with a group of frequent terms occurring in the *dateline* section of relevant documents in a particular topic domain as the long query. The reason for the use of frequent terms as query terms is that these frequent terms would return the most random sample of documents from the test datasets. In order to obtain long queries with the

Table 3 Average size of query description for the Reuters-21578 and the LookSmart Web Datasets

	Reuters–21578			LookSmart		
Collections	Min	Avg	Max	Min	Avg	Max
Short query (Title) (Words)	4	5.7	8	5	8.2	11
Long query (Title + frequent terms) (Words)	16	20.5	25	14	28.4	36

frequent terms for a specific topic, we use the query selection strategy reported in (Callan and Connell 2001). For each topic, a collection of relevant documents is chosen from the dataset (note that these documents are excluded from the training and test datasets used for the experiments). The selection of frequent terms is measured by average term frequency ($avg\_tf = ctf/df$), where ctf is collection term frequency and df is document frequency. Note that all the documents and the queries are preprocessed by removing stop words and employing the Porter stemming algorithm (Porter 1980). We hired undergraduate students to produce long queries. We provide them a list of frequent words for reference when they generated long queries. Table 3 below lists query set statistics for these two datasets.

As described earlier in Section 4.1, the documents in the two datasets had been labeled with relevant topics. Thus, the relevant assessment for the queries were based on the category label associated with each document in the database. The documents with no category label are treated as irrelevant. For example, each Reuters-21578 document contains the <*topic*> field that denotes the class label of this document. With the information provided by the <*topic*> field of the document, it is easy to know the number of relevant documents in a database with respect to a given query of a particular topic.

5. Experimental results

In this section, we present experimental results to test the robustness of using the smoothed two-stage language model for database selection.

5.1. The effects of various values of smoothing parameters in individual language models

To examine the effect of the smoothing parameter in each specific smoothing method on selection effectiveness, we vary the value of the smoothing parameters in a wide range in order to observe all possible difference in smoothing. In each run, we set of the smoothing parameters the same value across different data sets to report the average selection performance at each parameter value. Then, we compare selection performance by plotting the average precision P_{Class} or the $\hat{R}_k(E, B)$ metric against the variation in these values.

Jelinek-Mercer (JM) Smoothing: We compare the precision of the JM model at different settings of the smoothing parameter λ_1 (Recall Eq. (11)). In order to carry out an exhaustive search on the parameter space of λ_1, we firstly tried 10 values {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95}. We note that, when λ_1 varies in the range between 0.5∼0.8, the results are statistically significant improvement in precision. This means that, if we expect that the JM model performs reasonably well, we had better set λ_1 at some *desired* range with the value of 0.5–0.8. However, it is hard to determine the optimal λ_1 value, because the set of the

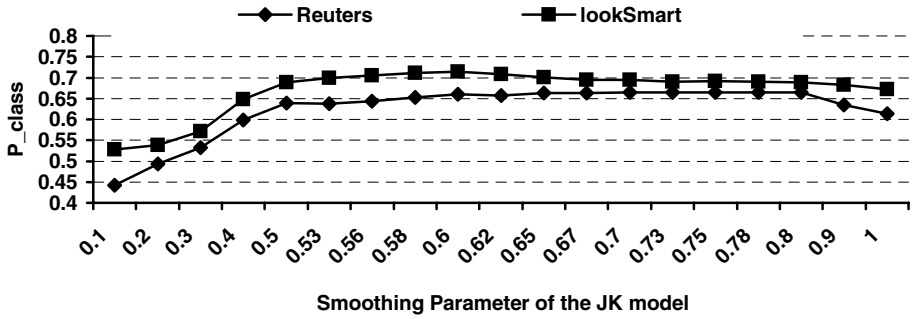

Fig. 2 The effect of different values of the JM coefficient λ_1 on selection performance

optimal λ_1 varies with individual queries and topic classes throughout the training process. To further obtain the optimal λ_1 on separate datasets, we used the EM algorithm (Recall Section 3.3.1) to learn the optimal value over the *desired* parameter space with the value of 0.5–0.8.

The plots in Figure 2 show that the average precision for different values of λ_1 over two datasets. It is easily noted that the choice of the appropriate smoothing parameter λ_1 can have a large impact on selection performance. We can see that the optimal value of λ_1 is a little high (0.67 for the LookSmart and 0.75 for the Reuters), which suggests that although the parent-class model and the class-corpus model can help smooth the word sparseness, the basic class model $P(q \mid c_k)$, in practice, plays a major role in class selection.

Dirichlet Prior (DP) Smoothing: To see the detail change, we experimented with a wide range of value of the DP smoothing parameter α Recall Eq. (17). The plots in Figure 3 show the average $\hat{R}_k(E, B)$ value for different settings of the prior sample size α. It is observed from the results in Figure 3 that when the size of the database becomes large, there is a great chance for α to be at a small value. This is probably because α is a document-dependent parameter, which is affected by the length of the data collection. When the data collection becomes larger, α tends to be smaller in order to emphasize the impact of the original occurrence frequency of query term $C(t_i \mid S)$ on selection process. The effect of α is similar to the smoothing parameter λ_1 in the JM model. The optimal value of α seems to vary from dataset to dataset. However, in most cases, it is "safe" to be set a relative large value with the range between 1000–3000.

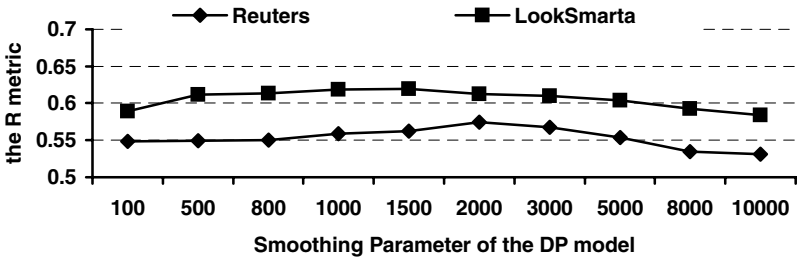

Fig. 3 The effect of different values of the DP parameter α on selection performance

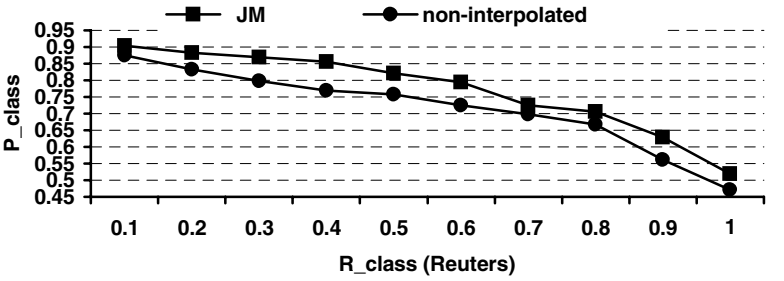

Fig. 4 Selection performance of the JM model and the non-interpolated model in the reuters 21578 data set

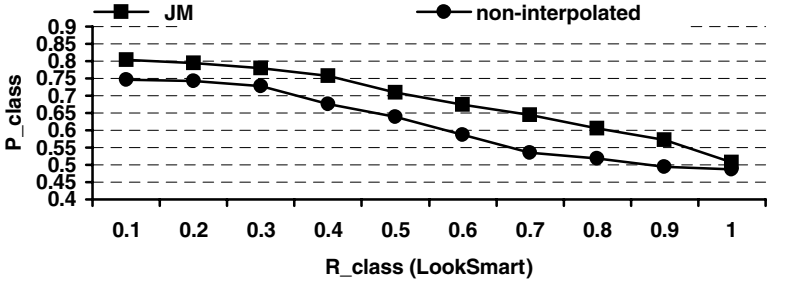

Fig. 5 Selection performance of the JM model and the non-interpolated model in LookSmart data set

5.2. The JM model vs the non-interpolated model on relevant topic selection

The class-based language model using the JM smoothing method is expected to perform better than the simple non-interpolated model. In order to prove it, we compare the precision-recall chart of the JM model with that of the non-interpolated model over two datasets. The 11 point precision-recall curves are shown in Figures 4 and 5. Note that the figure for the JM model uses the best choice of the smoothing parameter $\lambda_1(\lambda_1 = 0.7)$.

Figures 4 and 5 clearly show that on the 11 point precision-recall chart, the JM language modeling approach achieved better precision at all levels of recall. The precision is improved significantly by 7.54% and 9.45% on average respectively. This means that the linear interpolated approach is an attractive way in which the parent-class model and the class-corpus model help effectively smooth the word sparseness and therefore improve the selection effectiveness.

It is interesting to notice that the behavior of the JM model and the non-interpolated model varies in different data sets. For small-scale Reuters databases, the improvement of the JM model over the non-interpolated model is not as significant as that for the LookSmart data set. The performance gain in the Reuters data set achieves 7.54% improvement in contrast to 9.45% in the LookSmart data set. One possible reason for this difference might be because in the small-scale data corpus, there are more chances to assign a default higher corpus probability to a non-occurring query term in a specific database. It leads to the deterioration of selection performance. However, this is only a conjecture, the verification of which we leave for future work.

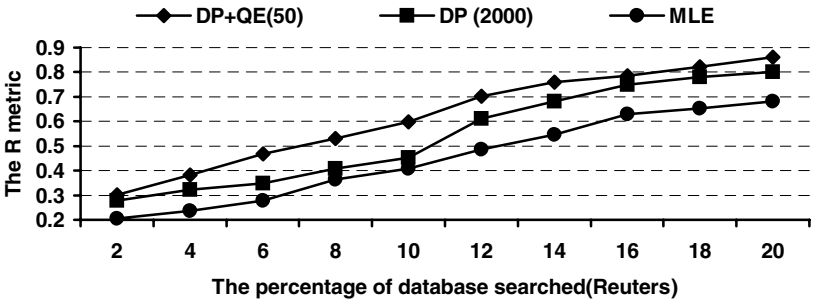

Fig. 6 The effects of different term-based language models on selection performance in the reuters data set

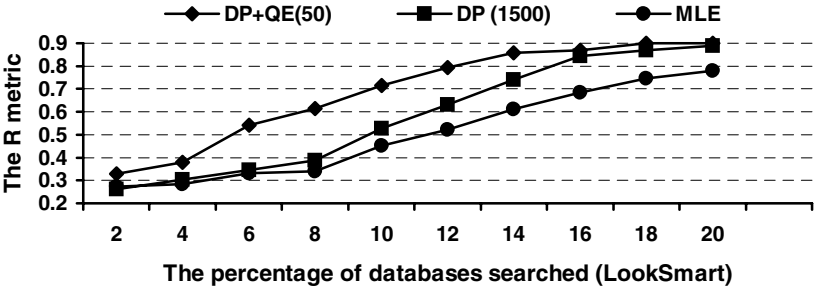

Fig. 7 The effects of different term-based language models on selection performance in the LookSmart data set

5.3. The effects of different term-based language models on selection performance

In this subsection, we study the behavior of three different term-based language models discussed in Section 3.3.2. These language models, the MLE model, the DP model and the DP + QE model are evaluated in our experiments. The selection performances of these three models are shown in Figures 6 and 7.

Compared with the behavior of the MLE model, the DP model and the DP + QE model improve selection performance significantly and consistently across both datasets. This is understandable because some query terms missed at the database lead to a bad probability estimate for the query in the MLE model.

As would be expected, the DP + QE model (Expansion terms are 50) performs the best among all the models. The average $\hat{R}_k(E, B)$ value (Recall Section 4.2) is increased by 42.9 and 41.23% on average against the MLE model in Figures 6 and 7. It suggests that query expansion does help improve the selection effectiveness if the number of expansion terms is chosen properly.

Note that the DP model has not performed as well as the DP + QE model in all our experiments. However, the results of the DP model are still very competitive when compared with the performances of the MLE model. As we see, in all results, the performance of the DP model smoothed with the optimal parameter value α ($\alpha = 1500$) is statistically better than the best performance of the MLE model, even though it is sometimes very close to the MLE model in specific cases.

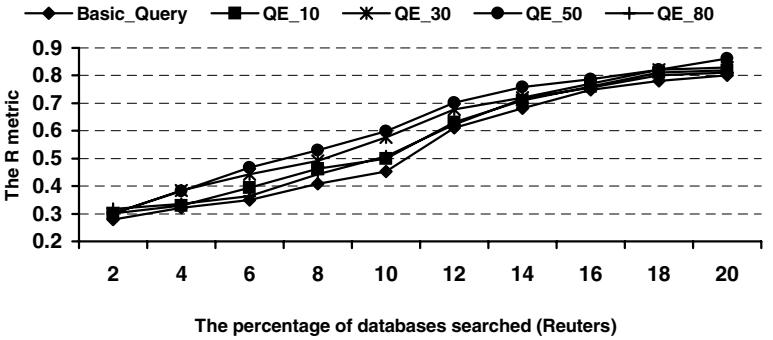

Fig. 8 The effect of query expansion size on selection performance in the reuters data set

5.4. The effects of query expansion size on selection performance

Obviously, the quality of query expansion might well be influenced by the quality of the features associated with topic classes. When expansion terms are selected, how many of them are actually relevant to the original query? To investigate this, we conducted experiments on different query expansion sizes over both data sets. It is interesting to observed how the number of expansion terms used affects selection performance (e.g., QE_10 refers to the query expansion with 10 terms). To observed this more clearly, we plot the $\hat{R}_k(E, B)$ value curves in Figures 8 and 9.

There are a number of interesting points to observe in Figures 8 and 9. First, the results of the translation model with query expansion are comparable to that of the base-query model. This is a clear indication that query expansion can be helpful in improving the selection performance using the translation modeling approach. Second, the greatest improvement can be seen when 50 terms are used for selection in both datasets. When more expansion terms take part in selection, the performance begins to deteriorate. The possible explanation for this phenomenon is that the most distinguished features for the class could be contained in the top 50 features. For this reason, selecting a larger number of features for query expansion does not bring any larger benefit.

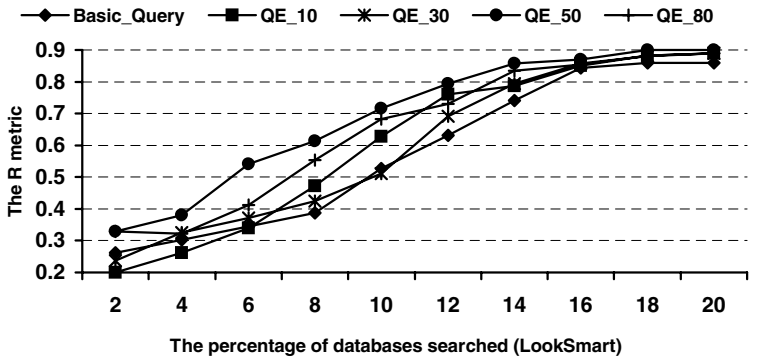

Fig. 9 The effect of query expansion size on selection performance in the LookSmart data set

5.5. The comparison of the two-stage database selection approach with the one-stage approaches on selection performance

The aim of the experiments in this subsection is to check the overall influence of our proposed two-stage database selection approach on selection performance. As mentioned earlier, one of the distinguish characteristics of our proposed database selection approach different from most current one-stage database selection approaches is that the search focuses on a small set of databases associated with the most related topic classes to the query. In the previous Sections 5.2 and 5.3, we separately evaluate the performance of the language models at different search stages, and compare their effectiveness of selection. Here, we deliberately conduct a set of experiments to compare the overall selection performance of the estimated two-stage language models with the best results achievable using the one-stage model—the CORI model in both test datasets.

Three types of language models are measured in the experiments: one two-stage model—the JM + QE + DP model, and two one-stage models—the QE + DP model and the baseline CORI model. To compare the selection performance of these three models, we select a group of best runs for each model on each testing dataset and compare the average $\hat{R}_k(E, B)$ value at the top 20 databases of the selected runs. The plots in Figures 10 and 11 show the average $\hat{R}_k(E, B)$ value for different database selection models.

In all the results, it seems to be a clear ordering among the three selection approaches in terms of the $\hat{R}_k(E, B)$ metrics. The performance difference is clearly significant for

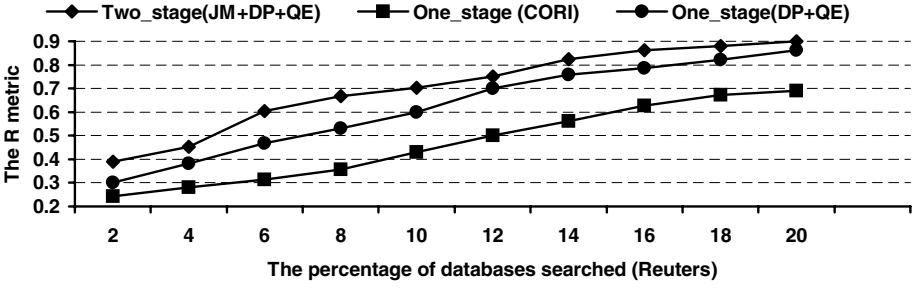

Fig. 10 The comparison of three different language models on selection performance in the reuters data set

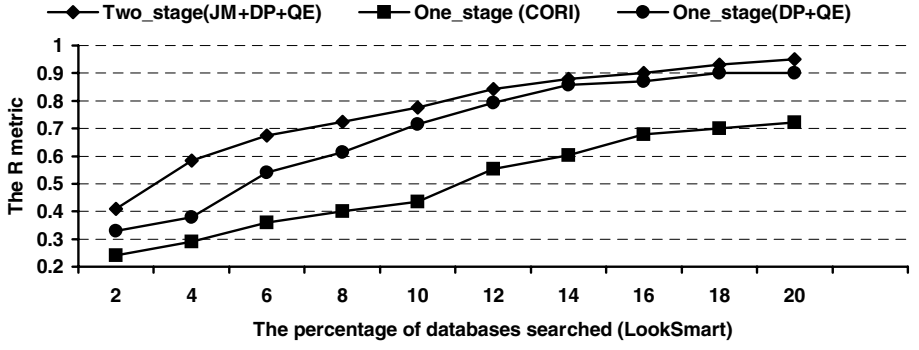

Fig. 11 The comparison of three different language models on selection performance in the LookSmart data set

the JM+QE + DP model and the QE + DP model compared to the baseline model—the CORI model. Figure 10 shows that, the overall $\hat{R}_k(E, B)$ value of these two models increase greatly with 51.67% and 31.54%, respectively, on the Reuters-21578 dataset. The same effect is also observed in the LookSmart dataset, as illustrated in Figure 11. It appears that the statistical language model is quite a robust language model for database selection.

We also notice that, the performance of the JM + QE + DP approach is consistently better than the best performance of the QE + DP approach. Indeed, some results in Figures 10 and 11 suggest that the category-specific search stage is usually helpful in improving the selection effectiveness on relevant databases.

6. Conclusions and future works

We have presented a novel way of looking at the problem of database selection based on two-stage language models that are rooted on the solid foundation of statistical natural language processing. For databases categorized into a topic hierarchy, instead of searching them in an ad hoc manner, we intentionally keep the database language models separated according to individual natures of different search stages. It makes our models easy to be understood and extended. We also conducted a number of experiments to prove the effectiveness and robustness of our language models. The experimental results have demonstrated that this approach holds great promise of improving search performance. Due to the simplicity of our models, we believe that our model can be easily extended to incorporate with any new language-based techniques under a general, well-ground framework.

We plan to investigate the following matters in the future work. First, due to time constraint, our current language models do not incorporate many familiar ideas into our selection system such as relevant feedback and semantic-based information retrieval. It is possible that additional knowledge added to the models will further improve our selection system. For example, using relevant feedback techniques, a user can provide more meaningful words which facilitate the formation of better queries.

Second, as mentioned previously, we simply take all the words occurring in the databases independently. Such unigram models completely ignore word phrases that are likely to be beneficial to probability estimate of the query. To compensate for this shortcoming, a possibility is to combine bigram or trigram models into our current language models. However, parameter estimation of these complex language models remains a major difficulty. Thus, we hope to borrow the techniques in other related areas to help solving the word phrase problem. For example, in the kernel-based SVM, Cancedda et al. proposed the use of *word-sequence* kernels, a novel way of computing document similarity based on matching sequences of words (Cancedda et al. 2003). The idea of sequence kernels to process document as a sequence of words is interesting, which is worth investigating in our future work. We wish that we could combine this technique into our current database selection model.

Third, we only employ simple smoothing methods in our current work. We are planning to explore other more elaborate smoothing methods to extend our models. It is also possible that this will improve selection results.

Fourth, in our experiments, we used the frequently occurring words as long queries. This approach to produce queries probably leads to the problem of a self-fulfilling good evaluation in database selection. To avoid this problem, one of the possible appropriate methods is the direct participation of the human being in query formation. The users will create more natural

queries with their own language. This kind of queries will be useful for our proposed language model used in the real-world applications. We will leave this for further research.

Finally, in our current datasets used for the experiments, we use the Reuters 21578 dataset which contains a very small set of homogeneous documents. This dataset is not enough for simulating a huge, complex, and heterogeneous real environment. Recently, a new Reuters dataset - Reuters Corpus Volume 1 (RCV1) is introduced by Lewis et al. (2004), which is a collection of over 800,000 documents manually categorized newswire stories made available by Reuters, Ltd for text categorization. One of the important features for this dataset is that this collection is constructed with the hierarchical category taxonomies. This feature makes RCV1 dataset suitable for our current research. We are planning to replace the Reuters 21578 dataset with the RCV1 dataset to further investigate the performance of our proposed two-stage approach in a more complex environment.

Acknowledgment

This research was supported by a university grant from Wollongong University under contract SDRG03-227381010. We thank Peter Eklund for his help and advices in the written expression of this paper. Finally, we would like to thank the anonymous reviewers for many valuable comments and suggestions that are useful for improving the quality of this paper.

References

Apte C, Damerau R and Weiss SM (1994) Automated learning of decision rules for text categorization. ACM Transactions on Information Systems, 12(3):233–251.

Baumgarten (1997) A probabilistic model for distributed information retrieval. In: Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, New York, pp. 258–266.

Baumgarten C (1999) A probabilistic solution to the selection and fusion problem. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Berkeley, California, USA, pp. 246–253.

Berger A and Lafferty J (1999) Information retrieval as statistical translation. In: Proceedings of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Berkeley, California, pp. 222–229.

Callan J (2000) Distributed information retrieval. In: W.B. Croft, (Ed.), Advances in Information Retrieval. Kluwer Academic Publishers, pp. 127–150.

Callan J and Connell M (2001) Query-based sampling of text databases. ACM Transactions on Information Systems, 19(2):97–130.

Cancedda N, Gaussier E, Goutte C and Renders JM (2003) Special issue on machine learning methods for text and images: Word sequence kernels. ACM Journal of Machine Learning Research, 3:1059–1082.

Craswell N, Baile P and Hawking D (2000) Server selection on the world wide web. In: Proceedings of the 5th International Conference on Digital Libraries, pp. 37–46.

D'Alession S, Murray M, Schiaffino R and Kershenbaum A (1998) Category levels in hierarchical text categorization. In: Proceedings of the Third Conference on Empirical Methods in Natural Language Processing.

David RH, Miller TL and Richard MS (1999) A hidden markov model information retrieval system. In: Proceedings of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Berkeley, California, United States, pp. 214–221.

Dempser AP, Laird NM and Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. International Journal of the Royal Statistical Society, 39(B):1–38.

D'Souza D, Thom J and Zobel J (2000) A comparison of techniques for selecting text collections. In: Proceedings of the 11th Australasian Database Conference. Canberra, Australia, pp. 28–32.

Dumais ST and Chen H (2000) Hierarchical classification of web content. In: N. Y. ACM Press, US, Eds. Proceedings of the 23rd ACM SIGIR International Conference on Research and Development in Information Retrieval. Athens, GR, pp. 256–263.

French JC, et al. (1999) Comparing the performance of database selection algorithms. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Berkeley, California, USA, pp. 238–245.

Gauch S, Wang G and Gomez M (1996) Profusion: Intelligent fusion from multiple, distributed search engines. The Journal of Universal Computer Science, 2(9):637–649.

Gravano L, Chang CK, García-Molina H and Paepcke A (1997) Starts: Stanford proposal for internet meta-searching. In: Proceedin of the 1997 ACM SIGMOD International Conference on Management of Data. New York, pp. 207–218.

Gravano L, Garcia-Molina H and Tomasic A (1999) *Gloss*: Text-source discovery over the internet. ACM Transactions on Database Systems, 24(2):229–264.

Gravano L, Ipeirotis PG and Sahami M (2003) Qprober: A system for automatic classification of hidden-web databases. ACM Transactions on Information Systems, 21(1):1–41.

Hawking D and Thistlewaite P (1999) Methods for information server selection. ACM Transaction on Information System, 17(1):40–76.

Hiemstra D (1998) A linguistically motivated probabilistic model of information retrieval. In: Proceedings of the 2nd European Conference on Digital Libraries. Heraklion, Crete, Greece, pp. 569–584.

Ipeirotis PG and Gravano L (2002) Distributed search over the hidden web: Hierarchical database sampling and selection. In: Proceedings of the 28th International Conference on Very Large Databases. Hong Kong, China, pp. 394–405.

Ipeirotis PG and Gravano L (2004) When one sample is not enough: Improving text database selection using shrinkage. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. Paris, France, pp. 767–778.

Ipeirotis PG, Gravano L and Sahami M (2001) Probe, count, and classify: Categorizing hidden web database. In: Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data. Santa Barbara, California, USA, pp. 67–78.

Jelinek F and Mercer R (1980) Interpolated estimation of marvok source parameters from sparse data. In: Patter Recognition in Practices. Amsterdam, Holland, pp. 381–402.

Jin R, Hauptman A and Zhai C (2002) Title language model for information retrieval. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Tampere, Finland, pp. 42–48.

Koller D and Sahami M. (1997) Hierarchically classifying documents using very few words. In: Proceedings of the Fourteenth International Conference on Machine Learning. Nashville, Tennessee, USA, pp. 170–178.

Kullback S and Leibler RA (1951) On information and sufficiency. Annals of Mathematical Statistics, 22:76–88.

Lafferty J and Zhai C (2001) Document language models, query models, and risk minimization for information retrieval. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New Orleans, Louisiana, USA, pp. 111–119.

Lewis DD, Yang Y, Rose TG and Li F (2004) Rcv1: A new benchmark collection for text categorization research. Journal of Machine Learning Research, 5:361–397.

Manber U and Bigot P (1997) The search broker. In: Proceedings of USENIX Symposium on Internet Technologies and System. Monterey, California.

Meng W, Liu KL, Yu C, Wang X and Chang Y (1998) Determining text databases to search in the internet. In: Proceedings of the 24th International Conference on Very Large Data Bases. New York, USA, pp. 14–25.

Meng W, Wang W, Sun H and Yu C (2002) Concept hierarchy based text database categorization. Journal of Knowledge and Information Systems, 4(2):132–150.

Miller DJ, Leek T and Schwartz RM (1999) A hidden markov model information retrieval system. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Berkeley, California, USA, pp. 214–221.

Mood AM and Graybill FA (1963) Introduction to the Theory of Statistics 2th Ed., McGraw-Hill.

Ponte JM and Croft WB (1998) A language modeling approach to information retrieval. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Melbourne, Australia, pp. 214–221.

Porter M (1980) An algorithm for suffix stripping. Program, 14(3):130–137.

Powell AL and French JC (2003) Comparing the performance of collection selection algorithms. ACM Transactions on Information systems, 21(4):412–456.

Robertson SE (1977) The probabilistic ranking principles in IR. International Journal on Document, 33:294–304.

Robertson SE and Sparck Jones K (1976) Relevance weighting of search terms. Journal of American Society of Information Science, 27:129–146.

Salton G and McGill M (1983) Introduction of modern information retrieval. McGrag-Hill, New York.
Si L, Jin R, Callan J and Ogilivie P (2002) A language modeling framework for resource selection and results merging. In: Proceedings of the Eleventh International Conference on Information and Knowledge Management. McLean, Virginia, USA, pp. 391–397.
Si L and Callan J (2003) The effect of database size distribution on resource selection algorithms. In: Proceedings of SIGIR 2003 Workshop on Distributed Information Retrieval. Toronto, Canada, pp. 31–42.
Song F and Croft WB (1998) A general language model for information retrieval. In: Proceedings of the Eighth International Conference on Information and Knowledge Management. Kansas City, Missouri, USA, pp. 316–321.
Turtle H and Croft WB (1990) Inference network for document retrieval. In: Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, pp. 1–24.
Van Rijsbergen CJ (1989) Towards an information logic. In: Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, pp. 77–86.
Van Rijsbergen CJ (1992) Probabilistic retrieval revisited. International Journal of Computation, 35:291–298.
Voorhees E, Gupta NK and Johnson-Laird B (1995) Learning collection fusion strategies. In: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, pp. 172–179.
Weighend AS, Wiener ED and Pedersen JO (1999) Exploiting hierarchy in text categorization. Information Retrieval, 1(3):193–216.
Wong SKM, Ziarko W, Raghavan VV and Wong PCH (1987) On modeling of informtion retrieval concepts in vector space. ACM Transaction Database System, 12:229–321.
Xu J and Croft WB (1999) Cluster-based language models for distributed retrieval. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Berkeley, California, USA, pp. 254–261.
Yang H and Zhang M (2004) Hierarchical classification for multiple, distributed web databases. International Journal of Computers and Their Applications, 11(2):118–130.
Yu C, Liu K, Wu W, Meng W and Rishe N (1999a), "A methodology to retrieve text documents from multiple databases," Technical report, University of Illinois at Chicago.
Yu C, Meng W, Liu KL, Wu W and Rishe N (1999b) Efficient and effective metasearch for a large number of text databases. In: Proceedings of the Eighth International Conference on Information and Knowledge Management. Kansas City, Missouri, USA, pp. 217–224.
Yuwono B and Lee DL (1997) Server ranking for distributed text retrieval systems on internet. In: Proceedings of the Conference on Database Systems for Advanced Applications, pp. 41–49.
Zhai C and Lafferty J (2001) A study of smoothing methods for language models applied to ad hoc information retrieval. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New Orleans, Louisiana, United States, pp. 334–342.
Zaragoza H, Hiemstra D and Tipping M (2003) Bayesian extension to the language model for ad hoc information retrieval. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval. Toronto, Canada, pp. 4–9.
Zobel J (1997) Collection selection via lexicon inspection. In: Proceedings of the 2nd Australian Document Computing Symposium. Melbourne, Australia, pp. 74–80.