Relevance Ranking using Kernels

Jun Xu[a], Hang Li[a], Chaoliang Zhong[b]

[a]*Microsoft Research Asia, No. 49 Zhichun Road, Beijing, China 100190*
[b]*Beijing University of Posts and Telecommunications, Beijing, China 100876*

Abstract

This paper is concerned with relevance ranking in search, particularly that using term dependency information. It proposes a novel and unified approach to relevance ranking using the kernel technique in statistical learning. In the approach, the general ranking model is defined as a kernel function of query and document representations. A number of kernel functions are proposed as specific ranking models in the paper, including BM25 Kernel, LMIR Kernel, and KL Divergence Kernel. The approach has the following advantages. (1) The (general) model can effectively represent different types of term dependency information and thus can achieve high performance. (2) The model has strong connections with existing models such as BM25 and LMIR. (3) It has solid theoretical background. (4) The model can be efficiently computed. Experimental results on web search dataset and TREC datasets show that the proposed kernel approach outperforms MRF and other baseline methods for relevance ranking.

Key words: Information retrieval, Relevance ranking, Kernel

1. Introduction

Relevance ranking is still a central issue of research in Information Retrieval. Traditionally, relevance ranking models are defined based on the bag-of-words assumption, i.e., query and document are viewed as two sets of words. These include Vector Space Model (VSM) [27], Probabilistic model BM25 [25, 24], and Language Models for IR (LMIR) [21, 35].

There is a clear trend in IR recently on study of models beyond bag-of-words, particularly under the notion of term dependency or proximity, which assumes terms are dependent and takes into consideration in relevance ranking the cooccurrences of query terms in documents. For example, if the query is 'machine learning book', then a document containing the phrase 'machine learning' and the term 'book' should be more relevant than a document in which the terms 'machine', 'learning', and 'book' occur separately. It is obvious from the example that dependency informaiton needs to be leveraged in relevance ranking. For instance, a model for using term dependency based on Markov Random Fields (MRF) has been proposed [18], in which the information on independent terms (unigrams), sequential terms (n-grams), and dependent terms is exploited. The MRF model performs well and is regarded as a state-of-the-art method for using term dependency. (For other work, see [23, 1, 14, 15, 2, 31, 7, 5].)

In our view, an ideal ranking model using term dependency should have the following characteristics. (1) It can incorporate various types of term dependency information to represent relevance, and thus can achieve high accuracy in ranking. (2) It has strong connections with existing models . (This is necessary, as existing models of BM25 and LMIR already perform quite well.) (3) It is based on solid theory. (4) It can be efficiently computed. Although previous work on term dependencies made progresses, further study on the problem still appears to be necessary. In this paper, we aim to conduct a comprehensive study on relevance ranking using term dependency, and to propose a general ranking model which has the advantages.

Specifically this paper proposes a novel approach to relevance ranking, on the basis of String Kernel in statistical learning. It defines the relevance ranking model as a kernel function of query string and document string. The original query string and document string are thus mapped into vectors in a higher dimensional Hilbert Space and relevance (similarity) between the two representations is defined as dot product between the mapped vectors in the Hilbert Space.

As case studies, we introduce two kernel functions, BM25 Kernel and LMIR Kernel, and use one existing kernel function, KL (Kullback-Leibler) Divergence Kernel, in this paper. The former two are asymmetric kernel functions and the last is a symmetric kernel function. They correspond to different ways of mapping the original query and document strings to the Hilbert Space.

(1) The kernel based models can naturally represent various types of term dependency information, for example, n-grams and n-dependent-terms. Since the essence of relevance is to compare the similarity between query and document, a richer model on the similarity between query and document can then be realized with the approach and high performance in relevance ranking can be achieved.

(2) BM25 Kernel and LMIR Kernel respectively include conventional BM25 model and LMIR model as special cases. (Note that many other relevance models can also be interpreted as kernel functions, for example, the traditional Vector Space Model.) Within the kernel framework, different ranking models can be easily studied and compared.

(3) Kernel methods are becoming very popular in statistical learning. It is attractive because of its powerful representability. The proposed approach using kernel thus has a solid theoretical background. Although String Kernel techniques have been used in other applications, as far as we know, this is the first work using String Kernel for relevance ranking.

(4) Kernel functions are essentially dot product. This enables an efficient computation of kernel functions in search.

The implication of the work is far beyond using term dependency. The proposed kernel-based ranking models are actually generalized versions of conventional ranking models. It provides a new view, namely kernel function, on relevance ranking.

We conducted experiments using data from a web search engine and TREC data, and found that BM25 Kernel, LMIR Kernel, and KL Kernel perform better than MRF and other bag-of-words models for relevance ranking. We conclude, therefore, that the kernel approach proposed in this paper really has advantages.

The rest of the paper is organized as follows. After a survey on related work in Section 2, we propose the kernel based ranking models in Section 3. Advantages of the kernel based ranking models are discussed in Section 4. Section 5 reports our experimental results and Section 6 concludes the paper.

2. Related work

Many models have been proposed for ranking in Information Retrieval. Vector Space Model (VSM) [27] represents the query and document as tf-idf vectors and takes the dot product between the vectors (cosine similarity) as relevance score. BM25 is a formula derived from the probabilistic approach, which models relevance ranking with a two-Poisson distribution [25]. Language Models for IR (LMIR) [21, 35] is another probabilistic approach to ranking, which assumes query is generated from a document and uses the likelihood of query to document as relevance score. All the models were originally proposed based on the bag-of-words assumption. Obviously such an assumption is too strong, as in many cases query terms (words) are related, and their relations (dependencies) should also be considered in relevance ranking.

The beyond bag-of-words efforts may be categorized into two groups. One approach considers the use of term dependency, specifically, it assumes that the query is composed of several units of terms (e.g., n-grams) and utilizes the occurrences of the units in the document in ranking. For example, Metzler and Croft [18] proposed using Markov Random Fields to characterize the term dependency relations. They incorporated different types of term dependency into the MRF model and found that it can almost always outperform the conventional bag-of-words models. The MRF approach is viewed as one of the state-of-the-art methods for using term dependency. (For other related work see [1, 9, 22, 5, 30, 4, 20, 3]). The other approach considers the use of degree of approximate match of query to document, for example, it looks at the minimum span of the query terms appearing in the document (i.e., the best approximate match). This is also referred to as proximity. Tao and Zhai [31] conducted a comprehensive study on proximity measures for relevance ranking and found that the proximity measures are useful for further improving relevance. (Also refer to [2, 14, 15, 11].)

In statistical learning, the kernel technique is widely used. Kernel function characterizes a kind of similarity between two representations $k(x, y)$, where x and y denote representations. For example, String Kernel is a special type of kernel function defined on text strings, i.e., x and y are strings [10, 16]. KL Divergence Kernel is a type of kernel function defined based on KL Divergence [19]. In kernel technique, instead of directly measuring the similarity between the two representations, we map the representations into vectors in a higher dimensional Hilbert Space and compute the dot product between the vectors. The mapping function usually is a nonlinear function and we do not need to explicitly define it. Because of the rich representability of kernel function as similarity measure, it is widely used in statistical learning. For example, it is combined with linear classification algorithms such as SVM to solve nonlinear classification problems [29]. There are two ways to create (or identify) a kernel function. The Mercer's theorem provides a sufficient and necessary condition of kernel function, that is, any continuous, symmetric, positive semi-definite function $k(x, y)$ is a kernel function. One can also define a kernel function in a constructive way, i.e., through definition of mapping function. Usually a kernel function is symmetric in the sense $k(x, y) = k(y, x)$. Asymmetric kernel functions have also been introduced, which satisfy $k(x, y) \neq k(y, x)$ [28, 32]. Obviously asymmetric kernel functions are more general than symmetric kernel functions.

Kernel functions have been applied into several tasks in IR. In [33], the KL Divergence Kernel was used for calculating the similarity between documents given a query. In [16], a method for text classification using String Kernels was studied. In [26], a kernel function for measuring similarity between short texts was proposed. In [17], a number of String Kernel functions were surveyed and implementations of the functions using suffix trees were also suggested.

Learning to rank is a new area in IR, which aims at constructing a ranking model by using training data and a supervised learning technique [12, 34, 6]. Usually in learning to rank, the ranking model contains a number of features. The kernel-based ranking models proposed in this paper can be utilized as features in learning to rank.

3. Kernel Approach

The essence of relevance is to compare the similarity between query and document, or the matching degree between query and document. Different ways of defining the similarity or matching degree lead to different ranking models. In this paper, we use kernel techniques in statistical learning to construct ranking models. We first give a general model, and then some specific examples.

3.1. General Model

There are three issues we need to consider when defining a ranking model: query representation, document representation, and similarity calculation between query and document representations. Here, we assume that query and document are two strings of words (terms). Such a representation can retain all the major information contained in the query and document. We then define the similarity function between the query and document as kernel function between query string and document string.

Suppose that $\mathcal{Q}$ is the space of queries (word strings), and $\mathcal{D}$ is the space of documents (word strings). (q, d) is a pair of query and document from $\mathcal{Q}$ and $\mathcal{D}$. There is a mapping function $\phi : \mathcal{Q} \mapsto \mathcal{H}$ from the query space to the Hilbert space where $\mathcal{H}$ denotes the Hilbert Space. $\phi(q)$ then stands for the mapped vector in $\mathcal{H}$ from q. Similarly, there is a mapping function $\phi' : \mathcal{D} \mapsto \mathcal{H}$ from the document space to the Hilbert Space $\mathcal{H}$. $\phi'(d)$ then stands for the mapped vector in $\mathcal{H}$ from d. The similarity function between q and d is defined as a kernel function between q and d,

$$k(q, d) = \langle \phi(q), \phi'(d) \rangle = \sum_i \phi(q)_i \cdot \phi'(d)_i. \tag{1}$$

The kernel function is actually the dot product between vectors $\phi(q)$ and $\phi'(d)$, respectively mapped from q and d.

We call the kernel function $k(q, d)$ in Eq. (1) (general) kernel-based ranking model. Note that $k(q, d)$ is an *asymmetric* function, which means $k(q, d) \neq k(d, q)$, because ϕ and ϕ' are not necessarily identical. When ϕ and ϕ' are identical, the kernel function becomes a symmetric function, i.e., $k(q, d) = k(d, q)$.

The general kernel-based ranking model can include most existing relevance ranking models such as VSM, BM25, LMIR, and even LSI [8], as will be made clearer later. For example, in VSM, the query string and the document string are mapped into vectors of tf-idf values and dot product between the vectors is calculated and used, which is a very simple example of the general ranking function in Eq. (1).

3.2. Model Construction

Let us introduce the way of creating a kernel-based ranking model. We first identify a 'type of dependent query terms' for which we care about the occurrences in documents. (For example, if the type is bigram, we decompose the query 'machine learning book' into two bigrams 'machine learning' and 'learning book', and we look at their occurrences in documents.) Next we assume to respectively map the query string and document string into vectors of the type in a Hilbert Space. (For example, we map query and document into vectors of bigrams.) We can then construct a kernel function for the type on the basis of the mapping functions. Finally, we can define a ranking model by combining the kernel functions in different types using the properties of kernel functions (cf., Section 4.3). That is to say, we actually constructively create kernel functions (ranking models).

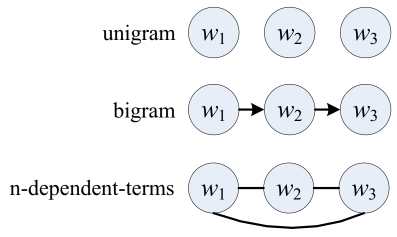

Figure 1: Illustration of unigram, n-gram, and n-dependent-terms.

As types, we can consider the occurrences of independent query terms in the document (unigrams), the occurrences of sequential query terms in the document (n-grams) , or the occurrences of dependent query terms in the document (n-dependent-terms). Then we can create a vector of unigrams, a vector of n-grams, or a vector of n-dependent-terms in the Hilbert Space. The relation between unigrams, n-grams, and n-dependent-terms is illustrated in Figure 1.

For each type, the set of possible units (unigrams, n-grams, or n-dependent-terms) is fixed, provided that the vocabulary is fixed. We denote the set of units as $\mathcal{X}$ and each element of it x.

In this paper we give three examples of kernel-based models. They are BM25 Kernel, LMIR Kernel, and KL Kernel.

3.3. BM25 Kernel

BM25 Kernel is a generalization of conventional BM25 model using String Kernel. It has a similar form as BM25, but is more general in the sense that it can make use of term dependencies in different types. It is an asymmetric kernel function.

The BM25 Kernel function between query q and document d in type t is defined as

$$\texttt{BM25-Kernel}_t(q,d) = \langle \phi_t^{\texttt{BM25}}(q), \phi_t'^{\texttt{BM25}}(d) \rangle,$$

where $\phi_t^{\texttt{BM25}}(q)$ and $\phi_t'^{\texttt{BM25}}(d)$ are two vectors of type t in the Hilbert Space mapped from q and d. Each dimension of the vector corresponds to a unit x:

$$\phi_t^{\texttt{BM25}}(q)_x = \sqrt{\texttt{IDF}_t(x)} \times \frac{(k_3+1)\times f_t(x,q)}{k_3 + f_t(x,q)}$$

and

$$\phi_t'^{\texttt{BM25}}(d)_x = \sqrt{\texttt{IDF}_t(x)} \times \frac{(k_1+1)\times f_t(x,d)}{k_1\left(1-b+b\frac{f_t(d)}{avgf_t}\right) + f_t(x,d)},$$

where k_1, k_3, and b are parameters, $f_t(x,q)$ is frequency of unit x with type t in query q , $f_t(x,d)$ is frequency of unit x with type t in document d, $f_t(d)$ is total number of units with type t in document d , and $avgf_t$ is average number of units $f_t(d)$ with type t per document within the whole collection. $\texttt{IDF}_t(x)$ is inverse document frequency of unit x with type t and is defined as

$$\texttt{IDF}_t(x) = \log \frac{df_t - df_t(x) + 0.5}{df_t(x) + 0.5},$$

where $df_t(x)$ is number of documents in which unit x with type t occurs, and df_t is total number of documents that have units with type t.

BM25 Kernel in type t then becomes

$$\texttt{BM25-Kernel}_t(q,d) = \sum_x \texttt{IDF}_t(x) \times \frac{(k_3+1)\times f_t(x,q)}{k_3 + f_t(x,q)} \times \frac{(k_1+1)\times f_t(x,d)}{k_1\left(1-b+b\frac{f_t(d)}{avgf_t}\right)+f_t(x,d)}.$$

Finally, BM25 Kernel is defined as sum of kernel functions over all the types:

$$\texttt{BM25-Kernel}(q,d) = \sum_t \texttt{BM25-Kernel}_t(q,d). \tag{2}$$

3.4. LMIR Kernel

LMIR Kernel is a generalization of LMIR and is an asymmetric kernel function. Smoothing is important for LMIR, and several methods have been proposed [35]. Among them, the Dirichlet smoothing method has been proved to be effective. In this paper, we employ Dirichlet smoothing in LMIR Kernel. Other smoothing methods can also be used.

The LMIR Kernel function between query q and document d in type t is defined as

$$\texttt{LMIR-Kernel}_t(q,d) = \langle \phi_t^{\texttt{LMIR}}(q), \phi_t'^{\texttt{LMIR}}(d)\rangle + f_t(q)\cdot \log\frac{\mu}{f_t(d)+\mu},$$

where μ is free parameter, and $\phi_t^{\texttt{LMIR}}(q)$ and $\phi_t'^{\texttt{LMIR}}(d)$ are two vectors of type t in the Hilbert Space mapped from query q and document d, respectively. Each dimension of the vector corresponds to a unit x:

$$\phi_t^{\texttt{LMIR}}(q)_x = f_t(x,q)$$

and

$$\phi_t'^{\texttt{LMIR}}(d)_x = \log\left(1+\frac{f_t(x,d)}{\mu P(x|t)}\right),$$

where $P(x|t)$ is probability of unit x with type t in the whole collection. $P(x|t)$ plays a similar role as inverse document frequency $\texttt{IDF}_t(x)$ in BM25 Kernel. Combining the two, the LMIR Kernel function in type t becomes

$$\texttt{LMIR-Kernel}_t(q,d) = \sum_x f_t(x,q)\cdot\log\left(1+\frac{f_t(x,d)}{\mu P(x|t)}\right) + f_t(q)\cdot\log\frac{\mu}{f_t(d)+\mu}.$$

Finally, we have

$$\texttt{LMIR-Kernel}(q,d) = \sum_t \texttt{LMIR-Kernel}_t(q,d). \tag{3}$$

3.5. KL Kernel

In this paper, we also use the KL Divergence kernel as a specific kernel-based ranking model, referred to as KL Kernel. Unlike BM25 Kernel and LMIR Kernel, KL Kernel is a *symmetric* kernel function. Another difference is that KL Kernel represents a mapping of the query and document into a Hilbert Space with an infinite number of dimensions. KL Kernel was previously used in document similarity comparison [19], and this is the first time it is used in relevance ranking.

In KL Kernel in a type t, the query and document are represented by distributions. We define a distribution $\mathbb{P}_t(q)$ of units x with type t in query q:

$$\mathbb{P}_t(q) = (P(x_1|t,q), P(x_2|t,q), \cdots, P(x_N|t,q)),$$

where $P(x_i|t,q)$ is probability of unit x_i given type t and query q, and N is size of $\mathcal{X}$. Similarly, we define a distribution $\mathbb{P}_t(d)$ of units x with type t in document d:

$$\mathbb{P}_t(d) = (P(x_1|t,d), P(x_2|t,d), \cdots, P(x_N|t,d).$$

Then, the KL Kernel function between query q and document d in type t is defined as

$$\texttt{KL-Kernel}_t(q,d) = \exp\{-D(\mathbb{P}_t(q)||\mathbb{P}_t(d)) - D(\mathbb{P}_t(d)||\mathbb{P}_t(q))\}.$$

Note that the kernel function represents dot product between two vectors in a Hilbert Space with infinite number of dimensions. (We can use Mercer's theorem to prove that it is a kernel function. [19]) $D(\mathbb{P}_t(q)||\mathbb{P}_t(d))$ is the KL divergence of $\mathbb{P}_t(d)$ from $\mathbb{P}_t(q)$:

$$D(\mathbb{P}_t(q)||\mathbb{P}_t(d)) = \sum_{i=1}^{N} P(x_i|t,q) \log \frac{P(x_i|t,q)}{P(x_i|t,d)}.$$

The KL Kernel function is defined as

$$\texttt{KL-Kernel}(q,d) = \prod_t \texttt{KL-Kernel}_t(q,d). \qquad (4)$$

According to the properties of kernel function, $\texttt{KL-Kernel}(q,d)$ is still a kernel function.

For efficiency consideration, we actually take logarithmic function of the kernel function. This will not change the ranking results.

$$\begin{aligned} \texttt{KL-Kernel}'(q,d) &= \log(\texttt{KL-Kernel}(q,d)) \\ &= \sum_t (-D(\mathbb{P}_t(q)||\mathbb{P}_t(d)) - D(\mathbb{P}_t(d)||\mathbb{P}_t(q))). \end{aligned}$$

Smoothing is also needed in estimation of the probability distributions in KL Kernel. In this paper we employ the Dirichlet smoothing method [35]:

$$P(x|t,d) = \frac{f_t(x,d) + \mu P(x|t)}{f_t(d) + \mu}.$$

4. Advantages of Kernel Approach

4.1. Representability

The kernel approach to ranking has rich representation ability. There are several orthogonal factors which one needs to consider when using term dependencies of different types. They are number of terms, order preservation, maximum number of skipping terms. Suppose that the query is 'machine learning book'. Number of terms indicates whether the term units we use consist of two terms or three terms, etc in documents ('machine learning' vs 'machine learning book'). Order preservation means we care about the order of query terms ('machine learning'

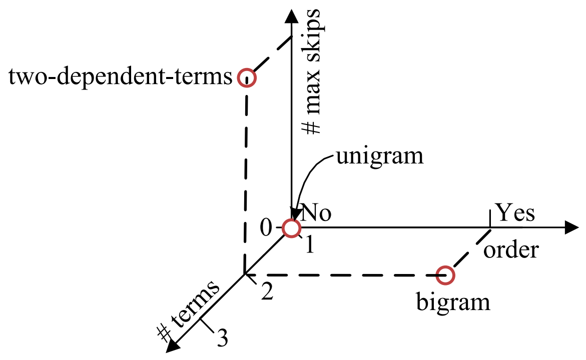

Figure 2: Three factors for term dependency.

vs 'learning machine') in documents. Maximum number of skipping terms includes the cases in which there are other terms occurring in between the terms in question ('machine book').

Combining the factors above can give us different types of dependencies, including some complex ones. The n-gram models belong to the cases in which different numbers of *ordered and non-skipping* terms are used. The n-dependent-terms models fall into the cases in which different numbers of *non-ordered and skipping* terms are used. The unigram model is the simplest. Figure 2 shows the relations among the factors. Each factor is represented as one axis; the first axis corresponds to order preservation, the second maximum number of skips, and the third number of terms. It further shows the positions of different types of dependencies.

One can immediately notice that the kernel approach proposed in this paper can easily utilize the different types of term dependencies (types) in its models. In the experiments in this paper, we make use of three types: unigram, bigram, and two-dependent-terms.

4.2. Relation with Conventional Models

It is easy to verify that BM25 Kernel and LMIR Kernel respectively include conventional BM25 and LMIR models as special cases. Specifically, when the type is unigram, the kernel functions become the existing ranking models.

The kernel approach to relevance ranking proposed in this paper, thus, provides a new interpretation of many IR ranking models. These include not only BM25 and LMIR, but also VSM and even LSI. Relevance ranking models are nothing but dot product (cosine similarity) between vectors which are nonlinearly mapped from query and document.

The kernel functions (BM25 Kernel, LMIR Kernel, and KL Kernel) have similar shapes as functions of unit frequency in document (e.g., term frequency in document for the case of unigram). (Note that unit frequency in query is usually fixed to one, because queries are short.) Figure 3 shows plots of the functions. From the figure we can see that although BM25 and LMIR were derived from different theories, they actually have similar shapes as functions of unit frequency. This can in part explain why BM25 and LMIR perform almost equally well in practice. This also strongly suggest kernel functions including KL Kernel would have similar performances in relevance ranking. As will be seen in Section 5, this is in fact true.

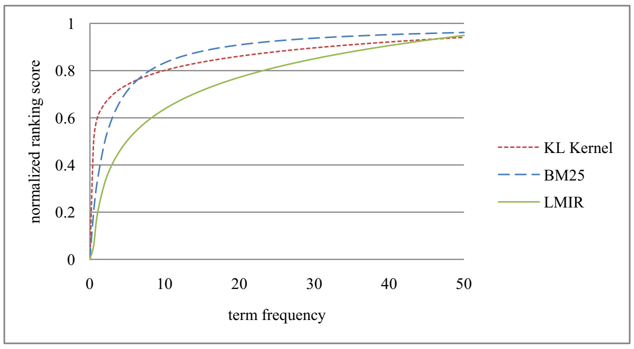

Figure 3: KL Kernel, BM25 Kernel, and LMIR Kernel as function of unit frequency (term frequency).

4.3. Theoretical Background

Another advantage of taking the kernel approach is that it has solid theoretical background. As a result, deeper understanding of the problem and broader use of the technique become possible.

For example, it is easy to verify that kernel functions have the following closure properties, which means several ways of combining kernel functions will still result in kernel functions. Let k_1 and k_2 be any two kernels. Then the function k given by

$$\begin{array}{ll} 1. & k(x,y) = k_1(y,x) \\ 2. & k(x,y) = k_1(x,y) + k_2(x,y) \\ 3. & k(x,y) = \alpha k_1(x,y), \text{ for all } \alpha > 0 \\ 4. & k(x,y) = k_1(x,y) \cdot k_2(x,y) \end{array}$$

are also kernels. Note that this is true even for *asymmetric* kernel functions. That means we can create different ranking models by taking sum or product of kernel functions.

4.4. Efficient Implementation

The kernel functions proposed above can be calculated efficiently. This is because kernel functions are essentially dot products and their computations only need to be performed on units which occur in query or document. (The values of the other units are just zero).

For example, for $D(\mathbb{P}_t(q)||\mathbb{P}_t(d))$ (with Dirichlet smoothing) in KL Kernel we actually only need to calculate

$$\begin{aligned} & \sum_{x:f_t(x,q)>0 \vee f_t(x,d)>0} P(x|t,q) \log \frac{P(x|t,q)}{P(x|t,d)} + \sum_{x:f_t(x,q)=0 \wedge f_t(x,d)=0} \frac{\mu P(x|t)}{\mu + f_t(q)} \log \frac{f_t(d)+\mu}{f_t(q)+\mu} \\ = & \sum_{x:f_t(x,q)>0 \vee f_t(x,d)>0} P(x|t,q) \log \frac{P(x|t,q)}{P(x|t,d)} + \frac{\mu}{\mu + f_t(q)} \log \frac{f_t(d)+\mu}{f_t(q)+\mu} \sum_{x:f_t(x,q)>0 \vee f_t(x,d)>0} (1 - P(x|t)) \,, \end{aligned}$$

where μ is the smoothing parameter.

5. Experiments

5.1. Experiment Setting

We conducted experiments to test the effectiveness of the proposed kernel approach, using a dataset from a web search engine and TREC data of OHSUMED [13] and AP. Specifically, we tested the performances of BM25 Kernel, LMIR Kernel, and KL Kernel. In the kernels, as types of term dependencies we used unigram, bigram, and two-dependent-terms. The final ranking score is defined as linear combination of the kernel function scores:

$$\text{Score} = (1 - \lambda_1 - \lambda_2) \cdot \text{kernel}_{unigram} + \lambda_1 \cdot \text{kernel}_{bigram} + \lambda_2 \cdot \text{kernel}_{two-dependent-terms},$$

where λ_1 and λ_2 are parameters. (The linear combination is still a kernel function, c.f., Section 4.3.)

BM25 and LMIR (with Dirichlet smoothing) were selected as baselines in the experiments, as representatives of bag-of-words models. We also viewed KL-Unigram as a baseline, in which only unigram is used in KL Kernel. MRF was also chosen as a baseline of using term dependencies. For the MRF model the same term dependency information (i.e., unigram, bigram, and two-dependent-terms) was used to ensure a fair comparison.

We adopted mean average precision (MAP) and NDCG@N ($N = 1, 3$, and 5) as evaluation measures. The web search dataset has five levels of relevance: 'Perfect', 'Excellent', 'Good', 'Fair', and 'Bad'. We considered the first three as relevant when calculating MAP. The OHSUMED dataset have three levels of relevance: 'definitely relevant', 'partially relevant', and 'not relevant'. We considered 'definitely relevant' and 'partially relevant' as relevant when calculating MAP.

5.2. Experiments on Web Search Dataset

In this experiment, we used the web search dataset. The dataset contains about 2.5 million web pages. Associated with the documents are 235 long queries randomly selected from query log. Each query has at least 8 terms. The average query length is 10.25 terms. For each query, the associated documents are labeled by several annotators. In total, there are 6,271 labeled query-document pairs. On average each query has 26.68 documents labeled.

We tested the accuracies of ranking models and obtained the results in Figure 4. We can observe an overall trend that the kernel based ranking models perform better than the bag-of-words models and the MRF model. Specifically, the kernel-based models outperform their counterparts (for example, BM25 Kernel works better than BM25, etc). There are always kernel-based models working better than MRF in different settings.

In the experiments, we tuned two parameters λ_1 and λ_2 for each kernel function. We investigated the sensitivity of the performances with respect to the two parameters. Specifically, we changed the values of a parameter and fixed the other at its optimal value (the optimal values are $\lambda_1 = 0.4$ and $\lambda_2 = 0.1$). The performances in terms of MAP with respect to parameter values are reported in Figure 5. We can see that the ranking accuracies are not very sensitive to the parameters of λ_1 and λ_2.

The KL kernel has another parameter μ. We also carried out experiments to investigate how this parameter impacts the performances. We changed the values of the parameter and observed the model's performances in terms of MAP. Figure 6 shows the performance curve. We can see that there is a flat region near the optimal value ($\mu = 4.0$), which means the model's performances are not sensitive to μ.

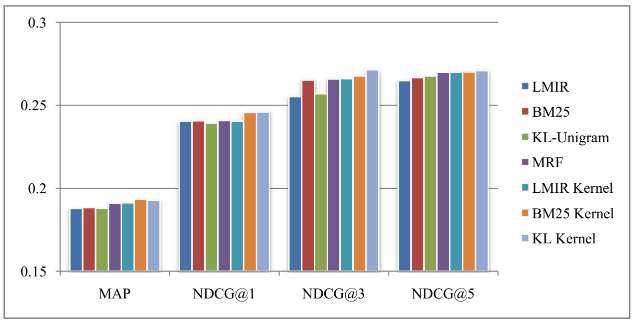

Figure 4: Ranking accuracies on web search data.

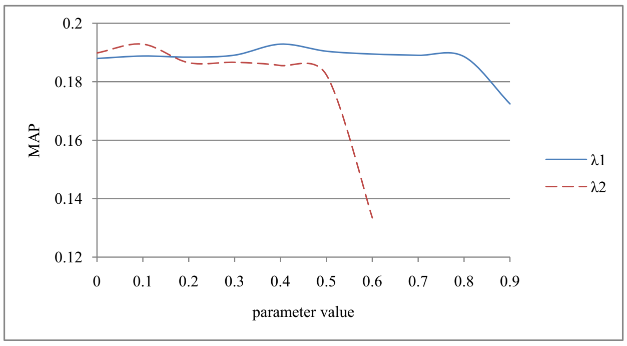

Figure 5: Ranking accuracies of KL Kernel w.r.t. λ_1 and λ_2.

5.3. Experiments on OHSUMED Dataset

In this experiment, we used the OHSUMED dataset [13] to test the performances of the ranking models. The OHSUMED dataset consists of 348,566 documents and 106 queries. There are in total 16,140 query-document pairs upon which relevance judgments are made. The relevance judgments are either ‘d’ (*definitely relevant*), ‘p’ (*possibly relevant*), or ‘n’(*not relevant*).

We used the Lemur toolkit[1] as our experiment platform. The OHSUMED dataset was indexed and queries were processed with Lemur. The experimental results are shown in Figure 7. Again, we can see that the kernel-based models work better than the bag-of-words models and MRF. However, the improvements are smaller than those on the web search data. This is probably because the query length of data is short (on average 4.96 terms per query).

5.4. Experiments on AP Dataset

We also tested the accuracies of ranking models on AP dataset of TREC ad-hoc retrieval track. The AP collection contains 158,240 articles of Associated Press in 1988 and 1990. 50 queries are selected from the TREC topics (No.51 ~ No.100). Each query has a number of

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

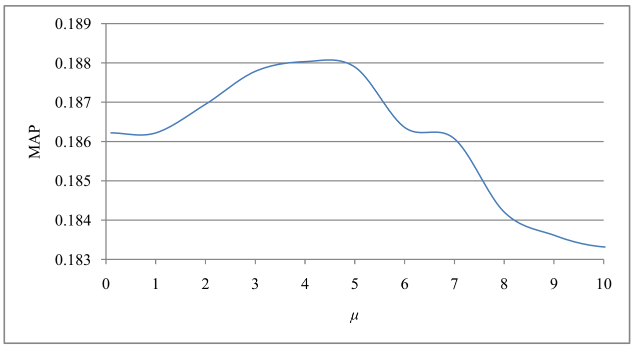

Figure 6: Ranking accuracies of KL Kernel w.r.t. μ.

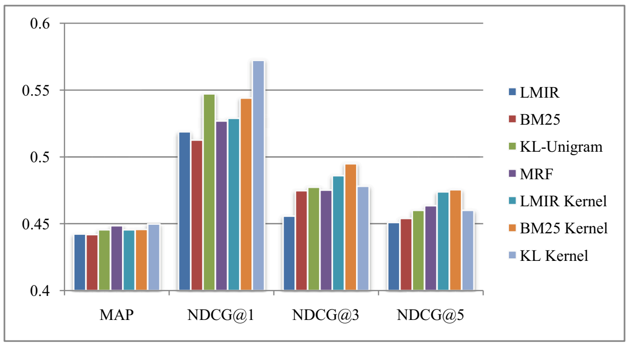

Figure 7: Ranking accuracies on OHSUMED data.

documents associated and they are labeled as ‘relevant’ or ‘irrelevant’. In total, there are 8,933 query-document pairs labeled. On average, each query has 178.66 labeled documents.

In the same way as in section 5.3, we indexed the AP datasets with the Lemur toolkit. The results are shown in Figure 8. We can draw similar conclusions as those on the OHSUMED dataset. That is, in general the kernel models perform better than the bag-of-words models and the MRF model. However, the improvements are smaller than those on the web search data. This is probably because the query length of the data is short (on average 3.22 terms per query).

5.5. Summary of Results

Table 1 and Table 2 show the ranking accuracies of the seven methods on the three datasets in terms of MAP and NDCG@3, respectively. Ranks of the seven methods based on their performances on the datasets are also given. The top ranked methods are highlighted. From the results, we can see that the kernel-based models generally work better than the baseline methods of bag-of-words models and MRF.

5.6. Discussions

We further conducted analysis on the results. Here we report them with the web search dataset as example.

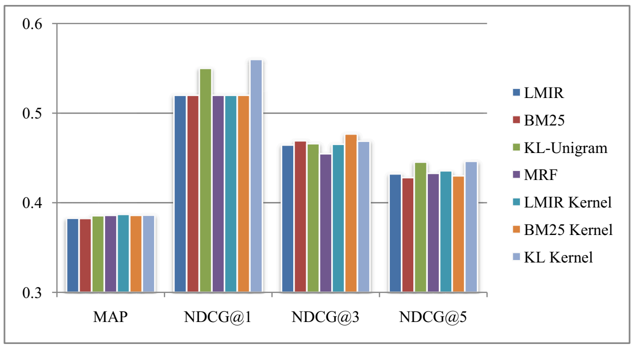

Figure 8: Ranking accuracies on AP data.

Table 1: Ranking accuracies in terms of MAP

	Search engine	OHSUMED	AP
LMIR	0.1877 (7)	0.4423 (6)	0.3827 (6)
BM25	0.1883 (5)	0.4418 (7)	0.3825 (7)
KL-Unigram	0.1880 (6)	0.4454 (5)	0.3856 (5)
MRF	0.1911 (4)	0.4486 (2)	0.3860 (3)
LMIR Kernel	0.1913 (3)	0.4455 (4)	**0.3871** (1)
BM25 Kernel	**0.1935** (1)	0.4458 (3)	0.3859 (4)
KL Kernel	0.1929 (2)	**0.4498** (1)	0.3862 (2)

First, we conducted experiments to verify how the use of term dependency information can improve ranking accuracy. Specifically, we tested the performances of BM25 Kernel (and also LMIR Kernel and KL Kernel) with unigram only, with unigram plus bigram, with unigram plus two-dependent-terms, and with unigram plus bigram and two-dependent-terms (denoted as ‘all’), in terms of MAP and NDCG. Figure 9 shows the results in terms of MAP. We can see that bigram and two-dependent-terms really help to improve the ranking accuracies in terms of MAP (we observe the same tendency for the results in terms of NDCG). If they are combined together, the ranking accuracies can be further improved. This indicates that the kernel-based model really has the ability of incorporating different types of dependency information.

Second, we conducted experiments to investigate how KL Kernel, BM25 Kernel, and LMIR Kernel can improve the ranking accuracies, especially on long queries. We calculated MAP (and NDCG) for BM25 and BM25 Kernel (also LMIR, LMIR Kernel, KL-Unigram, and KL Kernel) on each query, and grouped the queries by length. Figure 10, 11, and 12 show the results. From the results, we can see that BM25 Kernel, LMIR Kernel, and KL Kernel improve MAP scores when queries become longer. (The same tendency can be found for NDCG). This indicates that the kernel-based models are more effective when queries become longer.

In our experiments, we observed MRF model has similar performances as LMIR Kernel. Our explanation to it is that MRF is a combination of LMIR models with different types of term dependency (c.f., [18]), and thus it is likely that they perform similarly.

Table 2: Ranking accuracies in terms of NDCG@3

	Search engine	OHSUMED	AP
LMIR	0.2552 (7)	0.4558 (7)	0.4643 (6)
BM25	0.2652 (5)	0.4748 (6)	0.4691 (2)
KL-Unigram	0.2570 (6)	0.4774 (4)	0.4660 (4)
MRF	0.2659 (4)	0.4752 (5)	0.4547 (7)
LMIR Kernel	0.2661 (3)	0.4859 (2)	0.4653 (5)
BM25 Kernel	0.2676 (2)	**0.4949** (1)	**0.4767** (1)
KL Kernel	**0.2715** (1)	0.4780 (3)	0.4688 (3)

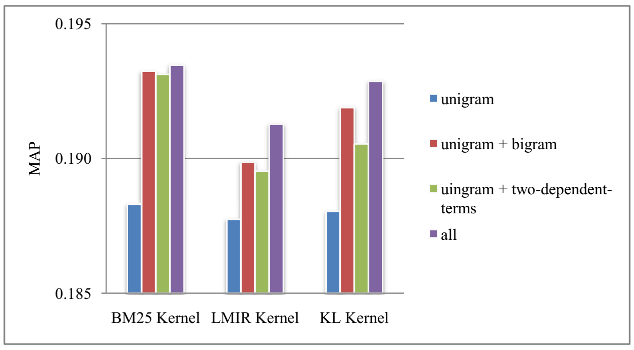

Figure 9: Ranking accuracies of BM25 Kernel, LMIR Kernel, and KL Kernel with different term dependencies.

6. Conclusion

In this paper, we have studied relevance ranking models, particularly, those using term dependencies. Our work is new and unique in that we employ the kernel technique in statistical learning in the analysis and construction of ranking models. We have defined three kernel based ranking models: BM25 Kernel, LMIR Kernel, and KL Kernel. The former two are generalization of BM25 and LMIR, and the latter is a new model for relevance ranking.

The general ranking model employed in the our approach has (1) rich representability for relevance ranking particularly for that using term dependencies (different types of term dependencies are summarized in the paper, and the kernel based ranking model can naturally represent them.), (2) strong connections with existing models, (3) solid theoretical background, and (4) efficient calculation.

Experimental results on web search data and TREC data show that (1) the kernel based ranking models almost always outperform the conventional bag-of-words models and the MRF model using the same information, (2) the kernel functions can really effectively incorporate different types of term dependency information (from unigram to bigram and two-dependent-terms), (3) the kernel functions work particularly well for long queries.

There are several issues which need further investigations as future work. (1) We used three types of term dependencies and three datasets to conduct experiments in this paper. We plan to use more types and more data to verify the effectiveness of the proposed approach. (2) The kernel based ranking models do not contain span-based models or other approximate match models (cf.,

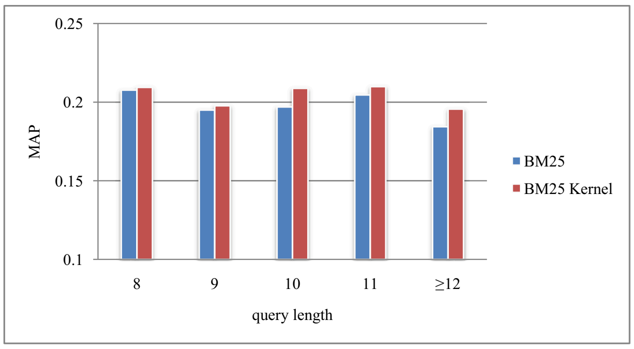

Figure 10: Ranking accuracies of BM25 and BM25 Kernel by query lengths.

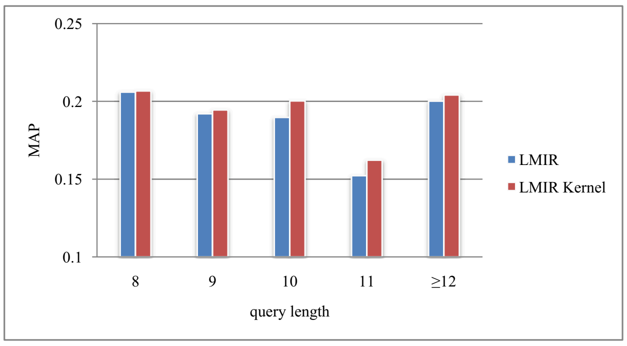

Figure 11: Ranking accuracies of LMIR and LMIR Kernel by query lengths.

Section 2). The relationship between them needs more studies. (3) Kernel functions can be used as features in learning to rank. A future step would be to combine the two together, that is, to learn the ranking model as well as the kernel functions at the same time.

References

[1] Bai, J., Chang, Y., Cui, H., Zheng, Z., Sun, G., Li, X., 2008. Investigation of partial query proximity in web search. In: Proceeding of the 17th international conference on World Wide Web. pp. 1183–1184.

[2] Beigbeder, M., Mercier, A., 2005. An information retrieval model using the fuzzy proximity degree of term occurences. In: Proceedings of the 2005 ACM symposium on Applied computing. pp. 1018–1022.

[3] Broschart, A., Schenkel, R., 2008. Proximity-aware scoring for xml retrieval. In: Proceedings of the 31st annual international ACM SIGIR conference. pp. 845–846.

[4] Buttcher, S., Clarke, C. L. A., 2005. Effciency vs. effectiveness in terabyte-scale information retrieval. In: Proceedings of the 14th Text REtrieval Conference.

[5] Büttcher, S., Clarke, C. L. A., Lushman, B., 2006. Term proximity scoring for ad-hoc retrieval on very large text collections. In: Proceedings of the 29th annual international ACM SIGIR conference. pp. 621–622.

[6] Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y., Hon, H.-W., 2006. Adapting ranking SVM to document retrieval. In: Proceedings of the 29th annual international ACM SIGIR conference. pp. 186–193.

[7] Clarke, C. L. A., Cormack, G. V., 2000. Shortest-substring retrieval and ranking. ACM Trans. Inf. Syst. 18 (1), 44–78.

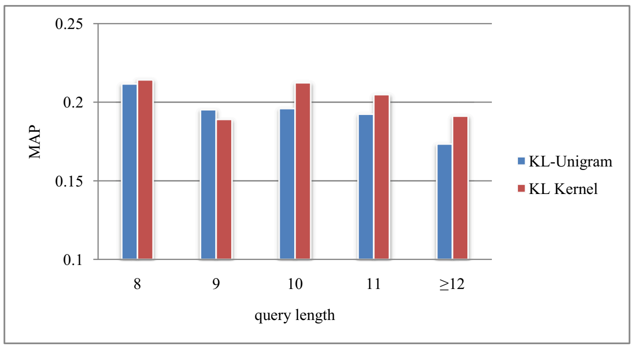

Figure 12: Ranking accuracies of KL-Unigram and KL Kernel by query lengths.

[8] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R., 1990. Indexing by latent semantic analysis. Journal of the American Society for Information Science 41, 391–407.

[9] Gao, J., Nie, J.-Y., Wu, G., Cao, G., 2004. Dependence language model for information retrieval. In: Proceedings of the 27th annual international ACM SIGIR conference. pp. 170–177.

[10] Haussler, D., 1999. Convolution kernels on discrete structures. Tech. rep., Dept. of Computer Science, University of California at Santa Cruz.

[11] Hawking, D., Thistlewaite, P. B., 1995. Proximity operators - so near and yet so far. In: Proceedings of the 4th Text Retrieval Conference.

[12] Herbrich, R., Graepel, T., Obermayer, K., 2000. Large Margin rank boundaries for ordinal regression. MIT Press, Cambridge, MA.

[13] Hersh, W., Buckley, C., Leone, T. J., Hickam, D., 1994. Ohsumed: an interactive retrieval evaluation and new large test collection for research. In: Proceedings of the 17th annual international ACM SIGIR conference. pp. 192–201.

[14] Keen, E. M., 1991. The use of term position devices in ranked output experiments. Journal of Documentation 47 (1), 1–22.

[15] Keen, E. M., 1992. Some aspects of proximity searching in text retrieval systems. J. Inf. Sci. 18 (2), 89–98.

[16] Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C., 2002. Text classification using string kernels. J. Mach. Learn. Res. 2, 419–444.

[17] Martins, A., 2006. String kernels and similarity measures for information retrieval. Tech. rep., Priberam, Lisbon, Portugal.

[18] Metzler, D., Croft, W. B., 2005. A markov random field model for term dependencies. In: Proceedings of the 28th annual international ACM SIGIR conference. pp. 472–479.

[19] Moreno, P. J., Ho, P. P., Vasconcelos, N., 2003. A kullback-leibler divergence based kernel for svm classification in multimedia applications. In: Advances in Neural Information Processing Systems 16. MIT Press.

[20] Na, S.-H., Kim, J., Kang, I.-S., Lee, J.-H., 2008. Exploiting proximity feature in bigram language model for information retrieval. In: Proceedings of the 31st annual international ACM SIGIR conference. pp. 821–822.

[21] Ponte, J. M., Croft, W. B., 1998. A language modeling approach to information retrieval. In: Proceedings of the 21st annual international ACM SIGIR conference. pp. 275–281.

[22] Rasolofo, Y., Savoy, J., 2003. Term proximity scoring for keyword-based retrieval systems. In: Proceedings 25th European Conference on IR Research. pp. 207–218.

[23] Rijsbergen, C. V., 1977. A theoretical basis for the use of co-occurrence data in information retrieval. Journal of Documentation 33 (2), 106 – 119.

[24] Robertson, S., Zaragoza, H., Taylor, M., 2004. Simple bm25 extension to multiple weighted fields. In: Proceedings of the thirteenth ACM international conference on Information and knowledge management. pp. 42–49.

[25] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M., Gatford, M., 1994. Okapi at trec-3. In: Proceedings of the 3rd Text REtrieval Conference.

[26] Sahami, M., Heilman, T. D., 2006. A web-based kernel function for measuring the similarity of short text snippets. In: Proceedings of the 15th international conference on World Wide Web. pp. 377–386.

[27] Salton, G., Wong, A., Yang, C. S., 1975. A vector space model for automatic indexing. Commun. ACM 18 (11), 613–620.

[28] Scholkopf, B., Mika, S., Burges, C. J. C., robert Mller, K., Ratsch, G., Smola, A. J., 1999. Input space versus

feature space in kernel-based methods. IEEE Transactions on Neural Networks 10, 1000–1017.

[29] Shawe-Taylor, J., Cristianini, N., 2004. Kernel Methods for Pattern Analysis. Cambridge University Press, New York, NY, USA.

[30] Song, F., Croft, W. B., 1999. A general language model for information retrieval. In: Proceedings of the eighth international conference on Information and knowledge management. pp. 316–321.

[31] Tao, T., Zhai, C., 2007. An exploration of proximity measures in information retrieval. In: Proceedings of the 30th annual international ACM SIGIR conference. pp. 295–302.

[32] Tsuda, K., 1999. Support vector classifier with asymmetric kernel functions. In: European Symposium on Artificial Neural Networks. pp. 183–188.

[33] Xie, Y., Raghavan, V. V., 2007. Language-modeling kernel based approach for information retrieval. J. Am. Soc. Inf. Sci. Technol. 58 (14), 2353–2365.

[34] Xu, J., Li, H., 2007. Adarank: a boosting algorithm for information retrieval. In: Proceedings of the 30th annual international ACM SIGIR conference. pp. 391–398.

[35] Zhai, C., Lafferty, J., 2004. A study of smoothing methods for language models applied to information retrieval. ACM Trans. Inf. Syst. 22 (2), 179–214.