Journal of Machine Learning Research 12 (2011) 1429-1458 Submitted 6/10; Revised 12/10; Published 5/11

Learning a Robust Relevance Model for Search Using Kernel Methods

Wei Wu[*] V-WEW@MICROSOFT.COM
MOE -Microsoft Key Laboratory of Statistics and Information Technology
Department of Probability and Statistics
Peking University
No.5 Yiheyuan Road, Haidian District, Beijing, 100871, P. R. China

Jun Xu JUNXU@MICROSOFT.COM
Hang Li HANGLI@MICROSOFT.COM
Microsoft Research Asia
13F Building 2
No. 5 Danling Street, Haidian District, Beijing, 100080, P.R. China

Satoshi Oyama [*] OYAMA@IST.HOKUDAI.AC.JP
Graduate School of Information Science and Technology
Hokkaido University
Kita 14, Nishi 9, Kita-ku, 060-0814, Japan

Editor: Corinna Cortes

Abstract

This paper points out that many search relevance models in information retrieval, such as the Vector Space Model, BM25 and Language Models for Information Retrieval, can be viewed as a similarity function between pairs of objects of different types, referred to as an S-function. An S-function is specifically defined as the dot product between the images of two objects in a Hilbert space mapped from two different input spaces. One advantage of taking this view is that one can take a unified and principled approach to address the issues with regard to search relevance. The paper then proposes employing a kernel method to learn a robust relevance model as an S-function, which can effectively deal with the term mismatch problem, one of the biggest challenges in search. The kernel method exploits a positive semi-definite kernel referred to as an S-kernel. The paper shows that when using an S-kernel the model learned by the kernel method is guaranteed to be an S-function. The paper then gives more general principles for constructing S-kernels. A specific implementation of the kernel method is proposed using the Ranking SVM techniques and click-through data. The proposed approach is employed to learn a relevance model as an extension of BM25, referred to as Robust BM25. Experimental results on web search and enterprise search data show that Robust BM25 significantly outperforms baseline methods and can successfully tackle the term mismatch problem.

Keywords: search, term mismatch, kernel machines, similarity learning, s-function, s-kernel

1. Introduction

There are many applications such as search, collaborative filtering, and image annotation, that can be viewed as a task employing a similarity function defined on pairs of instances from two different spaces. For example, search is a task as follows. Given a query, the system retrieves documents

*. This work was conducted when the first and fourth authors visited Microsoft Research Asia.

relevant to the query and ranks the documents based on the degree of relevance. The relevance of a document with respect to a query can be viewed as a kind of similarity, and the search task is essentially one based on a similarity function between query and document pairs, where query and document are instances from two spaces: query space and document space.

In this paper, we formally define the similarity function as the dot product of the images of two objects in a Hilbert space mapped from two different spaces. For simplicity, we call the similarity function S-function. In fact, the state-of-the-art relevance models in information retrieval (IR), such as the Vector Space Model (VSM) (Salton and McGill, 1986), BM25 (Robertson et al., 1994) and Language Models for Information Retrieval (LMIR) (Ponte and Croft, 1998; Zhai and Lafferty, 2004), are all S-functions. We prove some properties of the S-function and show that it becomes a positive semi-definite kernel under certain conditions. One advantage of taking this view to search is that it provides us with a unified and principled approach to using and learning relevance models.

In this paper, we focus on the learning of a robust relevance model as an S-function, to deal with term mismatch, one of the critical challenges for search. We show that we can define a new type of positive semi-definite kernel function called S-kernel and learn a robust relevance model using a kernel method based on S-kernel. Recently, the learning of similarity function has emerged as a hot research topic in machine learning (cf., Abernethy et al., 2009; Grangier and Bengio, 2008). Our work is novel and unique in that it learns a similarity function for search using a kernel method .

The conventional relevance models are all based on term matching. That is, they look at the matched words in a query and document, and calculate the similarity (relevance) based on the degree of matching. A good match at term level does not necessarily mean high relevance, however, and vice versa. For example, if the query is "NY" and the document only contains "New York", then the BM25 score of the query and document pair will be low (i.e., the two will be viewed less relevant), although the query and document are relevant. Similar problems occur with LMIR and other relevance models. This is the so-called term mismatch problem, which all existing relevance models suffer from. In other words, the scores from the relevance models may not be reliable and the question of how to learn a more robust similarity function for search arises; this is exactly the problem we want to address in this paper.

In this paper, we tackle the term mismatch problem with a kernel method based on the notion of S-function. Intuitively, we calculate a more reliable score between a query document pair by using the scores between the pairs of similar query and similar document. Our kernel method exploits a special positive semi-definite kernel, referred to as S-kernel, defined based upon the S-function.

An S-kernel is formally defined as a positive semi-definite kernel such that the reproducing kernel Hilbert space (RKHS) generated by the kernel is also a space of S-functions. Therefore, the model learned by a kernel method is guaranteed to be an S-function. We further give general principles for constructing S-kernels, and thus offer a formulation for learning similarity functions with S-kernels. An S-kernel can be viewed as an extension of the hyper kernel proposed by Ong et al. (2005).

We provide a method for implementing the kernel method using the Ranking SVM techniques and click-through data. The method is used to train a relevance model named 'Robust BM25' to deal with term mismatch, as an extension of BM25. The learned Robust BM25 model determines the relevance score of a query document pair on the basis of not only the BM25 score of the query document pair, but also the BM25 scores of similar query and similar document pairs. All calculations are naturally incorporated in the kernel method. Experimental results on two large scale data sets show that Robust BM25 can indeed solve term mismatch and significantly outperform the baselines.

This paper has the following contributions: 1) proposal of a kernel method for dealing with term mismatch in search, 2) proposal of a unified view to search using S-function, 3) proposal of a family of kernel functions, S-kernel.

The rest of the paper is organized as follows. A survey of related work is conducted in Section 2, and then the definition of S-function and interpretation of traditional relevance models as S-functions are given in Section 3. Section 4 first introduces the term mismatch problem in search, and then proposes a kernel method for learning a robust relevance model to deal with the problem, such as Robust BM25. Section 5 defines S-kernel and proposes learning a similarity function with S-kernel. Section 6 describes how to implement the learning Robust BM25 method. Section 7 reports experimental results and Section 8 concludes this paper.

2. Related Work

Kernel methods, including the famous Support Vector Machines (SVM) (Vapnik, 1995), refer to a class of algorithms in machine learning which can be employed in a variety of tasks such as classification, regression, ranking, correlation analysis, and principle component analysis (Hofmann et al., 2008; Schölkopf and Smola, 2002). Kernel methods make use of kernel functions which map a pair of data in the input space (Euclidean space or discrete set) into the feature space (Hilbert space) and compute the dot product between the images in the feature space. Many kernels have been proposed for different applications (Zhou, 2004; Vishwanathan and Smola, 2004; Haussler, 1999; Watkins, 1999; Gartner et al., 2003; Kashima et al., 2004). Conventional kernels are defined over one single input space and are symmetric and positive semi-definite. The kernel function is called Mercer kernel when it is continuous. The similarity function, S-function, which we define in this paper, is related to the conventional kernel function. An S-function is defined as the dot product in a Hilbert space between the images of inputs from two spaces, and a conventional kernel function is defined as the dot product in a Hilbert space between the images of inputs from the same input space. If the two spaces in an S-function are the same, the S-function becomes a kernel function.

Koide and Yamashita (2006) defined a similarity function called asymmetric kernel and applied it to Fisher's linear discriminant. The asymmetric kernel defined by Koide and Yamashita (2006) is similar to S-function. We use the term S-function instead of asymmetric kernel in this paper, because further investigation of the properties of S-function (or asymmetric kernel), particularly the necessary and sufficient condition, is still necessary.

The learning of a similarity function between pairs of objects has been studied. When the pair of objects are from the same space, the similarity function becomes a positive semi-definite kernel; a typical approach is kernel learning (cf., Lanckriet et al., 2002; Bach et al., 2004; Ong et al., 2005; Micchelli and Pontil, 2005; Bach, 2008; Cortes, 2009; Varma and Babu, 2009). Lanckriet et al. (2002) as well as Bach et al. (2004) have proposed methods for multiple kernel learning, in which the optimal kernel (similarity function) is selected from a class of linear combinations of kernels. Besides this, Ong et al. (2005) have proposed learning a kernel function (similarity function) by using kernel methods, in which the optimal kernel is chosen from RKHS generated by the 'hyperkernel'. Our method can be viewed as an extension of Ong et al.'s method. Recently, the learning of a similarity function between pairs of objects from two different spaces has also emerged as a hot research topic (cf., Abernethy et al., 2009; Grangier and Bengio, 2008). In this paper, we propose a kernel approach for performing the learning task.

Term mismatch is one of the major challenges for search, because most of the traditional relevance models, including VSM (Salton and McGill, 1986), BM25 (Robertson et al., 1994), and LMIR (Ponte and Croft, 1998; Zhai and Lafferty, 2004), are based on term matching and the ranking result will be inaccurate when term mismatch occurs. To solve the problem, heuristic methods of query expansion or (pseudo) relevance feedback (cf., Salton and Buckley, 1997; Xu and Croft, 1996; Salton and McGill, 1986; Baeza-Yates and Ribeiro-Neto, 1999; Mitra et al., 1998; Broder et al., 2009; Zhuang and Cucerzan, 2006) and Latent Semantic Indexing (LSI) (Deerwester et al., 1990) or Probabilistic Latent Semantic Indexing (PLSI) (Hofmann, 1999) have been proposed and certain improvements have been made. The former approach tackles the problem at the term level and the latter at the topic level. In this paper, we demonstrate that we can learn a relevance model Robust BM25 to address the term mismatch challenge at the term level. The learned Robust BM25 is also an S-function.

Click-through data, which records the URLs clicked by users after their query submissions at a search engine, has been widely used in web search (Agichtein et al., 2006; Joachims, 2002; Craswell and Szummer, 2007). For example, click-through data has been used in the training of a Ranking SVM model, in which preference pairs on documents given queries are derived from click-through data (Joachims, 2002). Click-through data has also been used for calculating query similarity, because queries which link to the same URLs in click-through data may represent the same search intent (Beeferman and Berger, 2000; Cui et al., 2003; Wen et al., 2002). In this paper, we use click-through data for training a Robust BM25 as well as calculating query similarity.

Learning to rank refers to supervised learning techniques for constructing ranking models using training data (cf., Liu, 2009). Several approaches to learning to rank have been proposed and it has become one of the important technologies in the development of modern search engines (e.g., Herbrich et al., 1999; Joachims, 2002; Crammer and Singer, 2001; Agarwal and Niyogi, 2005; Freund et al., 2003; Rudin et al., 2005; Burges et al., 2006; Cao et al., 2006; Xu and Li, 2007; Cao et al., 2007). The method for learning Robust BM25 in this paper can also be viewed as a learning to rank method. Robust BM25 runs on the top of conventional learning to rank methods. Specifically, it trains a 're-ranking' model online to deal with term mismatch, while conventional learning to rank methods train a ranking model offline for basic ranking. The method of learning Robust BM25 is similar to Ranking SVM proposed by Herbrich et al. (1999) and Joachims (2002), a popular learning to rank algorithm. However, there are some differences. For example, the Robust BM25 method uses a different kernel function.

3. Similarity Function

This section describes the definition and properties of S-function, the similarity function between pairs of objects of different types.

3.1 Definition

An S-function measures the similarity between two objects from two different spaces. It is in fact the dot product between the images in the feature space mapped from two objects in the two input spaces.

Definition 1 (S-function) *Let $\mathcal{X}$ and $\mathcal{Y}$ be two input spaces, and $\mathcal{H}$ be a feature space (Hilbert space). S-function is a function $k : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, satisfying $k(x,y) = \langle \varphi_X(x), \varphi_Y(y) \rangle_{\mathcal{H}}$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, where φ_X and φ_Y are mapping functions from $\mathcal{X}$ and $\mathcal{Y}$ to $\mathcal{H}$, respectively.*

A positive semi-definite kernel is defined as a function $K(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which satisfies that there is a mapping $\phi(\cdot)$ from $\mathcal{X}$ to a Hilbert space $\mathcal{H}$ with inner product $< \cdot,\cdot >_{\mathcal{H}}$, such that $\forall x, x' \in \mathcal{X}$, $K(x,x') =< \phi(x), \phi(x') >_{\mathcal{H}}$. A positive semi-definite kernel measures the similarity of pairs of objects in a single space by using the dot product of their images in a Hilbert space. In contrast, S-function measures the similarity between pairs of objects in two different spaces. If the two input spaces (also the two mapping functions) are identical in Definition 1, then S-function becomes a positive semi-definite kernel. Moreover, S-function also has some properties similar to those of positive semi-definite kernels, as shown below.

3.2 Properties

S-function has properties as shown below; they are similar to those in conventional positive semi-definite kernels, but there are also differences. Note that for a conventional kernel, α must be non-negative in property (1) of Lemma 2. The properties will enable us to construct more complicated S-functions from simple S-functions.

Lemma 2 (Properties of S-function) *Let $k_1(x,y)$ and $k_2(x,y)$ be S-functions on $\mathcal{X} \times \mathcal{Y}$, then the following functions $k : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ are also S-functions: (1) $\alpha \cdot k_1$ (for all $\alpha \in \mathbb{R}$), (2) $k_1 + k_2$, (3) $k_1 \cdot k_2$.*

Proof Since $k_1(x,y)$ and $k_2(x,y)$ are S-functions, suppose that $k_1(x,y) = \langle \varphi_X^1(x), \varphi_Y^1(y) \rangle_1$ and $k_2(x,y) = \langle \varphi_X^2(x), \varphi_Y^2(y) \rangle_2$, where $\langle \cdot,\cdot \rangle_1$ is the dot product in N_1-dimensional Hilbert space and $\langle \cdot,\cdot \rangle_2$ is the dot product in N_2-dimensional Hilbert space. N_1 and N_2 can be finite or infinite.

Let $\varphi_{Xi}^1(\cdot)$ and $\varphi_{Yi}^1(\cdot)$ be the i^{th} elements of vectors $\varphi_X^1(\cdot)$ and $\varphi_Y^1(\cdot)$, respectively ($i = 1, 2, \ldots, N_1$), and $\varphi_{Xi}^2(\cdot)$ and $\varphi_{Yi}^2(\cdot)$ be the i^{th} elements of vectors $\varphi_X^2(\cdot)$ and $\varphi_Y^2(\cdot)$, respectively ($i = 1, 2, \ldots, N_2$).

(1) Let ${\varphi_X^1}'(x) = \alpha \cdot \varphi_X^1(x)$, we obtain $\alpha \cdot k_1(x,y) = \langle {\varphi_X^1}'(x), \varphi_Y^1(y) \rangle_1$, which proves that $\alpha \cdot k_1$ is an S-function, $\forall \alpha \in \mathbb{R}$.

(2) Let $\varphi_X(x) = (\varphi_X^1(x), \varphi_X^2(x))$, and $\varphi_Y(y) = (\varphi_Y^1(y), \varphi_Y^2(y))$, we obtain $\langle \varphi_X(x), \varphi_Y(y) \rangle = \langle \varphi_X^1(x), \varphi_Y^1(y) \rangle_1 + \langle \varphi_X^2(x), \varphi_Y^2(y) \rangle_2 = k_1(x,y) + k_2(x,y)$, which proves that $k_1 + k_2$ is an S-function.

(3) Let $\varphi_X(x) = \varphi_X^1(x) \otimes \varphi_X^2(x)$ and $\varphi_Y(y) = \varphi_Y^1(y) \otimes \varphi_Y^2(y)$. $\varphi_X(x)$ is a vector whose elements are $\{\varphi_{Xi}^1(x)\varphi_{Xj}^2(x)\}$, $1 \leqslant i \leqslant N_1$, $1 \leqslant j \leqslant N_2$ and $\varphi_Y(y)$ is a vector whose elements are $\{\varphi_{Yi}^1(y)\varphi_{Yj}^2(y)\}$, $1 \leqslant i \leqslant N_1$, $1 \leqslant j \leqslant N_2$. We obtain

$$\begin{aligned}
\langle \varphi_X(x), \varphi_Y(y) \rangle &= \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \varphi_{Xi}^1(x)\varphi_{Xj}^2(x)\varphi_{Yi}^1(y)\varphi_{Yj}^2(y) \\
&= \sum_{i=1}^{N_1} \varphi_{Xi}^1(x)\varphi_{Yi}^1(y) \sum_{j=1}^{N_2} \varphi_{Xj}^2(x)\varphi_{Yj}^2(y) \\
&= \sum_{i=1}^{N_1} \varphi_{Xi}^1(x)\varphi_{Yi}^1(y) k_2(x,y) \\
&= k_1(x,y)k_2(x,y),
\end{aligned}$$

which proves that $k_1 \cdot k_2$ is an S-function.

■

3.3 Relevance Models as Similarity Functions

Traditional relevance models, including VSM (Salton and McGill, 1986), BM25 (Robertson et al., 1994) and LMIR (Ponte and Croft, 1998; Zhai and Lafferty, 2004), can be viewed as S-functions.[1] In fact, all these models measure the similarity of a query and document from query space and document space. In VSM, query space and document space are treated as the same space, while in the other two models, query space and document space are two different spaces.

3.3.1 VSM

Let $\mathcal{Q}$ and $\mathcal{D}$ denote query and document spaces. Each dimension in the two spaces corresponds to a term, and query and document are respectively represented as vectors in the two spaces. Let $\mathcal{H}$ denote a Hilbert space endowed with dot product $\langle \cdot, \cdot \rangle$ (it is in fact an n-dimensional Euclidean space where n is the number of unique terms).

Given query $q \in \mathcal{Q}$ and document $d \in \mathcal{D}$, VSM is calculated as

$$\text{VSM}(q,d) = \langle \varphi_Q^{\text{VSM}}(q), \varphi_D^{\text{VSM}}(d) \rangle,$$

where $\varphi_Q^{\text{VSM}}(q)$ and $\varphi_D^{\text{VSM}}(d)$ are mappings to $\mathcal{H}$ from $\mathcal{Q}$ and $\mathcal{D}$, respectively.

$$\varphi_Q^{\text{VSM}}(q)_t = idf(t) \cdot tf(t,q)$$

and

$$\varphi_D^{\text{VSM}}(d)_t = idf(t) \cdot tf(t,d),$$

where t is a term, $tf(t,q)$ is the frequency of term t in query q, $tf(t,d)$ is the frequency of term t in document d, $idf(t)$ is the inverse document frequency of term t. That is to say, VSM is a linear positive semi-definite kernel, and is an S-function as well.

3.3.2 BM25

Given query $q \in \mathcal{Q}$ and document $d \in \mathcal{D}$, BM25 is calculated as

$$\text{BM25}(q,d) = \langle \varphi_Q^{\text{BM25}}(q), \varphi_D^{\text{BM25}}(d) \rangle, \tag{1}$$

where $\varphi_Q^{\text{BM25}}(q)$ and $\varphi_D^{\text{BM25}}(d)$ are mappings to $\mathcal{H}$ from $\mathcal{Q}$ and $\mathcal{D}$, respectively.

$$\varphi_Q^{\text{BM25}}(q)_t = \frac{(k_3+1) \times tf(t,q)}{k_3 + tf(t,q)}$$

and

$$\varphi_D^{\text{BM25}}(d)_t = idf(t) \frac{(k_1+1) \times tf(t,d)}{k_1 \left(1 - b + b \cdot \frac{\text{len}(d)}{\text{avgDocLen}}\right) + tf(t,d)},$$

where $k_1 \geq 0$, $k_3 \geq 0$, and $b \geq 0$ are parameters. Moreover, $\text{len}(d)$ is the length of document d and avgDocLen is the average length of documents in the collection.

1. “ The matching function between a query and document should be defined as an asymmetric function.” - Stephen Robertson, personal communication.

3.3.3 LMIR

We use Dirichlet smoothing as an example. Other smoothing methods such as Jelinek-Mercer (JM) can also be used. Given query $q \in \mathcal{Q}$ and document $d \in \mathcal{D}$, the LMIR with Dirichlet smoothing is calculated as

$$\text{LMIR}(q,d) = \langle \varphi_Q^{\text{LMIR}}(q), \varphi_D^{\text{LMIR}}(d) \rangle,$$

where $\phi_Q^{\text{LMIR}}(q)$ and $\phi_D^{\text{LMIR}}(d)$ are $(n+1)$-dimensional mappings to $\mathcal{H}$ from $\mathcal{Q}$ and $\mathcal{D}$, respectively. For $t = 1,2,\ldots,n$, $\phi_Q^{\text{LMIR}}(q)_t$ and $\phi_D^{\text{LMIR}}(d)_t$ are defined as

$$\varphi_Q^{\text{LMIR}}(q)_t = tf(t,q)$$

and

$$\varphi_D^{\text{LMIR}}(d)_t = \log\left(1 + \frac{tf(t,d)}{\mu P(t)}\right),$$

where $\mu > 0$ is a smoothing parameter, $P(t)$ is the probability of term t in the whole collection. $P(t)$ plays a similar role as inverse document frequency $idf(t)$ in VSM and BM25. The $(n+1)^{th}$ entries of $\varphi_Q^{\text{LMIR}}(q)$ and $\varphi_D^{\text{LMIR}}(d)$ are defined as

$$\varphi_Q^{\text{LMIR}}(q)_{n+1} = \text{len}(q)$$

and

$$\varphi_D^{\text{LMIR}}(d)_{n+1} = \log\frac{\mu}{\text{len}(d)+\mu},$$

where $\text{len}(q)$ and $\text{len}(d)$ are the lengths of query q and document d, respectively.

There are several advantages of applying the similarity function view to search. First, it gives a general and unified framework to relevance models. Although BM25 and LMIR are derived from different probability models, they work equally well in practice. It was difficult to understand the phenomenon. The S-function interpretation of the relevance models can give a better explanation of it. BM25 and LMIR are nothing but similarity functions representing query and document matching with different formulations. Second, it is easy to make an extension of the conventional relevance models based on the S-function definition. In (Xu et al., 2010), we show that the conventional relevance models can be naturally extended from unigram based models to n-gram based models to improve search relevance, with the S-function interpretation. In this paper, we demonstrate that we can deal with the term mismatch problem in a principled way on the basis of S-function.

An S-function measures the similarity of pairs of objects from two different spaces. It is an essential model not only for search, but also for many other applications such as collaborative filtering (Abernethy et al., 2009) and image retrieval (Grangier and Bengio, 2008). In the tasks, there exist two spaces and given an object in one space the goal is to find the most similar (relevant) objects in the other space. The spaces are defined over query and document, user and item, and image and text, respectively. In all these problems, the model can be represented as an S-function.

4. Learning a Robust Relevance Model

In this section, we first describe term mismatch, then propose using Robust BM25 to deal with term mismatch, and finally propose employing a kernel method to learn Robust BM25.

yutube	yuotube	yuo tube
ytube	youtubr	yu tube
youtubo	youtuber	youtubecom
youtube om	youtube music videos	youtube videos
youtube	youtube com	youtube co
youtub com	you tube music videos	yout tube
youtub	you tube com yourtube	your tube
you tube	you tub	you tube video clips
you tube videos	www you tube com	wwww youtube com
www youtube	www youtube com	www youtube co
yotube	www you tube	www utube com
ww youtube com	www utube	www u tube
utube videos	our tube	utube
u tube	my tube	toutube

Table 1: Example queries representing search intent “finding YouTube website”.

4.1 Term Mismatch in Search

Search is basically based on term match. For example, if the query is “soccer” and the term “soccer” occurs several times in the document, then the document is regarded as ‘relevant’. The relevance models of VSM, BM25 and LMIR will give high scores to the document and the document will be ranked highly. This term matching paradigm works quite well. However, the so-called term mismatch problem also inevitably occurs. That is, even if the document and the query are relevant, but they do not match at term level, in other words, they do not share a term, then they will not be viewed as relevant. For example, if the query is “New York” and the document contains “NY”, then the document will not be regarded relevant. Similarly, “aircraft” and “airplane” refer to the same concept; but if one of them is used in the query and the other in the document, then the document will be considered irrelevant. Term mismatch due to the differences in expressions including typos, acronyms, and synonyms can easily happen and deteriorate the performance of search.

In web search, users are more diverse and so are the web contents. The term mismatch problem becomes more severe than traditional search. Although modern search engines exploit more sophisticated models for retrieval and ranking, they still heavily rely on the term matching paradigm. Therefore, term mismatch is still one of the most critical challenges for web search. For example, we have observed over 200 different forms for representing the same search intent “finding YouTube website” from the query log of a commercial web search engine. Table 1 lists some examples.

The relevance models of VSM, BM25, LMIR are all based on term frequencies of $tf(t,d)$ and $tf(t,q)$. If the query and document share a term t, then the term frequencies will be non-zero values and the relevance score between the query and document will become high. That is, the value of the S-function between the query and document will be large. When term mismatch occurs, either $tf(t,d)$ or $tf(t,q)$ will be zero, and the relevance score will be low, although it should not be so. In that case, the value of S-function will be unnecessarily small.

More generally, term mismatch corresponds to the fact that some S-function values are reliable while the others are not. The question is whether it is possible to ‘smooth’ the S-function values based on some training data and to do it in a theoretically sound way. The kernel approach that we propose in this paper can exactly solve the problem.

4.2 Robust BM25 Model

We try to learn a more reliable relevance model (S-function) from data. The model is an extension of BM25 but more robust to term mismatch. We call the model 'Robust BM25'. Without loss of generality, we use BM25 as the basic relevance model; one can easily extend the techniques here to other relevance models.

We give the definition of Robust BM25 and then explain why it has the capability to cope with term mismatch.

Robust BM25 (RBM25) is defined as follows

$$k_{RBM25}(q,d) = \sum_{i=1}^{N} \alpha_i \cdot k_{BM25}(q,d)k_Q(q,q_i)k_D(d,d_i)k_{BM25}(q_i,d_i), \tag{2}$$

where $k_{BM25}(q,d)$ is the BM25 model, $k_Q : \mathcal{Q} \times \mathcal{Q} \to \mathbb{R}$ and $k_D : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ are positive semi-definite kernels in query space and document space, which represent query similarity and document similarity, respectively. N is the number of training instances. $\{\alpha_i\}_{i=1}^{N}$ are weights and can be learned from training data. In fact, Robust BM25 is also an S-function that measures the similarity of query q and document d through a dot product in a Hilbert space, as will be explained in Section 5.

Here we assume that $k_{BM25}(q,d) > 0, \forall q \in \mathcal{Q}, d \in \mathcal{D}$, otherwise, we can add a small positive value ε to Equation (1). Furthermore, we assume that $0 \leq k_Q(\cdot,\cdot) \leq 1$ and $0 \leq k_D(\cdot,\cdot) \leq 1$.

Robust BM25 is actually a linear combination of BM25 scores of similar queries and similar documents. Because it is based on smoothing, it can be more robust, particularly when the weights are learned from data.

Figure 1 gives an intuitive explanation on why Robust BM25 can effectively deal with term mismatch. Suppose that the query space contains queries as elements and has the kernel function k_Q as a similarity function. Given query q, one can find its similar queries q_i based on $k_Q(q,q_i)$ (its neighbors). Similarly, the document space contains documents as elements and has the kernel function k_D as a similarity function. Given document d, one can find its similar documents d_i based on $k_D(d,d_i)$ (its neighbors). The relevance model BM25 is defined as an S-function between query and document over the two spaces. Term mismatch means that the BM25 score $k_{BM25}(q,d)$ is not reliable.

One possible way to deal with the problem is to use the neighboring queries q_i and documents d_i to smooth the BM25 score of q and d, as in the k-nearest neighbor algorithm (Cover and Hart, 1967; Dudani, 1976). In other words, we employ the k-nearest neighbor method in both the query and document spaces to calculate the final relevance score (cf., Figure 1). This is exactly what Robust BM25 does. More specifically, Robust BM25 determines the ranking score of query q and document d, not only based on the relevance score between q and d themselves (i.e., $k_{BM25}(q,d)$), but also based on the relevance scores between similar queries q_i and similar documents d_i (i.e., $k_{BM25}(q_i,d_i)$), and it makes a weighted linear combination of the relevance scores (2).

To help further understand why Robust BM25 can tackle the term mismatch problem, we give an example. If q is "NY", and d is about "New York", then $k_{BM25}(q,d)$ will fail to match them because q and d do not share any term. On the other hand, if q' is "New York", and we know that q and q' are similar ($k_Q(q,q')$ is high), and $k_{BM25}(q',d)$ should have a high matching score, then

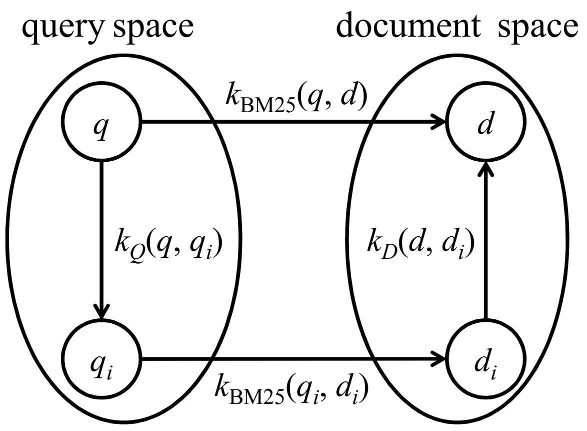

Figure 1: Robust BM25 deals with term mismatch by using the neighbors in query space and document space.

we can use $k_{BM25}(q',d)$ to boost $k_{BM25}(q,d)$. Note here that we assume $d = d'$ and $k_D(d,d') = 1$. Therefore, Robust BM25 can overcome the term mismatch problem and outperform conventional IR models.

4.3 Learning Robust BM25

We learn the weights $\{\alpha_i\}_{i=1}^N$ in Robust BM25 by using training data and a kernel method. In the kernel method, we use the following kernel on $\mathcal{Q} \times \mathcal{D}$:

$$\bar{k}_{HBM25}((q,d),(q',d')) = k_{BM25}(q,d)k_Q(q,q')k_D(d,d')k_{BM25}(q',d'), \tag{3}$$

where $k_{BM25}(q,d)$ is the BM25 model, k_Q and k_D are the query and document similarity kernels.

Suppose that the reproducing kernel Hilbert space (RKHS) generated by $\bar{k}_{HBM25}$ is $\mathcal{H}_{\bar{k}_{HBM25}}$. Given some training data $\{(q_i,d_i,r_i)\}_{i=1}^N$ where r_i represents the relevance degree between query q_i and document d_i, the learning problem is then as follows

$$\underset{k\in\mathcal{H}_{\bar{k}_{HBM25}}}{\arg\min}\ \frac{1}{N}\sum_{i=1}^N l(k(q_i,d_i),r_i) + \frac{\lambda}{2}\|k\|^2_{\mathcal{H}_{\bar{k}_{HBM25}}}, \tag{4}$$

where $l(\cdot,\cdot)$ is a loss function and $\|\cdot\|_{\mathcal{H}_{\bar{k}_{HBM25}}}$ is the norm defined in $\mathcal{H}_{\bar{k}_{HBM25}}$.

According to the representer theorem of kernel methods (Hofmann et al., 2008; Schölkopf and Smola, 2002), the optimal relevance model $k^*(q,d)$ has exactly the same form as Robust BM25 in Equation (2).

Robust BM25 (2) is also an S-function, because $\bar{k}_{HBM25}$ (3) belongs to a specific kernel class referred to as S-kernel in this paper.

5. S-kernel

In this section, we give the definition of S-kernel and also explain the kernel method of learning an S-function using an S-kernel.

Suppose that we are given training data $\mathcal{S} = \{(x_i, y_i), t_i\}_{i=1}^N$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ are a pair of objects, and $t_i \in \mathcal{T}$ is their response. The training data can be that for classification, regression, or ranking. Suppose that the hypothesis space $\mathcal{K}$ is a space of S-functions. Our goal is to learn the optimal S-function from the hypothesis space given the training data. We consider employing a kernel method to perform the learning task. That is, we specifically assume that the hypothesis space is also an RKHS generated by a positive semi-definite kernel.

The learning problem then becomes the following optimization problem:

$$\arg\min_{k \in \mathcal{K}} \frac{1}{N} \sum_{i=1}^{N} l(k(x_i, y_i), t_i) + \frac{\lambda}{2} \|k\|_{\mathcal{K}}^2, \tag{5}$$

where $\lambda > 0$ is a coefficient, $\mathcal{K}$ is a subspace of S-functions endowed with norm $||\cdot||_{\mathcal{K}}$, and $||k||_{\mathcal{K}}$ denotes regularization on space $\mathcal{K}$. Here $\mathcal{K}$ is also an RKHS generated by a positive semi-definite kernel $\overline{k} : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$, that is, for each S-function $k(x,y) \in \mathcal{K}$, $k(x,y) = \langle k(\cdot,\cdot), \overline{k}((\cdot,\cdot),(x,y))\rangle_{\mathcal{K}}$.

According to the representer theorem of kernel methods, the optimal solution of problem (5) is in the form

$$k^*(x,y) = \sum_{i=1}^{N} \alpha_i \overline{k}((x_i, y_i), (x,y)),$$

where $\alpha_i \in \mathbb{R}, 1 \le i \le N$, and N denotes the number of training instances.

The question then is whether there exists space $\mathcal{K}$, or equivalently kernel $\overline{k}$. We show below that it is the case and refer to the kernel $\overline{k}$ as S-kernel.

We formally define S-kernel and give two families of S-kernels.

Definition 3 (S-kernel) *Let $\mathcal{X}$ and $\mathcal{Y}$ be two input spaces. $\overline{k}((x,y),(x',y'))$ is called S-kernel, if it has the following properties. (1) $\overline{k} : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$ is a positive semi-definite kernel. (2) All the elements in the RKHS generated by $\overline{k}$ are S-functions on $\mathcal{X}$ and $\mathcal{Y}$.*

If the two input spaces $\mathcal{X}$ and $\mathcal{Y}$ are identical in Definition 3, then S-kernel degenerates to the hyperkernel proposed by Ong et al. (2005).

We give two families of S-kernels based on power series and multiple kernels.

Theorem 4 (Power Series Construction) *Given two Mercer kernels $k_X : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $k_Y : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, for any S-function $g(x,y)$ and $\{c_i\}_{i=0}^{\infty} \subset \mathbb{R}^+$, $\overline{k}_P$ defined below is an S-kernel.*

$$\overline{k}_P((x,y),(x',y')) = \sum_{i=0}^{\infty} c_i \cdot g(x,y) \left(k_X(x,x') k_Y(y,y')\right)^i g(x',y'), \tag{6}$$

where the convergence radius of $\sum_{i=0}^{\infty} c_i \xi^i$ is R, $|k_X(x,x')| < \sqrt{R}, |k_Y(y,y')| < \sqrt{R}$, for any x, x', y, y'.

Theorem 5 (Multiple Kernel Construction) *Given two finite sets of Mercer kernels $K_X = \left\{k_i^X(x,x')\right\}_{i=1}^n$ and $K_Y = \left\{k_i^Y(y,y')\right\}_{i=1}^n$. For any S-function $g(x,y)$ and $\{c_i\}_{i=1}^n \subset \mathbb{R}^+$, $\overline{k}_M$ defined below is an S-kernel.*

$$\overline{k}_M((x,y),(x',y')) = \sum_{i=1}^{n} c_i \cdot g(x,y) k_i^X(x,x') k_i^Y(y,y') g(x',y'). \tag{7}$$

rank	URL	click
1	`www.walmart.com`	Yes
2	`en.wikipedia.org/wiki/Wal*Mart`	No
3	`www.walmartstores.com`	Yes
4	`instoresnow.walmart.com`	No
5	`mp3.walmart.com`	No

Table 2: A record of click-through data for query "walmart". Only the top 5 URLs are shown.

Proofs of Theorem 4 and Theorem 5 are given in Appendix A and Appendix B, respectively.

Note that if space $\mathcal{X}$ and space $\mathcal{Y}$ are identical, k_X and k_Y are identical, k_i^X and k_i^Y are identical for any $1 \leqslant i \leqslant n$, and $g(x,y) = 1$, then $\overline{k_P}$ and $\overline{k_M}$ are exactly the hyperkernels given in Section 4.1 and Section 4.3 respectively by Ong et al. (2005).

With the theorems one can easily verify that the following kernel is an S-kernel.

$$g(x,y)k_X(x,x')k_Y(y,y')g(x',y'), \tag{8}$$

where $g(x,y)$ is an S-function, and $k_X(x,x')$ and $k_Y(y,y')$ are positive semi-definite kernels on spaces $\mathcal{X}$ and $\mathcal{Y}$, respectively. In fact, the S-kernel in Equation (8) is a member of the families of S-kernels in both Equation (6) and Equation (7).

It is obvious that in the learning of Robust BM25 (4), we specify $g(x,y)$ as BM25 (an S-function), and k_X and k_Y as query similarity kernel k_Q and document similarity kernel k_D, respectively. Therefore, the learning problem (4) is a specific case of learning with S-kernel (5), and Robust BM25 (2) is an S-function.

Basilico and Hofmann (2004) propose a pairwise kernel for collaborative filtering. The pairwise kernel is defined as $\overline{k_C}((u,i),(u',i')) = k_U(u,u') \cdot k_I(i,i')$, where k_U and k_I are kernels defined on the spaces of users and items, respectively. Obviously, $\overline{k_C}$ is an S-kernel and their learning problem is another specific case of learning with S-kernel (5).

6. Implementation

In this section, we describe a specific implementation to learn Robust BM25 (4).

To learn Robust BM25, we need to decide the query similarity $k_Q(q,q')$, document similarity $k_D(d,d')$, training data, and optimization technique. We explain one way of implementing them.

Click-through data has been proven to be useful for improving search relevance (cf., Cui et al., 2003; Joachims, 2002). An instance of click-through data consists of a query, a ranked list of URLs, and a user's clicks. Table 2 shows a click-through instance. In this case, the user submitted the query "walmart" and received the ranked list of URLs, and the user clicked on the URLs at ranks 1 and 3 but skipped the URLs at ranks 2, 4, and 5. Every time when a search is conducted using a search engine, this kind of data is recorded. The amount of click-through data is usually extremely large. Obviously users do not click on URLs at random, but based on their relevance judgments. Though click-through data is noisy, it still conveys users' implicit feedback to search results.

To calculate query similarity, we represent query and URL click-through relationships in a bipartite graph in which queries and URLs are nodes in two sets and clicks are edges between nodes in the two sets. A weight is associated with each edge representing the total number of times that the URL is clicked after the query is issued. Figure 2 illustrates a click-through bipartite graph.

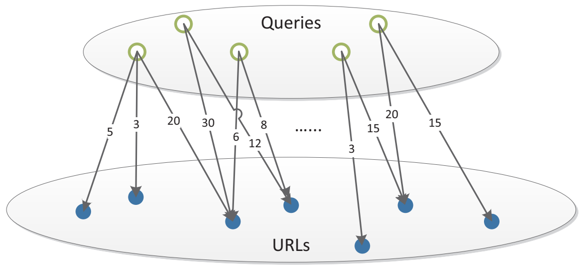

Figure 2: Click-through bipartite graph.

We specifically define query similarity using co-clicked URLs in the click-through bipartite graph. Intuitively, if two queries share many clicked URLs, then they will be regarded as similar. Since queries with the same search intent tend to be linked to the same URLs, query similarity defined in this way actually represents the degree of being the same search intent. We calculate the query similarity function $k_Q(q,q')$ as a Pearson Correlation Coefficient between the co-clicked URLs of two queries:

$$k_Q(q,q') = \frac{\sum_{i=1}^{n}(u_i-\bar{u})(v_i-\bar{v})}{\sqrt{\sum_{i=1}^{n}(u_i-\bar{u})^2}\sqrt{\sum_{i=1}^{n}(v_i-\bar{v})^2}}, \tag{9}$$

where u_i and v_i denote the numbers of clicks on URL i by queries q and q' respectively, $\bar{u}$ and $\bar{v}$ denote the average numbers of clicks of q and q' respectively, and n denotes the total number of clicked URLs by q and q'. Note that query similarity $k_Q(q,q')$ defined in Equation (9) is a positive semi-definite kernel, because it is the dot product of two vectors $(\frac{u_1-\bar{u}}{\sqrt{\sum_{i=1}^{n}(u_i-\bar{u})^2}},\ldots,\frac{u_n-\bar{u}}{\sqrt{\sum_{i=1}^{n}(u_i-\bar{u})^2}})$ and $(\frac{v_1-\bar{v}}{\sqrt{\sum_{i=1}^{n}(v_i-\bar{v})^2}},\ldots,\frac{v_n-\bar{v}}{\sqrt{\sum_{i=1}^{n}(v_i-\bar{v})^2}})$ in $\mathbb{R}^n$.

Our experimental results also show that by using the similarity function, one can really find similar queries with high quality.[2] Table 3 shows some examples of similar queries found by using our method. In fact, with the use of click-through bipartite and query similarity measure, different types of similar queries can be found, including spelling error (e.g., “wallmart” v.s. “walmart”), word segmentation (“ironman” v.s. “iron man”), stemming (e.g., “knives” v.s. “knifes” and “knife”), synonym (e.g., “aircraft for sale” v.s. “airplanes for sale”), and acronym (e.g., “ucsd” v.s. “university of california san diego”).

Document similarity $k_D(d,d')$ is simply defined as the cosine similarity between the titles and URLs of two documents, which is certainly a kernel (cosine similarity is the dot product in an Euclidean space).

Following the proposal given by Joachims (2002), we generate pairwise training data from click-through data. More precisely, for each query q_i we derive preference pairs (d_i^+,d_i^-), where d_i^+ and d_i^- mean that document d_i^+ is more preferred than d_i^- with respect to query q_i (e.g., d_i^- is skipped even though it is ranked higher than d_i^+).

Finally, we take the pairwise training data as input and learn the optimal S-function, Robust BM25. We use hinge loss as the loss function, the learning problem (4) then becomes

$$\arg\min_{k\in\mathcal{H}_{k_{HBM25}}} \sum_{i=1}^{M} \left[1-(k(q_i,d_i^+)-k(q_i,d_i^-))\right]_+ + \frac{\lambda}{2}\|k\|^2_{\mathcal{H}_{k_{HBM25}}}, \tag{10}$$

2. We evaluated the precision of several similar measures. the Pearson Correlation Coefficient and the Jensen-Shannon Divergence work the best, followed by the Jaccard Coefficient.

original query	**similar queries**
wallmart	wall mart, walmart, wal mart, walmarts
ironman	iron man, ironman movie, irnman, www.iron man.com
knives	knifes, knives.com, knife outlet, knife
aircraft for sale	aircraft sales, airplanes for sale, used airplanes for sale, used planes for sale
ucsd	ucsd.edu, uc san diego, uscd, university of california san diego

Table 3: Similar queries extracted from web search click-through data.

where M is the number of preference pairs in the training data. Note that this is similar to Ranking SVM (Herbrich et al., 1999). The major difference is that in our case the kernel function used is an S-kernel.

According to the representer theorem and Equation (2), when using pairwise training data, the optimal solution is given as follows

$$k_{RBM25}(q,d) = k_{BM25}(q,d) \cdot \sum_{i=1}^{M} \theta_i \cdot k_Q(q,q_i)\left[k_{BM25}(q_i,d_i^+)k_D(d_i^+,d) - k_{BM25}(q_i,d_i^-)k_D(d_i^-,d)\right], \tag{11}$$

where θ_i is a parameter to learn.

Reformulating the non-constrained optimization in Equation (10) as a constrained optimization by using Equation (11) and slack variables $\{\xi_i\}$, we obtain the following primal problem:

$$\arg\min_{\{\theta_i\}_{i=1}^M} \sum_{i=1}^{M} \xi_i + \frac{\lambda}{2}\sum_{i,j=1}^{M} \theta_i\theta_j \mathcal{W}(i,j) \tag{12}$$
$$k_{RBM25}(q_i,d_i^+) - k_{RBM25}(q_i,d_i^-) \geqslant 1-\xi_i, \quad \xi_i \geqslant 0\ \forall i,$$

where calculating $\mathcal{W}(i,j)$ using the reproducing kernel property is given by

$$\begin{aligned}\mathcal{W}(i,j) = k_Q(q_i,q_j)\cdot[&k_D(d_i^+,d_j^+)k_{BM25}(q_i,d_i^+)k_{BM25}(q_j,d_j^+) - k_D(d_i^+,d_j^-)k_{BM25}(q_i,d_i^+)k_{BM25}(q_j,d_j^-)\\ &- k_D(d_i^-,d_j^+)k_{BM25}(q_i,d_i^-)k_{BM25}(q_j,d_j^+) + k_D(d_i^-,d_j^-)k_{BM25}(q_i,d_i^-)k_{BM25}(q_j,d_j^-)].\end{aligned}$$

With Lagrange multipliers $\{\beta_i\}_{i=1}^M$ and $\{\gamma_i\}_{i=1}^M$, the objective function becomes

$$\begin{aligned}L &= \sum_{i=1}^{M}\xi_i + \frac{\lambda}{2}\sum_{i,j=1}^{M}\theta_i\theta_j\mathcal{W}(i,j) + \sum_{i=1}^{M}\beta_i\left[1-\xi_i-\left(k_{RBM25}(q_i,d_i^+) - k_{RBM25}(q_i,d_i^-)\right)\right] - \sum_{i=1}^{M}\gamma_i\xi_i\\ &= \sum_{i=1}^{M}\xi_i + \frac{\lambda}{2}\sum_{i,j=1}^{M}\theta_i\theta_j\mathcal{W}(i,j) + \sum_{i=1}^{M}\beta_i\left[1-\xi_i-\sum_{j=1}^{M}\theta_j\mathcal{W}(i,j)\right] - \sum_{i=1}^{M}\gamma_i\xi_i,\end{aligned}$$

Differentiating L by ξ_i and θ_i, we have

$$\frac{\partial L}{\partial \xi_i} = 1-\beta_i-\gamma_i = 0, \tag{13}$$

and

$$\frac{\partial L}{\partial \theta_i} = \sum_{j=1}^{M} (\lambda\theta_j - \beta_j)\mathcal{W}(i,j) = 0. \tag{14}$$

Thus, according to Equation (13), we have

$$\gamma_i = 1 - \beta_i.$$

Since $\beta_i \geqslant 0$ and $\gamma_i \geqslant 0$, we have $0 \leqslant \beta_i \leqslant 1$. According to Equation (14), we have

$$\sum_{j=1}^{M} \lambda\theta_j \mathcal{W}(i,j) = \sum_{j=1}^{M} \beta_j \mathcal{W}(i,j).$$

Substituting the above two formulas into L, we obtain the dual problem:

$$\underset{\{\beta\}_{i=1}^{M}}{\arg\max} \sum_{i=1}^{M} \beta_i - \frac{1}{2\lambda}\sum_{i=1}^{M}\sum_{j=1}^{M} \beta_i\beta_j \mathcal{W}(i,j) \quad s.t. \quad 0 \leq \beta_i \leq 1. \tag{15}$$

By solving the dual problem (15) we obtain the optimal values $\{\beta_i^\star\}_{i=1}^{M}$. We can further get the optimal values $\{\theta_i^\star\}_{i=1}^{M}$ by solving equation (14), and using $\theta_i^\star = \frac{1}{\lambda}\beta_i^\star$. Note that when $(\mathcal{W}(i,j))_{M\times M}$ is not strictly positive, the solution of (14) is not unique. In such a case, we can still take $\theta_i^\star = \frac{1}{\lambda}\beta_i^\star$ as a solution for simplicity, because all solutions will make the objective function achieve the same minimum (12).

In online search, given a query, we first retrieve the queries similar to it, then individually retrieve documents with the original query and similar queries, combine the retrieved documents, train a Robust BM25 model using click-through data, and rank the documents with their Robust BM25 scores (note that a Robust BM25 model is trained for each query). When training Robust BM25, we solve the dual problem (15) using a standard QP solver LOQO.[3] The time complexity is of order $O(M^2)$, where M is the number of preference pairs. Since the number of retrieved documents is small, a search with Robust BM25 can be carried out efficiently. In our experiments, we observe that on average it takes about 1.5 seconds per query to train a model on a workstation with Quad-Core Intel Xeon E5410 2.33GHz CPU and 16GB RAM.

7. Experiments

We conducted experiments to test the performances of Robust BM25.

7.1 Experimental Data

In our experiments, we used two large scale data sets from a commercial web search engine and an enterprise search engine running in an IT company. The two data sets consist of query-URL pairs and their relevance judgments. The relevance judgments can be ‘Perfect’, ‘Excellent’, ‘Good’, ‘Fair’, or ‘Bad’. Besides this, we also collected large scale click-through data from both search engines. Table 4 shows the statistics on the two data sets. The click-through data in both data sets was split into two parts, one for learning query similarity and the other for learning Robust BM25.

3. LOQO can be found at `http://www.princeton.edu/~rvdb/loqo/LOQO.html`.

	Web search	Enterprise search
# of judged queries	8,294	2,864
# of judged query-URL pairs	1,715,844	282,130
# of search impressions in click-through	490,085,192	17,383,935
# of unique queries in click-through	14,977,647	2,368,640
# of unique URLs in click-through	30,166,304	2,419,866
# of clicks in click-through	2,605,404,156	4,996,027

Table 4: Statistics on web search and enterprise search data sets.

7.2 Baselines

BM25 was selected as a baseline, whose parameters were tuned by using the validation set. Query expansion (Xu and Croft, 1996) was also chosen as a baseline. Query expansion is a state-of-the-art technique to tackle term mismatch in search. The key idea in query expansion is to add into the original query terms extracted from relevant queries or documents. Thus, even though the original query and document do not share a term, after expansion, the query is enriched and it is likely to be matched with relevant documents. On the other hand, query expansion may also suffer from the so-called topic drift problem. That is, irrelevant terms can be added to the original query. As a result, the accuracy of search may drop, rather than improve. In contrast, our method can effectively address the problem. First, similar queries mined from click-through data are used in search, which represent the same or similar intent. Thus, the documents retrieved are more likely to be relevant. Second, the final ranking of results is based on Robust BM25 which is trained specifically for the query using click-through data. Therefore, the accuracy of the final ranking will be high.

In our experiment, we tried several different ways to conduct query expansion and chose the one performing the best as the baseline. In our method, we first use the title of the most clicked URL in the retrieved result to do expansion. If there is no such a URL, we use the terms of the most similar query to do expansion.

The pairwise kernel, which is initially proposed for collaborative filtering (Basilico and Hofmann, 2004), was also chosen as a baseline. The difference between our method and the pairwise kernel is that the pairwise kernel does not use a traditional relevance model $k_{BM25}(q,d)$.

7.3 Evaluation Measures

As evaluation measures, we used Mean Average Precision (MAP) (Baeza-Yates and Ribeiro-Neto, 1999) and Normalized Discounted Cumulative Gain (NDCG) (Jarvelin and Kekalainen, 2000) at positions 1, 3, and 5, which are standard measures in IR.

MAP assesses the accuracy of a ranking algorithm by looking at how well it ranks relevant documents against irrelevant documents. MAP denotes mean average precision (AP). Average precision is defined as

$$AP(q) = \frac{\sum_{r=1}^{N_q} P(r) \times rel(r)}{\sum_{r=1}^{N_q} rel(r)},$$

where N_q is the number of documents retrieved, $rel(r) \in \{0,1\}$, and if the document ranked at position r is relevant, $rel(r) = 1$, otherwise, $rel(r) = 0$. $P(r)$ is precision at position r:

$$P(r) = \frac{\sum_{i=1}^{r} rel(i)}{r}.$$

Finally, MAP is defined as

$$\text{MAP} = \frac{\sum_q AP(q)}{\#q},$$

where $\#q$ is the number of queries. If relevant documents are ranked higher than irrelevant documents, the value of MAP will be high.

NDCG is usually used to assess a ranking algorithm when documents have multiple relevance grades (e.g., “Bad”, “Good”, “Fair”, “Excellent”, and “Perfect”). Given a query q, NDCG at position n is defined as

$$\text{NDCG@}n(q) = \frac{\text{DCG@}n(q)}{\text{IDCG@}n(q)},$$

where DCG@$n(q)$ is defined as

$$\text{DCG@}n(q) = \sum_{i=1}^{n_q} \frac{2^{rel(i)} - 1}{\log_2(i+1)},$$

where $rel(i)$ is the relevance grade of a document ranked at position i. The DCG@$n(q)$ score is normalized by IDCG@$n(q)$, which is an ideal DCG@$n(q)$ score when documents are ranked in decreasing order of their relevance grades.

Finally, NDCG is averaged over queries.

$$\text{NDCG@}n = \frac{\sum_q \text{NDCG@}n(q)}{\#q}$$

A high NDCG score means that relevant documents are ranked higher in the ranking list than irrelevant documents.

In our experiment, when calculating MAP, we view the documents with judgments ‘Perfect’ and ‘Excellent’ as relevant and the documents with the other three judgments as irrelevant.

7.4 Experimental Results

We trained a model for each query, as described in Section 6. On average, about 207.6 and 174.7 training pairs were used for each query in web search data and enterprise data, respectively. The only parameter λ in Equation (15) was heuristically set as 1. In fact, we found that λ does not affect the results so much. Table 5 reports the results on the web search data and enterprise data. We can see that Robust BM25 outperforms the baselines, in terms of all measures on both data sets. We conducted significant tests (t-test) on the improvements. The results show that the improvements are all statistically significant (p-value < 0.05). We conducted analysis on the cases in which Robust BM25 performs better and found that the reason is that Robust BM25 can indeed effectively address the term mismatch problem. The pairwise kernel outperforms BM25 and query expansion, which indicates that it is better to learn a relevance model in search. However, its performance is still lower than Robust BM25, suggesting that it is better to include BM25 in the final relevance model, as in Robust BM25.

		MAP	NDCG@1	NDCG@3	NDCG@5
Web search	Robust BM25	0.1192	0.2480	0.2587	0.2716
	Pairwise Kernel	0.1123	0.2241	0.2418	0.2560
	Query Expansion	0.0963	0.1797	0.2061	0.2237
	BM25	0.0908	0.1728	0.2019	0.2180
Enterprise search	Robust BM25	0.3122	0.4780	0.5065	0.5295
	Pairwise Kernel	0.2766	0.4465	0.4769	0.4971
	Query Expansion	0.2755	0.4076	0.4712	0.4958
	BM25	0.2745	0.4246	0.4531	0.4741

Table 5: Ranking accuracies on web search and enterprise search data.

Query	wal**l**mart
Similar queries	wall mart, walmart, wal mart, walmarts
Page	`http://www.walmart.com`
Title	Walmart.com: Save money. Live better
Rate	Perfect

Table 6: Example 1 from web search.

7.5 Discussions

We investigated the reasons that Robust BM25 can outperform the baselines, using the experiments on web search data as examples. It seems that Robust BM25 can effectively deal with term mismatch with its mechanisms: using query similarity and document similarity.

Our approach can effectively deal with term mismatch with similar queries. Table 6 gives an example. The query, web page, and label are respectively "wallmart", which is a typo, "`http://www.walmart.com`" with title "Walmart.com: Save money. Live better", and "Perfect", which means that the page should be ranked in first position. There is a mismatch between query and page, the basic relevance model BM25 cannot give a high score to the page (note that there is a difference between the query term "wallmart" and the document term "walmart".). Query expansion cannot rank the page high, either. The web page "`http://www.walmartstores.com`" with title "Walmartstores.com" is the most clicked web page with respect to the original query in the click-through data. Query expansion uses the title to conduct term expansion, that is, uses the words in the title. Because it does not have sufficient knowledge to break "Walmartstores" into "walmart" and "stores", query expansion cannot add good terms to the original query. When query expansion adds more terms to the original query, "walmart" will appear, but at the same time noise terms will also be included. In contrast, our approach can effectively leverage similar queries such as "walmart", "wal mart", and "walmarts" and rank the web page to first position.

Table 7 gives another example. The query is "mensmagazines", which is a tail query and does not have a similar query found in the click-through data. The web page is "`http://en.wikipedia.org/wiki/List/_of/_men′s/_magazines`" (referred to as Page1) and the relevance label is "Excellent". There is a mismatch, because there is not sufficient knowledge to break query "mensmagazines" into "mens" and "magazines". As a result, BM25 cannot rank Page1 high. In contrast, Robust BM25 uses similar documents to calculate the relevance. Specifically, it uses a similar web page "`http://www.askmen.com/links/sections/mensmagazines.html`" (referred to

Query	mensmagazines
Page1	`http://en.wikipedia.org/wiki/List/_of/_men's/_magazines`
Title1	List of men's magazines - Wikipedia, the free encyclopedia
Rate1	Excellent
Page2	`http://www.askmen.com/links/sections/mensmagazines.html`
Title2	AskMen.com - Men's magazines

Table 7: Example 2 from web search.

Query	southwest airlines
Page1	`http://www.southwest-airlines.net`
Title1	Southwest Airlines
Rate1	Perfect
Page2	`http://www.southwestvacations.com/index.asp`
Title2	Southwest Vacations - Vacation Packages - Cheap Airline Tickets, Hotels, Rental Cars, Activities & Attractions
Rate2	Fair

Table 8: Example 3 from web search.

as Page2), which contains the term "mensmagazines" in its URL. The original query can match well with Page2. Besides, Page1 and Page2 are also similar because they have common terms "men" and "magazines" in titles. Therefore, Robust BM25 can assign a high score to Page1.

Compared with the pairwise kernel, Robust BM25 successfully leveraged the traditional matching model BM25 when its score is reliable to reflect relevance between query and document. We show an example in Table 8. The query is "southwest airlines". The two web pages are "`http://www.southwest-airlines.net`" (referred to as Page1) with label "Perfect" and "`http://www.southwestvacations.com/index.asp`" (referred to as Page2) with label "Fair". In the pairwise kernel, the ranking score of Page2 is larger than Page1. In Robust BM25, however, the ranking score of Page1 is larger. This is because the pairwise kernel does not consider the match between query and documents using BM25, while Robust BM25 does.

8. Conclusion and Future Work

We have formally defined a similarity function between pairs of objects from two different spaces and named it S-function. We have shown that traditional relevance models in search proposed in information retrieval can be viewed as S-functions. We have proposed a new kernel method for learning a robust relevance model as an S-function for search. The learned model can deal with the term mismatch problem which traditional relevance models suffer from. The kernel method employs a new kernel called S-kernel. An S-kernel is a kernel that can generate an RKHS which is also a space of S-functions. We have provided a theoretical basis for constructing S-kernels. Finally, we have shown that we can apply our method to learn a Robust BM25 model to deal with term mismatch in search.

There are several directions for future research from the current work:

1. **S-function as generalization of kernel:** In this paper, we give a formal definition of S-function and show that it is related to a positive semi-definite kernel. S-function is also similar to the asymmetric kernel defined by Koide and Yamashita (2006). To make S-function a generalization of a positive semi-definite kernel, there are still some open questions that we need to answer. For example, what is a necessary and sufficient condition for a two-argument function over two spaces to be an S-function? Is there a theorem like the Mercer theorem for S-function?

2. **S-kernel:** We define two families of S-kernels in this paper, that is, to give two sufficient conditions for a positive semi-definite kernel to be an S-kernel. It is still an open question: what is a necessary and sufficient condition for a positive semi-definite kernel to be an S-kernel?

3. **Similarity function learning:** We employ a kernel method to learn a similarity function for search. An interesting research direction is to study the general problem of similarity function learning, particularly, the learning of a similarity function for pairs of objects from two different spaces. The learning task can be applied to a wide range of applications and is becoming a popular research topic.

4. **Learning of S-Kernel:** Our kernel method employs S-kernel which contains free parameters. How to automatically learn the parameters from data, and thus a better S-function is also an interesting issue.

Acknowledgments

We would like to thank Stephen Robertson for his suggestions on this work. We also acknowledge Corinna Cortes, the anonymous reviewers, and Matt Callcut for their many valuable comments on this paper.

Appendix A. Proof of Theorem 4

Theorem 4 *Given two Mercer kernels $k_X : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $k_Y : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, for any S-function $g(x,y)$ and $\{c_i\}_{i=0}^{\infty} \subset \mathbb{R}^+$, $\overline{k}_P$ defined below is an S-kernel.*

$$\overline{k}_P((x,y),(x',y')) = \sum_{i=0}^{\infty} c_i \cdot g(x,y) \left(k_X(x,x')k_Y(y,y')\right)^i g(x',y'),$$

where the convergence radius of $\sum_{i=0}^{\infty} c_i \xi^i$ is R, $|k_X(x,x')| < \sqrt{R}, |k_Y(y,y')| < \sqrt{R}$, for any x,x',y,y'.

According to Definition 3, to prove Theorem 4, we first need to prove that $\overline{k}_P((x,y),(x',y'))$ is a positive semi-definite kernel. Note that $g(x,y)g(x',y')$ is a positive semi-definite kernel, since it is symmetric and $\forall\{\alpha\}_{i=1}^n, \{(x_i,y_i)\}_{i=1}^n$, $\sum_{i,j=1}^n \alpha_i\alpha_j g(x_i,y_i)g(x_j,y_j) = (\sum_{i=1}^n \alpha_i g(x_i,y_i))^2 \geq 0$. Moreover, for any $i \in \mathbb{Z}^+$, $c_i \in \mathbb{R}^+$, $c_i\left(k_X(x,x')k_Y(y,y')\right)^i$ is also a positive semi-definite kernel. Hence, $c_i g(x,y)\left(k_X(x,x')k_Y(y,y')\right)^i g(x',y')$ is a positive semi-definite kernel. Since the summation of positive semi-definite kernels is also a positive semi-definite kernel, we conclude that $\overline{k}_P((x,y),(x',y'))$ is a positive semi-definite kernel.

Second, we need to prove that all elements in the reproducing kernel Hilbert space $\overline{\mathcal{K}_P}$ generated by $\overline{k_P}$ are S-functions. We need two lemmas:

Lemma 6 *Suppose that $g(x,y)$ is an S-function, and k_X and k_Y are Mercer kernels. Given any finite example set $\{(x_j,y_j)\}_{j=1}^N \subset \mathcal{X}\times\mathcal{Y}$, and any $\{\alpha_j\}_{j=1}^N \subset \mathbb{R}$, $\sum_{j=1}^N \alpha_j \overline{k_P}((x,y),(x_j,y_j))$ is an S-function.*

Proof

$$\begin{aligned}\sum_{j=1}^N \alpha_j \overline{k_P}((x,y),(x_j,y_j)) &= \sum_{j=1}^N \alpha_j g(x,y) \sum_{i=0}^\infty c_i \left(k_X(x,x_j)k_Y(y,y_j)\right)^i g(x_j,y_j)\\ &= g(x,y)\sum_{j=1}^N \alpha_j \tilde{\overline{k}}_P((x,y),(x_j,y_j)),\end{aligned}$$

where

$$\tilde{\overline{k}}_P((x,y),(x_j,y_j)) = \sum_{i=0}^\infty c_i \left(k_X(x,x_j)k_Y(y,y_j)\right)^i g(x_j,y_j).$$

Since $g(x,y)$ is an S-function, according to Lemma 2, to prove $\sum_{j=1}^N \alpha_j \overline{k_P}((x,y),(x_j,y_j))$ is an S-function, we only need to show that $\sum_{j=1}^N \alpha_j \tilde{\overline{k}}_P((x,y),(x_j,y_j))$ is an S-function.

For any $i \geqslant 0$, $i \in \mathbb{Z}^+$, since $\sqrt{c_i}k_X^i(x,x')$ and $\sqrt{c_i}k_Y^i(y,y')$ are both Mercer kernels, we obtain

$$\sqrt{c_i}k_X^i(x,x') = \langle \psi_X^i(x), \psi_X^i(x')\rangle_{\mathcal{H}_X^i}$$

and

$$\sqrt{c_i}k_Y^i(y,y') = \langle \psi_Y^i(y), \psi_Y^i(y')\rangle_{\mathcal{H}_Y^i},$$

where $\psi_X^i(\cdot) : \mathcal{X} \to \mathcal{H}_X^i$ and $\psi_Y^i(\cdot) : \mathcal{Y} \to \mathcal{H}_Y^i$ are feature mappings, and $\mathcal{H}_X^i$ and $\mathcal{H}_Y^i$ are Hilbert spaces with respect to ψ_X^i and ψ_Y^i, respectively.

Let $\mathcal{H}_i = \mathcal{H}_X^i$, $\varphi_X^i(x) = \psi_X^i(x)$, and $\varphi_{YN}^i(y) = \sum_{j=1}^N \alpha_j g(x_j,y_j)\psi_X^i(x_j){\psi_Y^i}^\top(y_j)\psi_Y^i(y)$. Note that $\varphi_{YN}^i = \Gamma_N^i \psi_Y^i$, where $\Gamma_N^i = \sum_{j=1}^N \alpha_j g(x_j,y_j)\psi_X^i(x_j){\psi_Y^i}^\top(y_j)$ is a linear operator from $\mathcal{H}_Y^i$ to $\mathcal{H}_X^i = \mathcal{H}_i$. Thus, we have

$$\sum_{j=1}^N \alpha_j c_i (k_X(x,x_j)k_Y(y,y_j))^i g(x_j,y_j) = \langle \varphi_X^i(x), \varphi_{YN}^i(y)\rangle_{\mathcal{H}_i}.$$

Let $\mathcal{H} = \mathcal{H}_0 \times \mathcal{H}_1 \times \ldots \mathcal{H}_k \times \ldots$,

$$\begin{aligned}\varphi_X :& \mathcal{X} \to \mathcal{H}\\ & x \mapsto (\varphi_X^0(x), \varphi_X^1(x), \ldots, \varphi_X^k(x), \ldots),\end{aligned}$$

and

$$\begin{aligned}\varphi_{YN} :& \mathcal{Y} \to \mathcal{H}\\ & y \mapsto (\varphi_{YN}^0(y), \varphi_{YN}^1(y), \ldots, \varphi_{YN}^k(y), \ldots),\end{aligned}$$

we have

$$\begin{aligned}\sum_{j=1}^{N}\alpha_j\widetilde{k_P}((x,y),(x_j,y_j)) &= \sum_{j=1}^{N}\alpha_j\sum_{i=0}^{\infty}c_i(k_X(x,x_j)k_Y(y,y_j))^i g(x_j,y_j)\\ &= \sum_{i=0}^{\infty}\sum_{j=1}^{N}\alpha_j c_i(k_X(x,x_j)k_Y(y,y_j))^i g(x_j,y_j)\\ &= \sum_{i=0}^{\infty}\langle\varphi_X^i(x),\varphi_{YN}^i(y)\rangle_{\mathcal{H}_i}\\ &= \langle\varphi_X(x),\varphi_{YN}(y)\rangle_{\mathcal{H}},\end{aligned}$$

where the inner product in $\mathcal{H}$ is naturally defined as $\sum_{i=0}^{\infty}\langle\cdot,\cdot\rangle_{\mathcal{H}_i}$. Note that $\sum_{i=0}^{\infty}\langle\cdot,\cdot\rangle_{\mathcal{H}_i}$ can be defined only when $(z_0,\dots,z_k,\dots)\in\mathcal{H},\sum_{i=0}^{\infty}\langle z_i,z_i\rangle_{\mathcal{H}_i}<\infty$. Obviously, for any $x\in\mathcal{X}$ and $y\in\mathcal{Y}$, $\varphi_X(x)$ and $\varphi_{YN}(y)$ satisfy this condition. ■

Lemma 7 *Given any two positive semi-definite kernels $k_X:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$ and $k_Y:\mathcal{Y}\times\mathcal{Y}\to\mathbb{R}$. Suppose $\psi_Y:\mathcal{Y}\to\mathcal{H}_Y$ is the feature mapping of $k_Y(\cdot,\cdot)$. $\mathcal{H}_Y$ is a Hilbert space endowed with inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}_Y}$. Given any sets $\{x_i\}_{i=1}^N\subset\mathcal{X}$ and $\{y_i\}_{i=1}^N\subset\mathcal{Y}$, for an arbitrary $z\in\mathcal{H}_Y$, the following matrix inequality holds:*

$$\left(k_X(x_i,x_j)\langle\psi_Y(y_i),z\rangle_{\mathcal{H}_Y}\langle\psi_Y(y_j),z\rangle_{\mathcal{H}_Y}\right)_{N\times N}\preccurlyeq\left(k_X(x_i,x_j)k_Y(y_i,y_j)\langle z,z\rangle_{\mathcal{H}_Y}\right)_{N\times N}.$$

Proof Since $k_X(\cdot,\cdot)$ is a positive semi-definite kernel, following the conclusion given in Proposition 4 by Hofmann et al. (2008), we only need to prove

$$\left(k_Y(y_i,y_j)\langle z,z\rangle_{\mathcal{H}_Y}-\langle\psi_Y(y_i),z\rangle_{\mathcal{H}_Y}\langle\psi_Y(y_j),z\rangle_{\mathcal{H}_Y}\right)_{N\times N}$$

is positive semi-definite, which means given any $\{\alpha_i\}_{i=1}^N\subset\mathbb{R}$, we need to prove

$$\sum_{i,j=1}^{N}\alpha_i\alpha_j\left(k_Y(y_i,y_j)\langle z,z\rangle_{\mathcal{H}_Y}-\langle\psi_Y(y_i),z\rangle_{\mathcal{H}_Y}\langle\psi_Y(y_j),z\rangle_{\mathcal{H}_Y}\right)\geqslant 0.$$

Since

$$\begin{aligned}\sum_{i,j=1}^{N}\alpha_i\alpha_j k_Y(y_i,y_j)\langle z,z\rangle_{\mathcal{H}_Y} &= \sum_{i,j=1}^{N}\alpha_i\alpha_j\langle\psi_Y(y_i),\psi_Y(y_j)\rangle_{\mathcal{H}_Y}\langle z,z\rangle_{\mathcal{H}_Y}\\ &= \left\langle\sum_{i=1}^{N}\alpha_i\psi_Y(y_i),\sum_{i=1}^{N}\alpha_i\psi_Y(y_i)\right\rangle_{\mathcal{H}_Y}\langle z,z\rangle_{\mathcal{H}_Y},\end{aligned}$$

and

$$\sum_{i,j=1}^{N}\alpha_i\alpha_j\langle\psi_Y(y_i),z\rangle_{\mathcal{H}_Y}\langle\psi_Y(y_j),z\rangle_{\mathcal{H}_Y}=\left(\langle\sum_{i=1}^{N}\alpha_i\psi_Y(y_i),z\rangle_{\mathcal{H}_Y}\right)^2,$$

according to the Cauchy inequality, we reach the conclusion. ■

With Lemma 6 and Lemma 7, we can complete the proof of Theorem 4:

Proof Given a function $k(x,y)$ in $\mathcal{K}_P$, there is a sequence $\{k_N(x,y)\}$ in $\mathcal{K}_P$ such that

$$\begin{aligned} k_N(x,y) &= \sum_{j=1}^{N} \alpha_j \bar{k}_P((x,y),(x_j,y_j)) \\ &= \sum_{j=1}^{N} \alpha_j g(x,y) \sum_{i=0}^{\infty} c_i \left(k_X(x,x_j)k_Y(y,y_j)\right)^i g(x_j,y_j) \\ &= g(x,y) \sum_{j=1}^{N} \alpha_j \tilde{\bar{k}}_P((x,y),(x_j,y_j)); \\ k(x,y) &= \lim_{N\to\infty} k_N(x,y), \end{aligned}$$

where $\tilde{\bar{k}}_P((x,y),(x_j,y_j)) = \sum_{i=0}^{\infty} c_i \left(k_X(x,x_j)k_Y(y,y_j)\right)^i g(x_j,y_j)$.

We try to prove that $k(x,y)$ is an S-function. Let $\tilde{k}_N(x,y) = \sum_{j=1}^{N} \alpha_j \tilde{\bar{k}}_P((x,y),(x_j,y_j))$, $k(x,y) = \lim_{N\to\infty} k_N(x,y) = \lim_{N\to\infty} \tilde{k}_N(x,y)g(x,y) = \tilde{k}(x,y)g(x,y)$, where $\tilde{k}(x,y) = \lim_{N\to\infty} \tilde{k}_N(x,y)$.

According to Lemma 2, we only need to prove that $\tilde{k}(x,y)$ is an S-function. From the proof of Lemma 6, we know $\tilde{k}_N(x,y) = \langle \varphi_X(x), \varphi_{YN}(y)\rangle_{\mathcal{H}}$, where $\mathcal{H}$ is a Hilbert space determined by $\{\sqrt{c_i}k_X^i\}_{i=0}^{\infty}$.

$$\varphi_{YN}(y) = (\varphi_{YN}^0(y), \varphi_{YN}^1(y), \ldots, \varphi_{YN}^k(y), \ldots),$$

and we define $\mathcal{H}_Y = \mathcal{H}_Y^0 \times \ldots \mathcal{H}_Y^k \times \ldots$, $\mathcal{H}_X = \mathcal{H}_X^0 \times \ldots \mathcal{H}_X^k \times \ldots$, and

$$\begin{aligned} \Gamma_N : & \mathcal{H}_Y \to \mathcal{H}_X \\ & z = (z_0, \ldots, z_k, \ldots) \mapsto \Gamma_N(z) = (\Gamma_N^0 z_0, \ldots, \Gamma_N^k z_k, \ldots), \end{aligned}$$

where $\mathcal{H}_X^i$ and $\mathcal{H}_Y^i$ are the Hilbert spaces with respect to feature mappings $\psi_X^i(\cdot)$ and $\psi_Y^i(\cdot)$ defined by Mercer kernels $\sqrt{c_i}k_X^i$ and $\sqrt{c_i}k_Y^i$, respectively, $\langle\cdot,\cdot\rangle_{\mathcal{H}_Y^i}$ is the inner product defined in $\mathcal{H}_Y^i$, and $\Gamma_N^k z_k = \sum_{j=1}^{N} \alpha_j g(x_j,y_j)\psi_X^k(x_j)\langle \psi_Y^k(y_j), z_k\rangle_{\mathcal{H}_Y^k}$. The inner products for $\mathcal{H}_X = \mathcal{H}_X^0 \times \ldots \mathcal{H}_X^k \times \ldots$ and $\mathcal{H}_Y = \mathcal{H}_Y^0 \times \ldots \mathcal{H}_Y^k \times \ldots$ are naturally defined as $\sum_{i=0}^{\infty}\langle\cdot,\cdot\rangle_{\mathcal{H}_X^i}$ and $\sum_{i=0}^{\infty}\langle\cdot,\cdot\rangle_{\mathcal{H}_Y^i}$, respectively. Note that to make the inner products well defined, we require that input $(z_0,\ldots,z_k,\ldots)$ satisfies $\sum_{i=0}^{\infty}\langle z_i, z_i\rangle_{\mathcal{H}_Y^i} < \infty$. From the following proof, we will see that this condition will guarantee that $\sum_{i=0}^{\infty}\langle \Gamma_N^i z_i, \Gamma_N^i z_i\rangle_{\mathcal{H}_X^i} < \infty$. Thus, Γ_N is well defined.

Then the key point we need to prove is that $\{\Gamma_N\}$ is a Cauchy sequence. $\forall z \in \mathcal{H}_Y$, $||z||_{\mathcal{H}_Y} < \infty$,

$$\| \Gamma_N(z) \|_{\mathcal{H}_X}^2 = \sum_{i=0}^{\infty} \sum_{k,j=1}^{N} \alpha_k \alpha_j g(x_k,y_k) g(x_j,y_j) \sqrt{c_i} k_X^i(x_k,x_j) \langle \psi_Y^i(y_k), z_i\rangle_{\mathcal{H}_Y^i} \langle \psi_Y^i(y_j), z_i\rangle_{\mathcal{H}_Y^i}.$$

Using the conclusion given by Lemma 7, we have

$$\begin{aligned}
&\sum_{i=0}^{\infty}\sum_{k,j=1}^{N}\alpha_k\alpha_j g(x_k,y_k)g(x_j,y_j)\,\sqrt{c_i}k_X^i(x_k,x_j)\langle\psi_Y^i(y_k),z_i\rangle_{\mathcal{H}_Y^i}\langle\psi_Y^i(y_j),z_i\rangle_{\mathcal{H}_Y^i}\\
\leqslant&\sum_{i=0}^{\infty}\sum_{k,j=1}^{N}\alpha_k\alpha_j g(x_k,y_k)g(x_j,y_j)c_i(k_X(x_k,x_j)k_Y(y_k,y_j))^i\langle z_i,z_i\rangle_{\mathcal{H}_Y^i}\\
\leqslant&\left(\sum_{i=0}^{\infty}\sum_{k,j=1}^{N}\alpha_k\alpha_j g(x_k,y_k)g(x_j,y_j)c_i(k_X(x_k,x_j)k_Y(y_k,y_j))^i\right)\left(\sum_{i=0}^{\infty}\langle z_i,z_i\rangle_{\mathcal{H}_Y^i}\right).
\end{aligned}$$

Thus,

$$\parallel\Gamma_N(z)\parallel^2_{\mathcal{H}_X}\leqslant\sum_{i=0}^{\infty}\sum_{k,j=1}^{N}\alpha_k\alpha_j g(x_k,y_k)g(x_j,y_j)c_i(k_X(x_k,x_j)k_Y(y_k,y_j))^i\parallel z\parallel^2_{\mathcal{H}_Y}.$$

Therefore,

$$\begin{aligned}
\parallel\Gamma_N\parallel^2\leqslant&\sum_{i=0}^{\infty}\sum_{k,j=1}^{N}\alpha_k\alpha_j g(x_k,y_k)g(x_j,y_j)c_i(k_X(x_k,x_j)k_Y(y_k,y_j))^i\\
=&\sum_{k,j=1}^{N}\alpha_k\alpha_j\overline{k}_P((x_k,y_k),(x_j,y_j)).
\end{aligned}$$

Note that $\sum_{k,j=1}^{N}\alpha_k\alpha_j\overline{k}_P((x_k,y_k),(x_j,y_j))$ is just the square of the norm of $k_N(x,y)$ in $\mathcal{K}_P$. From the fact that $\{k_N(x,y)\}$ is a Cauchy sequence in $\mathcal{K}_P$, we know that $\{\Gamma_N\}$ is also a Cauchy sequence. Then there is a linear operator Γ which satisfies $\Gamma=\lim_{N\to\infty}\Gamma_N$. The convergence is in the norm. Thus, for every $(x,y)\in\mathcal{X}\times\mathcal{Y}$, $\lim_{N\to\infty}\tilde{k}_N(x,y)=\langle\varphi_X(x),\varphi_Y(y)\rangle_{\mathcal{H}}=\tilde{k}(x,y)$, where φ_Y is given by

$$(\Gamma^0\psi_Y^0,\ldots,\Gamma^k\psi_Y^k,\ldots).$$

■

Appendix B. Proof of Theorem 5

Theorem 5 *Given two finite sets of Mercer kernels $K_X=\left\{k_i^X(x,x')\right\}_{i=1}^n$ and $K_Y=\left\{k_i^Y(y,y')\right\}_{i=1}^n$. For any S-function $g(x,y)$ and $\{c_i\}_{i=1}^n\subset\mathbb{R}^+$, $\overline{k}_M$ defined below is an S-kernel.*

$$\overline{k}_M((x,y),(x',y'))=\sum_{i=1}^{n}c_i\cdot g(x,y)k_i^X(x,x')k_i^Y(y,y')g(x',y').$$

First, since $\forall i$, $c_ig(x,y)k_i^X(x,x')k_i^Y(y,y')g(x',y')$ is a positive semi-definite kernel, and the summation of positive semi-definite kernels is also a positive semi-definite kernel, we know that $\overline{k}_M((x,y),(x',y'))$ is a positive semi-definite kernel on $(\mathcal{X}\times\mathcal{Y})\times(\mathcal{X}\times\mathcal{Y})$.

Second, we need one more lemma to prove that all elements in the reproducing kernel Hilbert space $\mathcal{K}_M$ generated by $\overline{k}_M$ are S-functions:

Lemma 8 *Suppose that $g(x,y)$ is an S-function, and k_X and k_Y are Mercer kernels. Given any $\{(x_j,y_j)\}_{j=1}^N \subset \mathcal{X}\times\mathcal{Y}$, and any $\{\alpha_j\}_{j=1}^N \subset \mathbb{R}$, $\sum_{j=1}^N \alpha_j \bar{k}_M((x,y),(x_i,y_i))$ is an S-function.*

Proof

$$\begin{aligned}\sum_{j=1}^N \alpha_j \bar{k}_M((x,y),(x_j,y_j)) &= \sum_{j=1}^N \alpha_j g(x,y) \sum_{i=1}^n c_i k_i^X(x,x_j) k_i^Y(y,y_j) g(x_j,y_j)\\ &= g(x,y)\sum_{j=1}^N \alpha_j \tilde{k}_M((x,y),(x_j,y_j)).\end{aligned}$$

Since $g(x,y)$ is an S-function, according to Lemma 2, to prove $\sum_{j=1}^N \alpha_j \bar{k}_M((x,y),(x_j,y_j))$ is an S-function, we only have to prove that $\sum_{j=1}^N \alpha_j \tilde{k}_M((x,y),(x_j,y_j))$ is an S-function.

For any $0 \leqslant i \leqslant n, i \in \mathbb{Z}$, since $\sqrt{c_i}k_i^X(x,x')$ and $\sqrt{c_i}k_i^Y(y,y')$ are both Mercer kernels, we obtain

$$\sqrt{c_i}k_i^X(x,x') = \langle \psi_X^i(x), \psi_X^i(x')\rangle_{\mathcal{H}_X^i}, \qquad \sqrt{c_i}k_i^Y(y,y') = \langle \psi_Y^i(y), \psi_Y^i(y')\rangle_{\mathcal{H}_Y^i},$$

where $\psi_X^i(\cdot) : \mathcal{X} \to \mathcal{H}_X^i$ and $\psi_Y^i(\cdot) : \mathcal{Y} \to \mathcal{H}_Y^i$ are feature mappings, and $\mathcal{H}_X^i$ and $\mathcal{H}_Y^i$ are Hilbert spaces with respect to ψ_X^i and ψ_Y^i, respectively.

Let $\mathcal{H}_i = \mathcal{H}_X^i$, $\varphi_X^i(x) = \psi_X^i(x)$, and $\varphi_{YN}^i(y) = \sum_{j=1}^N \alpha_j g(x_j,y_j)\psi_X^i(x_j){\psi_Y^i}^\top(y_j)\psi_Y^i(y)$. Note that $\varphi_{YN}^i = \Gamma_N^i \psi_Y^i$, where $\Gamma_N^i = \sum_{j=1}^N \alpha_j g(x_j,y_j)\psi_X^i(x_j){\psi_Y^i}^\top(y_j)$ is a linear operator from $\mathcal{H}_Y^i$ to $\mathcal{H}_X^i = \mathcal{H}_i$. Thus, we have

$$\sum_{j=1}^N \alpha_j c_i k_i^X(x,x_j) k_i^Y(y,y_j) g(x_j,y_j) = \langle \varphi_X^i(x), \varphi_{YN}^i(y)\rangle_{\mathcal{H}_i}.$$

Let $\mathcal{H} = \mathcal{H}_1 \times \mathcal{H}_2 \times \dots \mathcal{H}_n$,

$$\begin{aligned}\varphi_X(x) :& \mathcal{X} \to \mathcal{H}\\ & x \mapsto (\varphi_X^1(x), \varphi_X^2(x), \dots, \varphi_X^n(x)),\end{aligned}$$

and

$$\begin{aligned}\varphi_{YN}(y) :& \mathcal{Y} \to \mathcal{H}\\ & y \mapsto (\varphi_{YN}^1(y), \varphi_{YN}^2(y), \dots, \varphi_{YN}^n(y)),\end{aligned}$$

we have

$$\begin{aligned}\sum_{j=1}^N \alpha_j \tilde{k}_M((x,y),(x_j,y_j)) &= \sum_{j=1}^N \alpha_j \sum_{i=1}^n c_i k_i^X(x,x_j) k_i^Y(y,y_j) g(x_j,y_j)\\ &= \sum_{i=1}^n \sum_{j=1}^N \alpha_j c_i k_i^X(x,x_j) k_i^Y(y,y_j) g(x_j,y_j)\\ &= \sum_{i=1}^n \langle \varphi_X^i(x), \varphi_{YN}^i(y)\rangle_{\mathcal{H}_i}\\ &= \langle \varphi_X(x), \varphi_{YN}(y)\rangle_{\mathcal{H}},\end{aligned}$$

where the inner product in $\mathcal{H}$ is naturally defined as $\sum_{i=1}^n \langle \cdot, \cdot \rangle_{\mathcal{H}_i}$. ■

We prove that $\bar{k}_M$ is an S-kernel on the basis of Lemma 8 and Lemma 7:

Proof Given a function $k(x,y)$ in $\mathcal{K}_M$, there is a sequence $\{k_N(x,y)\}$ in $\mathcal{K}_M$ such that

$$\begin{aligned}
k_N(x,y) &= \sum_{j=1}^{N} \alpha_j \bar{k}_M((x,y),(x_j,y_j)) \\
&= \sum_{j=1}^{N} \alpha_j g(x,y) \sum_{i=1}^{n} c_i k_i^X(x,x_j) k_i^Y(y,y_j) g(x_j,y_j) \\
&= g(x,y) \sum_{j=1}^{N} \alpha_j \tilde{k}_M((x,y),(x_j,y_j)); \\
k(x,y) &= \lim_{N\to\infty} k_N(x,y).
\end{aligned}$$

We try to prove that $k(x,y)$ is an S-function. Let $\tilde{k}_N(x,y) = \sum_{j=1}^{N} \alpha_j \tilde{k}_M((x,y),(x_j,y_j))$, $k(x,y) = \lim_{N\to\infty} k_N(x,y) = \lim_{N\to\infty} \tilde{k}_N(x,y) g(x,y) = \tilde{k}(x,y) g(x,y)$, where $\tilde{k}(x,y) = \lim_{N\to\infty} \tilde{k}_N(x,y)$.

According to Lemma 2, we only have to prove that $\tilde{k}(x,y)$ is an S-function. From Lemma 8, we know $\tilde{k}_N(x,y) = \langle \varphi_X(x), \varphi_{YN}(y) \rangle_{\mathcal{H}}$, where $\mathcal{H} = \mathcal{H}_1 \times \mathcal{H}_2 \times \ldots \mathcal{H}_n$ is a Hilbert space and $\mathcal{H}_i$ is the Hilbert space of the feature mapping of $\sqrt{c_i} k_i^X(\cdot,\cdot)$.

$$\varphi_{YN}(y) = (\varphi_{NY}^1(y), \varphi_{NY}^2(y), \ldots, \varphi_{NY}^n(y)),$$

and we define $\mathcal{H}_Y = \mathcal{H}_Y^1 \times \ldots \mathcal{H}_Y^n$, $\mathcal{H}_X = \mathcal{H}_X^1 \times \ldots \mathcal{H}_X^n$, and

$$\begin{aligned}
\Gamma_N : & \mathcal{H}_Y \to \mathcal{H}_X \\
& z = (z_1, \ldots, z_n) \mapsto \Gamma_N(z) = (\Gamma_N^1 z_1, \ldots, \Gamma_N^n z_n),
\end{aligned}$$

where $\mathcal{H}_X^i$ and $\mathcal{H}_Y^i$ are the Hilbert spaces with respect to feature mappings $\psi_X^i(\cdot)$ and $\psi_Y^i(\cdot)$ of Mercer kernels $\sqrt{c_i} k_i^X$ and $\sqrt{c_i} k_i^Y$, respectively, $\langle \cdot,\cdot \rangle_{\mathcal{H}_Y^i}$ is the inner product defined in $\mathcal{H}_Y^i$, and $\Gamma_N^i(z_i) = \sum_{j=1}^{N} \alpha_j g(x_j,y_j) \psi_X^i(x_j) \langle \psi_Y^i(y_j), z_i \rangle_{\mathcal{H}_Y^i}$.

Then the key point we need to prove is that $\{\Gamma_N\}$ is a Cauchy sequence. $\forall z \in \mathcal{H}_Y$,

$$\| \Gamma_N(z) \|_{\mathcal{H}_X}^2 = \sum_{i=1}^{n} \sum_{k,j=1}^{N} \alpha_k \alpha_j g(x_k,y_k) g(x_j,y_j) \sqrt{c_i} k_i^X(x_k,x_j) \langle \psi_Y^i(y_k), z_i \rangle_{\mathcal{H}_Y^i} \langle \psi_Y^i(y_j), z_i \rangle_{\mathcal{H}_Y^i}.$$

Using the conclusion given by Lemma 7, we have

$$\begin{aligned}
& \sum_{i=1}^{n} \sum_{k,j=1}^{N} \alpha_k \alpha_j g(x_k,y_k) g(x_j,y_j) \sqrt{c_i} k_i^X(x_k,x_j) \langle \psi_Y^i(y_k), z_i \rangle_{\mathcal{H}_Y^i} \langle \psi_Y^i(y_j), z_i \rangle_{\mathcal{H}_Y^i} \\
\leqslant & \sum_{i=1}^{n} \sum_{k,j=1}^{N} \alpha_k \alpha_j g(x_k,y_k) g(x_j,y_j) c_i k_i^X(x_k,x_j) k_i^Y(y_k,y_j) \langle z_i, z_i \rangle_{\mathcal{H}_Y^i} \\
\leqslant & \left(\sum_{i=1}^{n} \sum_{k,j=1}^{N} \alpha_k \alpha_j g(x_k,y_k) g(x_j,y_j) c_i k_i^X(x_k,x_j) k_i^Y(y_k,y_j) \right) \left(\sum_{i=1}^{n} \langle z_i, z_i \rangle_{\mathcal{H}_Y^i} \right).
\end{aligned}$$

Thus,

$$\| \Gamma_N(z) \|^2_{\mathcal{H}_X} \leqslant \sum_{i=1}^{n} \sum_{k,j=1}^{N} \alpha_k \alpha_j g(x_k, y_k) g(x_j, y_j) c_i k_i^X(x_k, x_j) k_i^Y(y_k, y_j) \| z \|^2_{\mathcal{H}_Y} .$$

Therefore,

$$\| \Gamma_N \|^2 \leqslant \sum_{i=1}^{n} \sum_{k,j=1}^{N} \alpha_k \alpha_j g(x_k, y_k) g(x_j, y_j) c_i k_i^X(x_k, x_j) k_i^Y(y_k, y_j) = \sum_{k,j=1}^{N} \alpha_k \alpha_j \bar{k}_M((x_k, y_k), (x_j, y_j)).$$

Note that $\sum_{k,j=1}^{N} \alpha_k \alpha_j \bar{k}_M((x_k, y_k), (x_j, y_j))$ is just the square of the norm of $k_N(x,y)$ in $\mathcal{K}_M$. From the fact that $\{k_N(x,y)\}$ is a Cauchy sequence in $\mathcal{K}_M$, we know that $\{\Gamma_N\}$ is also a Cauchy sequence. Then there is a linear operator Γ which satisfies $\Gamma = \lim_{N\to\infty} \Gamma_N$. The convergence is in the norm. Thus, for every $(x,y) \in \mathcal{X} \times \mathcal{Y}$, $\tilde{k}(x,y) = \lim_{N\to\infty} \tilde{k}_N(x,y) = \langle \varphi_X(x), \varphi_Y(y) \rangle_{\mathcal{H}}$, where φ_Y is given by

$$(\Gamma^1 \psi_Y^1, \ldots, \Gamma^n \psi_Y^n).$$

■

References

J. Abernethy, F. Bach, T. Evgeniou, and J.P. Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *JMLR '09*, 10:803–826, 2009.

S. Agarwal and P. Niyogi. Stability and generalization of bipartite ranking algorithms. In *COLT'05*, pages 32–47, 2005.

E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06*, pages 19–26, 2006.

F. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *NIPS'08*, pages 105–112, 2008.

F. Bach, G. Lanckriet, and M. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML'04*, 2004.

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *ICML '04*, pages 65–72, 2004.

D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD'00*, pages 407–416, 2000.

A. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, D. Metzler, L. Riedel, and J. Yuan. Online expansion of rare queries for sponsored search. In *WWW '09*, pages 511–520, 2009.

C. Burges, R. Ragno, and Q. Le. Learning to rank with nonsmooth cost functions. In *NIPS'06*, pages 395–402. 2006.

Y. Cao, J. Xu, T.Y. Liu, H. Li, Y. Huang, and H.W. Hon. Adapting ranking SVM to document retrieval. In *SIGIR '06*, pages 186–193, 2006.

Z. Cao, T. Qin, T.Y. Liu, M.F. Tsai, and H. Li. Learning to rank: From pairwise approach to listwise approach. In *ICML '07*, pages 129–136, 2007.

C. Cortes. Invited talk: Can learning kernels help performance? In *ICML'09*, page 161, 2009.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. on Information Theory*, 13(1):21–27, 1967.

K. Crammer and Y. Singer. Pranking with ranking. In *NIPS'01*, pages 641–647, 2001.

N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR' 07*, page 246, 2007.

H. Cui, J. Wen, J. Nie, and W. Ma. Query expansion by mining user logs. *IEEE Trans. on Knowl. and Data Eng.*, 15(4):829–839, 2003.

S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *JASIS*, 41:391–407, 1990.

S. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(4):325–327, April 1976.

Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *JMLR'03*, 4:933–969, 2003.

T. Gartner, P. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *COLT '03*, page 129, 2003.

D. Grangier and S. Bengio. A discriminative kernel-based model to rank images from text queries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1371–1384, 2008.

D. Haussler. Convolution kernels on discrete structures. *Technical Report UCSC-CRL-99-10, Computer Science Dept., UC Santa Cruz.*, 1999.

R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. *NIPS'99*, pages 115–132, 1999.

T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR' 99*, pages 50–57, 1999.

T. Hofmann, B. Scholkopf, and A.J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36(3):1171, 2008.

K. Jarvelin and J. Kekalainen. Ir evaluation methods for retrieving highly relevant documents. In *SIGIR' 00*, pages 41–48, 2000.

T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02*, pages 133–142, 2002.

H. Kashima, K. Tsuda, and A. Inokuchi. Kernels for graphs. *Kernel Methods in Computational Biology*, pages 155–170, 2004.

N. Koide and Y. Yamashita. Asymmetric kernel method and its application to Fisher's discriminant. In *ICPR '06*, pages 820–824, 2006.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. In *ICML'02*, pages 323–330, 2002.

T.Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, 2009. ISSN 1554-0669.

C. Micchelli and M. Pontil. Learning the kernel function via regularization. *JMLR'05*, 6:1099–1125, 2005.

M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *SIGIR '98*, pages 206–214, 1998.

C.S. Ong, A.J. Smola, and R.C. Williamson. Learning the kernel with hyperkernels. *JMLR '05*, 6: 1043–1071, 2005.

J.M. Ponte and W.B. Croft. A language modeling approach to information retrieval. In *SIGIR' 98*, pages 275–281, 1998.

S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *TREC*, 1994.

C. Rudin, C. Cortes, M. Mohri, and R. E. Schapire. Margin-based ranking meets boosting in the middle. In *COLT'05*, pages 63–78, 2005.

G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Readings in Information Retrieval*, pages 355–364, 1997.

G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986. ISBN 0070544840.

B. Schölkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. the MIT Press, 2002.

V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8.

M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *ICML'09*, page 134, 2009.

S.V.N. Vishwanathan and A.J. Smola. Binet-cauchy kernels. In *NIPS '04*, 2004.

C. Watkins. Dynamic alignment kernels. In *NIPS '99*, 1999.

J.R. Wen, J.Y. Nie, and H.J. Zhang. Query clustering using user logs. *ACM Trans. Inf. Syst.*, 20(1): 59–81, 2002. ISSN 1046-8188.

J. Xu and W.B. Croft. Query expansion using local and global document analysis. In *SIGIR '96*, pages 4–11, 1996.

J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *SIGIR’ 07*, pages 391–398, 2007.

J. Xu, H. Li, and Z.L. Zhong. Relevance ranking using kernels. In *AIRS ’10*, 2010.

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

S.K. Zhou. Trace and determinant kernels between matrices. In *NIPS ’04*, 2004.

Z.M. Zhuang and S. Cucerzan. Re-ranking search results using query logs. In *CIKM ’06*, pages 860–861, 2006.