Modeling Term Associations for Probabilistic Information Retrieval

JIASHU ZHAO, JIMMY XIANGJI HUANG, and ZHENG YE, Information Retrieval and Knowledge Management Research Lab, York University

7

Traditionally, in many probabilistic retrieval models, query terms are assumed to be independent. Although such models can achieve reasonably good performance, associations can exist among terms from a human being's point of view. There are some recent studies that investigate how to model term associations/dependencies by proximity measures. However, the modeling of term associations theoretically under the probabilistic retrieval framework is still largely unexplored. In this article, we introduce a new concept *cross term*, to model term proximity, with the aim of boosting retrieval performance. With cross terms, the association of multiple query terms can be modeled in the same way as a simple unigram term. In particular, an occurrence of a query term is assumed to have an impact on its neighboring text. The degree of the query-term impact gradually weakens with increasing distance from the place of occurrence. We use shape functions to characterize such impacts. Based on this assumption, we first propose a bigram CRoss TErm Retrieval ($CRTER_2$) model as the basis model, and then recursively propose a generalized n-gram CRoss TErm Retrieval ($CRTER_n$) model for n query terms, where $n > 2$. Specifically, a bigram cross term occurs when the corresponding query terms appear close to each other, and its impact can be modeled by the intersection of the respective shape functions of the query terms. For an n-gram cross term, we develop several distance metrics with different properties and employ them in the proposed models for ranking. We also show how to extend the language model using the newly proposed cross terms. Extensive experiments on a number of TREC collections demonstrate the effectiveness of our proposed models.

Categories and Subject Descriptors: H.1.1 [**Models and Principles**]: Systems and Information Theory—*Value of information*; H.2.8 [**Database Management**]: Database Application—*Data mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval models*; I.5.4 [**Pattern Recognition**]: Applications—*Text processing*

General Terms: Theory, Experimentation, Algorithms, Performance

Additional Key Words and Phrases: Cross term, BM25, probabilistic information retrieval, kernel, term association, N-gram

ACM Reference Format:
Zhao, J., Huang, J. X., and Ye, Z. 2014. Modeling term associations for probabilistic information retrieval. *ACM Trans. Inf. Syst.* 32, 2, Article 7 (April 2014), 47 pages.
DOI: http://dx.doi.org/10.1145/2590988

1. INTRODUCTION AND MOTIVATION

Many traditional Information Retrieval (IR) models assume that query terms are independent of each other. For those models, a document is normally represented as a bag of words/terms and their frequencies. Although traditional retrieval models can achieve

This work is substantially supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, the Early Researcher Award/Premiers Research Excellence Award, and the IBM Shared University Research (SUR) Award. This work is also affiliated with Information Retrieval and Knowledge Management Research Laboratory.
Authors' addresses: J. Zhao, Information Retrieval and Knowledge Management Research Lab, Computer Science and Engineering Department, York University, Canada; email: jessie@cse.yorku.ca; J. X. Huang (corresponding author) and Z. Ye, Information Retrieval and Knowledge Management Research Lab, School of Information Technology, York University, Canada; emails: {jhuang, yezheng}@yorku.ca.

DOI: http://dx.doi.org/10.1145/2590988

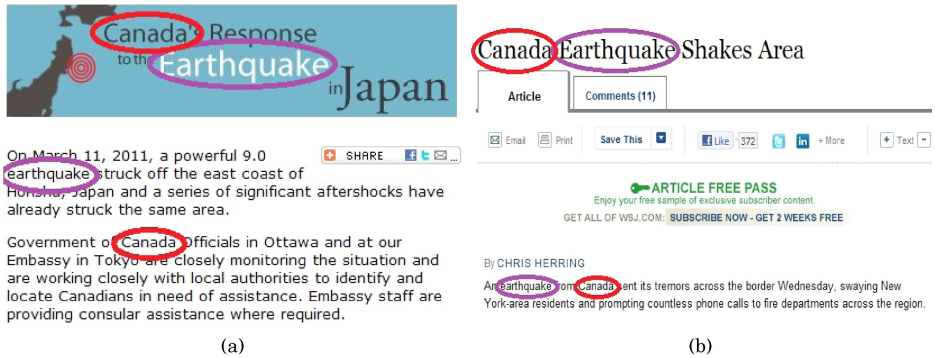


Fig. 1. An example of term association for "Earthquake in Canada".

reasonably good performance in many applications, the corresponding independence assumption has limitations. The terms may have some dependencies on each other. In other words, there might be some implied associations among the query terms. In order to provide more relevant documents to the users, associations among query terms should be also considered.

For example, given an input query "Earthquake in Canada", there exists an association between the query terms. Users are looking for neither other events in Canada nor earthquakes in other countries. In Figure 1, we show two documents with both "earthquake" and "Canada" occurring twice. A traditional IR model would assign these two documents the same weights. However, if we read the documents in detail, we can see that the first document (in Figure 1(a)) reports an earthquake in Japan, and the second document (in Figure 1(b)) reports an earthquake in Canada. Obviously, the second document is more relevant to the user's query than the first document. Therefore, it is necessary to reward the document in which the matched query terms have a stronger association.

Many recent studies in IR consider not only the occurrences of a single query term in documents, but also how the query terms associate with each other. A document is rewarded if the matched query terms have strong associations with each other. These approaches have been shown to be effective in many IR applications [Beigbeder and Mercier 2005; Gao et al. 2004; Hawking and Thistlewaite 1995]. However, the nature of the associations among query terms still awaits further study. Some proximity approaches only consider adjacency [Song and Croft 1999; Srikanth and Srihari 2002], while non-adjacent terms may also have associations. N-gram models [Ahmed and Nurnberger 2009; Mayfield and McNamee 2003] consider n word sequences, which expand the radius of matching. It is yet hard to determine the optimal n, and the complexity usually grows exponentially with the growth of n. Other proximity-based probabilistic weighting models, such as those of Broschart and Schenkel [2008] and Buttcher et al. [2006], add proximity information into their weighting functions in a heuristic manner. However, their experiments are not conclusive, and their retrieval functions are not shown to be effective and robust enough [Tao and Zhai 2007].

In this article, we provide a new perspective for addressing these problems. In particular, we focus on proposing models to integrate the associations among multiple query terms in probabilistic retrieval models. They are the bigram Cross TErm Retrieval model ($CRTER_2$) as the basis model for two-query terms, and the n-gram Cross TErm Retrieval model ($CRTER_n$) for n query terms, where $n > 2$. We also extend cross terms on the basis of language models and propose a bigram Cross TErm Retrieval model ($CRTER_2^{LM}$) under the language model framework. A *cross term* is a pseudo term that

is generated by two or more query terms occurring close to each other. By proposing the concept of cross term, we can model term association by computing the weight of the corresponding cross term in a manner that is theoretically similar to that used for a single query term.

We assume that an occurrence of a query term has an impact on its neighboring text. This impact attenuates when the position of a neighboring term is farther away. If we try to characterize this impact with a mathematic function, intuitively, the function should satisfy the following properties: nonnegative, continuous, symmetric, monotonic, identity (see details in Section 2.1). We use kernel functions that have been brought to proximity retrieval [de Kretser and Moffat 1999; Lv and Zhai 2009] to estimate query-term occurrence impact. They are Gaussian kernel, triangle kernel, circle kernel, and cosine kernel. In this article, we investigate three more kernel functions that satisfy these properties: quartic kernel, Epanechnikov kernel, and triweight kernel. The kernel functions are normalized by scaling between 0 and 1.

We first illustrate the idea of a cross term for two query terms as a special case. We say a bigram cross term $q_{i,j}$ occurs when two query terms, q_i and q_j, occur in close proximity in a document, and therefore their impact shape functions have an intersection. q_i and q_j are called the *generating terms* of $q_{i,j}$. The corresponding impact shape function value at this intersection is bigram cross term $q_{i,j}$'s occurrence value, which ranges from 0 to 1. According to the impact shape function properties, the closer two query terms' occurrences are, the higher the generated bigram cross term occurrence value is. Therefore, the bigram cross term occurrence value indicates to what extent two query terms are correlated.

Many previous term-association approaches treat term association as an extra part, for instance, adding an extra score to the weighting function [Tao and Zhai 2007]. In this article, we evaluate the term association directly by regarding the cross term as a special term in the existing probabilistic weighting models. Terms are the most fundamental element in IR. By manipulating cross terms, we will have a better understanding of term association and a better approach to controlling its effects. In a probabilistic weighting model, some variants change from term to term, namely, the within-document term frequency (tf), the number of documents containing the term (nd), and the within-query term frequency (qtf), respectively. We define the corresponding variants for cross terms: $tf(q_{i,j}, D)$, $nd(q_{i,j})$, and $qtf(q_{i,j})$. For $tf(q_{i,j}, D)$, as a pseudo term, the traditional counting method of the occurrences in a document does not make much sense, especially when we aim to give more weight to occurrences of cross terms where the generating query terms are closer. Instead of accumulating the number of occurrences, we accumulate a cross term's value in a document as its $tf(q_{i,j}, D)$, which is small when q_i and q_j are far away, and large when q_i and q_j are close to each other. Please note that $tf(q_{i,j}, D)$ is always smaller than the number of occurrences of a cross term in the document. In order to balance other variants with $tf(q_{i,j}, D)$, we define $nd(q_{i,j})$ and $qtf(q_{i,j})$, correspondingly. $nd(q_{i,j})$ is the accumulated cross term average value on each document over the collection. In a query, since it is normally short and only contains query terms, we assume that terms in a query are densely distributed. If two terms exist in a query, they are regarded as being adjacent. Then $qtf(q_{i,j})$ is a simplified case of within-document frequency by treating the query as a document.

We further extend the concept of cross term for multiple terms in order to be able to capture the association of more than two query terms. When there are n query terms ($n > 2$), there is no single intersection where all the term impact functions cross each other. Therefore, we have to find an alternative way to define an n-gram cross term. A basic component for bigram cross term is the distance between two query terms. In the higher-dimensional case, we propose several advanced distance metrics to characterize

the occurrences of multiple query terms, and thereby define n-gram cross terms. With the defined cross term variants, we integrate cross terms into the traditional BM25 weighting model [Robertson et al. 1996] by treating them as special terms. Cross term weights are computed and linearly combined with query term weights.

In summary, the main contributions of this article are as follows. First, we propose a novel concept of cross term to model multiple query-term association. Second, we incorporate such cross terms into the probabilistic retrieval framework and language model framework theoretically, resulting in the proposed n-gram-based term-association model. Third, we introduce a number of functions to compute the statistics of a cross term (e.g., the term frequency and distance of an n-gram cross term). Finally, we conduct extensive experiments to validate the proposed models on a relatively large set of representative TREC collections and suggest empirical settings of important parameters in our models.

The remainder of this article is organized as follows. In Section 2, we introduce the concept of cross term, define its variants, and extend the cross term idea for modeling multi-term associations. In Section 3, we propose the n-gram cross term retrieval ($CRTER_n$) model utilizing both BM25 and language model as the basic weighting function, and analyze the algorithm and its computational complexity. In Section 4, we set up our experimental environment on six TREC collections and describe major state-of-the-art proximity models for comparison. In Section 5, we test the proposed models, analyze the parameters, and compare them with existing major proximity models. In Section 6, we investigate possible reasons for the effectiveness of CRTER models and introduce the practical impact of using cross terms. In Section 7, we discuss prior related work. In Section 8, we conclude with a discussion of our findings and future work.

2. USING CROSS TERM FOR MODELING MULTI-TERM ASSOCIATIONS

We first introduce a new concept, *cross term*, which is not an actual existing term but a pseudo term. The association among query terms is formalized as a cross term. To have a better understanding of cross term, we start with a special case when two query terms are considered, namely, bigram cross term. In particular, we formalize the notion of bigram cross term in Section 2.1 and illustrate how to compute the statistics of a bigram cross term (e.g., term frequency and inverse document frequency) in Section 2.2. Then we further extend the concept of cross term to n-gram in Section 2.3, where the associations among n query terms are investigated. We also present the kernel functions for measuring term proximities in Section 2.5.

2.1. A New Pseudo Term: Bigram Cross Term

In this section, we formally define the notion of a bigram cross term and the method of computing a bigram cross term generated by two query terms. Suppose we have a query, $Q = \{q_1, q_1, ..., q_n\}$ and a document D, where pos is one of the positions of query term q_i in document D. The term q_i will influence the positions between $pos + k$ and $pos - k$. We use an impact function $f_i(pos, k)$ to capture the impact of a matching term q_i at position pos. In order to better describe this impact, $f_i(pos, k)$ should satisfy the following properties.

Property 2.1. Let $f_i(pos, k)$ be the impact function of a query term q_i at position $pos + k$. We assume that $f_i(pos, k)$ follow five properties.

(1) Nonnegative. $f_i(pos, k) > 0$, the impact of a term towards its neighborhood is always nonnegative. We only consider the positive influence among query terms.
(2) Continuous. $|f_i(pos, k) - f_i(pos, k + 1)|$ is small, that is, there is a slight difference between two neighboring positions.

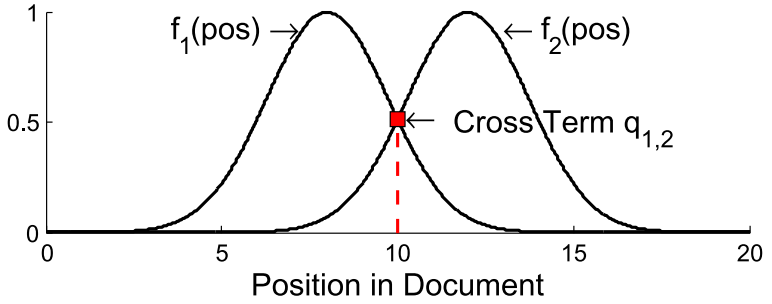


Fig. 2. An example of bigram cross term.

(3) Symmetric. $f_i(pos, -k) = f_i(pos, k)$, the term has the same impact towards two equal-distance positions.
(4) Monotonic. $f_i(pos, k) > f_i(pos, k+1)$, where $k > 0$. The influence decreases with the increase of $|k|$.
(5) Identity. $f_i(pos, 0) = 1$, set one as standard influence.

The nonnegative property ensures that the closely-occurred query terms only boost each other. At the current stage, we do not take into account the negative impact, although some words may have negative influence, such as "no" and "without". In this article, we will not discuss this issue, which could be our future work. The continuous property is intuitive in that the impact changes gradually. In practice, the domain is the cartesian product of natural numbers and integers, that is, *pos* is a natural number and k is an integer. The symmetric property is a compromise for the lack of prior knowledge towards the query terms. The semantic meaning of words could be very complicated to simulate. Therefore, we simplify the individual meanings by assuming their impact functions are identical and symmetric. The monotonic property and the symmetric property ensure that a closer position always has higher impact. The identify property ensures that the impact function is normalized to the range of 0 to 1.

Gaussian kernel, circle kernel, and triangle kernel are widely used kernel functions and satisfy all of the preceding properties [Lv and Zhai 2009]. We will discuss more about various kernel functions in Section 2.5. Without previous domain knowledge of a query and a collection, we assume that the query terms are identically distributed, that is, the query terms have the same impact shape functions. When two query terms are close enough, their impact shape functions will join. According to the preceding properties, we can see that the point of intersection will have a higher value when two query terms occur closer, and the value of the intersection will be lower when two query terms are farther away. The point of intersection of the impact functions reflects the association between these two query terms. We create a new concept of bigram cross term to quantify the association between two query terms. This bigram cross term will be generated when two query term impact functions have an intersection.

Definition 2.2. Given two query terms q_i and q_j, if there exists at least one point of intersection for impact functions $f_i(pos_1, k_1)$ and $f_j(pos_2, k_2)$, we say that a *bigram cross term* occurs, denoted as $q_{i,j}$. We call q_i and q_j *generating terms* of $q_{i,j}$.

Definition 2.3. When a bigram cross term $q_{i,j}$ occurs, the *bigram cross term value* is the impact function's value at the intersection.

A bigram cross term is always located in the middle of its generating query terms, and has higher value when the query terms are closer to each other. Figure 2 shows

an example of a bigram cross term. Two query terms, q_1 and q_2, are located at the 8th position and the 12th position in the document. More intuitively, we adopt the Gaussian kernel as the impact shape function. Their impact shape functions are located at $f_1(pos_1, 2)$ and $f_2(pos_2, -2)$, respectively, in the document. We can see that two shape functions cross over each other, and there exists an intersection at the 10th position. There is no threshold incorporated into the definitions of bigram cross term and its value. For a continuous kernel function, there would always be an intersection if two terms occur in one document. For some kernel functions, its value naturally becomes 0 when a position is far enough from the center point.

2.2. Estimations for Bigram Cross Term Variants

Based on the previous definition of a bigram cross term, we can see that it is impossible to evaluate it the same as a regular query term. In this section, we propose reestimating the variants of a bigram cross term. Several term-dependent variants for this pseudo term are defined accordingly.

2.2.1. Within-Document Frequency Estimation of Bigram Cross Term. Here we will define the counting method for the frequency. The within-document frequency is the rate at which a term occurs in a document. For a single query term, its frequency in document D equals the number of times it occurs in D. There are two reasons that we need to redefine the within-document frequency for a bigram cross term. First, simply counting the occurrences of a cross term can not show the degree of the association between two generating query terms. For example, if two query terms q_i and q_j's impact functions have one point of intersection in both D_1 and D_2 with values 0.1 and 0.9, respectively, the occurrences of $q_{i,j}$ in both D_1 and D_2 are 1. However, we can see that there is a stronger association between q_i and q_j in D_2 than that in D_1. Second, the influence of $q_{i,j}$ is overemphasized, if we use the value of occurrence as cross term within-document frequency. For example, if two query terms q_i and q_j's impact functions in D have 5 points of intersections with values of $\{0.1, 0.2, 0.1, 0.3, 0.1\}$, then the occurrences of $q_{i,j}$ in D equals 5. However, the influence of each occurrence of $q_{i,j}$ is lower than the influence of a query term.

We introduce a new estimation of a bigram cross term within-document frequency. Naturally we adopt the bigram cross term value in estimating its within-document frequency. Suppose the positions of q_i in a document are $\{pos_{1,i}, pos_{2,i}, ..., pos_{tf_i,i}\}$, where tf_i is the term frequency of q_i. Correspondingly, the positions of q_j in the document are $\{pos_{1,j}, pos_{2,j}, ..., pos_{tf_j,j}\}$, where tf_j is the term frequency of q_j. Then the within-document term frequency of $q_{i,j}$ is defined as follows.

Definition 2.4. The frequency of $q_{i,j}$ in D is the accumulation of $q_{i,j}$'s value.

$$tf\left(q_{i,j}, D\right) = \sum_{k_1=1}^{tf_i} \sum_{k_2=1}^{tf_j} Kernel\left(\frac{1}{2} dist(pos_{k_1,i}, pos_{k_2,j})\right), \tag{1}$$

where tf is the term frequency of $q_{i,j}$ in D, $Kernel(\cdot)$ is the kernel function adopted in query term's impact function, and $dist(\cdot)$ is the distance between two positions.

$$dist(pos_{k_1,i}, pos_{k_2,j}) = |pos_{k_1,i} - pos_{k_2,j}|. \tag{2}$$

Please note that the frequency of a bigram cross term might not be an integer. Meanwhile, various kernel functions will be studied in Section 2.5.

2.2.2. Document-Frequency Estimation of Bigram Cross Term. To evaluate the number of documents containing a bigram cross term $q_{i,j}$, it is not reasonable to simply count

how many documents in which $q_{i,j}$ occurs. The contribution from a query term and a bigram cross term is different. For a query term q_i, an occurrence means its frequency accumulates 1, and the number of documents containing q_i is

$$nd(q_i) = \sum_{D \in Index} \mathbf{1}_{\{qi \in D\}},$$

where $\mathbf{1}_{\{qi \in D\}}$ is an indicator function, which is equal to 1 if $q_i \in D$ and equal to 0 otherwise. On the other hand, an occurrence of a bigram cross term adds a value less than 1 to its frequency. A bigram cross term's value could be various and ranges from 0 to 1. For example, if a bigram cross term $q_{i,j}$ occurs only once in D with a value of 0.03 at this occurrence, it's term frequency equals 0.03 according to Formula (1). In this case, there is a very small amount of association between q_i and q_j in D. Therefore, D is more likely to contribute a value less than 1 to $nd(q_{i,j})$. We accumulate the average value of $q_{i,j}$ on each document in $nd(q_{i,j})$, as shown in Definition 2.5.

Definition 2.5. The number of documents containing a bigram cross term $q_{i,j}$ is the sum of $q_{i,j}$'s average value on each document, shown as follows,

$$nd(q_{i,j}) = \sum_{D \in Index, Occur(q_{i,j}, D) \neq 0} \frac{tf(q_{i,j}, D)}{Occur(q_{i,j}, D)}, \tag{3}$$

where *Occur* is the number of occurrence of $q_{i,j}$, which is

$$Occur(q_{i,j}, D) = \sum_{k_1=1}^{tf_i} \sum_{k_2=1}^{tf_j} \mathbf{1}_{\{Kernel(\frac{1}{2} dist(pos_{k_1,i}, pos_{k_2,j})) \neq 0\}}.$$

2.2.3. Within-Query Frequency Estimation of Bigram Cross Term. To evaluate a bigram cross term's within-query term frequency, we can track each position of q_i and q_j the same way as within-document frequency by the sum of all possible intersections. Moreover, different from in documents, query terms distribute densely in a query, so we can assume that query terms are adjacent to each other and let $dist(q_i, q_j)$ =1.

Definition 2.6. The within-query frequency of $q_{i,j}$ is

$$\begin{aligned} qtf(q_{i,j}) &= Kernel(\frac{1}{2} \cdot dist(q_i, q_j)) \cdot min\{(qtf(q_i), qtf(q_j)\} \\ &= Kernel(\frac{1}{2}) \cdot min\{(qtf(q_i), qtf(q_j)\}, \end{aligned} \tag{4}$$

where $qtf(q_i)$ and $qtf(q_j)$ are within query-term frequencies of q_i and q_j, and *Kernel* is the same kernel function utilized in $tf(q_{i,j}, D)$.

2.3. A New Pseudo Term: N-Gram Cross Term

In the previous section, we have defined the concept of bigram cross term, which is generated from two query terms. Here we further discuss the association among n query terms ($n > 2$). When there are n query terms, there doesn't exist an intersection where all the term impact functions cross each other. Figure 3 shows an example when there are multiple query terms that occur in one document. f_1, f_2, and f_3 represent the impact functions for the occurrences of three query terms. f_1 and f_2 have one intersection. f_2 and f_3 have another intersection. However, we can see that there is no nonzero point through which three query term impact functions all pass, no matter where these query terms occur in the document. Therefore, we have to find an alternative way to define n-gram cross terms. As we can see from the previous section,

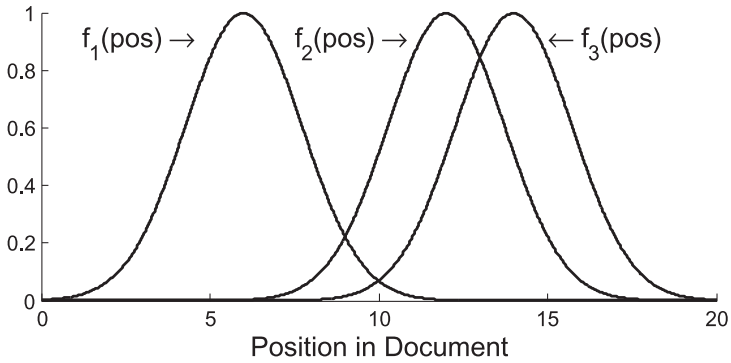


Fig. 3. An example of cross term by multiple query terms.

a basic component in all the definitions is the distance between two query terms. In the higher-dimensional case, we utilize the distance for n query-term occurrences to define the n-gram cross term. A formal definition of n-gram cross term is shown in Definition 2.7.

Definition 2.7. Given n query terms $q_{i_1}, q_{i_2}, ..., q_{i_n}$ occurring at positions pos_{k_1,i_1}, $pos_{k_2,i_2}, ..., pos_{k_n,i_n}$, an *n-gram cross term* occurs if its value $Kernel\left(\frac{1}{2}dist(pos_{k_1,i_1}, pos_{k_2,i_2}, ..., pos_{k_n,i_n})\right)$ is larger than 0. We call an n-gram cross term *nCT*, denoted as $q_{i_1,i_2,...,i_n}$.

In Definition 2.7, $dist(\cdot)$ is a distance metric for n query-term occurring positions which will be described in Sections 2.4.1–2.4.3, and $Kernel(\cdot)$ is the kernel function adopted in the query-term impact function, which will be presented in Section 2.5.

2.4. Estimations for N-Gram Cross-Term Variants

We have defined how to do the counting for bigram cross term in Sections 2.2.1, 2.2.2, and 2.2.3. One identical part in Formulas (1), (3), and (4) is the distance between the co-occurring positions of two query terms. We redefine the variants for an nCT by integrating the distance among the co-occurring positions of n query terms.

Definition 2.8. Given a nCT, $q_{i_1,i_2,...,i_n}$, its variants are defined as follows.

— The within-document nCT frequency,

$$tf\left(q_{i_1,i_2,...,i_n}, D\right) = \sum_{k_1=1}^{tf_1} \sum_{k_2=1}^{tf_2} ... \sum_{k_n=1}^{tf_n} Kernel\left(\frac{1}{2}dist(pos_{k_1,i_1}, pos_{k_2,i_2}, ..., pos_{k_n,i_n})\right), \quad (5)$$

where $tf\left(q_{i_1,i_2,...,i_n}, D\right)$ is the term frequency of $q_{i_1,i_2,...,i_n}$ in D, and tf_j is the term frequency of query term q_{i_j}.

— The number of documents containing $q_{i_1,i_2,...,i_n}$,

$$nd(q_{i_1,i_2,...,i_n}) = \sum_{D \in Index, Occur(q_{i_1,i_2,...,i_n}, D) \neq 0} \frac{tf(q_{i_1,i_2,...,i_n}, D)}{Occur(q_{i_1,i_2,...,i_n}, D)}, \quad (6)$$

where $Occur(q_{i_1,i_2,...,i_n}, D)$ is the number of occurrence of $q_{i_1,i_2,...,i_n}$, which is

$$Occur(q_{i_1,i_2,...,i_n}, D) = \sum_{k_1=1}^{tf_1} \sum_{k_2=1}^{tf_2} ... \sum_{k_n=1}^{tf_n} \mathbf{1}_{\{Kernel(\frac{1}{2}dist(pos_{k_1,1},pos_{k_2,1},...,pos_{k_n,n}))\neq 0\}}.$$

— The within-query nCT frequency,

$$\begin{aligned} qtf(q_{i_1,i_2,...,i_n}) &= Kernel(\frac{1}{2} \cdot dist(q_{i_1}, q_{i_2}, ..., q_{i_n})) \cdot min\{qtf(q_{i_1}), qtf(q_{i_2}), ..., qtf(q_{i_n})\} \\ &= Kernel(\frac{1}{2}) \cdot min\{qtf(q_{i_1}), qtf(q_{i_2}), ..., qtf(q_{i_n})\}, \end{aligned} \tag{7}$$

where $qtf(q_{i_1,i_2,...,i_n})$ is the within query-term frequency of $q_{i_1,i_2,...,i_n}$, and $qtf(q_{i_j})$ is the query-term frequency of query term q_{i_j}. Here we assume that query terms are adjacent to each other and let $dist(q_{i_1}, q_{i_2}, ..., q_{i_n}) = 1$.

2.4.1. Lp-Norm Distances for N Terms. Traditional distance definitions evaluate how far away two points are. We newly define several distances for multi-terms in order to consider the association among n query terms. Suppose we have n query terms q_{i_1}, q_{i_2}, q_{i_3}, ..., q_{i_n} that occur in a document D with frequency of tf_1, tf_2, tf_3, ..., tf_n. For a set of co-occurring query terms, we record the positions of their occurrences in D as $p_1, p_2, p_3, ..., p_n$. The distance metrics for a set of query term occurrences are defined as follows.

One way to naturally define multi-term distance is to learn from distance concepts in geometry. In the Euclidean space $\Re^n$, the p-norm distance evaluates how far apart two points are. For a vector $(x_1, x_2, ..., x_n)$ and a vector $(y_1, y_2, ..., y_n)$, a generalized form of Lp-Norm distance is $||x-y||_p = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$. The Lp-Norm is commonly used for values 1, 2, and infinity. We redefine L1-Norm, L2-Norm, and L∞-Norm distances for query-term co-occurrences as follows.

Definition 2.9 *L1-Norm-Based Distance*.

$$dist_{L1}(p_1, p_2, p_3, ..., p_N) = \sum_{i \neq j} |p_i - p_j|. \tag{8}$$

Definition 2.10 *L2-Norm-Based Distance*.

$$dist_{L2}(p_1, p_2, p_3, ..., p_N) = \sqrt{\sum_{i \neq j} (p_i - p_j)^2}. \tag{9}$$

Definition 2.11 *L∞-Norm-Based Distance*.

$$dist_{L\infty}(p_1, p_2, p_3, ..., p_N) = \max_{i \neq j} |p_i - p_j|. \tag{10}$$

Please note that these definitions are different from the distance definitions in geometry, since we are not evaluating the distance between two vectors. L1-norm-based distance is the average distance of all possible different position pairs. L2-norm-based distance is the root sum squares (RSS) of the difference of all possible position pairs. L∞-norm-based distance is the maximum distance of all possible different position pairs. When n = 2, all these distance metrics have the same form as the distance for the bigram cross term in Formula (2).

2.4.2. Pairwise Distance. We also redefine two previously-utilized distance metrics in IR. Tao and Zhai [2007], define the minimum pair distance and the maximum pair distance as the smallest/largest distance value of all pairs of unique matched query terms among all the occurrences of two query terms . We modify the concept in this article to characterize the pair distance for n co-occurring query terms as follows.

Definition 2.12 *Pairwise Minimum Distance*.

$$dist_{min}(p_1, p_2, p_3, ..., p_N) = \min_{i \neq j} (p_i - p_j). \tag{11}$$

Definition 2.13 *Pairwise Maximum Distance*.

$$dist_{max}(p_1, p_2, p_3, ..., p_N) = \max_{i \neq j} (p_i - p_j). \tag{12}$$

2.4.3. Altitude- and Hypotenuse-Based Distance. The goal of a distance metric for n query terms is to have a lower value when the query terms are close and a higher value when the query terms are far apart. We naturally build two more metrics based on the altitude and hypotenuse of triangles defined as follows.

Definition 2.14 *Altitude-Based Distance*.

$$dist_{alti}(p_1, p_2, p_3, ..., p_n) = \sqrt{\prod_{i=1,2,...,n-1} (p'_{i+1} - p'_i)}, \tag{13}$$

where $p'_1, p'_2, p'_3, ..., p'_n$ is a ranked list generated from $p_1, p_2, p_3, ..., p_n$, such that $p'_1 \leq p'_2 \leq p'_3 \leq ... \leq p'_n$ and $p'_i \in \{p_1, p_2, p_3, ..., p_n\}$ for all i.

Definition 2.15 *Hypotenuse-Based Distance*.

$$dist_{hypo}(p_1, p_2, p_3, ..., p_n) = \sqrt{\sum_{i=1,2,...,n-1} (p'_{i+1} - p'_i)^2}. \tag{14}$$

THEOREM 2.16. *Altitude-based distance and hypotenuse-based distance have the following same properties.*

(*1*) *Nonnegative.* $dist_{alti}(p_1, p_2, p_3, ..., p_n) \geq 0$; $dist_{hypo}(p_1, p_2, p_3, ..., p_n) \geq 0$.
(*2*) *Identity.* $dist_{alti}(p_1, p_2, p_3, ..., p_N) = 0$; $dist_{hypo}(p_1, p_2, p_3, ..., p_n) = 0$, *when* $p_1 = p_2 = ... = p_n$.
(*3*) *Monotonic to* p_1. *When* $p'_1 < p_1$, $dist_{alti}(p'_1, p_2, p_3, ..., p_n) > dist_{alti}(p_1, p_2, p_3, ..., p_n)$ *and* $dist_{hypo}(p'_1, p_2, p_3, ..., p_n) > dist_{hypo}(p_1, p_2, p_3, ..., p_n)$.
(*4*) *Monotonic to* p_N. *When* $p'_n > p_n$, $dist_{alti}(p_1, p_2, p_3, ..., p'_n) > dist_{alti}(p_1, p_2, p_3, ..., p_n)$ *and* $dist_{hypo}(p_1, p_2, p_3, ..., p'_N) > dist_{hypo}(p_1, p_2, p_3, ..., p_n)$.

These properties are the same for both altitude-based distance and hypotenuse-based distance. It is easy to prove that both distances are larger than or equal to zero. When $p_1 = p_2 = ... = p_N$, altitude-based distance and hypotenuse-based distance are equal to zero. Due to the speciality of multiple inputs, it would be impossible to verify the triangle inequality property and symmetry property. The monotonic property means the distance values are larger when the span of the positions is wider, which is intuitively reasonable for a distance function in IR.

THEOREM 2.17. *Besides the preceding properties, altitude-based distance and hypotenuse-based distance have the following different properties.*

(*1*) *Hypotenuse-based distance has a higher value than altitude-based distance:* $dist_{hypo}(p_1, p_2, p_3, ..., p_N) > dist_{alti}(p_1, p_2, p_3, ..., p_N)$.
(*2*) *Suppose the span* $p_N - p_1$ *is fixed, the median positions affect altitude-based distance and hypotenuse-based distance differently: altitude-based distance has a higher value, while hypotenuse-based distance has a lower value, if the median positions tend to spread evenly, that is,* $p_2 - p_1 = p_3 - p_2 = ... = p_N - p_{N-1}$.

For the proof of these two theorems, more details can be found in the Appendix. In this section, we define seven different distance measures from Formula (8) to (14) for multi-terms, which could also be applied in other models and domains. Meanwhile, possible distance measures not listed previously may also be applicable in our proposed models.

2.5. Kernel Functions

The impact of query-term occurrence is characterized by a kernel function. Here we present seven kernel functions that satisfy the query-term impact function properties (Property 2.1). Kernel functions have been studied in IR to characterize term propagation [de Kretser and Moffat 1999]. We first apply four kernel functions that have been brought into IR applications. Among them, the Gaussian kernel is widely used in statistics and machine-learning algorithms, such as Support Vector Machines. Moreover, the triangle kernel, circle kernel, and cosine kernel come from basic genomic graphics, which are applied to estimate proximity-based density distribution for the positional language model [Lv and Zhai 2009]. In addition, since it is difficult to determine which kernel functions can better simulate the term impacts, we investigate more kernel functions in this article. Among all the other kernel functions that satisfy the query-term impact function properties (Property 2.1), we introduce three most commonly-used kernel functions: quartic kernel, Epanechnikov kernel, and triweight kernel to estimate proximity-based term impacts.

— Gaussian kernel:

$$Kernel(u) = exp[\frac{-u^2}{2\sigma^2}], \tag{15}$$

— Triangle kernel:

$$Kernel(u) = (1 - \frac{u}{\sigma}) \cdot \mathbf{1}_{\{u \leq \sigma\}}, \tag{16}$$

— Circle kernel:

$$Kernel(u) = \sqrt{1 - (\frac{u}{\sigma})^2} \cdot \mathbf{1}_{\{u \leq \sigma\}}, \tag{17}$$

— Cosine kernel:

$$Kernel(u) = \frac{1}{2}[1 + cos(\frac{u\pi}{\sigma})] \cdot \mathbf{1}_{\{u \leq \sigma\}}, \tag{18}$$

— Quartic kernel:

$$Kernel(u) = (1 - (\frac{u}{\sigma})^2)^2 \cdot \mathbf{1}_{\{u \leq \sigma\}}, \tag{19}$$

— Epanechnikov kernel:

$$Kernel(u) = (1 - (\frac{u}{\sigma})^2) \cdot \mathbf{1}_{\{u \leq \sigma\}}, \tag{20}$$

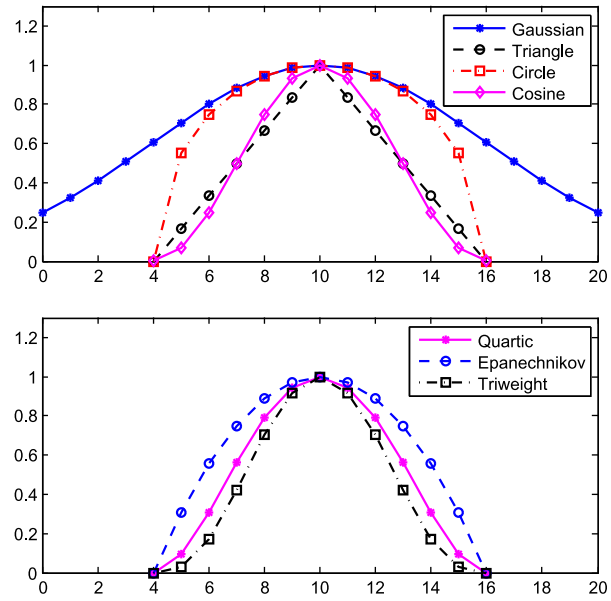


Fig. 4. Kernel functions.

— Triweight kernel:

$$Kernel(u) = (1 - (\frac{u}{\sigma})^2)^3 \cdot \mathbf{1}_{\{u \leq \sigma\}}, \tag{21}$$

where σ is a normalization parameter, and $\mathbf{1}_{\{u \leq \sigma\}}$ is indicator function, which equal 1 if $u \leq \sigma$ and equals 0 otherwise.

Figure 4 shows an example of the included kernel functions. The curves represent the impact of a term occurring at position 10 with various kernel functions. The distance between a position and position 10 is the input u, and parameter σ equals 6 in this example. We can see that most of the kernel functions have similar shapes with different gradients. They have positive values in the interval $u \in (-\sigma, \sigma)$, and a value of 0 otherwise. Parameter σ determines the range of positions considered. A special kernel function is Gaussian kernel, which has positive values even when $u \notin (-\sigma, \sigma)$. With Gaussian kernel, the impact function of query terms will all meet in a document. When query terms are far away, the value at the intersection will be very small. The parameter σ for Gaussian kernel controls the extent of the impact considered. The performance of using all the kernel functions will be investigated in the experiments in Section 5.

3. CROSS-TERM RETRIEVAL (CRTER) MODEL

We now propose cross-term retrieval models based on the concept of cross term defined in Section 2. First, we adopt the BM25 probabilistic model as the basic weighting model in Section 3.1, and propose a bigram CRoss TErm Retrieval ($CRTER_2$) model as a basis case in Section 3.2. Then we recursively define an n-gram CRoss TErm Retrieval ($CRTER_n$) model for $n > 2$ in Section 3.3. Further, we present a detailed analysis on the algorithms and complexities in Section 3.4. BM25 is a classical weighting function employed by the Okapi system [Robertson et al. 1996]. As shown in previous TREC experiments, BM25 usually provides very effective retrieval performance on the TREC collections that are used in Voorhees and Harman [2005] and Ye et al. [2009].

Please note that the proposed model can also easily extend the language model [Ponte and Croft 1998] in a similar way as BM25. Statistical language models have been used in many natural language processing applications. A language model is associated with a document, and the documents are ranked based on the probability of generating the query terms from the document's language model. We propose extending the language model (LM) using cross terms and describe several corresponding well-known open source IR systems in Section 3.5.

3.1. Unigram Model: BM25

In BM25, the weight of a search term is assigned based on its within-document term frequency and query-term frequency. The corresponding weighting function is as follows.

$$w(q_i, D) = \frac{(k_1 + 1) * tf(q_i, D)}{K + tf(q_i, D)} * \frac{(k_3 + 1) * qtf(q_i)}{k_3 + qtf(q_i)} * \log \frac{N - nd(q_i) + 0.5}{nd(q_i) + 0.5}, \quad (22)$$

where $w(\cdot)$ is the weight of a query term q_i in a document. The variants in this formula can be grouped into two categories as follows.

— The first category of variants is query independent. In this category, N is the number of indexed documents in the collection. k_1 and k_3 are tuning constants which depend on the dataset used and possibly on the nature of the queries. K equals $k_1 * ((1 - b) + b * dl/avdl)$, dl is the length of the document, and $avdl$ is the average document length.
— The values of the other group of variants change from term to term. In this category, $nd(q_i)$ is the number of documents containing a specific term. $tf(q_i, D)$ is within-document term frequency. $qtf(q_i)$ is within-query term frequency.

Finally, a document's weight for a query is given by the sum of its weight for each terms in the query,

$$BM25(D) = \sum_{i=1}^{|Q|} w(q_i, D), \quad (23)$$

where w is the term weight obtained from Equation (22), and $|Q|$ is the length of the query Q.

3.2. Bigram Cross Term Retrieval Model: $CRTER_2$

We propose a bigram cross term retrieval ($CRTER_2$) model utilizing the bigram cross term as a special term. In this model, a weighting score is calculated based on both the query terms and the bigram cross terms. The association among query terms is naturally imported into the weighting model by bigram cross terms. A new combined weighting model for a document is

$$CRTER_2(D) = (1 - \lambda_2) \cdot \sum_{1 \le i \le |Q|} w(q_i, D) + \lambda_2 \cdot \sum_{1 \le i < j \le |Q|} w_2(q_{i,j}, D), \quad (24)$$

where $w(\cdot)$ is the weighting function of query terms in the query Q, $w_2(\cdot)$ is bigram cross term $q_{i,j}$'s weight, which has the following form:

$$w_2(q_{i,j}, D) = \frac{(k_1 + 1) * tf(q_{i,j}, D)}{K + tf(q_{i,j}, D)} * \frac{(k_3 + 1) * qtf(q_{i,j})}{k_3 + qtf(q_{i,j})} * \log \frac{N - nd(q_{i,j}) + 0.5}{nd(q_{i,j}) + 0.5}, \quad (25)$$

where we replace the term-dependent variants with the bigram cross term variants defined in Equation (1), Equation (3), and Equation (4).

In the $CRTER_2$ model, λ_2 is a parameter balancing the query terms and bigram cross terms. When λ_2 is equal to 0, the retrieval model uses query terms only, which is the standard BM25 weighting model. When λ_2 is equal to 1, the retrieval model uses bigram cross terms only. Since the weights of query terms and bigram cross terms are normalized independently, the value of λ_2 reflects the influence of using bigram cross terms.

3.3. N-Gram Cross Term Retrieval Model: $CRTER_n$

We further extend the BM25 model by considering the association among multiple query terms, namely, the n-gram cross term. An n-gram CRoss TErm Retrieval ($CRTER_n$) model is built recursively based on $CRTER_{n-1}$ and n-gram cross terms $q_{i_1,\ldots,i_n}$ as follows:

$$CRTER_n(D) = (1-\lambda_n)\cdot CRTER_{n-1}(D) + \lambda_n \cdot \sum_{1\le i_1<i_2<\ldots<i_n\le K} w_n(q_{i_1,\ldots,i_n}, D),$$

where w_n is the weight of an n-gram cross term $q_{k_1,\ldots,k_n}$.

$$w_n(q_{i_1,i_2,\ldots,i_n}, D) = \frac{(k_1+1)*tf(q_{i_1,\ldots,i_n}, D)}{K+tf(q_{i_1,\ldots,i_n}, D)} * \frac{(k_3+1)*qtf(q_{i_1,\ldots,i_n})}{k_3+qtf(q_{i_1,\ldots,i_n})} * \log\frac{N-nd(q_{i_1,\ldots,i_n})+0.5}{nd(q_{i_1,\ldots,i_n})+0.5},$$

where we replace the term-dependent variants with the n-gram cross term variants defined in Equation (5), Equation (6), and Equation (7).

In the $CRTER_n$ model, λ_n is a parameter balancing the influence of the n-gram cross term. When λ_n is equal to 0, the association among n query terms is not considered, and $CRTER_n$ becomes $CRTER_{n-1}$. When λ_n is equal to 1, $CRTER_n$ only accumulates the weights of n-gram cross terms. This recursive model has $n-1$ parameters in total: $\lambda_2, \ldots, \lambda_n$. To simplify the parameter optimization process, we only optimize λ_n in $CRTER_n$, and use the $\lambda_2, \ldots, \lambda_{n-1}$ obtained from $CRTER_{n-1}$.

3.4. Algorithm and Time Complexity

In this section, we present the $CRTER_2$ and $CRTER_n$ algorithms with analyses on their time complexities. In order to shorten the extra retrieval time for cross terms, we build an offline vocabulary that stores cross term information. All the possible cross terms in the vocabulary are generated from the queries. We first calculate and permanently store the cross term within-document term frequency and the number of documents containing the cross term. Meanwhile, the variants for query terms are stored in the index as well. It is a common technique used in many existing IR experiments.

In Figure 5, we show the algorithm for $CRTER_2$ model. Suppose the number of query terms in a query Q is $|Q|$, and the total number of documents in an index is $|Index|$. The first eight steps process the query, where the within-query term frequency is computed for both query terms and bigram cross terms. Meanwhile, $nd(q_i)$ and $nd(q_{i,j})$ are obtained from the index. The time complexity of this part is $O(|Q|+|Q|^2)$. The remaining steps are documents processing through the index. Query terms and bigram cross terms are computed separately. In steps 9 to 22, for each document, we read the within-document query term frequency and the within-document bigram cross term frequency from the hard disk, and compute the corresponding term weights. The time complexity of this part is $O(|Index|\cdot(|Q|+|Q|^2))$. In steps 23 to 27, the weights are

1: **for all** $q_i \in Q$ **do**
2: Compute $qtf(q_i)$
3: Get $nd(q_i)$
4: **end for**
5: **for all** $q_i, q_j \in Q$ **do**
6: Compute $qtf(q_{i,j})$
7: Get $nd(q_{i,j})$
8: **end for**
9: $w(D) = 0$
10: $w_2(D) = 0$
11: **for all** $D \in Index$ **do**
12: **for all** $q_i \in Q$ **do**
13: Get $tf(q_i, D)$
14: Compute $w(q_i, D)$
15: $w(D) = w(D) + w(q_i, D)$
16: **end for**
17: **for all** $q_i, q_j \in Q$ **do**
18: Get $tf(q_{i,j}, D)$
19: Compute $w_2(q_{i,j}, D)$
20: $w_2(D) = w_2(D) + w_2(q_{i,j}, D)$
21: **end for**
22: **end for**
23: Normalize $w(D)$
24: Normalize $w_2(D)$
25: **for all** $D \in Index$ **do**
26: Compute $CRTER_2(D) = (1 - \lambda_2) * w(D) + \lambda_2 * w_2(D)$
27: **end for**

Fig. 5. Algorithm for $CRTER_2$.

normalized and linearly combined, where the time complexity is $O(|Index|)$. This algorithm for $CRTER_2$ model has the time complexity as follows

$$O(|Q| + |Q|^2 + |Index| \cdot (|Q| + |Q|^2) + |Index|). \tag{26}$$

We can see from Formula (26) that the processing time for the query in $CRTER_2$ is $O(|Q| + |Q|^2)$ within both the documents and the queries, while a traditional weighting model time complexity for processing the query is $O(|Q|)$. In most of the applications, the queries are usually short. Therefore, the time increased is not very significant.

To analyze the time complexity of $CRTER_n$ model, we first show the algorithm for the $CRTER_3$ model as a special case in Figure 6. In $CRTER_3$, the main procedures are the same as $CRTER_2$ with a few modifications. In steps 9 to 12, $qtf(q_{i_1,i_2,i_3})$ is calculated, and $nd(q_{i_1,i_2,i_3})$ is obtained from the index. In steps 27 to 31, we read $tf(q_{i_1,i_2,i_3})$ from index and compute $w_3(D)$. Finally, $w_3(D)$ is normalized in step 35, and the weight of $CRTER_3$ is calculated based on $W(D)$, $w_2(D)$, and $w_3(D)$ in steps 36 to 38. For the $CRTER_n$ model, we add the corresponding nCT processing procedures similarly. The time complexity increases for query processing within the query and the documents. Therefore, the generalized time complexity for $CRTER_n$ is

$$O(\sum_{l=1}^{l \le n} |Q|^l + |Index| \cdot \sum_{l=1}^{l \le n} |Q|^l + |Index|). \tag{27}$$

for all $q_i \in Q$ **do**
  Compute $qtf(q_i)$
  Get $nd(q_i)$
end for
for all $q_{i_1}, q_{i_2} \in Q$ **do**
  Compute $qtf(q_{i_1,i_2})$
  Get $nd(q_{i_1,i_2})$
end for
for all $q_{i_1}, q_{i_2}, q_{i_3} \in Q$ **do**
  Compute $qtf(q_{i_1,i_2,i_3})$
  Get $nd(q_{i_1,i_2,i_3})$
end for
$w(D) = 0$
$w_2(D) = 0$
$w_3(D) = 0$
for all $D \in Index$ **do**
  for all $q_i \in Q$ **do**
    Get $tf(q_i, D)$
    Compute $w(q_i, D)$
    $w(D) = w(D) + w(q_i, D)$
  end for
  for all $q_{i_1}, q_{i_2} \in Q$ **do**
    Get $tf(q_{i_1,i_2}, D)$
    Compute $w_2(q_{i_1,i_2}, D)$
    $w_2(D) = w_2(D) + w_2(q_{i_1,i_2}, D)$
  end for
  for all $q_{i_1}, q_{i_2}, q_{i_3} \in Q$ **do**
    Get $tf(q_{i_1,i_2,i_3}, D)$
    Compute $w_3(q_{i_1,i_2,i_3}, D)$
    $w_3(D) = w_3(D) + w_3(q_{i_1,i_2,i_3}, D)$
  end for
end for
Normalize $w(D)$
Normalize $w_2(D)$
Normalize $w_3(D)$
for all $D \in Index$ **do**
  Compute $CRTER_3(D) = (1 - \lambda_3) * ((1 - \lambda_2) * w(D) + \lambda_2 * w_2(D)) + \lambda_3 * w_3(D)$
end for

Fig. 6. Algorithm for $CRTER_3$.

The time complexity increases exponentially with the number of grams considered. When n is fixed, Formula (27) grows polynomially with the increase of query size $|Q|$. Therefore, it is a trade-off between considering a greater number of grams and spending more time for retrieval.

For the queries that have not been processed and the corresponding cross terms that have not been stored in the index, we need to perform preliminary steps to process the cross terms. For $CRTER_2$, we show the algorithm to compute $tf(q_{i,j}, D)$ and $nd(q_{i,j})$

```
1:  for all q_i, q_j ∈ Q do
2:    nd(q_{i,j}) = 0
3:    for all D ∈ Index do
4:    tf(q_{i,j}, D) = 0
5:    Occur(q_{i,j}, D) = 0
6:      for k_1 < tf(q_i, D)&k_2 < tf(q_j, D)
7:        Kernel_temp = Kernel(1/2 |pos_{k_1,i} − pos_{k_2,j}|)
8:        if Kernel_temp ≠ 0
9:          tf(q_{i,j}, D) += Kernel_temp
10:         Occur(q_{i,j}, D) += 1
11:       end if
12:     end for
13:     nd(q_{i,j}) += tf(q_{i,j},D) / Occur(q_{i,j},D);
14:   end for
15: end for
```

Fig. 7. Algorithm for computing $tf(q_{i,j}, D)$ and $nd(q_{i,j})$.

```
1:  for all q_{i_1}, q_{i_2}, ...q_{i_n} ∈ Q do
2:    nd(q_{i_1,...,i_n}) = 0
3:    for all D ∈ Index do
4:    tf(q_{i_1,...,i_n}, D) = 0
5:    Occur(q_{i_1,...,i_n}, D) = 0
6:      for k_1 < tf(q_{i_1}, D) & k_2 < tf(q_{i_2}, D)  & ... & k_n < tf(q_{i_n}, D)
7:        Kernel_temp = Kernel(1/2 dist(pos_{k_1,i_1}, pos_{k_2,i_2}, ..., pos_{k_n,i_n}))
8:        if Kernel_temp ≠ 0
9:          tf(q_{i_1,...,i_n}, D) += Kernel_temp
10:         Occur(q_{i_1,...,i_n} += 1
11:       end if
12:     end for
13:     nd(q_{i_1,...,i_n}) += tf(q_{i_1,...,i_n},D) / Occur(q_{i_1,...,i_n},D);
14:   end for
15: end for
```

Fig. 8. Algorithm for computing $tf(q_{i_1,...,i_n}, D)$ and $nd(q_{i_1,...,i_n})$.

in Figure 7. Here the term positions are stored in the index. For each pair of query terms, the corresponding cross term term frequencies in each document are calculated, and the average term frequencies are added up to be $nd(q_{i,j})$. The total time spent in this process is $O(|Q|^2 \cdot |Index| \cdot \overline{tf}^2)$, where $\overline{tf}$ represents the within-document term frequency. For $CRTER_n$, Figure 8 shows the algorithm for computing $tf((q_{i_1,...,i_n}, D)$ and $nd((q_{i_1,...,i_n})$. The time spent in this process is $O(|Q|^n \cdot |Index| \cdot \overline{tf}^n)$. When n is fixed, the time complexity is a polynomial function to $|Q| \cdot \overline{tf}$ in a real-life setting, since cross terms need to be extracted from the queries dynamically.

Reranking [Cormack et al. 2011; Kurland and Lee 2004] is an approach to reduce the computational complexity and reduce the running time in information retrieval. We rerank the top-$k_{reranking}$ documents instead of the whole collection in our experiments. Then the additional time spent on $CRTER_2$ is $O(k_{reranking} \cdot |Q|^2 \cdot \overline{tf}^2)$ for processing cross terms, and $O(|Q| + |Q|^2 + k_{reranking} \cdot (|Q| + |Q|^2) + k_{reranking})$ for reranking, respectively. Similarly, the overall additional time spent on the $CRTER_n$ is $O(\sum_{l=1}^{l \le n} k_{reranking} \cdot |Q|^2 \cdot \overline{tf}^n)$ for processing all grams of cross terms, and $O(\sum_{l=1}^{l \le n} |Q|^l + k_{reranking} \cdot \sum_{l=1}^{l \le n} |Q|^l + k_{reranking})$ for reranking, respectively. With a fixed constant $k_{reranking}$, the time spent on larger collections will be reduced significantly. We use $k_{reranking} = 2000$ in our experiments. In the future, we will study how to further reduce the computational complexity for the proposed algorithms in Figure 5–8.

3.5. Language-Model-Based Cross Terms Retrieval Models

In order to evaluate the effect of cross terms, we also compare with several state-of-the-art language models. To have a fair comparison, here we extend cross terms on the basis of the language model. In a language model [Zhai and Lafferty 2001], a document is ranked by

$$LM(D) = \sum_{1 \le i \le |Q|} log \frac{P(q_i|D)}{P(q_i|C)}, \tag{28}$$

where $P(q_i|C) = \frac{cf(q_i)}{|C|}$ is the collection language model, C represents the collection, $|C|$ is the total number of terms in the collection, $cf(q_i)$ is the frequency of q_i over the collection, and $P(q_i|D)$ is the document language. Several approaches smooth $P(q_i|D)$ to solve the zero probability problem. Dirichlet smoothing is one of the most popular approaches and is reported to have the best average precision in most cases [Zhai and Lafferty 2001, 2004]. The conditional probability $P(q_i|D)$ is smoothed by

$$P(q_i|D) = \frac{tf(q_i, D) + \mu \frac{cf(q_i)}{|C|}}{|D| + \mu}, \tag{29}$$

where μ is the Dirichlet smoothing parameter. μ is tuned to be optimal in our experiments.

A bigram language model is usually linearly combined to a unigram language model [Song and Croft 1999]. Language model with bigram cross term is

$$CRTER_2^{LM}(D) = (1 - \lambda_2^{LM}) \cdot LM(D) + \lambda_2^{LM} \cdot \sum_{1 \le i < j \le |Q|} log \frac{P(q_{i,j}|D)}{P(q_{i,j}|C)}, \tag{30}$$

where $P(q_{i,j}|D) = \frac{tf(q_{i,j},D) + \mu \frac{cf(q_{i,j})}{|C|}}{|D| + \mu}$ and $P(q_{i,j}|C) = \frac{cf(q_{i,j})}{|C|}$. $tf(q_{i,j}, D)$ is defined in Formula (1), and $cf(q_{i,j})$ is $q_{i,j}$'s frequency over the collection which can be derived by accumulating $tf(q_{i,j})$ over the collection

$$cf(q_{i,j}) = \sum_{D \in C} tf(q_{i,j}, D).$$

Table I. Overview of the TREC Collections Used

Collection Name	# of Docs	Topics	# of Topics
TREC8	528,155	401–450	50
Robust	528,155	301–450 & 601–700	250
AP88-89	164,597	51–100	50
WT2G	247,491	401–450	50
WT10G	1,692,096	451–550	100
.GOV2	25,178,548	701–850	150
Blog06	3,215,171	851–950 & 1000–1050	150

We naturally further extend the language model with n-gram cross terms. The weighting function is recursively defined as

$$CRTER_n^{LM}(D) = (1-\lambda_n^{LM}) \cdot CRTER_{n-1}^{LM}(D) + \lambda_n^{LM} \cdot \sum_{1 \le i_1 < i_2 < \ldots < i_n \le K} \frac{P(q_{i_1,i_2,\ldots,i_n}|D)}{P(q_{i_1,i_2,\ldots,i_n}|C)}, \tag{31}$$

where $P(q_{i_1,i_2,\ldots,i_n}|D) = \frac{tf(q_{i_1,i_2,\ldots,i_n},D)+\mu\frac{cf(q_{i_1,i_2,\ldots,i_n})}{|C|}}{|D|+\mu}$, $P(q_{i_1,i_2,\ldots,i_n}|C) = \frac{cf(q_{i_1,i_2,\ldots,i_n})}{|C|}$, $tf(q_{i_1,i_2,\ldots,i_n},D)$ is defined in Formula (5), and $cf(q_{i_1,i_2,\ldots,i_n})$ is $q_{i_1,i_2,\ldots,i_n}$'s frequency over the collection which can be derived by accumulating $tf(q_{i,j})$ over the collection

$$cf(q_{i_1,i_2,\ldots,i_n}) = \sum_{D \in C} tf(q_{i_1,i_2,\ldots,i_n},D).$$

In the rest of this article, we use the notation $CRTER_2^{LM}$ to represent the language model with bigram cross terms. For implementation, there are several well-known open-source IR systems supporting the language model. For instance, the Lemur project [Lavrenko and Croft 2001; Strohman et al. 2005] develops the Lemur Toolkit and the Indri search engine which combines the inference nets and language modeling in an architecture designed for large-scale applications. The Terrier search engine [Ounis et al. 2006a, 2007] implements indexing and retrieval functionalities, combining ideas from probabilistic theory, statistical analysis, and data compression techniques. $CRTER_2^{LM}$ can be implemented on any of these IR systems. The corresponding settings will be discussed in Section 5.5.

4. EXPERIMENTAL SETTINGS

Here we first present the datasets and the related evaluation measures used in our experiments in Section 4.1. Second, we describe our Okapi experimental platform in Section 4.2. Finally, we describe four major probabilistic and language models for comparison in Section 4.3.

4.1. Datasets and Evaluation Measures

We present six standard TREC collections used in our experiments, the statistics of which are shown in Table I. These collections are diverse in both size and content, which facilitate a thorough evaluation of our proposed models. The TREC8 contains newswire articles from various sources, such as the Financial Times (FT), the Federal Register (FR), etc., which are usually considered as high-quality text data with little noise. Robust has the same data collection as TREC8, and Robust includes more topics 301–450 and 601–700. We use both Robust and TREC8 to test the retrieval models on the same collection with different topic sets in Section 6.4. AP88-89 contains articles

<top>
<num> Number: 725
<title> Low white blood cell count
<desc> Description:
What would cause a lowered white blood cell count?
<narr> Narrative:
A relevant document will describe a condition or disease that causes a lowered white blood cell count. Lowered white blood cell counts caused by HIV infection, bone marrow failure and chemotherapy are relevant. A low count caused by a treatment or medication would also be relevant.
</top>

Fig. 9. An example of a standard TREC topic.

published by Association Press from years 1988 to 1989. The WT2G collection is a 2G-size crawl of Web documents. The WT10G collection is a medium-size crawl of Web documents, which was used in the TREC9 and TREC10 Web tracks. It contains 10GB of uncompressed data. The .GOV2 collection, which has 426GB of uncompressed data, is a crawl from the .gov domain. This collection has been employed in the TREC14 (2004), TREC15 (2005), and TREC16 (2006) Terabyte tracks. The Blog06 collection includes 100,649 blog feeds collected over an 11-week period from December 2005 to February 2006. Following the official TREC settings [Ounis et al. 2006b], we index only the permalinks, which are the blog posts and their associated comments. For all test collections used, each term is stemmed using Porter's English stemmer, and standard English stopwords are removed.

A topic usually contains three topic fields, namely, title, description, and narrative. Figure 9 shows an example of a standard TREC topic. We only use the title topic field that contains very few keywords related to the topic. The title-only queries are usually short, which is a realistic snapshot of real user queries in practice. On each collection, we evaluate our proposed model by a ten-fold cross-validation. The test topics associated to each collection are randomly split into ten equal subsets. In each fold, nine subsets of the test topics are used for training, and the remaining subset is used for testing. The overall retrieval performance is averaged over all ten test subsets of topics.

We use the TREC official evaluation measures in our experiments, namely, the topical MAP on Blog06 [Ounis et al. 2006b] and the mean average precision (MAP) on the other six collections [Voorhees and Harman 2005]. To put emphasis on the top retrieved documents, we also include P@5 and P@20 in the evaluation measures. All statistical tests are based on two-tailed Wilcoxon Matched-pairs Signed-rank test with the significance level of 0.05.

4.2. The Okapi System

In our experiments, we use Okapi BSS (basic search system) [Robertson and Walker 1994] as our main search system and conduct our information retrieval experiments using the improved Okapi system [Fan et al. 2006; Huang and Hu 2009; Huang et al. 2000, 2005, 2006a, 2006b, 2013; Miao et al. 2012; Yin et al. 2013]. Okapi is an information retrieval system based on the probability model of Robertson and Sparck Jones [Beaulieu et al. 1997; Robertson and Walker 1994], which is one of the most established and best-performing systems in information retrieval research and experimentation [Voorhees and Harman 2005]. The retrieval documents are ranked in the order

of their probabilities of relevance to the query. A search term is assigned weight based on its within-document term frequency and query-term frequency. The weighting function used is BM25. BM25 is a classical probabilistic model based on the two-Poisson approximation of the term-frequency distribution. It assigns the relevance score for a document d with respect to a given query Q by Robertson et al. [1996], which is detailed in Section 3.1. In our experiments, the values of k_1, k_3 in Formula (22) default to 1.2 and 8, respectively, which is the recommended setting in Robertson et al. [1996]. The parameter b is set to 0.35, which is shown to be optimal in our preliminary experiments.

4.3. Major Probabilistic and Language Models for Comparison

Here we briefly describe four proximity models used for comparison. We choose two probabilistic BM25-based proximity models, namely, PPM [Song et al. 2011] and BM25TP [Buttcher et al. 2006], as well as two language model (LM)-based term dependency models, namely, MRF [Metzler and Croft 2005] and PLM [Lv and Zhai 2009]. PPM and PLM are based on the same assumptions as our proposed *CRTER* model and use kernel functions to simulate term-influence propagation. BM25TP is a probabilistic BM25-based proximity model with good performance, and it is parameter free. MRF is a one of the most famous LM-based dependency models and was widely investigated by many IR researchers. In experiments, we will compare with these four models, and we will also discuss more BM25-based and LM-based dependency models in the related work (Section 7).

— *Proximity Probabilistic Model (PPM)*. In PPM, a position-dependent term count is related to both the number of occurrences of a term and the term counts propagated from other terms. Each query term has a pseudo-term frequency, which is the original query-term frequency added by accumulating term counts propagated from the nearest occurrences of the other terms.
— *BM25TP*. A proximity accumulator is associated with each query term in BM25PT. In a document, once there is a posting of a query term, both the current query-term and the previous query-term accumulators are incremented (if they are different query terms).
— *Markov Random Fields (MRF)*. MRF uses Markov random fields to model the joint distribution over queries and documents. A graph is constructed to represent the query dependencies to the document, and a set of potential functions over the cliques of this graph are defined. The potential functions are expected to satisfy $\psi(q_{i_1}, .., q_{i_n}, D) > \psi(q'_{j_1}, .., q'_{j_n}, D)$ when $q_{i_1}, .., q_{i_n}$ are much more compatible with document D than the terms $q'_{j_1}, .., q'_{j_n}$. Three variants are considered and linearly combined in MRF: full independency (FI) is a smoothed unigram language model estimate; sequential dependence (SD) estimates all subphrases; full dependence (FD) is a proximity-based estimate which is implemented with the co-occurrences of two or more query terms within a window.
— *Positional Language Model (PLM)*. PLM is also a kernel-based approach in which a term at each position can propagate its occurrence at that position to other positions. The PLM at each position can then be estimated based on all the propagated counts of all the words. A language model is defined for each position of a document.

5. EXPERIMENTAL RESULTS

Here we conduct a series of experiments for our proposed models, $CRTER_2$ and $CRTER_n$, respectively. For $CRTER_2$, we implement it with either BM25 or Dirichlet

LM as the basis model. In order to distinguish them, we denote $CRTER_2^{LM}$ as the LM-based $CRTER_2$ model and $CRTER_2$ as the BM25-based $CRTER_2$ model. We first present the results, investigate the impact of important parameters, and analyze the robustness of the $CRTER_2$ model in Section 5.1, 5.2, and 5.3. Then we compare $CRTER_2$ with two state-of-the-art probabilistic proximity approaches in Section 5.4, and compare $CRTER_2^{LM}$ with two state-of-the-art LM proximity approaches in Section 5.5. Moreover, a group of experiments are conducted to illustrate the $CRTER_n$ in Section 5.6. Finally, a case study to further demonstrate the effectiveness of our proposed models is presented in Section 6.

5.1. Experimental Results of $CRTER_2$

In our evaluation, we first investigate the performance of our proposed bigram cross term retrieval ($CRTER_2$) model compared to the basic weighting model BM25. Specifically, we use BM25 with optimal settings as our baseline. The parameter b is set to 0.35, which is shown to be optimal in our preliminary experiments. The related experimental results are presented in Table II. Seven different kernel functions are applied to instantiate the $CRTER_2$ model, including Gaussian, triangle, circle, cosine, quartic, Epanechnikov, and triweight kernels. All the results are evaluated by MAP, P@5, and P@20. The percentage of how much $CRTER_2$ outperforms BM25 is also listed. The best result obtained on each collection is marked bold. The experiments conducted on .GOV2 and Blog06 are more comprehensive than those presented in Zhao et al. [2011]. As shown by the results, our proposed $CRTER_2$ model outperforms BM25 on all six collections used. The advantage of $CRTER_2$ over BM25 is especially evident on the relatively larger collections, WT10G, .GOV2, and Blog06, where statistically-significant improvement is observed with all seven kernel functions used. Moreover, according to the results in Table II, each kernel function has its advantage on some aspects. There is no single kernel function that can outperform all the other kernel functions on all the datasets. Figure 10 shows the winnings of the kernel functions on different datasets under different evaluation metrics, which further illustrates Table II. We can see that the triangle kernel performs the best (wins seven times) in terms of winning times.

5.2. Parameter Sensitivity

An important issue that may affect the robustness of the $CRTER_2$ model is the sensitivity of its parameters λ (in Equation (26)) and σ (in Equation (4)–(10)) to retrieval performance. The parameter λ balances the influence of the query terms and the bigram cross terms. When λ is equal to 0, the retrieval model uses query terms only, which is the standard BM25 weighting model. When λ is equal to 1, the retrieval model uses bigram cross terms only. Since the weights of query terms and bigram cross terms are normalized independently, the value of λ reflects the influence of using bigram cross terms. The kernel parameter σ controls the range of a query term's impact. When σ is small, a bigram cross term occurs only if its generating term is very close. When σ is large, query terms far away from each other can generate a bigram cross term. But the bigram cross term value will be different according to the distance between query terms.

Figure 11 and Figure 12 plot the evaluation metrics MAP, P@5, and P@20 obtained by $CRTER_2$ over λ values ranging from 0 to 1 on all the datasets. In addition, a group of different settings of σ are applied, namely, $\sigma = 2, 5, 10, 20, 50, 75, 100$. The general tendency on each evaluation metric is similar. As we can see from these two figures, $CRTER_2$'s retrieval performance decreases with large λ values. $CRTER_2$ generally performs well over different datasets when λ falls between 0 and 0.2. Overall, a λ value

Table II. Comparison between BM25 Baseline and $CRTER_2$ with Different Kernel Functions

	Eval Metric	TREC8	AP88-89	WT2G	WT10G	.GOV2	Blog06
BM25	MAP	0.2561	0.2710	0.3156	0.2119	0.3039	0.3246
	P@5	0.4920	0.4360	0.5280	0.3800	0.6134	0.6400
	P@20	0.4000	0.3860	0.3930	0.2670	0.5426	0.5997
$CRTER_2$ Gaussian	MAP	0.2604 (+1.679%)	0.2787 (+2.841%)	0.3354* (+6.274%)	0.2213* (+4.436%)	**0.3342*** (+9.970%)	0.3505* (+7.979%)
	P@5	0.5040 (+2.439%)	0.4520 (+3.670%)	**0.5480** (+3.788%)	**0.4080*** (+7.368%)	**0.6550*** (+6.782%)	0.6560 (+2.500%)
	P@20	0.4190 (+4.750%)	0.3900 (+1.036%)	0.4070 (+3.562%)	0.2775 (+3.933%)	0.5809* (+7.059%)	0.6260* (+4.386%)
$CRTER_2$ Triangle	MAP	**0.2606** (+1.757%)	**0.2789** (+2.915%)	**0.3359*** (+ 6.432%)	0.2207* (+4.153%)	0.3339* (+9.871%)	0.3512* (+8.195%)
	P@5	0.5040 (+2.439%)	0.4520 (+3.670%)	**0.5480** (+3.788%)	**0.4080*** (+7.368%)	0.6537* (+6.570%)	**0.6733*** (+5.203%)
	P@20	0.4190 (+4.750%)	0.3890 (+0.777%)	**0.4100*** (+4.326%)	0.2775 (+3.933%)	0.5792* (+6.745%)	0.6340 (+5.720%)
$CRTER_2$ Circle	MAP	0.2599 (+1.484%)	0.2783 (+2.694%)	**0.3359*** (+ 6.432%)	**0.2227*** (+5.97%)	0.3331* (+9.608%)	0.3515* (+8.287%)
	P@5	0.5040 (+2.439%)	**0.4600*** (+5.505%)	0.5440 (+ 3.030%)	0.4060* (+6.842%)	0.6523* (+ 6.342%)	0.6653 (+3.953%)
	P@20	0.4190 (+4.750%)	0.3890 (+0.777%)	0.4080* (+ 3.817%)	**0.2785*** (+4.307%)	0.5782* (+6.561%)	**0.6357*** (+6.003%)
$CRTER_2$ Cosine	MAP	0.2599 (+1.484%)	**0.2789** (+2.915%)	0.3358* (+ 6.401%)	0.2216* (+4.578%)	0.3339* (+9.872%)	0.3515* (+8.287%)
	P@5	0.5040 (+2.439%)	0.4560 (+4.587%)	0.5440 (+3.030 %)	**0.4080*** (+7.368%)	0.6537* (+6.570%)	**0.6733*** (+5.203%)
	P@20	0.4190 (+4.750%)	**0.3910** (+1.295%)	0.4090 (+ 4.071%)	0.2765 (+3.558%)	**0.5812*** (+7.114%)	0.6343* (+5.770%)
$CRTER_2$ Quartic	MAP	0.2599 (+1.484%)	0.2787 (+2.841%)	0.3352* (+6.210%)	0.2212* (+4.389%)	0.3338* (+9.839%)	**0.3517*** (+8.349%)
	P@5	0.5040 (+2.439%)	0.4560 (+4.587%)	0.5440 (+ 3.030%)	**0.4080*** (+7.368%)	0.6537* (+6.570%)	**0.6733*** (+5.203%)
	P@20	0.4170 (+4.250%)	0.3920 (+1.554%)	**0.4100*** (+ 4.326%)	0.2780* (+4.120%)	0.5805* (+6.985%)	0.6343* (+5.770%)
$CRTER_2$ Epanechnikov	MAP	0.2602 (+1.601%)	0.2787 (+2.841%)	0.3343* (+ 5.925%)	0.2217* (+4.625%)	0.3330* (+9.576%)	0.3512* (+8.195%)
	P@5	**0.5080** (+3.252%)	0.4520 (+3.670%)	0.5440 (+ 3.030%)	**0.4080*** (+7.368%)	0.6523* (+6.342%)	0.6720* (+5.000%)
	P@20	**0.4200*** (+5.000%)	0.3890 (+0.777%)	0.4070 (+ 3.562%)	0.2785* (+4.307%)	0.5802* (+6.930%)	0.6340* (+5.720%)
$CRTER_2$ Triweight	MAP	0.2601 (+1.562%)	0.2788 (+2.878%)	0.3356* (+ 6.337%)	0.2225* (+5.002%)	0.3341* (+9.937%)	**0.3517*** (+8.349%)
	P@5	**0.5080** (+3.252%)	**0.4600*** (+5.505%)	0.5440 (+ 3.030%)	0.4060* (+6.842%)	**0.6550*** (+6.782%)	0.6720* (+5.000%)
	P@20	0.4180 (+4.500%)	**0.3910** (+1.295%)	0.4070 (+ 3.562%)	0.2780* (+4.120%)	0.5809* (+7.059%)	0.6350* (+5.886%)

Note: BM25 parameter b is initialized to be 0.35. All the results are evaluated by MAP, P@5, and P@20. $CRTER_2$ outperforms BM25 on all collections. "*" means the improvements over the BM25 are statistically significant ($p < 0.05$ with Wilcoxon Matched-pairs Signed-rank test).

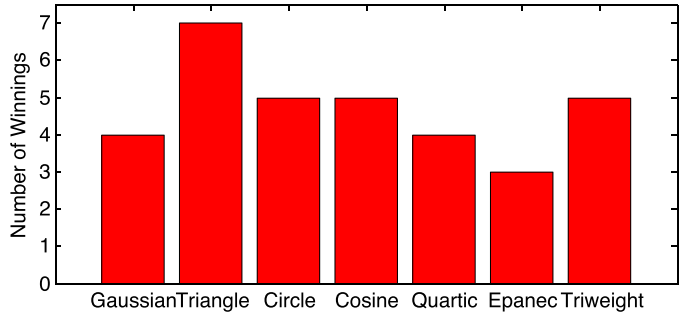
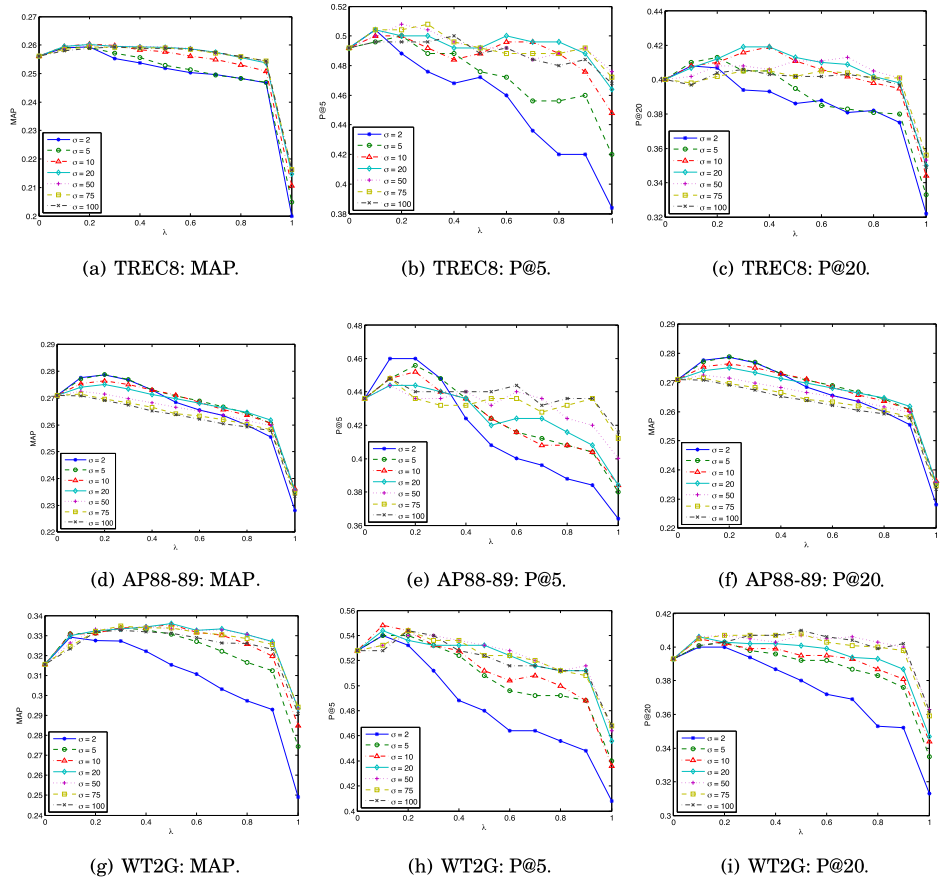


Fig. 10. Comparison among kernel functions.



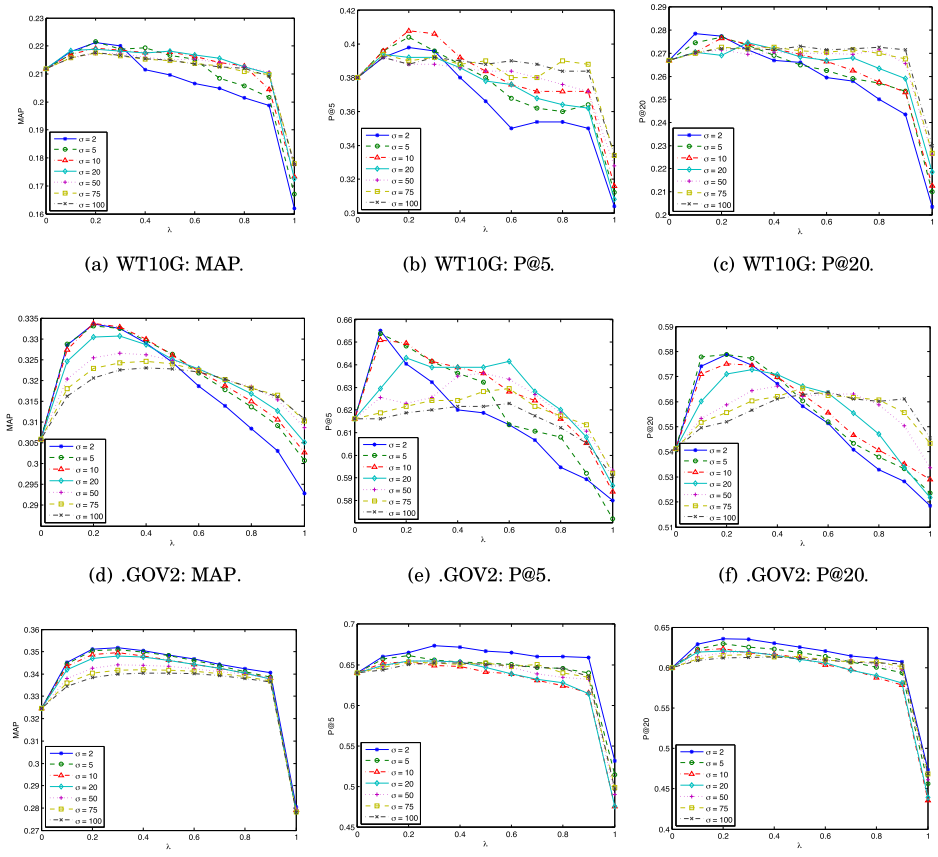
(a) TREC8: MAP. (b) TREC8: P@5. (c) TREC8: P@20.

(d) AP88-89: MAP. (e) AP88-89: P@5. (f) AP88-89: P@20.

(g) WT2G: MAP. (h) WT2G: P@5. (i) WT2G: P@20.

Fig. 11. Sensitivity to $CRTER_2$ parameter λ with different kernel parameters on TREC8, AP88-89, and WT2G.

between 0 and 0.2 is recommended, as it is shown to be reliable. In addition, σ is recommended to be between 3 and 25, considering both effectiveness and efficiency. The number of terms between 3 and 25 is consistent with the length of a sentence.



(a) WT10G: MAP. (b) WT10G: P@5. (c) WT10G: P@20.

(d) .GOV2: MAP. (e) .GOV2: P@5. (f) .GOV2: P@20.

(g) Blog06: MAP. (h) Blog06: P@5. (i) Blog06: P@20.

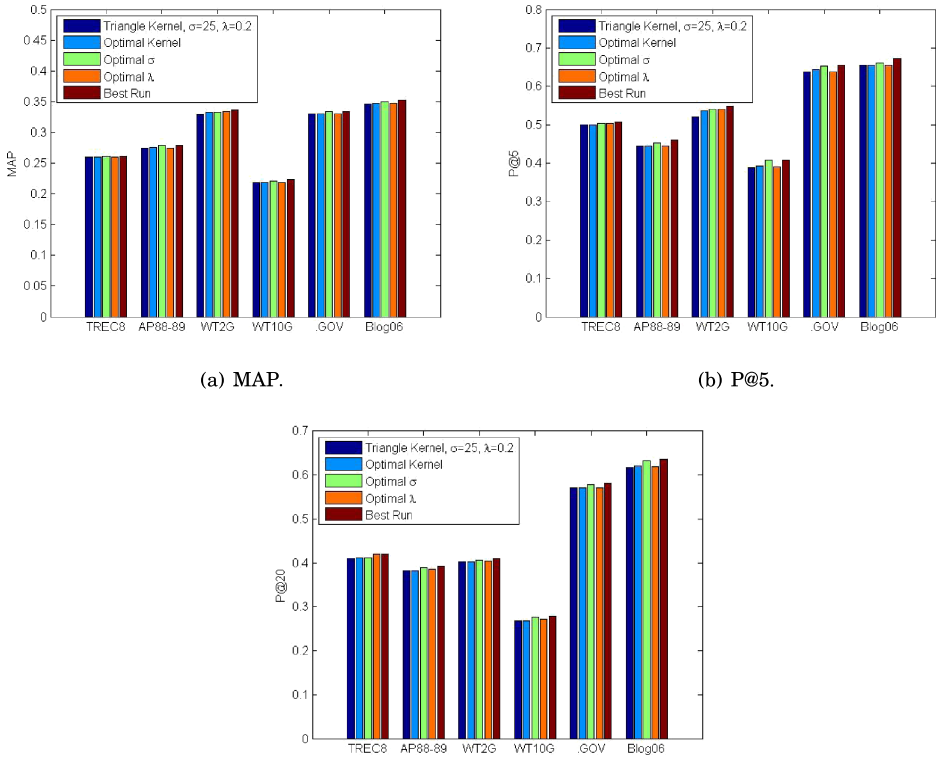
Fig. 12. Sensitivity to $CRTER_2$ parameter λ with different kernel parameters on WT10G, .GOV, and Blog06.

5.3. Robustness of $CRTER_2$

The proposed $CRTER_2$ model's robustness is important to its applications in practice. Ideally, we would like to have reliable retrieval performance using $CRTER_2$ on various datasets with its parameters within a stable safe range. This issue is particularly crucial for a given new dataset without training data.

We first fix BM25's parameter b = 0.35 and evaluate the robustness of $CRTER_2$ provided by an empirical setting, obtained from previous observations, namely, triangle kernel, $\sigma = 25$, and $\lambda = 0.2$. We compare this empirical setting with the following optimization strategies. First, optimize the kernel function, σ or λ individually while setting the other parameters to the empirical values. Second, optimize all three parameters: kernel functions, σ, and λ together. From Figure 13, we can see that the performance obtained by the empirical setting is comparable to the retrieval performance obtained by the optimized parameter settings on all datasets used.

We also test $CRTER_2$'s performance under the same empirical setting with BM25 over different BM25 parameters on all six collections. In BM25, b ranges from 0.15 to 0.95. In Figure 14, MAP of BM25 changes over different b values, and $CRTER_2$ can boost BM25 under all b's settings and over all the collections. More specifically, for .GOV2 dataset, we can see that BM25's performance is very sensitive to b, its MAP



(a) MAP.

(b) P@5.

(c) P@20.

Fig. 13. Robustness of $CRTER_2$. Compare $CRTER_2$'s retrieval performance provided by an empirical setting, namely, triangle kernel, $\sigma = 25$, and $\lambda = 0.2$, with the following optimization strategies. First, optimize the kernel function, σ, or λ individually while setting the other parameters to the empirical values. Second, optimize all three parameters: kernel functions, σ, and λ together.

gradually increases with the increment of b at the beginning and sharply decreases later. $CRTER_2$, on the other hand, boosts basic BM25 over different b values and tends to stabilize the retrieval performance.

In general, $CRTER_2$ performs robustly and has strong generalized performance. Without much knowledge of a new dataset, a group of parameters is recommended for $CRTER_2$: triangle kernel, $\sigma = 25$, and $\lambda = 0.2$.

5.4. Comparison with Major Probabilistic Proximity Models

In order to further evaluate our proposed model, we study how the proposed $CRTER_2$ model performs compared to the state-of-the-art probabilistic proximity approaches. We first compare the proposed $CRTER_2$ model with two probabilistic BM25-based proximity models, namely, the Proximity Probabilistic Model (*PPM*) [Song et al. 2011] and *BM25TP* [Buttcher et al. 2006], respectively.

In order to establish a fair comparison, we implement *PPM* and *BM25TP* with the same preprocessing techniques and the same optimal BM25 parameter settings. For *PPM*, we use the same kernel functions in Song et al. [2011] and tune parameters to be optimal by using cross-validation. For *BM25TP*, we follow the same experimental procedures reported in Buttcher et al. [2006]. We test $CRTER_2$, *PPM*, and *BM25TP* on the same collections with the same queries. The experimental results are shown

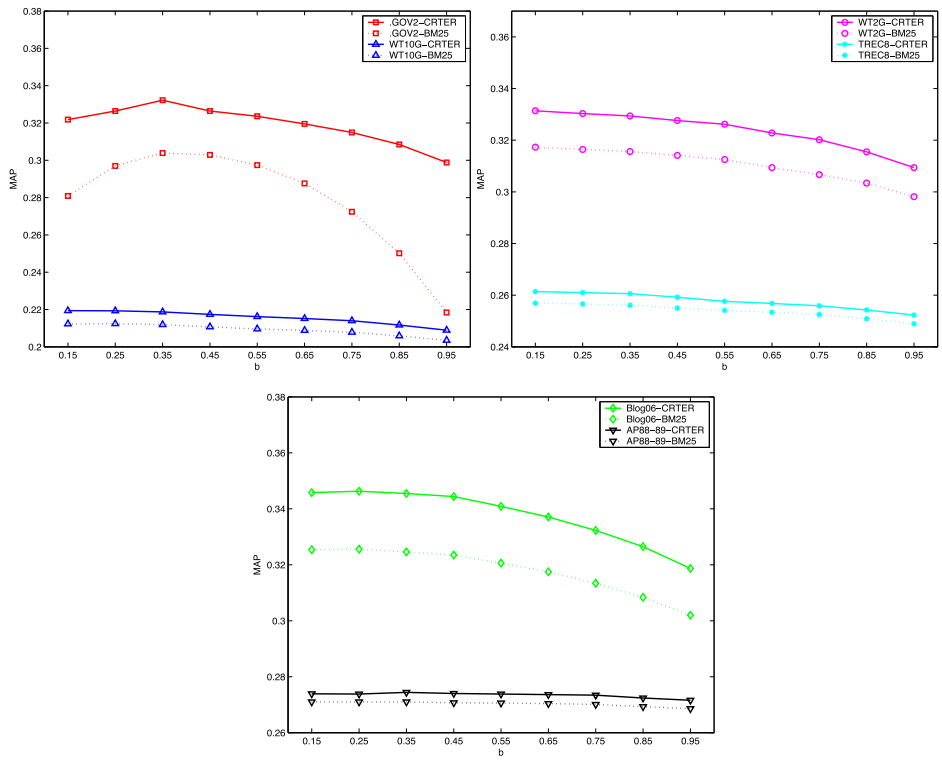


Fig. 14. Generalized performance of $CRTER_2$. Compare MAP between $CRTER_2$ and BM25 with the change of b ($CRTER_2$ uses fixed parameter: triangle kernel, $\sigma = 25$, $\lambda = 0.2$).

in Table III. In general, we can see that our proposed $CRTER_2$ shows improvement on all six data collections, and it is at least comparable to, if not better than, *PPM* and *BM25TP*. To illustrate the results in Table III graphically, we re-plot the improvement rates of these three models over BM25 in Figure 15, in which the x-axis shows the collections and the y-axis is the improvement rate. 0% means there is no improvement over BM25. We find that all three models improve BM25 in terms of evaluation metrics MAP and P@20. From Figure 15, we can see that $CRTER_2$ provides stable performance for all the evaluation metrics. In most of the cases, $CRTER_2$ outperforms *PPM* and *BM25TP*, especially on the top-ranked documents (e.g., top 5 and 20). This could be explained by $CRTER_2$'s capability of integrating more information into the retrieval process and considering the collection information (e.g., the number of documents containing a cross term). In PPM, a query-term occurrence is only propagated to other query-term nearest occurrences. In BM25TP, an occurrence of a query term contributes to the query terms at its previous posting and its following posting. Suppose we have an example query $\{q_1, q_2\}$, and the following document

$$\{...q_1^{(1)} q_2^{(1)} ...q_2^{(2)} ...q_1^{(2)} ...\},$$

where $q_1^{(1)}$ and $q_1^{(2)}$ are the first and second occurrences of q_1, and $q_2^{(1)}$ and $q_2^{(2)}$ are the first and second occurrences of q_2. Both *PPM* and *BM25TP* consider the associations: $< q_1^{(1)}, q_2^{(1)} >$ and $< q_2^{(2)}, q_1^{(2)} >$. On the other hand, $CRTER_2$ considers the associations: $< q_1^{(1)}, q_2^{(1)} >$, $< q_1^{(1)}, q_2^{(2)} >$, $< q_1^{(2)}, q_2^{(1)} >$, and $< q_1^{(2)}, q_2^{(2)} >$. This could give

Table III. Comparison among Three BM25-Based Proximity Models: *PPM*, *BM25TP*, and $CRTER_2$

	Eval Metric	TREC8	AP88-89	WT2G	WT10G	.GOV2	Blog06
BM25	MAP	0.2561	0.2710	0.3156	0.2119	0.3039	0.3246
	P@5	0.4920	0.4360	0.5280	0.3800	0.6134	0.6400
	P@20	0.4000	0.3860	0.3930	0.2670	0.5426	0.5997
PPM	MAP	0.2604* (+1.679%)	0.2763 (+1.956%)	0.3230 (+2.345%)	0.2213* (+4.436%)	0.3166* (+4.179%)	0.3403* (+4.837%)
	P@5	**0.5200** (+5.691%)	0.4360 (+0.000%)	0.5280 (+0.000%)	0.3880 (+2.105%)	0.6188 (+0.880%)	0.6400 (+0.000%)
	P@20	0.4070 (+1.750%)	0.3860 (+0.000%)	0.3990 (+1.527%)	0.2755 (+3.184%)	0.5584* (+2.912%)	0.6133* (+2.268%)
BM25TP	MAP	0.2582 (+0.820%)	0.2781 (+2.620%)	0.3276 (+3.802%)	0.2161 (+1.982%)	**0.3346*** (+10.102%)	0.3419 (+5.330%)
	P@5	0.4920 (+0.000%)	0.4440 (+1.835%)	0.5120 (−3.030%)	0.3700 (−2.632%)	0.6443 (+5.037%)	0.6507 (+1.672%)
	P@20	0.3960 (−1.000%)	0.3870 (+0.259%)	0.3960 (+0.763%)	0.2750 (+2.996%)	**0.5852*** (+7.851%)	0.6147 (+2.501%)
$CRTER_2$	MAP	**0.2606** (+1.757%)	**0.2789** (+2.915%)	**0.3359*** (+6.432%)	**0.2227*** (+5.097%)	0.3342*† (+9.970%)	**0.3517***†‡ (+8.349%)
	P@5	0.5080 (+3.252%)	**0.4600*** (+5.505%)	**0.5480**‡ (+3.788%)	**0.4080***†‡ (+7.368%)	**0.6550***† (+6.782%)	**0.6733***† (+5.203%)
	P@20	**0.4200***‡ (+5.000%)	**0.3910** (+1.295%)	**0.4100*** (+4.326%)	**0.2785*** (+4.307%)	0.5812*† (+7.114%)	**0.6357***†‡ (+6.003%)

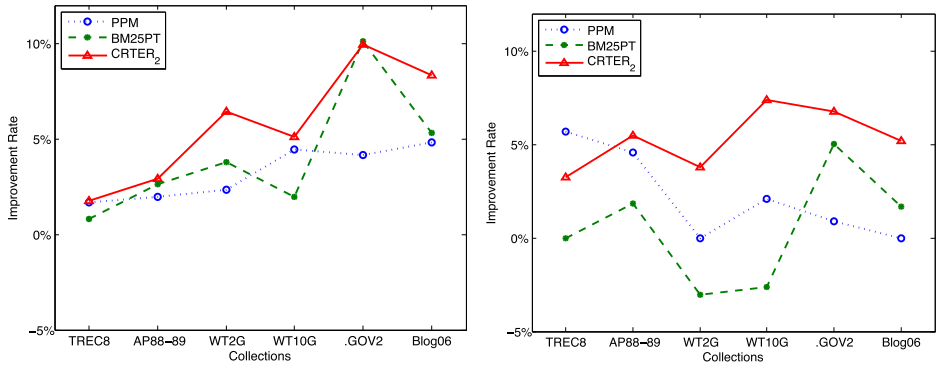
Note: BM25 parameter b is initialized to be 0.35. All the results are evaluated in terms of MAP, P@5, and P@20. "*" means the improvements over BM25 are statistically significant; "†" means the improvement over *PPM* is significant; and "‡" means the improvement over *BM25TP* is significant ($p < 0.05$ with Wilcoxon Matched-pairs Signed-rank test).

us an explanation as to why our $CRTER_2$ model performs better in most of the cases. Therefore, the retrieval performance could be boosted via integrating the associations between all query-term postings properly.

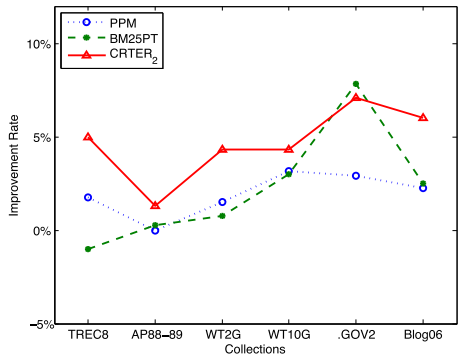
5.5. Comparison with Major Proximity Language Models

As we have discussed previously in Section 3.5, $CRTER_2$ can be extended on the basis of language models (LM). We use Dirichlet LM as the baseline model, which is a language model with Dirichlet smoothing technique [Zhai and Lafferty 2001]. We test the performance of $CRTER_2^{LM}$ by comparing Dirichlet $CRTER_2^{LM}$ with Markov Random Field (*MRF*) [Metzler and Croft 2005] and Positional Language Model (*PLM*) [Lv and Zhai 2009], which are described in Section 4.3.

First, we implement *MRF* under the same experimental environment and the same datasets as Dirichlet $CRTER_2^{LM}$. Both of these two models uses Dirichlet LM as the basic model, and the parameter μ is tuned to be optimal on each dataset, as shown in Table IV. The corresponding results are shown in Table V. The other parameters are optimized by hill climbing, as in Metzler and Croft [2005]. There is only one data collection, and the corresponding query set is exactly the same as the one in Metzler and Croft [2005], which is WT10G. Our implemented *MRF* has a very similar MAP on WT10G as the one reported in Metzler and Croft [2005]. We can see from Table V that $CRTER_2^{LM}$ and *MRF* have very comparable results. To illustrate the results in Table V graphically, we re-plot the improvement rates of *MRF* and $CRTER_2^{LM}$ over Dirichlet LM in Figure 16. Under the MAP evaluation metric, *MRF* slightly outperforms $CRTER_2^{LM}$ on most collections. For the top-5 and top-20 documents,



(a) MAP. (b) P@5.



(c) P@20.

Fig. 15. Improvement rates over BM25. Compare *PPM*, *BM25PT*, and $CRTER_2$ over six collections.

Table IV. Parameter μ for Dirichlet LM

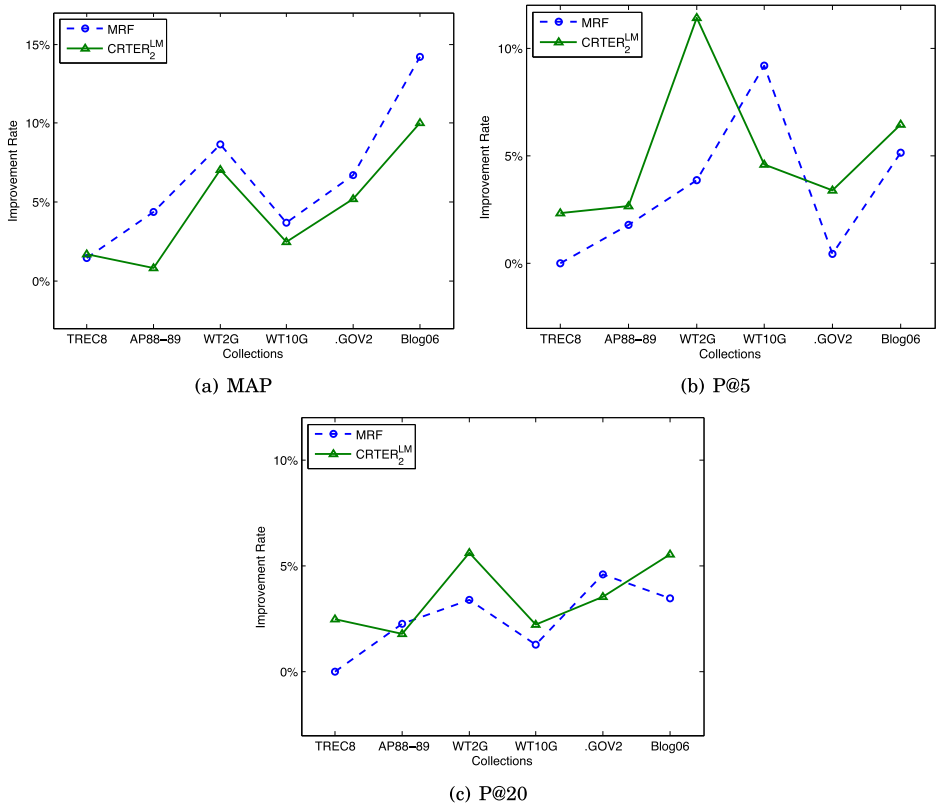
	TREC8	AP88-89	WT2G	WT10G	.GOV2	Blog06
μ	800	600	1,850	650	800	1,350

$CRTER_2^{LM}$ has better performance than *MRF* on most of the data collections. The difference between *MRF* and $CRTER_2^{LM}$ is not significant. Therefore, it is reasonable to state that $CRTER_2^{LM}$ favors top-ranked documents (e.g., top 5 and 20) compared to *MRF*. The experimental results show that $CRTER_2^{LM}$'s retrieval performance is comparable to *MRF*. One of the possible reasons could be that similar features are incorporated in $CRTER_2^{LM}$ and *MRF*. On the other hand, the modeling ideas of $CRTER_2^{LM}$ and *MRF* are quite different so that $CRTER_2^{LM}$ favors top-ranked documents. There are some cases in which $CRTER_2^{LM}$ can handle term dependency more specifically than *MRF*. For example, if each of q_1, q_2, q_3, q_4 occurs once in a document and $1 < dist_{q_1,q_2} < dist_{q_3,q_4} < N_{window}$, where N_{window} is the fixed window size, then *MRF* weights the dependency of $\{q_1, q_2, D\}$ and the dependency of $\{q_3, q_4, D\}$ the same. However, $CRTER_2^{LM}$ weights the dependency of $\{q_1, q_2, D\}$ higher than the dependency of $\{q_3, q_4, D\}$, because the distance between q_1 and q_2 is shorter than the distance between q_3 and q_4.

Table V. Comparison between Two Language Model-Based Proximity Models: $CRTER_2^{LM}$ with MRF

		TREC8	AP88-89	WT2G	WT10G	.GOV2	Blog06
Dirichlet LM	MAP	0.2552	0.2763	0.3060	0.2126	0.2983	0.3160
	P@5	0.5080	0.4531	0.5160	0.3480	0.5906	0.6187
	P@20	0.4020	0.4020	0.3810	0.2680	0.5268	0.5933
MRF	MAP	0.2589 (+1.450%)	**0.2884*** (+4.380%)	**0.3325*** (+8.660%)	**0.2204*** (+3.669%)	**0.3183*** (+6.705%)	**0.3609*** (+14.21%)
	P@5	0.5080 (+0.000%)	0.4612 (+1.788%)	0.5360 (+3.876%)	**0.3800*** (+9.195%)	0.5933 (+0.457%)	0.6507* (+5.172%)
	P@20	0.4020 (+0.000%)	**0.4112** (+2.289%)	0.3940 (+3.412%)	0.2715 (+1.306%)	**0.5510*** (+4.593%)	0.6140* (+3.489%)
Dirichlet $CRTER_2^{LM}$	MAP	**0.2595** (+1.684%)	0.2785 (+0.796%)	0.3275* (+7.026%)	0.2179 (+2.493%)	0.3138* (+5.196%)	0.3477* (+10.03%)
	P@5	**0.5200** (+2.362%)	**0.4653** (+2.693%)	**0.5750*** (+11.43%)	0.3640* (+4.598%)	**0.6107*** (+3.403%)	**0.6587*** (+6.465%)
	P@20	**0.4120** (+2.487%)	0.4092 (+1.791%)	**0.4025** (+5.643%)	**0.2740** (+2.238%)	0.5455 (+3.314%)	**0.6263*** (+5.562%)

Note: All the results are evaluated in terms of MAP, P@5, and P@20. "*" means the improvements over the Dirichlet LM are statistically significant ($p < 0.05$ with Wilcoxon Matched-pairs Signed-rank test).



(a) MAP (b) P@5

(c) P@20

Fig. 16. Improvement rates over Dirichlet LM. Compare *MRF* and $CRTER_2^{LM}$ over six collections.

Further, the effectiveness of $CRTER_2^{LM}$ is evaluated by a cross-comparison with *PLM* [Lv and Zhai 2009]. *PLM* estimates a language model for every single position in a document. Among various kernel functions tested, the *PLM* model is shown to

Table VI. Direct MAP Comparison with PLM

	TREC8	AP88-89	WT2G
Passage LM	0.2518	0.2154	0.3249
PLM	0.2550 (+1.3%)	0.2198 (+2.0%)	0.3285 (+1.1%)
Dirichlet LM	0.2552	0.2763	0.3060
$CRTER_2^{LM}$	0.2595 (+1.7%)	0.2785 (+0.8%)	0.3275 (+7.0%)
BM25	0.2561	0.2710	0.3156
$CRTER_2$	0.2606 (+1.8%)	0.2789 (+2.9%)	0.3359 (+6.4%)

Table VII. Overview of the TREC Collections with more than 3 Terms

Collection Name	# of Topics	# of Topics with ≥ 3 terms
TREC8	50	28
AP88-89	50	36
WT2G	50	24
WT10G	100	59
.GOV2	150	109
Blog06	150	35

provide the best retrieval performance with the Gaussian kernel. We directly use the results reported in Lv and Zhai [2009]. For *PLM*, the baseline is the fixed-length arbitrary passage retrieval method under the language modeling framework (e.g., Passage LM in Table VI). Since we are using different LM baselines, we compare *PLM* with LM-based $CRTER_2^{LM}$ and BM25-based $CRTER_2$ by the relative improvements in terms of percentage over the basic retrieval models for fair comparison. The comparison is conducted on the datasets used in Lv and Zhai [2009], including AP88-89, WT2G, and TREC8. As illustrated in Table VI, *PLM* improves the LM baseline by 1.3%, 2.0%, and 1.1% on TREC8, AP88-89, and WT2G, respectively. The $CRTER_2^{LM}$ improves Dirichlet LM by 1.7%, 0.8%, and 7.0% on TREC8, AP88-89, and WT2G, and the corresponding improvement over BM25 by $CRTER_2$ is 1.8%, 2.9%, and 6.4%, respectively. Overall, *PLM*, $CRTER_2$, and $CRTER_2^{LM}$ all boost their baselines. In particular, compared with *PLM*, $CRTER_2^{LM}$ have more improvement on TREC8 and WT2G, and less improvement on AP88-89. $CRTER_2$'s improvement rates over BM25 are all higher than the improvement rates of *PLM* over passage LM. We can conclude that $CRTER_2^{LM}$ and $CRTER_2$ have comparable (if not better) performance compared to *PLM*.

5.6. Experimental Results of $CRTER_n$ and Runtime Analysis

In previous sections, we have illustrated the effectiveness and robustness of $CRTER_2$. Now we further conduct experiments to evaluate the effectiveness of $CRTER_n$ on the same collections over different settings. Further, we compare and analyze the runtime of BM25, $CRTER_2$ and $CRTER_n$. The difference between $CRTER_n$ and $CRTER_2$ is that $CRTER_n$ imports nCT, which represents the association among more than two query terms. To compare $CRTER_n$ with $CRTER_2$, only the queries with at least three words would make sense. Therefore, we remove the queries that are shorter than three query terms. Table VII shows the query information in this section. For the documents, we use the original documents and process them in the same way as in Section 4.1.

As we can see from Table VII, around half of the utilized standard TREC queries contain three or more terms on most of the datasets. Further, as we have discussed in Section 3.4, the computational complexity of the n-gram model grows exponentially with the increase of n. Therefore, we focus on discussing the case when $n = 3$ in this section. We have come to the conclusion from Section 5.3 that a group of recommended

Table VIII. Comparison between $CRTER_2$ and $CRTER_3$ with Different Distance Metrics

	Eval Metric	TREC8	AP88-89	WT2G	WT10G	.GOV2	Blog06
$CRTER_2$	MAP	**0.2270**	**0.2286**	0.3338	0.2196	0.3322	0.2907
	P@5	**0.4286**	0.3944	**0.5500**	0.4475	0.6495	0.5543
	P@20	0.3554	**0.3222**	0.3687	0.3017	0.5830	0.5143
$CRTER_3$ with Pairwise-Min Distance	MAP	**0.2270**	**0.2286**	0.3338	0.2202	0.3322	0.2907
	P@5	**0.4286**	0.4000	**0.5500**	0.4508	0.6514	0.5657
	P@20	0.3589	**0.3222**	0.3687	0.3017	0.5853	0.5186
$CRTER_3$ with Pairwise-Max Distance	MAP	**0.2270**	**0.2286**	0.3338	0.2204	0.3322	0.2907
	P@5	**0.4286**	0.3944	**0.5500**	0.4508	0.6495	**0.5714***
	P@20	0.3607	**0.3222**	0.3687	0.3025	0.5858	0.5186
$CRTER_3$ with L_1 Distance	MAP	**0.2270**	**0.2286**	0.3381	0.2198	0.3329	0.2964
	P@5	**0.4286**	0.3944	**0.5500**	**0.4576**	0.6514	0.5657
	P@20	0.3643	**0.3222**	0.3687	0.3085	0.5835	0.5186
$CRTER_3$ with L_2 Distance	MAP	**0.2270**	**0.2286**	0.3388	0.2199	0.3332	0.2960
	P@5	**0.4286**	0.3944	**0.5500**	**0.4576**	0.6532	0.5657*
	P@20	0.3661	**0.3222**	0.3687	**0.3102**	0.5849	0.5143
$CRTER_3$ with L_∞ Distance	MAP	**0.2270**	**0.2286**	0.3395	0.2201	0.3330	0.2963
	P@5	**0.4286**	0.3944	**0.5500**	0.4508	**0.6569**	0.5657
	P@20	**0.3679**	**0.3222**	0.3687	0.3042	0.5853	0.5171
$CRTER_3$ with Altitude Distance	MAP	**0.2270**	**0.2286**	0.3440	**0.2211**	**0.3338**	**0.2994**
	P@5	**0.4286**	0.4000	**0.5500**	0.4542*	0.6550	0.5657
	P@20	0.3661	**0.3222**	0.3688	0.3042	0.5858	**0.5229**
$CRTER_3$ with Hypotenuse Distance	MAP	**0.2270**	**0.2286**	**0.3435**	0.2198	0.3332	0.2979
	P@5	**0.4286**	**0.4056**	**0.5500**	0.4508	0.6550	0.5657
	P@20	0.3607	**0.3222**	**0.3729**	0.3051	**0.5885**	**0.5229**

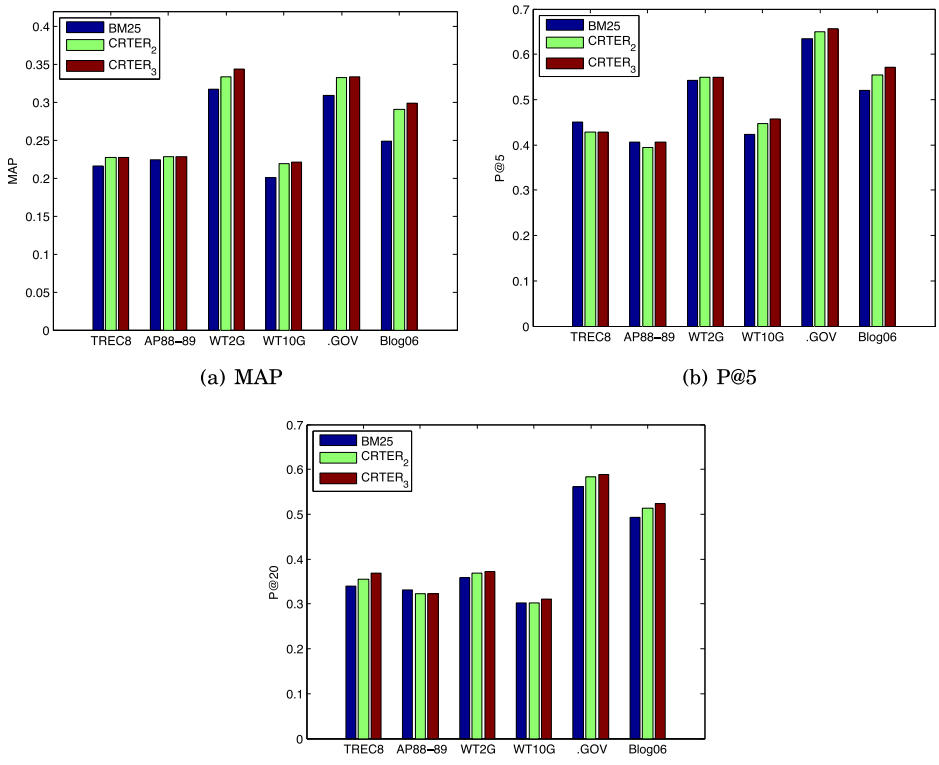
Note: Both use fixed parameters: triangle kernel, $\sigma = 25$, $\lambda = 0.2$. "*" means the improvements over $CRTER_2$ are statistically significant ($p < 0.05$ with Wilcoxon Matched-pairs Signed-rank test).

parameters of $CRTER_2$ are as follows: using triangle kernel as the kernel function, kernel parameter $\sigma = 25$ and balancing parameter $\lambda = 0.2$. The results of $CRTER_3$ with different distance metrics are compared with $CRTER_2$ using these recommended parameters.

Seven distance metrics are used for $CRTER_3$, which are the pairwise-min distance, pairwise-max distance, L_1-norm-based distance, L_2-norm-based distance, L_∞-norm-based distance, altitude-based distance, and hypotenuse-based distance. The experimental results are shown in Table VIII. Although only two cases are observed where $CRTER_3$ outperforms $CRTER_2$ significantly, there are still some small improvements of $CRTER_3$ over $CRTER_2$ in most of the cases. Comparing different distance metrics, more improvement is observed on $CRTER_3$ using hypotenuse-based distance.

Figure 17 shows the comparison between unigram model BM25, bigram model $CRTER_2$, and trigram model $CRTER_3$ on all the datasets using queries with equals to or more than three terms. All the experimental results are obtained by using a fixed set of parameters. In most of the cases, $CRTER_2$ improves BM25, and $CRTER_3$ improves $CRTER_2$. In general, considering the association among three terms in $CRTER_3$ is a useful addition to using two terms in $CRTER_2$ if the time and space conditions allow, although $CRTER_2$ is sufficient to provide a reliable retrieval performance on the collections used.

In Table IX, we present the runtime of BM25, $CRTER_2$, and $CRTER_3$ on each collection with the corresponding topic set. As a case study, the reported experiments are conducted on a computer with Intel 2.53GHz CPU and 8G memory. For a faster implementation, we rerank the top 2,000 documents instead of the whole collection. On each dataset, we report the average runtime of ten runs. To have an intuitive view over the runtime, we further plot the runtime of BM25, $CRTER_2$, and $CRTER_3$ divided by



(a) MAP

(b) P@5

(c) P@20

Fig. 17. Overview performance of unigram model BM25, bigram model $CRTER_2$, and trigram model $CRTER_3$.

Table IX. Runtime (in seconds)

	TREC8	AP88-89	WT2G	WT10G	.GOV2	Blog06
BM25	1.527	0.999	0.893	8.381	652.449	22.340
$CRTER_2$	1.935	1.942	1.270	9.411	670.452	23.094
$CRTER_3$	2.633	2.349	1.886	12.105	730.141	28.714

BM25's runtime in Figure 18. In general, $CRTER_2$ consumes more time than BM25, and $CRTER_3$ consumes more time than $CRTER_2$. There is a balance between time complexity and effectiveness. The improvements of $CRTER_2$ over $BM25$ are higher than the improvements of $CRTER_3$ over $CRTER_2$. $CRTER_3$ is still an option to further boost the performance, if a higher accuracy is required. However, we also have to take into account the additional complexity of $CRTER_3$. Since we are using an offline vocabulary and the top-2,000 reranking technique, the relative increments on larger datasets are smaller.

We further use Blog06 as a case study to show the trade-off between the effectiveness and the efficiency for reranking the top documents in Table X. We also show the time spent on both online and offline retrieval discussed in Section 3.4. In online retrieval, the cross terms have not been processed before retrieval. In offline retrieval, the information of cross terms generated by given queries is stored in the index. In real applications where the queries are not known, building the offline index for all possible cross terms on the whole collection would have a very high space complexity. Further feature selection techniques are required in such implementations, and we do

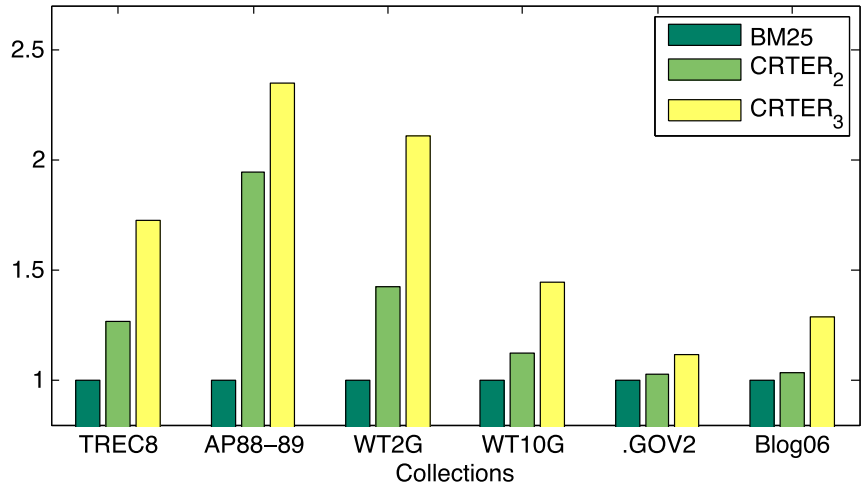


Fig. 18. Runtime of BM25, $CRTER_2$, and $CRTER_3$ (divided by BM25's runtime).

Table X. A Case Study: Trade-Off between Effectiveness and Efficiency for Reranking Top Documents on Blog06

	Online Retrieval (in Seconds)	Offline Retrieval (in Seconds)	MAP	P@5	P@20
Collection	7,520.00	38.74	0.3540	0.6627	0.6277
Top 2000	940.39	28.37	0.3477	0.6587	0.6257

not discuss them in detail in this article, which could be our future work. First of all, we can see that online retrieval takes much more time than offline retrieval. Second, reranking the top-2,000 documents rather than the whole collection would further reduce the retrieval time. Finally, we are able to reduce the retrieval time from 7,520.00 seconds to 28.37 seconds. On the other hand, reranking the top-2,000 documents sacrifices minor accuracy. Therefore, we need to balance the trade-off between effectiveness and efficiency in real applications.

6. FURTHER ANALYSIS AND DISCUSSION

In this section, we first investigate the possible reasons why the proposed *CRTER* models perform well with a case study. Then in Section 6.2 and Section 6.3, we present examples for bigram and n-gram cross terms for analyzing the proposed models. In Section 6.4, we test the stability of the proposed models by retrieving different topic sets on the same collection. The practical impact and further analysis about the proposed *CRTER* models are provided in Section 6.5. In conclusion, we summarize the advantages of using cross terms in Section 6.6.

6.1. A Case Study on the Manually-Judged Documents

We conduct a case study to investigate the possible reasons why the proposed bigram models are effective. In particular, out of the 550 test topics used in our experiments, we show a case study on topic 931 of the Blog06 collection. The title of topic 931 is "Fort McMurray", which refers to the Canadian oil boom town named Fort McMurray in Alberta. When we read each word individually, "Fort" usually represents a permanent army post, and "McMurray" is a family name. Therefore, the term association between "Fort" and "McMurray" would be very useful in finding the documents that are relevant to the topic "Fort McMurray". In detail, using the judged documents, namely, the golden standard provided by TREC, we explore how these models could

Table XI. A Case Study: Distribution of Terms on Relevant and Nonrelevant Documents (Topic 931 "Fort McMurray" on the Blog06 Collection)

Model	Description	Relevant	Nonrelevant	$Ratio(\frac{Relevant}{non-relevant})$
PPM	Position dependent Frequency	3.01	6.24	0.48
$CRTER_2$	Bigram Cross Term Frequency	2.96	0.42	7.11
BM25TP	Weight of term proximity	3.56	1.15	3.08
$CRTER_2$	Weight of bigram Cross Terms	140.23	24.23	5.79
MRF	Weight of phrases	23.21	12.64	1.84
$CRTER_2^{LM}$	Weight of bigram Cross Terms	11.89	3.17	3.75

Note: In the first row, *PPM* uses position dependent frequency with proximity in BM25 instead of the raw-term frequency. For $CRTER_2$, bigram cross term frequency is extracted to compare with *PPM*. In the second and third rows, we extract the additional proximity weight of each model for comparison.

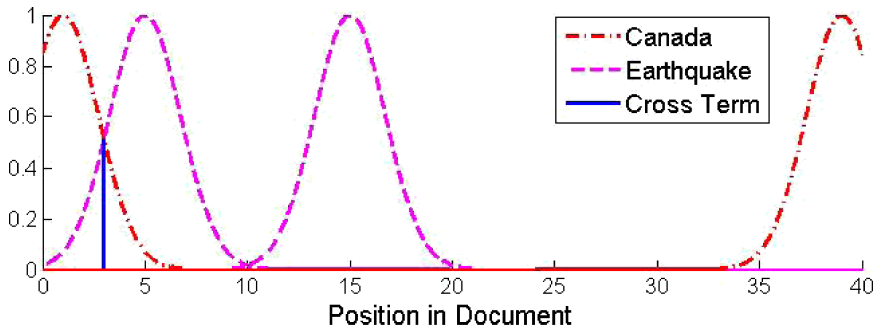
Table XII. Comparisons on the Judged Documents over the Blog06 Collection

# of Topics on Blog06 Collection	150
# of Topics with More Than One Term	101
# of Topics that $CRTER_2$ distinguishes better than *PPM*	66
# of Topics that $CRTER_2$ distinguishes better than *BM25TP*	73
# of Topics that $CRTER_2^{LM}$ distinguishes better than *MRF*	83

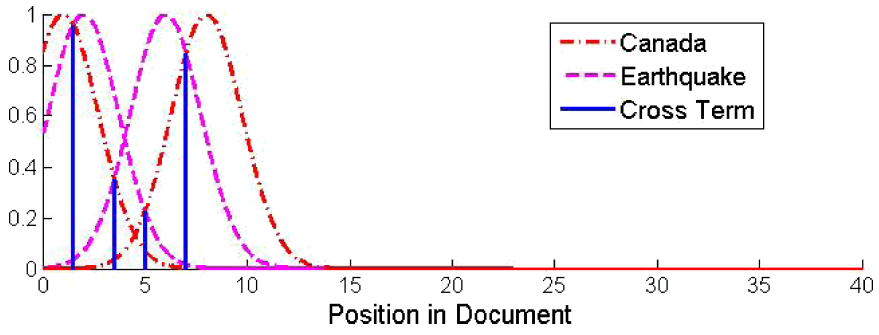
differentiate relevant documents from nonrelevant ones. In Table XI, we list our proposed bigram models and the proximity models that are presented in Sections 5.4 and 5.5, namely, *PPM*, $CRTER_2$, *BM25TP*, *MRF*, and $CRTER_2^{LM}$. We explore how these models could distinguish relevant documents from nonrelevant documents. For each model, we extract the part that is directly related to term association. For *PPM*, new-term frequency with proximity is utilized in BM25 instead of the raw-term frequency. A new-term frequency is related to both the number of occurrences of a term and the term counts propagated from other terms. We also extract the bigram cross term frequency in $CRTER_2$ for comparison with *PPM*. For both *BM25TP* and *MRF*, additional proximity weights are linearly combined into the original weighting functions. We also extract the cross term weights in $CRTER_2$ and $CRTER_2^{LM}$ for comparison. Table XI shows the average values on all the relevant documents and average values on nonrelevant documents. We also calculate the ratios of the results on relevant documents over the results on nonrelevant documents. A model with a higher ratio would be more effective in identifying relevant documents from nonrelevant ones for this topic.

From Table XI, we can see that $CRTER_2$'s ratio is much higher than *PPM*'s ratio in terms of frequencies. $CRTER_2$'s ratio is also higher than *BM25TP*'s ratio in terms of the additional weight. In addition, $CRTER_2^{LM}$'s ratio is higher than *MRF*'s ratio in terms of their additional weights. For this topic, $CRTER_2$ and $CRTER_2^{LM}$ can better distinguish relevant documents from nonrelevant documents compared with *PPM*, *BM25TP*, and *MRF* accordingly.

Similar patterns can be found on most of the topics. In Table XII, we show comparisons of the proximity models on all the topics of the Blog06 collection. The Blog06 collection includes 150 topics, where 49 topics contain one term. Here we analyze the rest of the 101 topics that contain two or more terms. We found that $CRTER_2$ better distinguishes relevant documents from nonrelevant documents than *PPM* on 66 out of 101 topics; $CRTER_2$ distinguishes better than *BM25TP* on 73 topics; and $CRTER_2^{LM}$ distinguishes better than *MRF* on 83 topics. From this case study, we can see that



(a) A nonrelevant document.



(b) A relevant document.

Fig. 19. A case study: example of bigram cross term.

our proposed *CRTER* models have better performance in distinguishing relevant documents from nonrelevant documents on most of the topics. This could give us a possible explanation why *CRTER* models can perform better than other models in retrieving.

6.2. An Analysis on the Bigram Cross Term Example

Figure 19 gives an example of a bigram cross term by query terms "Canada" and "Earthquake", which corresponds to the example documents in Figure 1. "Canada" and "Earthquake" occur twice in both the relevant document and the nonrelevant document. A traditional retrieval model will assign weights to both documents similarly. Here we visualize the bigram cross term generated by "Canada" and "Earthquake" in these two documents. In Figure 19, we plot the query-term impact shape functions and the corresponding bigram cross term values over the whole document. We use Gaussian kernel with a small σ for a better view of the cross terms. Figure 19(a) shows the nonrelevant document and Figure 19(b) relates the relevant document.

We can see that the bigram cross term occurs in the nonrelevant document (Figure 19(a)) only once, while it occurs in the relevant document (Figure 19(b)) four times. In particular, supposing we only have these two documents in the collection, we calculate the bigram cross term values, frequency, and weight in each document. These definitions are introduced in Section 2.1, 2.2, and 3.2. In Table XIII, we can see that the bigram cross term has a higher within-document frequency in the relevant document than that in the nonrelevant document. This is because the bigram cross term has higher value when two query terms occur closer and the bigram cross term value is

Table XIII. The Values Corresponding to the Nonrelevant and Relevant Documents in the Case Study

	Nonrelevant	Relevant
Cross Term occurrence	1	4
Cross Term values	0.5121	0.9567, 0.3519, 0.2225, 0.8442
Cross Term within-document frequency	0.5121	2.3753
Cross Term document frequency	1.1059	
Cross Term within-query frequency	0.9567	
Cross Term weight	0.2634	0.5723

(a) A nonrelevant document. (b) A relevant document.

Fig. 20. An example of term association for a trigram: "the U.S. election of 2008".

lower when two query terms are farther away. Eventually, the cross term weight in the relevant document is higher than that in the nonrelevant document. Since query-term weights of these two documents are similar, this relevant document's overall weight is higher than the nonrelevant document's overall weight. This example demonstrates that bigram cross term is a good measure for distinguishing relevant documents from nonrelevant documents.

6.3. The Usefulness of N-Gram Cross Terms

Further, considering only the bigram correlation may not be enough, so we also investigate the associations among n query terms. Figure 20 gives an example showing the usefulness of trigram dependency. In this figure, we show you an example of a three-term query, namely, "the U.S. election of 2008". After stop words are removed, it becomes "U.S. election 2008". Figure 20(a) shows a nonrelevant document and Figure 20(b) shows a relevant document. We can see that the query terms have the same frequencies in both documents, while three query terms occur more closely in the relevant document. In Figure 20(a), "U.S." is close to "election" at the beginning, and "elected" is close to "2008" at the end. However, three query terms do not occur together, as in Figure 20(b). Trigram cross term represents the association among all three query terms.

We calculate the trigram cross term values and frequencies in the nonrelevant and relevant documents, as shown in Table XIV. The corresponding definitions for the trigram cross term and its variants are presented in Sections 2.3 and 2.4. Here we use Gaussian kernel with parameter $\sigma = 15$. The trigram cross term occurs twice in each document and has higher values in the relevant document than that in the nonrelevant document. Therefore, the trigram cross term within-document frequency in the relevant document is higher than that in the nonrelevant document. Since the

Table XIV. The Values Corresponding to the Nonrelevant and Relevant Documents in the Trigram Example

	Nonrelevant	Relevant
Tri-gram Cross Term occurrences	2	2
Generating query term positions	(7, 8, 61), (7, 60, 61)	(2, 4, 6), (2, 12, 6)
Distances for query terms	53.009, 53.009	2.828, 7.211
Tri-gram Cross Term values	0.2099, 0.2099	0.9956, 0.9715
Tri-gram Cross Term within-document frequency	0.4198	1.9671

Table XV. Experimental Results on Robust Track

	TREC8			Robust		
	MAP	P@5	P@20	MAP	P@5	P@20
BM25	0.2561	0.4920	0.4000	0.2420	0.3458	0.4562
$CRTER_2$	0.2606	0.5080	0.4200	0.2452	0.3494	0.4586
	(+1.757%)	(+3.252%)	(+5.000%)	(+1.322%)	(+1.041%)	(+0.526%)
Dirichlet LM	0.2552	0.5080	0.4020	0.2562	0.4739	0.3653
MRF	0.2589	0.5080	0.4020	0.2651	0.4867	0.3691
	(+1.450%)	(0.000%)	(0.000%)	(+3.474%)	(+2.701%)	(+1.040%)
$CRTER_2^{LM}$	0.2595	0.5200	0.4120	0.2666	0.4827	0.3741
	(+1.684%)	(+2.362%)	(+2.487%)	(+4.059%)	(+1.857%)	(+2.409%)

other variants of these two documents are very similar, the document in which three query terms occur closely is boosted by employing trigram cross term in retrieval. This example illustrates that $CRTER_3$ could possibly further boost $CRTER_2$.

6.4. The Stability of Using Different Topic Sets

Finally, we test our models on the same collection with different topic sets. Some studies in the past few years conducted experiments on the TREC Robust 2004 [Bendersky et al. 2010; Wang and Zhu 2009]. Robust and TREC8 have the same data collection containing newswire articles. TREC8 includes topics 401–450, and Robust includes more topics 301–450 and 601–700. The experimental results are shown in Table XV. We can see that the tendencies of the performance on TREC8 and Robust are very similar. $CRTER_2$ has better retrieval precision compared to BM25. $CRTER_2^{LM}$ and *MRF* outperform Dirichlet LM, and $CRTER_2^{LM}$'s retrieval performance is better than *MRF* in most of the cases. These results further confirm the effectiveness and stability of $CRTER_2$ and $CRTER_2^{LM}$.

6.5. Practical Impact of the Proposed Approaches

In this section, we discuss the practical impact of the proposed approaches and briefly introduce how other various domains can benefit from the proposed models. The exact models used might need to adapt to the specific problems in these domains.

The proposed concept of cross term can be useful in other text analysis realms. It is natural to study term associations in document clustering, document summarization, sentiment analysis, etc. With properly-defined cross terms, we can visualize and quantify these term associations for more precise results. For example, document clustering aims to combine a set of documents into clusters so that intracluster documents are more similar to each other than intercluster documents [Agrawal and Phatak 2013]. Thus we can measure the similarity between documents by matching the real terms and the cross terms. As another example, document summarization can represent the document with a short piece of text covering the main topics [He et al. 2012]. We can extract the most important cross terms in the original document and try to keep these

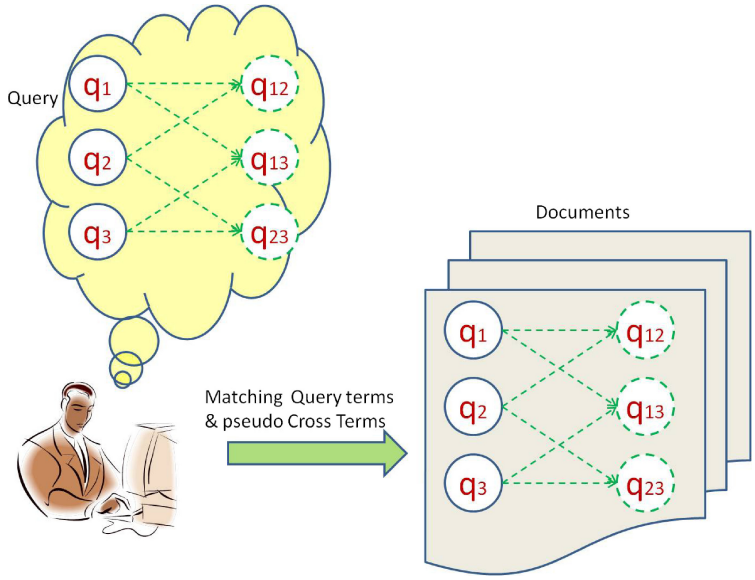


Fig. 21. Hidden information of using cross terms.

cross terms in the summarization. Cross terms can also be employed in sentiment analysis, where a review can be considered as a document generated by a number of hidden sentiment factors [Yu et al. 2012]. We can use the idea of cross terms to represent the hidden sentiment information in reviews.

The generalized concept of cross term can be applied in more domains, such as the medical data mining and image processing. Medical examinations have associations among each other [Zhao et al. 2012a, 2013]. For instance, if a patient is suspected to have diabetes, usually the doctor will assign both a hemoglobin test and a glucose fasting test for this patient. There exists an association between hemoglobin and glucose fasting with respect to some hidden information, diabetes in this case. We can use the idea of cross terms to simulate this hidden information among medical examinations/treatments. In image retrieval, it is usually difficult to match the actions in the images, and researchers often integrate other text features as alternatives [Ye et al. 2010]. The object boundaries and shapes can be detected by the approaches proposed in Ferrari et al. [2007]. The image objects at the form level and the content level can be represented by a logic-based model [Meghini et al. 1997]. We can use the idea of cross terms and kernel functions to investigate the correlations among the image objects and to identify the object actions.

6.6. Summary of Using Cross Terms

In this article, we focus on integrating proximity into the information retrieval process, which is similar to the other proximity models to some extent. However, the details vary significantly. We work on this problem from a new and novel perspective that is unique and different from previous methods. In detail, we abstract the concept of term association into a pseudo term, cross term, which does not exist but is very important for modeling term associations. In a user's query, there exists some hidden information underlying the query terms. The proposed concept of cross terms provides a promising avenue for modeling and mining associations among terms in information retrieval. During retrieval, this model matches the query terms in the topic with the terms in the documents and matches the cross terms in the topic with the cross terms in the

documents, as shown in Figure 21. As we know, the most fundamental elements in IR are terms. Therefore, proposing the cross term concept as a pseudo term can make us easily simulate and manipulate term associations in retrieval models. Term associations are embedded in *CRTER* models in a natural and global way. Compared to some other probabilistic models with proximity, we integrate proximity not only on the term level (i.e., changing some variables related to the individual terms), but also on the collection level (i.e., controlled by the number of documents containing a cross term, see Definitions 2.5 and 2.8). We implement the cross terms using proximity, which defines and estimates cross terms properly.

What is more, the cross term concept could be applied to other domains. These include document clustering, document summarization, sentiment analysis, and the medical data mining. The exact models used might have to be modified to accommodate the specific characteristics of the target domains.

7. RELATED WORK

Since the 1990s, some early researchers started to investigate term-association approaches in IR, and they found such approaches to be effective. The query-term associations have been modeled by different approaches according to the distance of the query terms in documents. For example, Allan et al. [1995] indexed phrases instead of terms with InQuery [Callan et al. 1992] and obtained improvements in TREC campaigns. This approach can only handle query terms that are adjacent to each other in the documents. Intuitively, however, the query terms that are not adjacent to each other but occur closely to each other may also carry some associations. A relaxed approach was proposed by Clarke et al. [1996], which introduced a "NEAR" operator to quantify the proximity of query terms. A similar approach was proposed by Hawking and Thistlewaite [1995], which evaluated "Span" proximity approaches on TREC datasets, which is the text segments containing all query-term instances. In addition to the "objective" statistical information, an IR system is also concerned to "subjectively" have the degrees of belief in the relevance of the document to the query [Sebastiani 1994]. White et al. [2002] showed using top-ranking sentences for relevance feedback can be effective and efficient.

In general, bigram IR models are popular in modeling terms associations and can make a better balance between effectiveness and complexity. A number of studies have been done under the language modeling framework. For example, Song and Croft [1999] proposed a general language model that combined bigram language models with several smoothing techniques, including a good-Turing estimate and corpus-based smoothing of unigram probabilities. The relative contributions of the different models to query-generation probability were determined empirically. Srikanth and Srihari [2002] proposed a biterm language model which approximated the biterm probabilities using the frequency of occurrence of terms. Biterm language models are similar to bigram language models except that the constraint of order in terms is relaxed. Alvarez et al. [2004] further proposed a language modeling approach that incorporated word pairs, without a constraint on adjacency or word order, where word pairs were determined by statistical relationships, or lexical affinities, between words. In addition, Pickens [2000] introduced an approach that used nonadjacent biterms, but the particular domain, musical documents, required an emphasis on the order of "words". In our previous preliminary work [Zhao et al. 2011], we also investigated the proximity between a pair of query terms. This article is a substantial extension which theoretically extends the previous work to model multi-term associations and propose an n-gram cross term retrieval model. In particular, our approach differs from previous studies where we propose the concept of a pseudo term, cross term, generated

by multiple query terms to investigate how multiple query-term occurrence changes together. Cross terms are naturally integrated into basic retrieval models as new terms and therefore incorporate proximity into the retrieval process. In addition, we mainly focus our research under the probabilistic retrieval framework.

N-gram term associations have been investigated in IR for years [Ahmed and Nurnberger 2009; Bendersky and Croft 2012; Hou et al. 2013; Mayfield and McNamee 2003; McNamee and Mayfield 2004]. Gao et al. [2004] introduced the linkage of a query as a hidden variable, which expressed the term associations within the query as an acyclic, planar, undirected graph. Beigbeder and Mercier [2005] proposed a mathematical model based on a fuzzy proximity degree of term occurrences, particularly for boolean queries. Bendersky and Croft [2008] extracted key concepts from long and verbose queries. Guo et al. [2008] proposed a conditional random field model to predict a sequence of refined query terms for a given sequence of ill-formed query terms. Bendersky et al. [2009] showed that query segmentation could reduce query latency without compromising effectiveness. Park et al. [2011] presented a term-dependency model which was inspired by a quasi-synchronous stochastic process for machine translation. A retrieval model based on Markov random field [Metzler and Croft 2005] was presented for developing a general retrieval framework for modeling bigram and n-gram term associations. They found that the sequential (bigram) dependence variant closely approximated the full (n-gram) dependence variant, which was similar to one of our conclusions. In Bendersky et al. [2010], the Markov random field model was further extended by assigning weights to concepts.

Researchers have developed various proximity approaches on different types of IR models. Some studies heuristically integrated word proximity or term location information into probabilistic weighting models (e.g., [Broschart and Schenkel 2008; Buttcher et al. 2006; He et al. 2011; Hu et al. 2012; Rasolofo and Savoy 2003; Tao and Zhai 2007; Wong and Yao 1993; Ye et al. 2012; Zhao et al. 2012b]). Some work also found that term proximity was a useful measure for selecting query expansion terms in probabilistic IR [Miao et al. 2012]. Song et al. [2011] used a position-dependent term count to represent both the number of occurrences of a term and the term counts propagated from other terms. However, it is still largely unexplored to theoretically model term associations under the probabilistic retrieval framework [Pickens 2000; Robertson and Walker 1994; Wang and Si 2008]. For statistical LM, some recent research work embedded the proximity information in LM [Wei and Croft 2007]. For example, Zhao and Yun [2009] viewed query-term proximate centrality as Dirichlet hyper-parameters for assigning weights to the parameters of the multinomial document language models. For discriminative models, dependency weights were assigned to term pairs [Shi and Nie 2010]. Our proposed *CRTER* models can be applied to any traditional IR models, since the term associations are regarded as pseudo terms, and terms are the most fundamental elements in IR.

Density functions based on proximity have been adopted to characterize term influence propagation [de Kretser and Moffat 1999; Kise et al. 2004; Lv and Zhai 2009; Mercier and Beigbeder 2005; Petkova and Croft 2007]. de Kretser and Moffat [1999] is early work that proposed propagating the $tf \cdot idf$ score of each query term to other positions, where triangle, cosine, circle, and arc contribution functions were discussed. The highest accumulated $tf \cdot idf$ score on all the positions was adopted as the document's score. Kise et al. [2004] used a hanning (cosine) window function to characterize the density of terms. Lv and Zhai [2009] proposed a positional language model that incorporates the term proximity in a model-based approach using four term-propagation functions: Gaussian, triangle, cosine, and circle. We employ these term-influence propagation functions for the concept of cross term in measuring term association and introduce more potential functions: Quartic, Epanechnikov, and triweight.

8. CONCLUSIONS AND FUTURE WORK

This article introduces the concept of cross term to study the context of the query-term association. A cross term is a pseudo term which measures the association of multiple query terms. The basic assumption is that query-term influence on the neighboring text is approximated by a kernel function, the degree of which gradually weakens with increasing distance from the place of the occurrence. In this article, we define bigram and n-gram cross terms, propose estimation methods for them, integrate them into basic probabilistic retrieval models, and evaluate the proposed models on several standard TREC data collections.

We first define a bigram cross term, which is the overlapped influence of two query terms. It occurs when two query terms appear close together in a document and is represented as an intersection of query-term influence functions. The value of a bigram cross term is defined as the value at the intersection, which is higher when the query terms are closer and lower when the query terms are farther apart. Thus the corresponding bigram cross term variants (e.g., within-document, within-query frequency of a bigram cross term, and the number of documents that contain the bigram cross term) are defined based on the value of the bigram cross term. one identical part in all of the bigram cross term variants is the distance between co-occurring positions of two query terms. for multiple query terms, we extend the concept to n-gram cross terms with several newly-defined high-dimensional distance metrics for the co-occurring positions of n query terms.

We propose a bigram Cross TErm Retrieval ($CRTER_2$) model as a basis model and an n-gram Cross TErm Retrieval ($CRTER_n$) model for n query terms, where $n > 2$. We incorporate the cross terms generated by different combinations of query terms into probabilistic models. Through extensive experiments on a relatively large set of representative TREC collections with various kernel functions, we show that the proposed models significantly outperform the BM25 baseline. By comparison, we conclude that the proposed $CRTER_2$ model is at least comparable to, if not better than, the state-of-the-art probabilistic proximity models, namely, *PPM* and *BM25TP*. We further extend LM by using cross terms and propose the corresponding $CRTER_2^{LM}$ model. The corresponding experiments show that the $CRTER_2^{LM}$ model is at least comparable to, if not better than, the state-of-the-art LM proximity models, *PLM* and *MRF*. In particular, compared to the other proximity models, both $CRTER_2$ and $CRTER_2^{LM}$ favor the top-ranked documents (e.g., top 5 and 20). Moreover, we discuss how parameter settings affect the retrieval performance of $CRTER_2$. A group of optimal parameters demonstrate the robustness of the $CRTER_2$ model on all collections used. We also experiment with the n-gram setting ($CRTER_n$) and find that $CRTER_n$ is a useful addition to $CRTER_2$.

As we know, the most fundamental elements in IR are terms. Therefore, proposing the concept of cross term as a pseudo term can allow us to simulate and manipulate term associations in any retrieval model. The implementation of cross terms using kernel functions provides a proper way to define and estimate cross terms.

The concept of cross term provides various possible future research directions. One direction is to further study cross-term distribution by examining other solutions to implement the cross term. It is also interesting to explore its usefulness in the process of query expansion when the association among the original query terms and the candidate expansion term is considered. Further, as observed in the experiments in which each kernel function has its advantage on some aspects, it is promising to adapt different kernel functions for different cross terms. Moreover, due to the complexity of semantic information, assigning different weights to term associations may also be

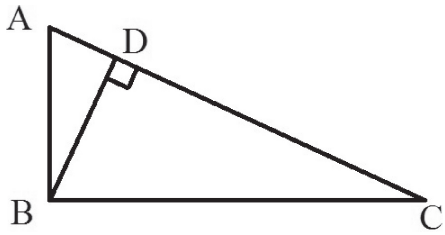


Fig. 22. Altitude- and hypotenuse-based distance (if we call the points at positions p_1, p_2, p_3 A, B, and C, then we build a right triangle $\triangle ABC$ using these three points. $dist_{alti}(p_1, p_2, p_3)$ is the altitude of $\triangle ABC$, and $dist_{hypo}(p_1, p_2, p_3)$ is the hypotenuse of $\triangle ABC$.)

one future direction. Finally, we will continue to study how to reduce the computational complexity of the model implementation, especially for real applications.

APPENDIX

THEOREM 10.1. *Altitude-based distance and hypotenuse-based distance have the following different properties.*

(1) Hypotenuse-based distance has a higher value than altitude-based distance: $dist_{hypo}(p_1, p_2, p_3, ..., p_N) > dist_{alti}(p_1, p_2, p_3, ..., p_N)$.

(2) Suppose the span $p_N - p_1$ *is fixed, the median positions affect altitude-based distance and hypotenuse-based distance differently: altitude-based distance has a higher value, while hypotenuse-based distance has lower value, if the median positions tend to spread evenly, that is,* $p_2 - p_1 = p_3 - p_2 = ... = p_N - p_{N-1}$.

PROOF. We take three positions p_1, p_2, p_3 as an example to discuss the detailed characteristics of the distances. If we call the points at positions p_1, p_2, p_3 A, B, and C, then we have $dist_{alti}(p_1, p_2, p_3) = \sqrt{AB \cdot BC}$ and $dist_{hypo}(p_1, p_2, p_3) = \sqrt{AB^2 + BC^2}$. In Figure 22, we build a right triangle $\triangle ABC$ using these three points. We can see that $dist_{alti}(p_1, p_2, p_3)$ is the altitude of $\triangle ABC$, and $dist_{hypo}(p_1, p_2, p_3)$ is the hypotenuse of $\triangle ABC$.

(1) The hypotenuse AC is always longer than altitude BD, therefore hypotenuse-based distance has a higher value than altitude-based distance.

(2) Suppose we have two groups of points, $\{A, B, C\}$ and $\{A', B', C'\}$, where the spans are the same, $AB + BC = A'B' + B'C'$, and the median point distributes differently, $A - B = \delta, A' - B' = \delta', \delta \neq \delta'$. If B is closer to the middle than B', then $\delta < \delta'$. We can deduce the following formulas:

$$AB \cdot BC - A'B' \cdot BC = \frac{\delta'^2 - \delta^2}{2} > 0,$$

$$(AB^2 + BC^2) - (A'B'^2 + B'C'^2) = \frac{\delta^2 - \delta'^2}{2} < 0.$$

Thus, we have

$$\sqrt{AB \cdot BC} > \sqrt{A'B' \cdot BC},$$

$$\sqrt{AB^2 + BC^2} < \sqrt{A'B'^2 + B'C'^2}.$$

Therefore, altitude-based distance has a higher value when B is closer to the middle. On the contrary, hypotenuse-based distance has a lower value when B is closer

to the middle. The proof is similar for the higher-dimensional case: $dist_{alti}(\cdot)$ is the product of altitudes; $dist_{hypo}(\cdot)$ is the hypotenuse built by lower-dimensional hypotenuses. □

ACKNOWLEDGMENTS

We gratefully appreciate the four anonymous reviewers and Professor Stephen Robertson for their valuable and detailed comments that greatly helped to improve the quality of this article. Special thanks go to Associate Editor Fabrizio Sebastiani for his hard work in reviewing and providing important feedback. The authors would like to thank Ben He for his help with the experiments at the early stage of research. We also would like to thank IBM Canada for providing IBM BladeCenter blade servers to conduct experiments. Finally, we greatly acknowledge Professor Stephen Robertson and Mladen Kovacevic from IBM Canada for their rigorous proofreading.

REFERENCES

Agrawal, R. and Phatak, M. 2013. A novel algorithm for automatic document clustering. In *Proceedings of the IEEE International Advance Computing Conference (IACC)*. IEEE, 877–882.

Ahmed, F. and Nurnberger, A. 2009. Evaluation of n-gram conflation approaches for Arabic text retrieval. *J. Amer. Soc. Inf. Sci. Technol. 60*, 7, 1448–1465.

Allan, J., Ballesteros, L., Callan, J., Croft, W. B., and Lu, Z. 1995. Recent experiments with inquery. In *Proceedings of the 4th Text Retrieval Conference*. 49–64.

Alvarez, C., Langlais, P., and Nie, J. 2004. Word pairs in language modeling for information retrieval. In *Proceedings of the 7th International Conference on Computer Assisted Information Retrieval*. 686–705.

Beaulieu, M., Gatford, M., Huang, X., Robertson, S., Walker, S., and Williams, P. 1997. Okapi at TREC-5. In *Proceedings of the 5th Text REtrieval Conference*. NIST Special Publication SP, 143–166.

Beigbeder, M. and Mercier, A. 2005. An information retrieval model using the fuzzy proximity degree of term occurences. In *Proceedings of the ACM Symposium on Applied Computing*. ACM, New York, NY, 1018–1022.

Bendersky, M. and Croft, W. B. 2008. Discovering key concepts in verbose queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 491–498.

Bendersky, M. and Croft, W. B. 2012. Modeling higher-order term dependencies in information retrieval using query hypergraphs. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'12)*. ACM, New York, NY, 941–950.

Bendersky, M., Croft, W. B., and Smith, D. A. 2009. Two-stage query segmentation for information retrieval. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 810–811.

Bendersky, M., Metzler, D., and Croft, W. B. 2010. Learning concept importance using a weighted dependence model. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*. ACM, New York, NY, 31–40.

Broschart, A. and Schenkel, R. 2008. Proximity-aware scoring for XML retrieval. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 845–846.

Buttcher, S., Clarke, C., and 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 on Research and Development in Information Retrieval*. ACM, New York, NY, 621–622.

Callan, J., Croft, W. B., and Harding, S. 1992. The INQUERY retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert Systems Applications*. 78–83.

Clarke, C., Cormack, G., and Burkowski, F. 1996. Shortest substring ranking (MultiText experiments for TREC-4). In *Proceedings of the 4th Text REtrieval Conference*. 295–304.

Cormack, G. V., Smucker, M. D., and Clarke, C. L. 2011. Efficient and effective spam filtering and re-ranking for large Web datasets. *Inf. Retrieval 14*, 5, 441–465.

de Kretser, O. and Moffat, A. 1999. Effective document presentation with a locality-based similarity heuristic. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 113–120.

Fan, Y., Huang, X., and An, A. 2006. York University at TREC 2006: Enterprise email discussion search. In *Proceedings of the 5th Text REtrieval Conference*. Special Publication 500–272.

Ferrari, V., Jurie, F., and Schmid, C. 2007. Accurate object detection with deformable shape models learnt from images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1–8.

Gao, J., Nie, J., Wu, G., and Cao, G. 2004. Dependence language model for information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 170–177.

Guo, J., Xu, G., Li, H., and Cheng, X. 2008. A unified and discriminative model for query refinement. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 379–386.

Hawking, D. and Thistlewaite, P. 1995. Proximity operators - So near and yet so far. In *Proceedings of the 4th Text Retrieval Conference*. 131–143.

He, B., Huang, J. X., and Zhou, X. 2011. Modeling term proximity for probabilistic information retrieval models. *Inf. Sci. J. 181*, 14, 3017–3031.

He, Z., Chen, C., Bu, J., Wang, C., Zhang, L., Cai, D., and He, X. 2012. Document summarization based on data reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence*. IEEE, 620–626.

Hou, Y., Zhao, X., Song, D., and Li, W. 2013. Mining pure high-order word associations via information geometry for information retrieval. *ACM Trans. Inf. Syst. 31*, 3.

Hu, Q., Huang, J., and Hu, X. 2012. Modeling and mining term association for improving biomedical information retrieval performance. *BMC Bioinformatics 13*, 9, 1–18.

Huang, X. and Hu, Q. 2009. A bayesian learning approach to promoting diversity in ranking for biomedical information retrieval. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieva*. ACM, New York, NY, 307–314.

Huang, X., Robertson, S., Cercone, N., and An, A. 2000. Probability-based Chinese text processing and retrieval. *Comput. Intell. 16*, 4, Wiley Online Library, 552–569.

Huang, X., Zhong, M., and Si, L. 2005. York University at TREC 2005: Genomics track. In *Proceedings of the 14th Text REtrieval Conference*.

Huang, X., Huang, Y. R., Wen, M., An, A., Liu, Y., and Poon, J. 2006a. Applying data mining to pseudo-relevance feedback for high performance text retrieval. In *Proceedings of the 6th IEEE International Conference on Data Mining*. IEEE, 295–306.

Huang, X., Wen, M., An, A., and Huang, Y. R. 2006b. A platform for okapi-based contextual information retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 728.

Huang, J. X., Miao, J., and He, B. 2013. High performance query expansion using adaptive co-training. *Inf. Process. Manage. 49*, 2, 441–453.

Kise, K., Junker, M., Dengel, A., and Matsumoto, K. 2004. Passage retrieval based on density distributions of terms and its applications to document retrieval and question answering. In *Reading and Learning: Adaptive Content Recognition*. Lecture Notes in Computer Science, vol. 2956. Springer-Verlag, Berlin, 306–327.

Kurland, O. and Lee, L. 2004. Corpus structure, language models, and ad hoc information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 194–201.

Lavrenko, V. and Croft, W. B. 2001. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 120–127.

Lv, Y. and Zhai, C. 2009. Positional language models for information retrieval. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 299–306.

Mayfield, J. and McNamee, P. 2003. Single n-gram stemming. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 415–416.

McNamee, P. and Mayfield, J. 2004. Character n-gram tokenization for European language text retrieval. *Inf. Retrieval 7*, 1, 73–97.

Meghini, C., Sebastiani, F., and Straccia, U. 1997. The terminological image retrieval model. In *Proceedings of the 9th International Conference on Image Analysis and Processing*. Lecture Notes in Computer Science, vol. 1311. Springer, 156–163.

Mercier, A. and Beigbeder, M. 2005. Fuzzy proximity ranking with boolean queries. In *Proceedings of the 5th Text REtrieval Conference*.

Metzler, D. and Croft, W. B. 2005. A Markov random field model for term dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 472–479.

Miao, J., Huang, J. X., and Ye, Z. 2012. Proximity-based Rocchio's model for pseudo relevance. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 535–544.

Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., and Lioma, C. 2006a. Terrier: A high performance and scalable information retrieval platform. In *Proceedings of the OSIR Workshop*. 18–25.

Ounis, I., De Rijke, M., Macdonald, C., Mishne, G., and Soboroff, I. 2006b. Overview of the TREC-2006 blog track. In *Proceedings of the 15th Text REtrieval Conference*.

Ounis, I., Lioma, C., Macdonald, C., and Plachouras, V. 2007. Research directions in terrier: A search engine for advanced retrieval on the Web. *CEPIS Upgrade J. 8*, 1.

Park, J. H., Croft, W. B., and Smith, D. A. 2011. A quasi-synchronous dependence model for information retrieval. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, 17–26.

Petkova, D. and Croft, W. B. 2007. Proximity-based document representation for named entity retrieval. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management*. ACM, New York, NY, 731–740.

Pickens, J. 2000. A comparison of language modeling and probabilistic text information retrieval approaches to monophonic music retrieval. In *Proceedings of the International Symposium on Music Information Retrieval*.

Ponte, J. M. and Croft, W. B. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 275–281.

Rasolofo, Y. and Savoy, J. 2003. Term proximity scoring for keyword-based retrieval systems. In *Proceedings of the 25th European Conference on Information Retrieval Research*. Lecture Notes in Computer Science, vol. 2633. Springer, 207–218.

Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M., and Gatford, M. 1996. Okapi at TREC-4. In *Proceedings of the 4th Text Retrieval Conference*. 73–97.

Robertson, S. E. and Walker, S. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 232–241.

Sebastiani, F. 1994. A probabilistic terminological logic for modelling information retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Springer, 122–130.

Shi, L. and Nie, J.-Y. 2010. Using various term dependencies according to their utilities. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, 1493–1496.

Song, F. and Croft, W. B. 1999. A general language model for information retrieval. In *Proceedings of the 8th International Conference on Information and Knowledge Management*. ACM, New York, NY, 316–321.

Song, R., Yu, L., Wen, J.-R., and Hon, H.-W. 2011. A proximity probabilistic model for information retrieval. Tech. rep., Microsoft Research.

Srikanth, M. and Srihari, R. 2002. Biterm language models for document retrieval. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 425–426.

Strohman, T., Metzler, D., Turtle, H., and Croft, W. B. 2005. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*. Vol. 2, 2–6.

Tao, T. and Zhai, C. 2007. An exploration of proximity measures in information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 295–302.

Voorhees, E. and Harman, D. 2005. *TREC: Experiment and Evaluation in Information Retrieval*. Vol. 32, MIT Press.

Wang, J. and Zhu, J. 2009. Portfolio theory of information retrieval. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 115–122.

Wang, M. and Si, L. 2008. Discriminative probabilistic models for passage based retrieval. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 419–426.

Wei, X. and Croft, W. B. 2007. Modeling term associations for ad-hoc retrieval performance within language modeling framework. In *Proceedings of the 29th European Conference on Information Retrieval Research*. Lecture Notes in Computer Science, vol. 4425. Springer, 52–63.

White, R., Ruthven, I., and Jose, J. 2002. Finding relevant documents using top ranking sentences: An evaluation of two alternative schemes. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 57–64.

Wong, S. K. M. and Yao, Y. Y. 1993. A probabilistic method for computing term-by-term relationships. *J. Am. Soc. Inf. Sci. 44*, 8, 431–439.

Ye, Z., Huang, J. X., and Miao, J. 2012. A hybrid model for ad-hoc information retrieval. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 1025–1026.

Ye, Z., Huang, X., He, B., and Lin, H. 2009. York University at TREC 2009: Relevance feedback track. In *Proceedings of the 18th Text REtrieval Conference*.

Ye, Z., Huang, X., Hu, Q., and Lin, H. 2010. An integrated approach for medical image retrieval through combining textual and visual features. In *Multilingual Information Access Evaluation II. Multimedia Experiments*. Lecture Notes in Computer Science, vol. 6242. Springer, 195–202.

Yin, X., Huang, J., Li, Z., and Zhou, X. 2013. A survival modeling approach to biomedical search result diversification using wikipedia. *IEEE Trans. Knowl. Data Eng. 25*, 6, 1201–1212.

Yu, X., Liu, Y., Huang, X., and An, A. 2012. Mining online reviews for predicting sales performance: A case study in the movie domain. *IEEE Trans. Knowl. Data Eng. 24*, 4, 720–734.

Zhai, C. and Lafferty, J. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 334–342.

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

Zhao, J. and Yun, Y. 2009. A proximity language model for information retrieval. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 291–298.

Zhao, J., Huang, J., and He, B. 2011. CRTER: Using cross terms to enhance probabilistic information retrieval. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 155–164.

Zhao, J., Huang, J. X., Hu, X., Kurian, J., and Melek, W. 2012a. A bayesian-based prediction model for personalized medical health care. In *Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine*. IEEE, 579–582.

Zhao, J., Huang, J., and Wu, S. 2012b. Rewarding term location information to enhance probabilistic information retrieval. In *Proceeding of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 1137–1138.

Zhao, J., Huang, J. X., and Hu, X. 2013. BPLT$^+$: A Bayesian-based personalized recommendation model for health care. *BMC Genomics J. 14(Suppl 4)*, S6.

Received November 2012; revised May 2013, September 2013, January 2014; accepted January 2014