Exploring and Exploiting Proximity Statistic for Information Retrieval Model

Yadong Zhu, Yuanhai Xue, Jiafeng Guo, Yanyan Lan,
Xueqi Cheng, and Xiaoming Yu

Institute of Computing Technology, Chinese Academy of Sciences
{zhuyadong,xueyuanhai,yuxiaoming}@software.ict.ac.cn
{guojiafeng,lanyanyan,cxq}@ict.ac.cn

Abstract. Proximity among query terms has been recognized to be useful for boosting retrieval performance. However, how to model proximity effectively and efficiently remains a challenging research problem. In this paper, we propose a novel proximity statistic, namely Phrase Frequency, to model term proximity systematically. Then we propose a new proximity-enhanced retrieval model named BM25PF that combines the phrase frequency information with the basic BM25 model to rank the documents. Extensive experiments on four standard TREC collections illustrate the effectiveness of the BM25PF model, and also shows the significant influence of the phrase frequency on retrieval performance.

Keywords: Phrase Frequency, Proximity Statistic, BM25.

1 Introduction

In the past decades, many different retrieval models have been proposed and successfully applied in document ranking, including vector space models [19], classical probabilistic models [16, 18], and statistical language models [14]. By assuming full independence between terms, most of them represent query and document as bag of words, and mainly exploit term based statistics such as within-document term frequency (tf), inverse document frequency (idf) and document length in ranking. However, the existing methods often lack a way to explicitly model the proximity among query terms within a document, which has been widely recognized to be useful for boosting retrieval performance [24].

Recently, many studies have been conducted to capture proximity in retrieval. One is to index phrases instead of terms which can capture proximity indirectly [5]. However, it is shown that the index methods may not perform consistently well across a variety of collections [12]. Another way is the dependency modeling [20, 22, 6, 11, 8], which considers the use of term dependency such as high-order n-grams, and integrates them into the existing retrieval models. The main problem of those dependency models lies in that the parameter space becomes very large, which will lead to difficulty in parameter estimation and sensitivity to data sparse and noise [26].

A simple but effective way is direct proximity modeling. Some early studies on direct proximity modeling try to add proximity factors into probabilistic ranking

Y. Hou et al. (Eds.): AIRS 2012, LNCS 7675, pp. 1–13, 2012.

models in a heuristic manner, but their experiment results are not conclusive and show limited success [3, 7, 15, 2]. Recent work [24] and [25] obtains a promising performance improvement essentially by a pairwise term distance measure. While considering only word pairs may not be appropriate for long queries, and many redundant proximity information may be involved which may be harmful to the final performance. For example, given a user query "second hand car in Detroit", word pairs such as "second car" or "hand car" can not convey the underlying concept of "second hand car" correctly. Therefore, how to model proximity effectively and efficiently remains a challenging research problem.

In this paper, we propose a novel proximity statistic named Phrase Frequency (pf). It reflects the statistical frequency information of a query phrase provided by a document, beyond the traditional term statistics such as tf and idf. It is defined based on the definition of Span_Cover. A Span_Cover is a basic document segment that covers each query term at least once within a limited window size, and no overlap between each other. It is different from the existing two span-based definitions: Span (or FullCover) and MinCover [24, 4]. Four proximity-based density functions are considered to produce proximity score of a span_cover instance, which is small when query terms within the span_cover are far away, and large when query terms are close to each other. The phrase frequency is then computed by accumulating the proximity scores within a document.

Based on the concept of phrase frequency, we propose a proximity-enhanced model referred as BM25PF. Through extensive experiments on different sizes and genres of TREC collections, we show that the BM25PF outperforms the BM25 baseline and is also comparable to the state-of-art approaches in general.

The rest of this paper is organized as follows. In Section 2, we introduce the previous related work. In Section 3, we introduce the concept of pf, corresponding calculating algorithm, and the practical application strategy. In Section 4, we propose a proximity-enhanced model named BM25PF based on pf. Then, in Section 5, we test the proposed model BM25PF on four TREC collections, and compare their performance with state-of-the-art approaches. Finally in Section 6, we conclude the paper with a discussion of our work and future work.

2 Related Work

Term proximity has been previously studied in several approaches. One approach tries to index phrases instead of terms. In such a way, dependence between words can be captured indirectly [5, 12]. However, the index method does not perform consistently well across a variety of collections.

Another way to capture proximity is the dependency modeling. Several work has been conducted to integrate term dependence into the language model. Song et al. propose a general model that combines bigram language model with unigram language model under several smoothing techniques [20]. Srikanth et al. introduce a biterm language model that takes into account dependency between unordered adjacent word pairs [22]. [13, 6] put forward dependency language models that consider the relation between terms existed in a dependency tree.

Metzler and Croft propose a general framework for model term dependencies based on Markov Random Field [11], in which term structures with different levels of proximity can be defined and captured. He et al. [8] extend the traditional BM25 model to a combination of a series of n-gram BM25 models that will take the proximity between query terms into account.

Direct proximity modeling has been considered widely recently. [3] and [7] appear to be the first to evaluate proximity on TREC data sets. Both of them try to measure proximity by using the shortest interval containing a match set. Recent work has attempted to heuristically incorporate proximity factor into probabilistic ranking functions [15, 2], but their experiments are not conclusive and show limited success. Song et al. represent term proximity by co-occurrences of terms in non-overlapping phrases [21]. Zhao and Yun's work views the query term proximity as the Dirichlet hyper-parameter that weights the parameters of the unigram document language model [26]. Lv and Zhai apply a series of kernel functions to estimate a language model for every position in a document [10]. Overall, direct proximity modeling is an economic and effective way to go beyond the "bag of words" assumption in retrieval. In our paper, we will also follow the way by introducing a novel proximity statistic.

Density functions based on proximity have been used to propagate term influence [10, 25]. Lv and Zhai [10] incorporate the term proximity by using four kernel functions: Gaussian, Triangle (Linear), Cosine, and Circle. [25] additionally introduces three more kernel functions: Quartic, Epanechnikov, Triweight. In our work, we utilize two of them: Gaussian and Linear, and introduce two more kernel functions containing Exponential and Negative Power.

3 A Novel Proximity Statistic: Phrase Frequency

In this section, we introduce the concept of Phrase Frequency and the associated calculating algorithm. Before giving the formal definition of Phrase Frequency, we will first define a new concept: *Span_Cover*, which is different from the previous notions of Span (or FullCover) and MinCover [24, 4].

3.1 Span_Cover

The traditional span-based notions: FullCover and MinCover, which can be viewed as two kinds of "boundary" definitions, while the Span_Cover can be viewed as a basic unit between them.

Definition 1. ***(Span_Cover)*** *Span_Cover is defined as a document segment that will have following properties:*

(1) Starting from a query term and ending with a query term;
(2) Covering each query phrase term at least once;
(3) Its length is no more than a limited maximal window size, which is usually adaptively set as integral times of the number of unique query terms;

(4) Only the shortest of several span_covers sharing a common starting points is counted as an instance
(5) There is no overlapped terms between any two span_cover instances under a certain scanning way;

Given a short document d as an example to explain our definitions [24].

$$d = t_1, t_2, t_1, t_3, t_5, t_4, t_2, t_3, t_4$$

Given a query: $\{t_1, t_2\}$, and setting the maximal windows size as 4 times the number of query terms, its corresponding Span_Cover instances would be $\{t_1, t_2\}$ and $\{t_1, t_3, t_5, t_4, t_2\}$ by a sequentially scanning way. The alert reader will notice that, the segment $\{t_2, t_1\}$ will be also a proper span_cover satisfying all the above properties. Usually, we will detect all the Span_Cover instances provided by a relevant document in a sequential scanning way. Certainly there may exist some other scanning ways, we will study them in our future work.

3.2 Phrase Frequency

The Phrase Frequency is a novel proximity statistic which reflects the statistical frequency information of a query phrase within a relevant document. Given a user query phrase that contains at least two terms and a candidate document, there exists several factors that may affect the relevance of a document:

1. Density: More tighter or shorter a Span_Cover instance, the greater the likelihood that the document is relevant.
2. Number: The more Span_Cover instances provided by a document, the greater the likelihood that the document is relevant.
3. Order: The order in a certain Span_Cover instance may also have some influences on query proximity.

Obviously, the traditional method of proximity modeling based on Mincover, which only considers the Density factor, may not be appropriate. Similarly, the method based on FullCover can not explicitly detect the number of Span_Cover instances or the density of a specific instance. In fact, the experiment results in [24] also show that they are not effective in general. Meanwhile, many previous work has shown that the order of appearances of query term is not important [11, 8]. Therefore, instead of strict order constraint, rather, we expect the query terms to appear ordered or unordered within a Span_Cover instance. And the definition of phrase frequency must systematically consider the density and number of Span_Cover instances.

Given a query phrase Q with K unique query terms and the allowed maximal windows size as wK (w usually set as positive integer). Supposing m Span_Cover instances have been detected in a candidate document D as follows:
$\{Span\_Cover_1, Span\_Cover_2, ..., Span\_Cover_m\}$

Definition 2. *The **Phrase Frequency** (pf) of a given query phrase Q in D will be calculated as the following.*

$$pf(Q, D) = \sum_{i=1}^{m} pf_i = \sum_{i=1}^{m} Density(|Span\_Cover_i| - K) \tag{1}$$

where $Density(\cdot)$ is a type of proximity-based density function that transforms a $Span\_Cover_i$ instance to a proximity score belonging to (0, 1], which is notated as pf_i. $|Span\_Cover_i|$ is the length of the ith Span_Cover: $|Span\_Cover_i| = pos_{end} - pos_{start} + 1$. The more tighter $Span\_Cover_i$, the higher pf_i will be.

While if there does not exist a legal Span_Cover instance in document D, we will assign a uniform minimal value for pf in the following way:

$$pf(Q, D) = Density(wK) \tag{2}$$

Obviously, $Density(\cdot)$ plays an important role in pf. It must follow several properties such as non-negative, monotonic [10, 25]. We will analyze and explore a number of proximity-based density functions in following subsection.

3.3 Proximity-Based Density Functions

A major technical challenge in the definition of pf is how to define the density functions $Density(\cdot)$. Following some previous work [24, 10, 25], we study four representative kernel functions: Gaussian, Linear, Exponential, and Negative Power. They are all non-increasing functions, and represent different ways in which the proximity information (pf_i) decreases when $|Span\_Cover_i|$ increases.

1. **Gaussian Kernel**
$$Kernel(x) = e^{-\frac{x^2}{2a^2}} \tag{3}$$
2. **Linear Kernel**
$$Kernel(x) = ax + 1,\ a < 0 \tag{4}$$
3. **Exponential Kernel**
$$Kernel(x) = e^{-ax},\ a > 0 \tag{5}$$
4. **Negative Power Kernel**
$$Kernel(x) = (ax + 1)^k,\ k < 0, a > 0 \tag{6}$$

The first two kernels have been used in [10, 25], and the last two kernel functions are firstly introduced in our work. The performance of using all the kernel functions will be investigated in the experiments. Obviously, when $|Span\_Cover_i| = K$, we will get the maximal value: $pf_i = 1$, which means a consecutive occurrence of a query phrase. a is a normalization parameter to make sure the output belonging to (0, 1]. It is usually determined by query length and the maximal windows size together.

3.4 Algorithm and Time Complexity

In this section, we present an effective and efficient calculating algorithm of pf by a sequential scanning way. Then we give an analysis of its time complexity. We show that its time complexity is close to linear in terms of N, which is the total number of occurrences of all the query terms within a document. The corresponding calculating algorithm is summarized in Algorithm 1.

Given a query phrase $Q = \{t_1, t_2, ..., t_K\}$, supposing that a candidate document D matches the K unique query terms, and the total number of occurrences

Algorithm 1. pf **Calculating using a Sequential Scanning**

Input:
K- query length in terms of unique words
N- the total number of query term occurrence
w- the parameter controlling the maximal windows size
$Hit[]$- a ordered chain of every query term occurrence
$List[]$- a list of length K
$kernel()$- a certain kernel function
$existpf$- a flag indicating whether pf exists or not
Output: pf score provided by a relevant document

```
1: for each i ∈ [0, K − 1] do
2:     List[i] = −1
3: end for
4: pf = 0
5: existpf = false
6: for i = 0 to N − 1 do
7:     for j = 0 to K − 1 do
8:         if Hit[i].term == t_j then
9:             List[j] = Hit[i].pos
10:            break
11:        end if
12:    end for
13:    start = min(List[0], List[1], ..., List[K − 1])
14:    end = Hit[i].pos
15:    if start ≠ −1 then
16:        cover_len = end − start + 1
17:        if cover_len ≤ w ∗ K then
18:            pf+ = kernel(cover_len − K)
19:            existpf = true
20:            for m = 0 to K − 1 do
21:                List[m] = −1
22:            end for
23:        end if
24:    end if
25: end for
26: if !existpf then
27:    pf = kernel(w ∗ K)
28: end if
```

of these K query terms is N. All the query term occurrences compose a chain of ordered hits: $\{t_{P_1}, t_{P_2}, ..., t_{P_N}\}$, and $t_{P_i} \in Q$, $\forall i \in \{1, 2..., N\}$. We record the position and the term information of every query term occurrence in a Hit set. While scanning, we maintain a list of length K: $List[0, ..., K-1]$, in which we store the temporary position of each seen query term. We set the allowed maximal windows size as $w * K$, w is a positive integer.

The first 5 steps are initialization. Specially we assign every position in $List[]$ a negative value of -1. For steps 7 to 12, we update the corresponding position information of a certain query term when scanning an occurrence sequentially. In steps 13 and 14, we record the *start* and *end* position information after every position updating. Then we will judge whether every position in $List$ has been updated in step 15, which corresponds with the second property of Span_Cover. In steps 16 and 17, we calculate the length of the segment, and then judge whether it satisfies the length constraint property. In steps 18 and 19, we accumulate the pf score provided by a certain Span_Cover instance and set the $existpf$ flag as true. From step 20 to 22, we reinitialize the position information in $List[]$ to make sure there is no overlapping between any two Span_Cover instances. Finally we will assign a uniform minimal value for pf from step 26 to 28 if there does not exist a legal Span_Cover instance. The algorithm for calculating pf has the time complexity: $O(N * K)$. Since K is often very small, the time complexity is close to linear in terms of N: $O(N)$, and N is the total number of query term occurrence within a document.

3.5 A Practical Application Strategy

In practical retrieval environment, there exists many long user queries (5 or more terms). A Span_Cover instance of a user query requires to cover each query term at least once within a limited windows size. This condition may be too strong for these long user queries. On the other hand, when user constructs a long query, he will often aggregate additional terms to the concepts which will make them less meaningful [1]. Ideally, we should select the meaningful term sequences, and consider the proximity information of them only. Therefore, we naturally consider to combine proximity modeling with the query segmentation techniques together. Query segmentation is an important task toward understanding queries accurately[9, 23], which is essential for improving search results in modern information retrieval. It aims to separate query words into segments so that each segment maps to a semantic component or sub-query phrases.

For a user query, if its query length is less than 5 terms, we calculate the pf score as Algorithm 1 normally, or else we need to do a user query structure analysis first by query segmentation techniques, and then calculate as follows:

$$pf(Q, D) = \sum_{sub_i \subseteq Q} wei_i \times pf(sub_i, D), \quad |sub_i| \geq 2,\ wei_i \in (0, 1] \qquad (7)$$

where sub_i is a sub-query phrase, which contains at least two query terms, wei_i is a normalization weight factor assigned for sub_i, which can reflect how meaningful

the sub-query phrase is. $pf(sub_i, D)$ denotes the pf score of a specific sub-query phrase, which will also be calculated by Algorithm 1.

There exists several methods for query segmentation, and the result sub-query phrases can be weighted correspondingly. In our paper, we use the method presented in [9], and define the wei_i as follows:

$$wei_i = \frac{connexity(sub_i)}{\sum_{sub_i \subseteq Q} connexity(sub_i)} \tag{8}$$

$$connexity(sub_i) = freq(sub_i) \times I(w_{i_1}...w_{i_{n-1}}, w_{i_2}...w_{i_n}) \tag{9}$$

where $connexity(sub_i)$ denotes the *connexity* score of the *ith* sub-query phrase (or segment)[9], which is defined as a product of the global frequency of the segment ($freq(sub_i)$) and the mutual information (I) between longest but complete subsequence of sub_i.

4 Proximity-Enhanced Retrieval Model

In this section, we will test whether our proposed proximity statistic can really boost the retrieval performance of the existing retrieval models, which are usually derived from bag-of-words assumption (e.g. Okapi BM25, KL-divergence language model). In this paper, we choose to combine the pf score with the representative state-of-the-art retrieval models: Okapi BM25 model [17, 18]. Previous extensive experiments show that BM25 can provide a robust and effective retrieval performance. The BM25 retrieval model can be expressed as follows:

$$BM25(Q, D) = \sum_{q_i \in q \cap d} (w_i \times \frac{(k_1 + 1) \times c(q_i, d)}{k_1((1 - b) + b\frac{|d|}{avdl}) + c(q_i, d)} \times \frac{(k_3 + 1) \times c(q_i, q)}{k_3 + c(q_i, q)})$$

where $c(q_i, d)$ is within-document term frequency, and $c(q_i, q)$ is within-query term frequency. Additionally, $|d|$ is document length, and $avdl$ is average document length. k_1, k_3 and b are tuning parameters. $w_i = ln\frac{N-n+0.5}{n+0.5}$ is q_i's term weight. N is the number of documents within a collection, n is the document frequency of term q_i.

And then we can define a new combined proximity-enhanced retrieval model as follows:

$$BM25PF(Q, D) = \lambda BM25(Q, D) + (1 - \lambda)pf(Q, D) \tag{10}$$

where λ is a trade-off parameter reflecting the influence of the proximity statistic. When λ equals to 1, the BM25PF model degenerates to the basic BM25 model.

5 Experiments

We conduct a series of experiments on four standard TREC collections: AP88-89, FR88-89, WT10G and ClueWebB. They represent different sizes and genres of

Table 1. Basic Data Set Statistics

	AP88-89	FR88-89	WT10G	ClueWeb B
Topics	51-100	51-100	451-550	rf.01-rf.50
# of Topics	50	21	100	50
# of Docs	164,597	45,820	1,692,096	49,375,681

Table 2. Performance comparison between BM25 and BM25PF with different kernels

	WT10G			ClueWeb B			AP88-89			FR88-89		
	MAP	P@5	P@10	MAP	P@5	P@10	MAP	P@5	P@10	MAP	P@5	P@10
BM25	0.2131	0.3864	0.291	0.2109	0.3780	0.2806	0.2670	0.4358	0.3970	0.2892	0.1670	0.1450
BM25PF_E	0.2210	0.4006	0.3019	0.2134	0.4034	0.2897	0.2725	0.4480	0.4153	0.2971	0.1741	0.1520
BM25PF_N	0.2226	0.4017	0.3023	0.2176	0.4053	0.2894	0.2730	0.4479	0.4200	0.2952	0.1743	0.1515
BM25PF_L	0.2231	0.4093	0.3051	0.2201	**0.4073***	0.2976*	0.2743	**0.4621***	0.4200	0.2973	0.1780*	0.1530
BM25PF_G	**0.226***	**0.4128***	**0.3057***	**0.2213**	0.4071*	**0.2981***	**0.2780**	0.4610*	**0.4208***	**0.30**	**0.1780***	**0.1539***

text collections. AP88-89 and FR88-89 are chosen as small homogeneous collections with little noise. The WT10G is a medium size crawl of Web documents. ClueWeb collection is the largest TREC test collection currently with a very large crawl of the Web. We use the category B of ClueWeb which contains about 50 million English Web pages. Queries are taken from the title field of the TREC topics. The basic statistics of the collections are illustrated in Table 1. We preform stemming with the Porter stemmer. And no stop words removing is done in both documents and queries to make sure the minimum preprocessing of them.

On each collection, we conduct a 2-fold cross-validation. The associated test topics are split into two equal subsets, referred as odd-number and even-number topics. In each fold, we use one subset of topics for training, and use the remaining subset for testing. The overall retrieval performance is averaged over the two test subsets of topics. Both top precision and average precision are used to evaluate the experiment result which include MAP/statMAP, P@5, P@10. All statistical tests are based on Wilcoxon matched-pairs signed-rank test at the 0.05 level.

5.1 Effectiveness of BM25PF

In this section, we evaluate the effectiveness of the proximity-enhanced model BM25PF. We use the optimal BM25 as our baseline. The BM25 has three main parameters. We set $k_1 = 1.2$ and $k_3 = 1,000$ suggested in [24] and tune b to be optimal. And the parameter b is set to be 0.3.

The related experimental results are presented in Table 2. We apply four kinds of kernel functions for instantiating BM25PF model, and notate them correspondingly as: BM25PF_G (BM25PF Gaussian), BM25PF_L (BM25PF Linear), BM25PF_E (BM25PF Exponential), BM25PF_N (BM25PF Negative Power). Additionally, the parameter a of them is set as: $w * K$, $-1/((w+1) * K)$, $w * K$ and 1, respectively. The parameter k in Equation 6 is set as -1 (in fact any negative value is allowed). All results are evaluated in terms of MAP, P@5, P@10.

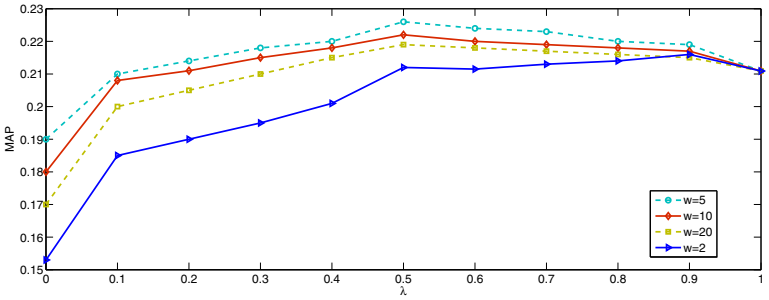

Fig. 1. Sensitivity to BM25PF parameter λ on ClueWeb B

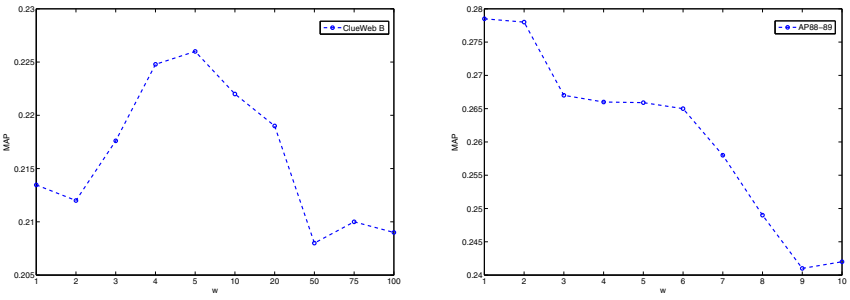

Fig. 2. Sensitivity to BM25PF proximity parameter w on ClueWeb B and AP88-89

Results with "$*$" indicate the improvement is significant at 0.05 level. The best result obtained on each collection is highlighted.

The results show that the proposed BM25PF model outperforms BM25 on all four data collections. Notablely, the proposed model has significant performance improvement especially on top retrieved documents in terms of P@5, P@10. And each kernel function shows a stable performance improvement especially Gaussian kernel, which is also consistent with the conclusion of the previous work [10]. The Gaussian kernel exhibits the following property: the density measurement would drop slowly when the Span_Cover length is small, but drop quickly as the length is in a middle range, and then drop slowly again when its length become large. Such an "S-shape" trend is reasonable for that: the dependent query terms are not always adjacent in documents, but can be a little far from each other, thus we would not like to make the density measurement of a Span_Cover instance so sensitive to the length when its length is small. However, when the Span_Cover length is just around the boundary of strong semantic associations, the density measurement should be more sensitive to the length change. Then as the Span_Cover length increases further, all query terms are presumably only loosely associated, and thus the density measurement again should not be so sensitive to the change of Span_Cover length.

We further compare the retrieval performance of the BM25PF with two state-of-the-art approaches proposed in [24] and [25] respectively. For the first approach, to be fair, we use the best retrieval formula suggested in [24] (BM25+MinDist), and tune parameter α to be optimal. We label this approach as 'BM25MD'. For the second approach, the CRTER model proposed in [25] which measures the association of every two query terms, we set their free parameters by using cross-validation, which is also the same way using for our approach. We conduct the performance comparison on all four data collections and report the comparison results in Table 3. Overall, the proposed BM25PF model shows a steady performance improvement on all four data collections. And it is at least comparable to, if no better than, the two state-of-art approaches.

Table 3. Performance comparison of different ranking models in terms of MAP

	WT10G	ClueWeb B	AP88-89	FR88-89
BM25	0.2131	0.2109	0.267	0.2892
BM25MD	0.2197	0.213	0.2713	0.2918
CRTER	0.2207	0.2138	0.2751	0.2926
BM25PF	**0.2261***	**0.2213**	**0.2780**	**0.3000**

5.2 Parameter Sensitivity of BM25PF

The BM25PF model introduces two parameters λ and w which will affect the retrieval performance. The parameter λ balances the influence of basic BM25 model and the proximity statistic. w is a proximity parameter for modeling different level proximity information. We choose the Gaussian for instantiating the BM25PF model based on the results shown in Table 2.

Figure 1 plot the sensitivity curves over λ values ranging from 0 to 1 in terms of MAP on ClueWeb data. And a group of value settings of parameter w are applied such as $w = 2, 5, 10, 20$. Overall, a reliable λ value is relatively insensitive to different w values, and $\lambda = 0.5$ seems to be best.

Figure 2 plot the sensitivity tendency over proximity parameter w on ClueWeb and AP88-89 respectively, in terms of MAP. These two data collections represent different data genres. Additionally, the λ is fixed at 0.5. From the first subfigure (ClueWeb B), we find that the BM25PF achieves its best performance when w value is around 5, which means the model capturing proximity information at a sentence-level. The second subfigure shows that when w is set a small value such as 1 or 2, the overall performance tend to be best. A larger windows size will bring noise, and the retrieval performance will decrease and be not stable. This may be due to the homogeneous, clean nature of the documents in these small newswire collections. And a strictly tighter matching Span_cover instances in the calculating of pf will capture high quality proximity information. While for the Web collections, which is heterogeneous and noisy collections, a larger windows size of sentence-level is appropriate because they can deal better with the noise inherent in Web documents.

6 Conclusions and Future Work

In this paper, we propose a novel proximity statistic, namely Phrase Frequency (pf), to model term proximity for boosting retrieval performance. It reflects the statistical frequency information of a query phrase within a document, and is approximated by a density function based on the definition of Span_Cover. pf systematically considers the density of a span_ cover instance and the numbers of span_cover within a document. In addition, we also present an efficient calculating algorithm of pf. Based on the concept of pf, we propose a proximity-enhanced retrieval model: BM25PF. Through extensive experiments on different sizes and genre of TREC collections, we show that the BM25PF outperforms the BM25 baseline and is also comparable to the state-of-art approaches in general.

The proposed proximity statistic pf has shown significant influence on retrieval performance. It is a novel and important proximity statistic as the same as the traditional term statistics such as tf and idf. In the future work, we can study how to eventually obtain a unified retrieval model with the incorporation of pf in a more soundly theoretical way.

Acknowledgements. This research work was funded by the National Natural Science Foundation of China under Grant No. 60933005, No. 61173008, No. 61003166, 973 Program of China under Grants No. 2012CB316303, and National Key Technology R&D program under Grant No. 2011BAH11B02.

References

[1] Bai, J., Chang, Y., Cui, H., Zheng, Z., Sun, G., Li, X.: Investigation of partial query proximity in web search. In: Proc. of the 17th WWW, pp. 1183–1184 (2008)
[2] Büttcher, S., Clarke, C.L.A., Lushman, B.: Term proximity scoring for ad-hoc retrieval on very large text collections. In: Proc. of the 29th SIGIR, pp. 621–622 (2006)
[3] Clarke, C.L.A., Cormack, G.V., Burkowski, F.J.: Shortest substring ranking (multitext experiments for trec-4). In: TREC, pp. 295–304 (1995)
[4] Cummins, R., O'Riordan, C.: Learning in a pairwise term-term proximity framework for information retrieval. In: Proc. of the 32nd SIGIR, pp. 251–258 (2009)
[5] Fagan, J.: Automatic phrase indexing for document retrieval. In: Proc. of the 10th ACM SIGIR, pp. 91–101 (1987)
[6] Gao, J., Nie, J.-Y., Wu, G., Cao, G.: Dependence language model for information retrieval. In: Proc. of the 27th ACM SIGIR, pp. 170–177 (2004)
[7] Hawking, D., Thistlewaite, P.B.: Proximity operators - so near and yet so far. In: TREC (1995)
[8] He, B., Huang, J.X., Zhou, X.: Modeling term proximity for probabilistic information retrieval models. Inf. Sci. 181, 3017–3031 (2011)
[9] Risvik, K.M., Mikolajewski, T., Boros, P.: Query segmentation for web search. In: Proc. of the 12th WWW (2003)
[10] Lv, Y., Zhai, C.: Positional language models for information retrieval. In: Proc. of the 32nd ACM SIGIR, pp. 299–306 (2009)

[11] Metzler, D., Croft, W.B.: A markov random field model for term dependencies. In: Proc. of the 28th ACM SIGIR, pp. 472–479 (2005)
[12] Mitra, M., Buckley, C., Singhal, A., Cardie, C.: An analysis of statistical and syntactic phrases. In: Proc. of 5th RIAO, pp. 200–214 (1997)
[13] Nallapati, R., Allan, J.: Capturing term dependencies using a language model based on sentence trees. In: Proc. of the 11th ACM CIKM, pp. 383–390 (2002)
[14] Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: Proc. of the 21st ACM SIGIR, pp. 275–281 (1998)
[15] Rasolofo, Y., Savoy, J.: Term Proximity Scoring for Keyword-Based Retrieval Systems. In: Sebastiani, F. (ed.) ECIR 2003. LNCS, vol. 2633, pp. 207–218. Springer, Heidelberg (2003)
[16] Robertson, S.E., Sparck Jones, K.: Relevance weighting of search terms. Journal of the American Society for Information Science 27(3), 129–146 (1976)
[17] Robertson, S.E., Walker, S., Hancock-Beaulieu, M.: Okapi at trec-7: Automatic ad hoc, filtering, vlc and interactive. In: TREC, pp. 199–210 (1998)
[18] Robertson, S.E., Zaragoza, H.: The probabilistic relevance framework: Bm25 and beyond. Foundations and Trends in Information Retrieval 3(4), 333–389 (2009)
[19] Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM 18, 613–620 (1975)
[20] Song, F., Croft, W.B.: A general language model for information retrieval. In: Proc. of the 8th ACM CIKM, pp. 316–321 (1999)
[21] Song, R., Taylor, M.J., Wen, J.-R., Hon, H.-W., Yu, Y.: Viewing Term Proximity from a Different Perspective. In: Macdonald, C., Ounis, I., Plachouras, V., Ruthven, I., White, R.W. (eds.) ECIR 2008. LNCS, vol. 4956, pp. 346–357. Springer, Heidelberg (2008)
[22] Srikanth, M., Srihari, R.: Biterm language models for document retrieval. In: Proc. of the 25th ACM SIGIR, pp. 425–426 (2002)
[23] Tan, B., Peng, F.: Unsupervised query segmentation using generative language models and wikipedia. In: Proc. of the 17th WWW, pp. 347–356 (2008)
[24] Tao, T., Zhai, C.: An exploration of proximity measures in information retrieval. In: Proc. of the 30th ACM SIGIR, pp. 295–302 (2007)
[25] Zhao, J., Huang, J.X., He, B.: Crter: using cross terms to enhance probabilistic information retrieval. In: Proc. of the 34th ACM SIGIR, pp. 155–164 (2011)
[26] Zhao, J., Yun, Y.: A proximity language model for information retrieval. In: Proc. of the 32nd ACM SIGIR, pp. 291–298 (2009)