Positional Language Models for Information Retrieval

Yuanhua Lv
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
ylv2@uiuc.edu

ChengXiang Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
czhai@cs.uiuc.edu

ABSTRACT

Although many variants of language models have been proposed for information retrieval, there are two related retrieval heuristics remaining "external" to the language modeling approach: (1) proximity heuristic which rewards a document where the matched query terms occur close to each other; (2) passage retrieval which scores a document mainly based on the best matching passage. Existing studies have only attempted to use a standard language model as a "black box" to implement these heuristics, making it hard to optimize the combination parameters.

In this paper, we propose a novel *positional language model* (PLM) which implements both heuristics in a unified language model. The key idea is to define a language model for each *position* of a document, and score a document based on the scores of its PLMs. The PLM is estimated based on propagated counts of words within a document through a proximity-based density function, which both captures proximity heuristics and achieves an effect of "soft" passage retrieval. We propose and study several representative density functions and several different PLM-based document ranking strategies. Experiment results on standard TREC test collections show that the PLM is effective for passage retrieval and performs better than a state-of-the-art proximity-based retrieval model.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms

Algorithms

Keywords

Positional language models, proximity, passage retrieval

1. INTRODUCTION

As a new generation of probabilistic retrieval models, language modeling approaches [23] to information retrieval (IR) have recently enjoyed much success for many different tasks [30]. In the past decade, many variants of language models have been proposed, mostly focusing on improving the estimation of query language models (e.g., [31, 15]) and document language models (e.g., [17]). Although these language models are motivated in a different way than a traditional model such as the vector-space model, they tend to boil down to retrieval functions that implement retrieval heuristics similar to those implemented in a traditional model, such as TF-IDF weighting and document length normalization [32]. With sound statistical foundation, these language models make it easier to set and optimize retrieval parameters and often outperform traditional retrieval models.

Although much work has been done in language models, there are two related retrieval heuristics remaining "external" to the language modeling approach: (1) proximity heuristic which rewards a document where the matched query terms occur close to each other; (2) passage retrieval which scores a document mainly based on the best matching passage. Existing studies have only attempted to use a standard language model as a black box to implement these heuristics, which means that these heuristics have not really been incorporated into a language model as a component and make it hard to leverage advantages of language models to optimize the combination parameters. For example, proximity heuristic has been studied in [28] where the authors proposed heuristic proximity measures and combined them with the scores of documents computed using standard language models. Also, passage retrieval has been studied in [16] where the authors explored different ways of segmenting text to create passages and then applied standard language models on top of the passages as if they were regular documents. A common deficiency of these studies is that the proximity and passage retrieval heuristics are not modeled from language modeling perspective, making it difficult to optimize the way of combining them with language models.

In this paper, we propose a novel *positional language model* (PLM) which implements both heuristics in a unified language model. The key idea is to define a language model for each *position* of a document (thus the name positional language model), and score a document based on the scores of its PLMs. This is in contrast with virtually all the existing work in which a document language model is generally defined for the *entire* document. An important advantage of introducing a language model for each position is that it can allow us to model the "best-matching position" in a document with probabilistic models, thus supporting "soft" passage retrieval naturally.

The PLM at a position of a document would be estimated based on the propagated word counts from the words at all other positions in the document. Specifically, we let each word at each position of a document to propagate the evidence of its occurrence to all other positions in the document so that positions close to the word would get more share of the evidence than those far away. This way, each position would receive propagated counts of words from all the words in the document with most propagated counts coming from words near the position. We can then estimate a language model for the position based on the propagated counts reaching the position.

A main technical challenge in implementing this idea is how to define the propagation function and estimate the PLM accordingly. We propose and evaluate several different proximity-based density functions for propagation. With some specific choices, we show that the PLM can cover the standard whole document language model and the fixed-window passage language model, as special cases. Since in all these density functions, close-by positions would receive more propagated counts than positions far away from the current word, the PLM also captures the proximity heuristics.

Once we have a language model estimated for each position, we can use one or multiple PLMs of a document as regular document language models to generate a score for the document. We propose and study three general document ranking strategies for combining different PLMs to score documents, including scoring based on the best PLM, combining scores from PLMs at multiple positions, and combining PLMs with different propagation ranges.

Experiment results on several standard test collections show that among all the proximity-based density functions, the Gaussian density kernel performs the best, and that combining PLMs with different propagation ranges is the best document ranking strategy. It is also observed that the proposed PLM not only outperforms the general document language model, but also outperforms the regular sliding-window passage retrieval method and a state-of-the-art proximity-based retrieval model. Overall, the PLM is shown to be able to achieve "soft" passage retrieval and capture proximity heuristic effectively in a unified probabilistic framework.

2. RELATED WORK

Term proximity in information retrieval has been previously studied in [11, 12, 7, 5, 24, 20, 2, 3, 28, 27]. Keen's work [11, 12] is among the earliest efforts, in which, a "NEAR" operator was introduced to address proximity in Boolean retrieval model. The shortest interval containing a match set was first used as a measure of proximity in [5, 7]. Recent work has attempted to heuristically incorporate proximity into an existing retrieval model (often through score combinations) [21, 24, 2, 3, 28]. A variety of proximity measures were proposed, e.g., minimum term span, minimum pair-wise distance, etc.; in [28], the authors systematically examined different measures and concluded that the minimum pair-wise distance is most effective.

An indirect way to capture proximity in the language modeling framework is to use high-order n-grams as units to represent text. For example, in [26], bigram and trigram language models were shown to outperform simple unigram language models. However, n-gram language models cannot capture dependency of non-adjacent terms (we may attempt to capture such proximity by increasing the length of an n-gram, but it is impractical to enumerate all lengths). A more general way to capture proximity through using appropriate "matching units" is Metzler and Croft's work on term dependency [20]. In that work, term structures with different levels of proximity can be defined in a general probabilistic model. Unfortunately, they only attempted to use a standard language model as a black box to implement the proximity heuristic.

Our work differs from the previous studies in two important aspects. First, we propose a new type of language model that incorporates the term proximity evidence in a model-based approach; thus, the existing language modeling techniques (e.g., mixture-model based feedback [31]) can be applied to our model naturally. Second, we capture term proximity directly based on proximity-based term propagation functions.

In passage retrieval [25, 4, 9, 16, 29], documents are often pre-segmented into small passages, which are then taken as units for retrieval. Also documents can be segmented in a more dynamic way defined at query time, referred to as arbitrary passages [9] ("arbitrary" means that a passage can start at any position in a document). Two subclasses are further defined: fixed-length arbitrary passages resemble overlapped windows but with an arbitrary starting point; variable-length arbitrary passages can be of any length. Fixed-length arbitrary passage retrieval was shown to be as effective as, but more efficient than variable-length arbitrary passages [9]. The proposed PLM covers the fixed-length arbitrary passage as a special case, and can be viewed as a "soft" fixed-length passage. However, different from general passage retrieval, which only models term position evidence indirectly using a standard language model as a black box, our model can incorporate term position information directly into the estimation of language models using a proximity-based propagation function.

Proximity-based density functions have been used to propagate term influence in [6, 13, 19, 22]. Kretser's work [6] proposed to propagate the $tf \cdot idf$ score of each query term to other positions, based on several proximity-based kernel functions. The document is scored using the position with the highest accumulative $tf \cdot idf$ score finally. But their methods have not been able to achieve effective retrieval performance. The studies [13, 19] are very similar to [6] and based on the vector space model and Boolean model respectively. In our work, we also evaluate the kernel functions proposed in these studies. In addition, we also propose several other density functions that are more effective than theirs. Compared with this previous work, our work also differs in that we use the language modeling framework, and incorporate such density functions into the estimation of language models.

Similar to our work, Petkova and Croft's work [22] proposed a proximity-based document representation for name entities. Their work emphasizes terms of proximity to entities by using a proximity-based density function, which is then used to build description for entities. Our work, however, proposes a positional language model for general document retrieval, and we evaluate the empirical performance of a number of proximity-based density functions systematically.

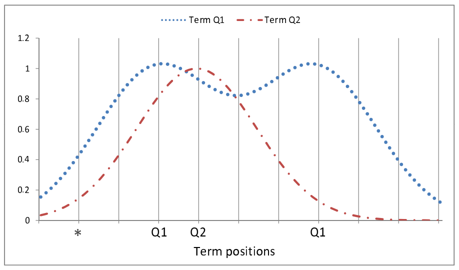

Figure 1: Examples of term propagation.

3. POSITIONAL LANGUAGE MODEL

In this section, we propose a positional language model (PLM) to incorporate term position information into the language model so that we can naturally implement retrieval heuristics such as proximity and passage retrieval.

In most existing work on language models, a document language model is estimated based on only the counts of words in a document, but not the position of words. The main idea of the PLM is to break this limitation and estimate language models based on the position-dependent counts of words. At a high-level, our idea is to define a language model for each word position in a document. This language model is intended to capture the content of the document at the position, which is roughly like a "fuzzy passage" centered at this position but can potentially cover all the words in the document with less weight on words far away from the position.

Specifically, we assume that a term at each position can *propagate* its occurrence at that position to other positions within the same document through a proximity-based density function, as shown in Figure 1. The idea is that if a word w occurs at position i, we would like to pretend that the same word has also occurred at all other positions with a discounted count such that if the position is closer to i, the propagated count for word w at that position would be larger than the propagated count at a position farther away, even though both propagated counts would be less than one, which is the count of w at position i.

The PLM at each position can then be estimated based on all the propagated counts of all the words that to the position as if all the words had appeared actually at the position with discounted counts. Such a PLM intuitively gives a position-specific view of the content of the document, and thus can naturally support passage retrieval. It can also implement the proximity heuristic because of the use of a proximity-based propagation function.

Once we obtain a PLM for each position in a document, we can use each PLM as a regular document language model for matching with a query. We can then score the document by using one or combining multiple PLMs as we will explain later.

We now present PLM more formally. We first introduce the following notations. Let $D = (w_1, ..., w_i, ..., w_j, ..., w_N)$ be a document, where 1, i, j, and N are absolute positions of the corresponding terms in the document, and obviously N is the length of the document.

c(w, i): the count of term w at position i in document D. If w occurs at position i, it is 1, otherwise 0.

k(i, j): the propagated count to position i from a term at position j (i.e., w_j). Intuitively, given w_j, $k(i,j)$ serves as a discounting factor and can be any non-increasing function of $|i-j|$, that is, $k(i,j)$ favors positions close to j. $k(i,j)$ plays an important role in PLMs, and we will analyze and explore a number of proximity-based density functions.

c′(w, i): the total propagated count of term w at position i from the occurrences of w in all the positions. That is, $c'(w,i) = \sum_{j=1}^{N} c(w,j)k(i,j)$. Thus even if $c(w,i)$ is 0, $c'(w,i)$ may be greater than 0. As shown in Figure 1, after propagation, position $'*'$ has a non-zero "count" of terms Q_2 and Q_1.

Based on term propagation, we have a term frequency vector $\langle c'(w_1,i), ..., c'(w_N,i)\rangle$ at position i, forming a *virtual* document D_i. We can see that term position information has been translated to term frequency information stored in this vector. Thus the language model of this virtual document can be estimated as:

$$p(w|D,i) = \frac{c'(w,i)}{\sum_{w' \in V} c'(w',i)} \tag{1}$$

where V is the vocabulary set. We call $p(w|D,i)$ a **Positional Language Model** (PLM) at position i.

Intuitively, we can imagine that the PLMs give us multiple representations of D. Thus given a query Q, we can adopt the KL-divergence retrieval model [14] to score each PLM as follows:

$$S(Q,D,i) = -\sum_{w \in V} p(w|Q) \log \frac{p(w|Q)}{p(w|D,i)} \tag{2}$$

where $p(w|Q)$ is an estimated query language model. We can estimate $p(w|Q)$ with the maximum likelihood estimate or through some pseudo relevance feedback algorithms (e.g., relevance model [15] or mixture model [31]).

Similar to a regular document language model, the PLM also needs to be smoothed to solve the zero probability problem and to penalize common terms [32]. We consider two popular smoothing methods: Dirichlet prior and Jelinek-Mercer. Dirichlet prior smoothing has proven an effective smoothing method for document language models and captures the document length normalization heuristic [32]. For a PLM, the length of the virtual document at position i is $Z_i = \sum_{w \in V} c'(w,i)$. We use the general collection language model $p(w|\mathcal{C})$ as our background model. Thus the smoothed model is given by:

$$p_\mu(w|D,i) = \frac{c'(w,i) + \mu p(w|\mathcal{C})}{Z_i + \mu} \tag{3}$$

where μ is a smoothing parameter. Although previous work has shown that Jelinek-Mercer does not work as well as Dirichlet prior [32], it is unclear whether the same conclusion holds for PLMs because the virtual document length Z_i at different positions are similar to each other [18]. We thus also consider it as an alternative smoothing method, which is given by:

$$p_\lambda(w|D,i) = (1-\lambda)p(w|D,i) + \lambda p(w|\mathcal{C}) \tag{4}$$

where λ is a smoothing parameter.

3.1 Proximity-based count propagation

Clearly, a major technical challenge in PLMs is how to define the propagation function $k(i,j)$. Following some previous work [6, 13, 22], we present here four representative

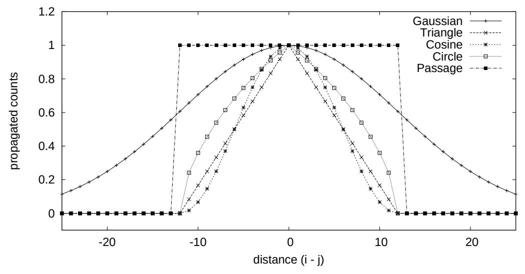

Figure 2: Proximity-based kernel functions. We set $\sigma = 12.5$ for all kernels.

kernel functions: Gaussian, Triangle, Cosine, and Circle, as shown in Figure 2. Different kernels lead to different PLMs.

1. Gaussian kernel

$$k(i,j) = \exp\left[\frac{-(i-j)^2}{2\sigma^2}\right] \tag{5}$$

2. Triangle kernel

$$k(i,j) = \begin{cases} 1 - \frac{|i-j|}{\sigma} & \text{if } |i-j| \le \sigma \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

3. Cosine (Hamming) kernel

$$k(i,j) = \begin{cases} \frac{1}{2}\left[1 + \cos\left(\frac{|i-j|\cdot\pi}{\sigma}\right)\right] & \text{if } |i-j| \le \sigma \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

4. Circle kernel

$$k(i,j) = \begin{cases} \sqrt{1 - \left(\frac{|i-j|}{\sigma}\right)^2} & \text{if } |i-j| \le \sigma \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

All these four kernels have one parameter σ to tune, which controls the spread of kernel curves, i.e., it restricts the propagation scope of each term. In general, the optimal setting of σ for a term may vary according to the term and may also depend on the query because some general terms presumably would have wider semantic scope in a document, thus requiring a higher value of σ, and similarly, some general query might match a longer relevant passage than a more specific query. Our definition of PLMs would in principle allow us to explore such options. However, as a first study of PLMs, in this paper, we simply assume that σ is set to the constant across all the terms and all the queries, leaving further optimization of σ as a future work.

As a baseline, we also present the following *non-proximity-based* Passage kernel:

5. Passage kernel:

$$k(i,j) = \begin{cases} 1 & \text{if } |i-j| \le \sigma \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

With the passage kernel, the PLM can recover the fixed-length arbitrary passage retrieval method [9] in the language modeling framework. We would use this kernel as a baseline to examine whether the proximity-based kernel functions perform better than this non-proximity-based kernel.

According to the proximity-based propagation property, $p(w|D,i)$ is mainly influenced by terms around the position i, all of which form a "soft" passage together. Hence, the PLM captures a "soft passage" naturally in the language modeling framework. Moreover, different from general passage retrieval, which only captures term position evidence indirectly, our model can measure term position information and incorporate it into a language model directly.

Furthermore, if we set σ to a very large or infinite value for any of the proposed kernels, we would have $k(i,j) = 1$ for all i and j. Thus, we have $c'(w,i) = \sum_{j=1}^{N} c(w,j)k(i,j) = c(w,D)$, which means that $p(w|D,i)$ degenerates to the basic whole document language model $p(w|D)$. This shows that the PLM can cover the basic language model as a special case. In general, we can balance the local term proximity evidence and the document level term statistics by tuning the parameter σ (a small σ would emphasize more on local term proximity). Thus, PLM captures term proximity information in the language modeling framework in a natural way.

3.2 PLM-Based Document Ranking

As discussed earlier, with PLMs, we can compute a position-specific score $S(Q,D,i)$ for each position i using the KL-divergence of the PLM at the position and the query language model. Such position-specific scores serve as the basis for computing an overall score for document D. We now discuss several different ways of doing this.

3.2.1 Best Position Strategy

Our first strategy is to simply score a document based on the score of its best matching position, formally,

$$S(Q,D) = \max_{i\in[1,N]} \{S(Q,D,i)\} \tag{10}$$

This strategy resembles most existing studies on passage retrieval, which generally considered evidences from the best matching passage [4, 9, 16].

3.2.2 Multi-Position Strategy

A more flexible alternative strategy is to first compute the scores of top-k positions separately, and then combine these scores together to take advantage of the evidence from several top ranked positions. Particularly, we can take the average of the top-k scores to score a document:

$$S(Q,D) = \frac{1}{k}\sum_{i\in TopK} S(Q,D,i) \tag{11}$$

where $TopK$ is the set of positions corresponding to the top-k highest scores of $S(Q,D,i)$.

3.2.3 Multi-σ Strategy

In this strategy, we compute the best position scores for several different σ values, and then combine these scores together as the final score for a document. The idea is to use different σ values to capture proximity at different propagation ranges.

$$S(Q,D) = \sum_{\sigma\in R} [\beta_\sigma \cdot \max\{S_\sigma(Q,D,i)\}] \tag{12}$$

where R is a predefined set of σ values, $S_\sigma(\cdot)$ is the score function for PLMs with parameter σ, β_σ is the weight on different σ ($\sum_{\sigma\in R}\beta_\sigma = 1$). In particular, if $R = \{\sigma_0, \infty\}$, this strategy equals to an interpolation of the PLM (with a parameter σ_0) and the regular document language model. Considering the efficiency issue, we only evaluate this special case of multi-σ strategy, defined formally as follows:

$$S(Q,D) = \gamma\cdot \max_{i\in[1,N]}\{S_{\sigma_0}(Q,D,i)\} + (1-\gamma)\cdot[-D(\theta_Q||\theta_D)] \tag{13}$$

3.3 Model Implementation

If the PLM is naively implemented, the cost of estimating and ranking PLMs can be extremely high, since the number of positions is much larger than the number of documents or predefined passages. Fortunately, with some mathematical transformation, we may significantly reduce the computational complexity. Below we will show that under reasonable assumptions, PLMs can be implemented similarly to the fixed-length arbitrary passage retrieval.

Given a query, suppose all terms in a document have the same propagation function with the same σ, and the curve of the kernel density function is symmetric. Then we have $k(i,j) = k(j,i)$. Since the most time-consuming part is to compute the normalized length $Z_i = \sum_{w \in V} c'(w,i)$, we rewrite it as:

$$\begin{aligned}\sum_{w \in V} c'(w,i) &= \sum_{w \in V} \sum_{j=1}^{N} c(w,j)k(i,j) = \sum_{j=1}^{N} \left(\sum_{w \in V} c(w,j) \right) k(i,j) \\ &= \sum_{j=1}^{N} k(i,j) = \sum_{j=1}^{N} k(j,i)\end{aligned}$$

This means the sum of propagated count *to* a position is equal to that propagated *from* the position. We show the computation of Z_i for the Gaussian kernel as an example:

$$\begin{aligned}\sum_{j=1}^{N} k(j,i) &= \sum_{j=1}^{N} \left(exp\left[\frac{-(j-i)^2}{2\sigma^2} \right] \right) \\ &\approx \sqrt{2\pi\sigma^2} \cdot \int_1^N \frac{1}{\sqrt{2\pi\sigma^2}} exp\left[\frac{-(x-i)^2}{2\sigma^2} \right] dx \\ &= \sqrt{2\pi\sigma^2} \cdot \left[\Phi\left(\frac{N-i}{\sigma} \right) - \Phi\left(\frac{1-i}{\sigma} \right) \right]\end{aligned}$$

where $\phi(\cdot)$ is the cumulative normal distribution and N is the document length. To calculate $\phi(\cdot)$, we can adopt some existing algorithms, such as the algorithm 26.2.17 [1]. For other kernels considered by us, it is also easy to obtain their cumulative distribution functions through integration.

From this analysis, we can see that our PLM can be implemented similarly to fixed-length arbitrary passage retrieval model. Thus, we can use the techniques proposed in [10] for passage retrieval to implement PLMs; with such an implementation, ranking documents based on PLMs has a complexity of the same order as regular document ranking.

4. EXPERIMENTS

4.1 Testing Collections and Evaluation

We used several standard TREC data sets in our study: AP88-89, FR, TREC8, and WT2G. They represent different sizes and genre of text collections. AP88-89 is chosen as a homogeneous collection. FR is selected as a collection of long documents, with a large variance in the document length. TREC8 is a relatively large heterogeneous collection, while WT2G is Web data. Queries are taken from the title field of the TREC topics [1]. Table 1 shows some basic statistics about these data sets. The preprocessing of documents and queries is minimum, involving only stemming with the Porter stemmer. No stop words have been removed.

In each experiment, we first use the baseline model (KL-divergence) to retrieve 2,000 documents for each query, and then use the PLM (or a baseline method) to re-rank them.

	AP88-89	FR	TREC8	WT2G
queries	51-100	51-100	401-450	401-450
#qry(with qrel)	49	21	50	50
mean(ql)	3.70	4.19	2.46	2.46
#total_qrel	4418	502	4728	2279
#documents	164,597	45,820	528,155	247,491
mean(dl)	462	1498	481	1056

Table 1: Document set characteristic

The top-ranked 1,000 documents for all runs are compared using the mean average precisions (MAP) as the main metric.

4.2 Best Position Strategy

We first examine the effectiveness of the Best Position Strategy for scoring documents based on PLM. Since the performance of this strategy is directly determined by the effectiveness of the kernel function used to estimate the PLM, we first compare the proposed four different proximity-based kernel functions to see which one performs the best. For this comparison, the initial retrieval results were obtained using the KL-divergence retrieval model with Dirichlet prior smoothing; since the *relative* performance of different kernel functions would presumably not be affected by the setting of the smoothing parameter in the initial retrieval, we did not tune the smoothing parameter and simply set it to 1,000. To compare different kernel functions, we follow [9] and systematically test a set of fixed σ values from 25 to 300 in increments of 25. For the sake of efficiency, positions start at 25-word intervals, which was shown by [8] to be an effective way for passage retrieval.

Since we also smooth an estimated PLM when computing retrieval scores, we test both Dirichlet prior smoothing (with parameter 1,000) and Jelinek-Mercer (with parameter 0.5) (see Equations 3 and 4). The results of comparing different kernel functions when using each smoothing method are shown in Table 2 and Table 3, respectively. The best result for each σ value is highlighted. Overall, we see that for all kernels, a relatively large σ value, e.g., 125, 175, and 275, often brings the best performance. It seems that the performance of all runs stabilizes after σ reaches 125. Considering the length of the soft passage is approximately 2σ (as shown in Figure 2), this result confirms the observation in recent studies of passage retrieval [9, 16] that setting passage length to a value around 350 often achieves the best performance. Among all the kernel functions, the Gaussian kernel clearly has the best performance; it contributed 15 best MAP scores out of 20 for Dirichlet prior smoothing and 13 out of 20 for Jelinek-Mercer. To see whether the setting of the smoothing parameter may have affected the relative performance of these kernel functions, we further compare them for a wide range of values of the Dirichlet prior parameter on TREC8 in Figure 3, where we fix $\sigma = 175$ for all kernels. The results clearly show that the Gaussian kernel is the winner among all the four functions. One way to explain why the Gaussian kernel performs the best is that it is the only one of all the functions that exhibits the following property: the propagated count would drop slowly when the distance value $|i-j|$ is small, but drop quickly as the distance value is in a middle range, and then drop slowly again when distance value becomes very large. Such an "S-shape" trend is reasonable for the following reason. Dependent terms are not always adjacent in documents, but can be a little far from each other, thus we would not like to make the propaga-

[1]Topic 110 in AP88-89 was left out accidently due to a format problem in preprocessing.

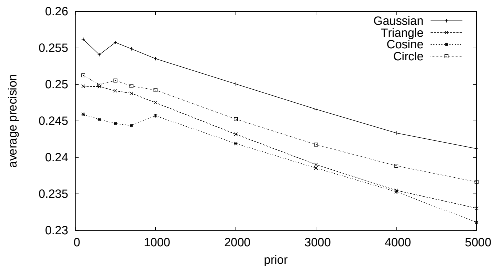

Figure 3: Sensitivity to Dirichlet smoothing parameter of different kernels over TREC8

tion so sensitive to the distance when the distance is small. However, when the distance is just around the *boundary* of strong semantic associations (semantic scope of a term), the propagated count should be more sensitive to the distance change. Then as the distance increases further, all terms are presumably only loosely associated, and thus the propagated term count again should not be so sensitive to the difference of distances.

Since the Gaussian kernel performs the best, in all the following experiments, we use this kernel function. In order to see whether PLM can effectively capture proximity and passage retrieval, we compare the performance of Gaussian kernel ($\sigma = 175$) with the baseline whole document language model using both Dirichlet prior smoothing and Jelinek-Mercer smoothing (both the PLM and the baseline would use the same smoothing method). The results are shown in Figure 4, where we vary the smoothing parameters for both smoothing methods on all the four data sets. We can observe that the PLM improves performance on WT2G and FR clearly and consistently, which shows that, similar to general passage retrieval, the PLM can bring added benefits to document retrieval when documents are relatively long. Some improvements are also found on TREC8, possibly because it is a heterogeneous data set as compared to AP88-89, which is homogeneous; a heterogeneous collection is relatively nosier, thus term dependence information may be more helpful. Unfortunately, the PLM does not seem to show its advantages on AP88-89 for Dirichlet prior smoothing even though it is so for Jelinek-Mercer smoothing.

By comparing the results of the two smoothing methods in Figure 4, we see that in general, Dirichlet prior performs better than Jelinek-Mercer for PLM, and the Dirichlet prior smoothing method seems to perform stably for a range of μ values around 500.

Figure 4 shows that PLM outperforms whole document LM baseline likely due to the use of proximity and passage-retrieval heuristics. We would like to further understand whether PLM, which captures "soft" passages, is also better than the fixed-length arbitrary passage retrieval method. Thus, we compare the PLM (using the Gaussian kernel, $\sigma = 175$, Dirichlet prior smoothing, $\mu = 500$) with the fixed-length arbitrary passage retrieval method [9] (i.e., the Passage kernel). The MAP scores are summarized in Table 4, where the best result for each σ is highlighted. We can observe that the PLM indeed outperforms the standard passage retrieval baseline significantly, which shows that modeling term proximity directly using a proximity-based density function is more effective and robust than assuming fixed lengths.

WT2G					
–	25	75	125	175	275
Psg	0.2962	0.3176	0.3225	0.3265	0.3249
PLM	**0.2989**	**0.3213**	**0.3286**$^+$	**0.3307**$^+$	**0.3285**
TREC8					
Psg	0.2358	0.2433	0.2492	0.2511	0.2518
PLM	**0.2364**	**0.2465**$^+$	**0.2503**	**0.2535**$^+$	**0.2550**$^+$
FR					
Psg	0.2899	**0.2704**	0.2878	**0.2887**	**0.2860**
PLM	**0.2913**	0.2679	**0.2895**	0.2880	0.2846
AP88-89					
Psg	0.1854	0.2054	0.2130	0.2142	0.2154
PLM	**0.1926**$^+$	**0.2112**$^+$	**0.2162**$^+$	**0.2177**$^+$	**0.2198**$^+$

Table 4: Comparison of fixed-length arbitrary passage retrieval (Psg) and PLM. $'+'$ means that improvements over the Psg are statistically significant.

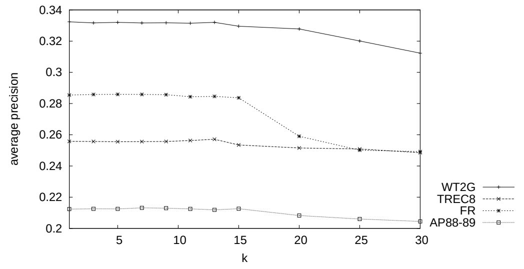

Figure 5: Sensitivity to the parameter k of multi-position strategy

4.3 Multi-position Strategy

We now evaluate the multi-position ranking strategy. Based on the observation in the previous section, we use the Gaussian kernel ($\sigma = 175$) with Dirichlet smoothing ($\mu = 500$) for the PLM. We vary parameter k and plot the MAP results of multi-position strategy in Figure 5. We see that the multi-position strategy does not lead to any noticeable improvement over the best position strategy (i.e., $k = 1$), and if we use a relatively large k, the performance can even degrade dramatically. Hence, given a single σ value, the best position strategy is a robust and reasonable method for document ranking.

4.4 Multi-σ Strategy

We now turn to the evaluation of the multi-σ strategy – in particular, the special case of interpolating a PLM with the whole document language model (i.e., $\sigma = \infty$). To test this special case of Multi-σ strategy, we fix one σ value to ∞, and vary the other one from 25 to 300 in increments of 25. For each σ value, we again use the Gaussian kernel and the Dirichlet prior smoothing method ($\mu = 500$). The results are presented in Table 5, where we tune the interpolation coefficient γ in the range of $[0.0, 1.0]$ to its optimal value for each σ. It shows that, when interpolated with document language models, the PLM performs more robustly and effectively. One possible explanation is that a locally focused PLM alone does not model document-level retrieval heuristics as effectively as the whole document language model does, even though the former captures term proximity heuristic better, thus balancing them will get better results. Another interesting observation is that the best results are always obtained when we use a smaller σ value, e.g. 25 or 75, which also suggests that the PLM is bet-

WT2G					
kernel$\diagdown\sigma$	25	75	125	175	275
Gaussian	**0.2989**	**0.3213**	**0.3286**	**0.3307**	0.3285
Triangle	0.2661	0.3028	0.3149	0.3211	**0.3288**
Cosine	0.2621	0.3007	0.3128	0.3181	0.3243
Circle	0.2797	0.3140	0.3225	0.3273	0.3267
FR					
Gaussian	**0.2913**	0.2679	0.2895	0.2880	0.2846
Triangle	0.2585	0.2898	0.2858	0.2682	**0.2897**
Cosine	0.2603	**0.2910**	**0.3000**	**0.2948**	0.2858
Circle	0.2685	0.2754	0.2673	0.2877	0.2873

TREC8					
kernel$\diagdown\sigma$	25	75	125	175	275
Gaussian	**0.2364**	**0.2465**	**0.2503**	**0.2535**	**0.2550**
Triangle	0.2244	0.2379	0.2438	0.2475	0.2500
Cosine	0.2257	0.2390	0.2430	0.2457	0.2486
Circle	0.2315	0.2401	0.2464	0.2492	0.2523
AP88-89					
Gaussian	**0.1926**	**0.2112**	**0.2162**	**0.2177**	**0.2198**
Triangle	0.1709	0.1987	0.2077	0.2117	0.2173
Cosine	0.1682	0.1969	0.2063	0.2107	0.2144
Circle	0.1801	0.2034	0.2093	0.2135	0.2159

Table 2: MAP Results of different kernel functions with Dirichlet smoothing method.

WT2G					
kernel$\diagdown\sigma$	25	75	125	175	275
Gaussian	**0.3024**	**0.3170**	0.3133	0.3096	0.3010
Triangle	0.2711	0.3057	0.3118	0.3170	0.3131
Cosine	0.2622	0.2855	0.2681	0.2452	0.2039
Circle	0.2813	0.3130	**0.3188**	**0.3179**	**0.3148**
FR					
Gaussian	**0.2639**	0.2606	0.2592	**0.2827**	0.2822
Triangle	0.2458	**0.2681**	0.2607	0.2610	**0.2834**
Cosine	0.2463	0.2476	0.2424	0.2249	0.1593
Circle	0.2512	0.2557	**0.2613**	0.2591	0.2833

TREC8					
kernel$\diagdown\sigma$	25	75	125	175	275
Gaussian	**0.2454**	**0.2510**	**0.2548**	**0.2575**	**0.2576**
Triangle	0.2335	0.2477	0.2491	0.2506	0.2562
Cosine	0.2335	0.2423	0.2356	0.2227	0.2058
Circle	0.2369	0.2456	0.2498	0.2528	0.2555
AP88-89					
Gaussian	**0.1892**	**0.2016**	**0.2054**	**0.2066**	0.2049
Triangle	0.1718	0.1933	0.1968	0.2002	**0.2051**
Cosine	0.1701	0.1910	0.1815	0.1636	0.1349
Circle	0.1735	0.1933	0.1962	0.2010	0.2049

Table 3: MAP Results of different kernel functions with Jelinek-Mercer smoothing method

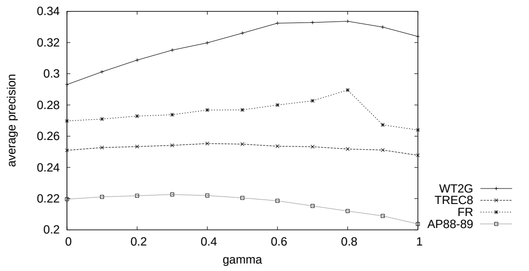

Figure 6: Sensitivity to γ value of multi-σ strategy.

ter at capturing local term proximity evidence rather than document-level evidence (e.g., term frequency).

method$\diagdown$data	WT2G	TREC8	FR	AP88-89
KL	0.2931	0.2509	0.2697	0.2196
$\sigma = 25$	0.3247^+	0.2562^+	**0.2936**	$\mathbf{0.2237}^+$
$\sigma = 75$	$\mathbf{0.3336}^+$	0.2553^+	0.2896^+	0.2227
$\sigma = 125$	0.3330^+	0.2559^+	0.2885	0.2201
$\sigma = 175$	0.3324^+	$\mathbf{0.2574}^+$	0.2858	0.2196
$\sigma = 275$	0.3255^+	0.2561^+	0.2852	0.2193

Table 5: The best performance of multi-σ strategy for different σ. $'+'$ means that improvements over the baseline KL method are statistically significant.

To further look into the sensitivity to γ, we set $R = \{75, \infty\}$ and vary γ on all the four data sets. The results are shown in Figure 6. Interestingly, for collections of long documents (i.e., FR and WT2G), we can rely more on PLMs (larger γ), likely because the whole document language models may contain much noise, but for collections of short-documents (i.e., TREC8 and AP88-89), the sensitivity curves are generally flatter and a relatively smaller γ seems working better, suggesting that regular document language models work reasonably well without term proximity information.

We finally compare our multi-σ strategy ($R = \{75, \infty\}$, $\gamma = 0.8$ for FR and WT2G, and $\gamma = 0.4$ for TREC8 and AP88-89) with a state-of-the-art proximity retrieval method proposed in [28]. Our parameter setting gives PLM near optimal performance. To be fair, we also use the best language modeling retrieval formula suggested in [28] (i.e., R_1 + MinDist), and tune their parameter α to its optimal value to re-rank documents. We label this run as 'R_1 + MinDist' and report the comparison results in Table 6. Interestingly, we see both methods perform similarly on short-document collections (i.e., TREC8 and AP88-89), but our method is clearly better on long-document collections (i.e., WT2G and FR), suggesting that the proposed PLM can capture the passage and proximity heuristics more effectively.

method$\diagdown$data	WT2G	TREC8	FR	AP88-89
R_1 + MinDist	0.3197	**0.2568**	0.2708	0.2220
$R = \{75, \infty\}$	**0.3336**	0.2553	**0.2896**	**0.2227**

Table 6: MAP Comparison of the multi-σ strategy and the best method proposed by Tao and Zhai.

5. CONCLUSIONS

In this paper, we proposed a novel positional language model which implements both proximity heuristic and passage retrieval in a unified language model. We proposed and studied four different proximity-based density functions to estimate PLMs. Experiment results show that the Gaussian density kernel performs the best, followed by Circle, Triangle, and Cosine. As for the smoothing of PLM, the Dirichlet smoothing method performs better than Jelinek-Mercer smoothing.

In addition, we further proposed three PLM-based document ranking strategies. We evaluated their performance and found that the multi-σ strategy performs the best. Our experiments on several standard test collections show that the proposed PLM not only outperforms the regular document language models, but also outperforms the fixed-length arbitrary passage retrieval method and a state-of-the-art proximity-based retrieval model.

As a new family of language models, the PLM opens up

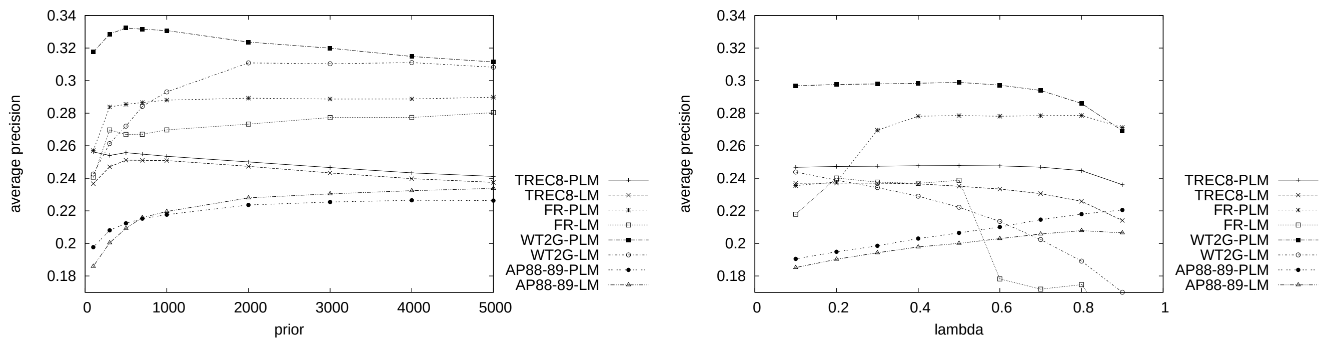

Figure 4: Sensitivity to the smoothing parameters of Dirichlet smoothing (left) and Jelinek-Mercer smoothing (right) of basic language modeling methods (LM) and the PLM on four collections.

many interesting future research directions. One of the most interesting directions is to further study whether setting a term-specific and/or query-specific σ can further improve performance. Another interesting direction is to study how to optimize σ automatically based on statistics such as IDF of terms and discourse structures of documents.

6. ACKNOWLEDGMENTS

We thank the anonymous SIGIR'09 reviewers for their useful comments. This material is based upon work supported by the National Science Foundation under Grant Numbers IIS-0347933, IIS-0713581, and IIS-0713571.

7. REFERENCES

[1] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover, New York, 1964.

[2] Stefan Buttcher and Charles L. A. Clarke. Efficiency vs. effectiveness in terabyte-scale information retrieval. In *Proceedings of TREC '05*, 2005.

[3] Stefan Buttcher, Charles L. A. Clarke, and Brad Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *Proceedings of SIGIR '06*, pages 621–622, 2006.

[4] James P. Callan. Passage-level evidence in document retrieval. In *Proceedings of SIGIR '94*, pages 302–310, 1994.

[5] Charles L. A. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. Shortest substring ranking (multitext experiments for trec-4). In *Proceedings of TREC '95*, pages 295–304, 1995.

[6] Owen de Kretser and Alistair Moffat. Effective document presentation with a locality-based similarity heuristic. In *Proceedings of SIGIR '99*, pages 113–120, 1999.

[7] David Hawking and Paul B. Thistlewaite. Proximity operators - so near and yet so far. In *Proceedings of TREC '95*, pages 500–236, 1995.

[8] Marcin Kaszkiel and Justin Zobel. Passage retrieval revisited. In *Proceedings of SIGIR '97*, pages 178–185, 1997.

[9] Marcin Kaszkiel and Justin Zobel. Effective ranking with arbitrary passages. *Journal of the American Society for Information Science and Technology*, 52(4):344–364, 2001.

[10] Marcin Kaszkiel, Justin Zobel, and Ron Sacks-Davis. Efficient passage ranking for document databases. *ACM Transactions on Information Systems*, 17(4):406–439, 1999.

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

[12] E. Michael Keen. Some aspects of proximity searching in text retrieval systems. *Journal of Information Science*, 18(2):89–98, 1992.

[13] Koichi Kise, Markus Junker, Andreas Dengel, and Keinosuke Matsumoto. *Passage Retrieval Based on Density Distributions of Terms and Its Applications to Document Retrieval and Question Answering*, volume 2956 of *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg, 2004.

[14] John D. Lafferty and Chengxiang Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR '01*, pages 111–119, 2001.

[15] Victor Lavrenko and W. Bruce Croft. Relevance-based language models. In *Proceedings of SIGIR '01*, pages 120–127, 2001.

[16] Xiaoyong Liu and W. Bruce Croft. Passage retrieval based on language models. In *Proceedings of CIKM '02*, pages 375–382, 2002.

[17] Xiaoyong Liu and W. Bruce Croft. Cluster-based retrieval using language models. In *Proceedings of SIGIR '04*, pages 186–193, 2004.

[18] David E. Losada and Leif Azzopardi. An analysis on document length retrieval trends in language modeling smoothing. *Information Retrieval*, 11(2):109-138, 2008.

[19] Annabelle Mercier and Michel Beigbeder. Fuzzy proximity ranking with boolean queries. In *Proceedings of TREC '05*, 2005.

[20] Donald Metzler and W. Bruce Croft. A markov random field model for term dependencies. In *Proceedings of SIGIR '05*, pages 472–479, 2005.

[21] Christof Monz. Minimal span weighting retrieval for question answering. In Rob Gaizauskas, Mark Greenwood, and Mark Hepple, editors, *SIGIR Workshop on Information Retrieval for Question Answering*, pages 23–30, 2004.

[22] Desislava Petkova and W. Bruce Croft. Proximity-based document representation for named entity retrieval. In *Proceedings of CIKM '07*, pages 731–740, 2007.

[23] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of SIGIR '98*, pages 275–281, 1998.

[24] Yves Rasolofo and Jacques Savoy. Term proximity scoring for keyword-based retrieval systems. In *Proceedings of ECIR '03*, pages 207–218, 2003.

[25] Gerard Salton, J. Allan, and Chris Buckley. Approaches to passage retrieval in full text information systems. In *Proceedings of SIGIR '93*, pages 49–58, 1993.

[26] Fei Song and W. Bruce Croft. A general language model for information retrieval. In *Proceedings of CIKM '99*, pages 316–321, 1999.

[27] Ruihua Song, Ji-Rong Wen, and Wei-Ying Ma. Viewing term proximity from a different perspective. In *Proceedings of ECIR '08*, 2008.

[28] Tao Tao and ChengXiang Zhai. An exploration of proximity measures in information retrieval. In *Proceedings of SIGIR '07*, pages 295–302, 2007.

[29] Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of SIGIR '03*, pages 41–47, 2003.

[30] ChengXiang Zhai. Statistical language models for information retrieval a critical review. *Found. Trends Inf. Retr.*, 2(3):137–213, 2008.

[31] ChengXiang Zhai and John D. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of CIKM '01*, pages 403–410, 2001.

[32] ChengXiang Zhai and John D. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR '01*, pages 334–342, 2001.