Unsupervised Query Segmentation Using Clickthrough for Information Retrieval

Yanen Li* [1], Bo-June (Paul) Hsu[2], ChengXiang Zhai[1], Kuansan Wang[2]

[1]Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801
{yanenli2, czhai}@illinois.edu

[2]Microsoft Research, One Microsoft Way Redmond, WA 98052
{paulhsu, kuansan.wang}@microsoft.com

ABSTRACT

Query segmentation is an important task toward understanding queries accurately, which is essential for improving search results. Existing segmentation models either use labeled data to predict the segmentation boundaries, for which the training data is expensive to collect, or employ unsupervised strategy based on a large text corpus, which might be inaccurate because of the lack of relevant information. In this paper, we propose a probabilistic model to exploit clickthrough data for query segmentation, where the model parameters are estimated via an efficient EM algorithm. We further study how to properly interpret the segmentation results and utilize them to improve retrieval accuracy. Specifically, we propose an integrated language model based on the standard bigram language model to exploit the probabilistic structure obtained through query segmentation. Experiment results on two datasets show that our segmentation model outperforms existing segmentation models. Furthermore, extensive experiments on a large retrieval dataset reveals that the results of query segmentation can be leveraged to improve retrieval relevance by using the proposed integrated language model.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query Segmentation and Language Modeling*

General Terms

Algorithms, Performance, Experimentation

*Work done as a summer intern at Microsoft Research

SIGIR'11, July 24–28, 2011, Beijing, China.

Keywords

Query Segmentation, Expectation Maximization Algorithm, Language Modeling, QSLM

1. INTRODUCTION

Since an accurate understanding of the user's query is critical to improving retrieval results, query segmentation is one of the most important tasks in modern information retrieval. For example, accurate query segmentation is the prerequisite for semantic retrieval models, phrase-based query reformulation and automatic relevance feedback. Supervised techniques have been used to solve the query segmentation problem in the past [3, 23]. However, they require lots of segmentation labels which are expensive to collect. An unsupervised approach based on a large text corpus and Wikipedia has been reported to achieve competitive performance [19]; but its accuracy without Wikipedia is still low, partly due to the lack of relevant information about the query structure.

In a modern search engine, there is a large amount of relevant data in the form of clickthroughs. Such data reflects users' implicit preference of documents, and can be leveraged to infer the underlying segmentation of the queries. In this paper, we propose a unsupervised probabilistic model to exploit user clickthroughs for query segmentation. Model parameters are estimated by an efficient EM algorithm. Segmentation results on a standard dataset demonstrate that our model significantly outperforms the EM model in [19] without the use of Wikipedia. Additionally, by combining more data from external resources, such as the Microsoft Web N-gram [22], our model can outperform state-of-the-art baselines.

One of the most important applications for query segmentation is to improve the retrieval models by incorporating query segmentation. Most information retrieval techniques, such as vector space models and language modeling approaches, rely on the bag-of-words assumption that every query term is independent in the relevance computation. But this assumption is over simplified; users have an order in mind when formulating queries to search for information. One of the reasons why bag-of-words based methods remain popular is because data sparsity makes it harder to estimate models imposing term dependencies [4, 5, 18]. Successful query segmentation has a great potential to lead to better

retrieval models that can utilize higher-order term dependencies.

However, query segmentation is ambiguous in nature – the same query can be segmented in difference ways by different people. Although several methods for query segmentation have been proposed, surprisingly little research has been performed to address the segmentation ambiguity and incorporate this information into retrieval models. In this paper, we propose a query segmentation model that quantifies the uncertainty in segmentation by probabilistically modeling the query and clicked document pairs. We further incorporate the probabilistic query segmentation into a unified language model for information retrieval. Experiments on a large web search dataset from a major commercial search engine show that the integrated language model with query segmentation (QSLM) outperforms both the BM25 model and other language models.

2. RELATED WORKS

Query segmentation models have been well studied in recent literature [10, 11, 3, 23, 19, 6]. Initially, the mutual information (MI) between adjacent words in a query is employed to segment queries with a cutoff [10, 11].The major drawback of MI based methods is that they are unable to detect multi-word or phrase based dependencies. Compared with MI based models, supervised query segmentation approaches can achieve higher accuracies [3, 23]. For example, by considering every boundary between two consecutive query words as a binary decision variable, Bersgma and Wang [3] trains the weights of a linear decision function with a set of syntactic and shallow semantic features extracted from the labeled data. However, its focus on noun phrase features may not be appropriate for the segmentation of web queries. Furthermore, acquiring training labels demands a great deal of manual effort that may not scale to the web. As another supervised learning approach, Yu and Shi [23] applies conditional random fields to obtain good query segmentation performance. However, it relies on field information features specific to databases, not available for general unstructured web queries. Moreover, the evaluation was conducted only on synthetic data, which is less desirable than real query data.

Tan and Peng [19] introduce a generative model in the unsupervised setting by adopting n-gram frequency counts from a large text corpus and computing the segment scores via expectation maximization (EM). It also utilizes Wikipedia as another term in the minimum description length objective function. Similarly our model in this paper also includes n-gram statistics and applies the EM algorithm to estimate the model parameters. However, we employ clickthrough query-document pairs to improve segmentation accuracy and further refine the retrieval model by utilizing probabilistic query segmentation. Similar probabilistic model is also proposed in [24], but this model focuses in parsing noun phrases thus not generally applicable to web queries.

This work is also closely related to the retrieval models that capture higher order dependencies of query terms. There are several research attempts to incorporate term dependency in query or document to retrieval models [12]. For example, some attempts have been made to add proximity heuristics to the vector space model or generative query LM model [13, 20]. However these methods rely on heuristics, which is not a principled way of incorporating term dependency. More unified higher-order language models have been studied by Srikanth *et al.* (Biterm LM) [18]. However, their assumption that every position is dependent is too strong. In fact, the word dependency is stronger within a semantic unit than across the unit, which is what we assume in our work. LM with query syntactics [5] assumes a structure on the query, but they are too complex to estimate accurately. More important, the query syntactic models usually take only the top (most likely) query structure in the modeling process. However, it is more appropriate to assess the probability for all possible segmentation if multiple structures have comparable probabilities to represent the query.

In addition, search log and clickthrough data have been reported to be able to improve the performance of personalized search [9, 1, 17]. For example, Joachims [9] first proposes to improve the retrieval quality of search engines by learning from the user clickthrough. Shen *et al.* [17] propose a decision-theoretic approach to improve search performance via user feedback. And Agichtein *et al.* [1] demonstrate that web search ranking can be improved by considering user behavior. We add to this line of work a novel way of exploiting the clickthrough data for query segmentation.

3. PROBLEM SETUP

The task of query segmentation is to separate the query words into disjoint segments so that each segment maps to a semantic unit. Given a query $Q = w_1, w_2, ..., w_n$ of length n, a segmentation $S = S_1 S_2 ... S_M$ of length M is consistent with the query Q if $S_m = w_{b_m} w_{b_m+1} ... w_{b_{m+1}-1}$ for $1 = b_1 < b_2 < ... < b_{M+1} = n + 1$. We define $B = b_1, b_2, ..., b_{M+1}$ as the segmentation partition, independent of the actual query words, and $\mathbb{B}_n$ as the set of all possible segmentation partitions consistent with any query of length n. There are a total of 2^{n-1} segmentation partitions in $\mathbb{B}_n$. Note that given a query Q, the segmentation partition B and the query segments S can be uniquely derived from each other.

Because query segmentation is potentially ambiguous, we are interested in assessing the probability of a query segmentation under some probability distribution: $P(S|\theta)$. With such a probabilistic model, we can then select those segmentations with high probabilities and use them to construct models for information retrieval. For example, for the query "*bank of america online banking*", {[*bank of america*] [*online banking*], 0.502}, {[*bank of america online banking*], 0.428} and {[*bank of*] [*america online*] [*banking*], 0.001} are all valid segmentations, where brackets [] are used to indicate segment boundaries and the number at the end is the probability of that particular segmentation. In this example, the first two segmentations are likely segmentations with high probabilities, whereas the last one is a rare segmentation, as reflected by the low probability. In the next section, we discuss how to compute the probability $P(S|Q)$ of a segmentation S given a query Q.

4. QUERY SEGMENTATION

The search log in a modern search engine usually contains a lot of user clickthrough data, where user-issued queries and

corresponding clicked documents are recorded. This kind of data contains rich information about users' preferences for each query. By carefully modeling the clickthroughs, we can assess the likelihood of a segmentation structure according to the collective user behavior. Table 1 shows examples of the clicked documents for two real-world queries from the search log. In these examples, although there are variations in the query words and documents, the sub-sequence "bank of america" remains intact in all clicked documents. The evidence strongly suggests that "*bank of america*" should be a segment. This observation motivates us to model the query segmentation using the query and clicked document pairs, a previously unexplored idea.

Table 1: Examples of Query and Clicked Documents

Query	Clicked document title
bank of america invesment	1. bank of america associate banking investments homepage 2. bank of america investment services inc investments overview 3. bank of america associate banking investments banking services ...
credit card bank of america	1. bank of america credit cards contact us overview 2. secured visa credit card from bank of america 3. credit cards overview find the right bank of america credit card for you ...

We now propose an unsupervised query segmentation model using user clickthroughs. We first describe the model for generating queries and will later extend it to query-click document pairs. The process of generating a query can be described as follows:

1. Pick a query length n under a length distribution.
2. Select a segmentation partition $B \in \mathbb{B}_n$, according to a segmentation partition model $P(B|n,\psi)$.
3. Generate query segments S_m consistent with B, according to a segment unigram model $P(S_m|\theta)$.

Recall that given a query Q of length n, the query segments S and the segmentation partition B can be derived from each other. Thus, we can compute the probability of a segmentation as:

$$P(S|Q,\theta,\psi) = P(B|Q,n,\theta,\psi) = \frac{P(Q|B,\theta)\cdot P(B|n,\psi)}{\sum_{B'\in\mathbb{B}_n} P(Q|B',\theta)\cdot P(B'|n,\psi)}, \quad (1)$$

$$P(Q|B,\theta) = \prod_{m=1}^{M} P(S_m|\theta) \quad (2)$$

where $P(Q|B,\theta)$ is the probability of generating a query Q given segmentation partition B. The $P(B|n)$ can be estimated by an expectation maximization algorithm described in the following section. However, in this paper we set $P(B|n)$ to a particular form by imposing an infinite strong prior that penalizes longer segments:

$$P(B|n,\psi) = \frac{\prod_{m=1}^{M} P(|S_m(B)|\ |\psi)}{\sum_{B'\in\mathbb{B}_n}\prod_{m'=1}^{M(B')} P(|S_{m'}(B')|\ |\psi)} \quad (3)$$

$$P(|S_m(B)|\ |\psi) = e^{-|S_m(B)|^f} \quad (4)$$

where $|S_m(B)|$ is the length of the m^{th} segment specified by B, and f is a factor controlling the segment length penalty. Note that the denominator is constant for a fixed length n. Since the probability of a segmentation is the product of all segment probabilities $P(S_m|\theta)$ and $P(B|n,\phi)$, such a segment length penalty is crucial to counter the bias for longer segments as they result in fewer segments and hence fewer terms in the final product. This need for segment length penalty is also discussed by Peng *et. al* in [14].

To extend the model to observed pairs of the query Q and the clicked document D, we consider Q to be generated from an interpolated model, consisting of the global component $P(S_m|\theta)$ and a document-specific component $P(S_m|\theta_D)$. Specifically, we redefine the query probability given a segmentation partition in Equation (2) as:

$$P(Q|B,\theta,\theta_D) \propto \prod_{m=1}^{M} P(S_m|\theta)P(S_m|\theta_D) \quad (5)$$

Mathematically, this is equivalent to generating each query segment using a log-linear interpolation of the global and document-specific models. Figure 1 illustrates the segmentation partition and the process of generating a query given the model.

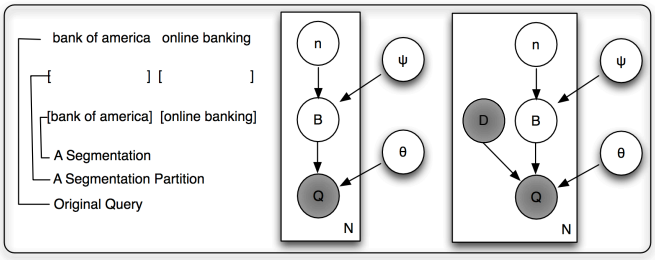

Figure 1: The Generative Model of Segmentation. Left: the query segmentation partition; middle: the process of generating a query Q; right: the process of generating query Q with clicked document D

For $P(S_m|\theta_D)$, we employ a smoothed bigram language model trained from the document D and interpolated with the global document collection statistics θ_C to model the probability of $S_m = w_{b_m} w_{b_m+1} ... w_{b_{m+1}-1}$:

$$P(S_m|\theta_D) = \prod_{l=b_m}^{b_{m+1}-1} P(w_l|w_{l-1},\theta_D) = \prod_{l=b_m}^{b_{m+1}-1} [(1-\lambda)P_{bi}(w_l|w_{l-1},\theta_D) + \lambda P_{bi}(w_l|w_{l-1},\theta_C)] \quad (6)$$

where

$$P_{bi}(w_l|w_{l-1},\theta_D) = \frac{f^D_{w_{l-1}w_l} + \mu_D \frac{f^D_{w_l}}{|D|}}{f^D_{w_{l-1}} + \mu_D},$$

$$P_{bi}(w_l|w_{l-1},\theta_C) = \frac{f^C_{w_{l-1}w_l} + \mu_C \frac{f^C_{w_l}}{|C|}}{f^C_{w_{l-1}} + \mu_C},$$

λ is the mixture weight, μ_C and μ_D are the bigram backoff weights, and f_{w_l}, $f_{w_{l-1}w_l}$ are the n-gram counts in document D or corpus C.

Overall, we want to estimate $\hat{\theta}$ to maximize the log likelihood of observing all the query-clicked document pairs in the dataset:

$$\log P(\mathbb{Q}|\theta,\theta_D) = \sum_l log \sum_{B\in\mathbb{B}_{n_l}} P(B|n_l) \cdot \prod_{m=1}^{M_l} [P(S_m(Q_l)|\theta) \cdot P(S_m(Q_l)|\theta_{D_l})] \quad (7)$$

With $\hat{\theta}$, we can compute the most probable segmentations for any query according to Equation (1).

4.1 Model Parameter Estimation by EM

Since the joint probability in Equation (7) involves the logarithm of a summation over hidden variables B, there is no exact analytical solution for $\hat{\theta}$. However, we can apply the expectation maximization (EM) algorithm to maximize the joint probability of all observed data. In the E-step, we evaluate the posterior probability of a valid segmentation of Q given the previous model parameter estimate $\theta^{(k-1)}$:

$$P(B|Q,\theta^{(k-1)},\theta_D,\psi) = \frac{P(Q|B,\theta^{(k-1)},\theta_D,\psi)\cdot P(B|n,\psi)}{\sum_{B'\in\mathbb{B}_n} P(Q|B',\theta^{(k-1)},\theta_D,\psi)\cdot P(B'|n,\psi)} \quad (8)$$

where

$$P(Q|B,\theta^{(k-1)},\theta_D,\psi) = \prod_{m=1}^{M} [P(S_m|\theta^{(k-1)})\cdot P(S_m|\theta_D)]$$

In the M-step, we update the estimate of θ according to:

$$P(w_1...w_r|\theta^{(k)}) = \frac{1}{Z}\cdot \sum_l \sum_{B\in\mathbb{B}_{n_l}} [P(B|Q_l,\theta^{(k-1)},\theta_D,\psi)\cdot \sum_{m=1}^{M_l} \delta(S_m(B,Q_l)=w_1...w_r)] \quad (9)$$

where Z is the normalization factor and $\delta()$ is an indicator function checking if segment S_m is equal to n-gram $w_1...w_r$. For a query of n keywords, a naive computation for the M-step requires summing of all 2^{n-1} possible segmentations, which is computationally impractical for longer queries. Fortunately, it can be computed efficiently using the Baum-Welch algorithm [2].

Here we introduce a graph representation for query segmentation. Given a query Q of length n, all segmentations consistent with Q can be represented by a graph G with $n+1$ nodes. Figure 2 illustrates a graph representation for two valid segmentations of the query "*bank of america online banking*". For a graph with $n+1$ nodes, there are a total of 2^{n-1} ways to connect node 1 to node $n+1$, each corresponding to a valid segmentation. For example, in the upper panel of Figure 2, there is a connection from node 1 to 4, corresponding to a segmentation boundary between *america* and *online*. In this case, the arc from node 1 to 4 corresponds to the segment "bank of america".

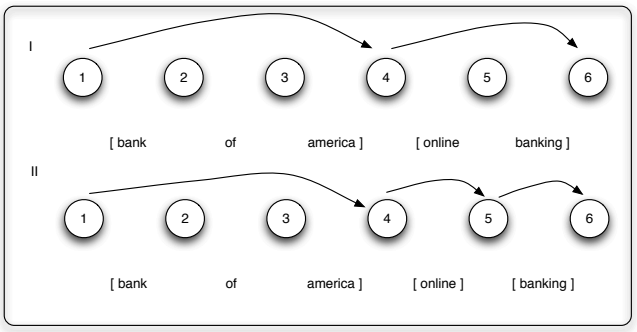

Figure 2: Graph Representation of Segmentations

Using this graph representation, Equation (9) in the M-step can be rewritten as:

$$P(w_1...w_r|\theta^{(k)}) = \frac{\sum_l \sum_i \sum_j \xi_l(i,j)\cdot\delta_l(S_{i\to j}=w_1...w_r)}{\sum_l \sum_i \sum_j \xi_l(i,j)} \quad (10)$$

where $\xi(i,j) = P(S_{i\to j}|Q,\theta^{(k-1)})$ is the probability of the segment S_m represented by the arc from node i to node j for the query Q, $\delta(S_{i\to j}=w_1...w_r)$ is an indicator function with a value of 1 when $S_{i\to j}=w_1...w_r$ and 0 otherwise.

In order to compute $\xi_l(i,j)$, we introduce $\alpha(i) = P(Q,i|\theta^{(t-1)})$, the probability of observing the query Q from the beginning of the graph to node i, and $\beta(j) = P(Q|j,\theta^{(t-1)})$, the probability of observing Q from node j to the end of the graph:

$$\alpha(i) = \sum_{k:k<i} \alpha(k)\cdot P(S_{k\to i}|\theta^{(t-1)})\cdot e^{-|S_{k\to i}|^f}\cdot P_D(S_{k\to i}),$$

$$\beta(j) = \sum_{k:k>j} \beta(k)\cdot P(S_{j\to k}|\theta^{(t-1)})\cdot e^{-|S_{j\to k}|^f}\cdot P_D(S_{j\to k}),$$

$$\xi_l(i,j) = \alpha_l(i)\cdot\beta_l(j)\cdot P(S_{i\to j}|\theta^{(t-1)})\cdot e^{-|S_{i\to j}|^f}\cdot P_{D_l}(S_{i\to j})$$

with the initial condition $\alpha_l(1)=1, \beta_l(n)=1$. Algorithm 1 summarizes the steps for estimating θ. For a set of queries with equal length, the computation complexity for each iteration is $O(Ln^2)$, where L is the number of input query-document pairs and n is the number of words in each query. Once the optimal θ is obtained, the probability of a segmentation $P(S|Q,\theta,\psi)$ can be computed by Equation (1).

4.2 Utilizing Other Resources

N-gram statistics from a very large scale of text resources can also be utilized to improve query segmentation. In fact in [19], the biggest improvement in segmentation accuracy is achieved by utilizing information from Wikipedia. In addition, [6] also reports a well-performing naive query segmentation method using Google Web N-gram. Here we propose a simple approach utilizing the Microsoft Web N-gram service. MS Web N-gram is essentially a distribution of n-gram probability θ' over the web. The probability of a segmenta-

Algorithm 1: N-gram concept probability estimation

input : A set of query-clicked document pairs $\mathbb{O} = \{< Q_l, D_l >\}, l \in [1, N]$

output: Optimal estimate of $\theta = \{P(S_m)\}$

Init $P(S_m|\theta^{(0)}) \leftarrow \frac{Count(S_m)}{Count(total\ Ngrams\ in\ query\ collection)}$;

for $t \leftarrow 1$ **to** T **do**

  $P(S_m|\theta^{(t)}) \leftarrow 0$;

  $\xi_{total} \leftarrow 0$;

  for $l \leftarrow 1$ **to** N **do**

    G_l is a graph representing query Q_l, with $n+1$ nodes; $\alpha_l(1) = 1$; $\beta_l(n+1) = 1$;

    for *node* $i \leftarrow 2$ **to** $n+1$ **do**

      $\alpha_l(i) \leftarrow \sum_{k:k<i} \alpha_l(k) \cdot P(S_m|\theta^{(t-1)}) \cdot e^{-|(S_{k\rightarrow i})|^f} \cdot P_{D_l}(S_{k\rightarrow i})$;

    for *node* $j \leftarrow n$ **to** 1 **do**

      $\beta_l(j) \leftarrow \sum_{k:k>j} \beta_l(k) \cdot P(S_{j\rightarrow k}|\theta^{(t-1)}) \cdot e^{-|S_{j\rightarrow k}|^f} \cdot P_{D_l}(S_{j\rightarrow k})$;

    for *node* $i \leftarrow 1$ **to** $n+1$ **do**

      for *node* $j \leftarrow i+1$ **to** $n+1$ **do**

        $\xi_l(i,j) \leftarrow \alpha_l(i) \cdot \beta_l(j) \cdot P(S_{i\rightarrow j}|\theta^{(t-1)}) \cdot e^{-|S_{i\rightarrow j}|^f} \cdot P_{D_l}(S_{i\rightarrow j})$;

        $P(S_m = S_{i\rightarrow j}|\theta^{(t)}) \leftarrow P(S_m = S_{i\rightarrow j}|\theta^{(t)}) + \xi_l(i,j)$;

        $\xi_{total} \leftarrow \xi_{total} + \xi_l(i,j)$;

  $P(S_m|\theta^{(t)}) \leftarrow \frac{P(S_m|\theta^{(t)})}{\xi_{total}}$;

return $\theta = \{P(S_m)\}$;

tion given Q is defined as:

$$P(B|Q,\theta',\psi') \propto P(Q|B,\theta',\psi') \cdot P(B|\psi') \quad (11)$$
$$= \prod_{m=1}^{M} P(S_m|\theta') \cdot P(B|\psi')$$
$$\propto \prod_{m=1}^{M} P(S_m|\theta') \cdot e^{-|S_m(B)|^{f'}}$$

Furthermore, we can combine our query segmentation model with clickthrough and the simple model with Web N-gram into an interpolated model:

$$logP(B|Q,\theta,\theta',\psi,\psi') = (1-\omega) \cdot logP(B|Q,\theta,\psi) + \omega \cdot logP(B|Q,\theta',\psi') \quad (12)$$

we find the setting of $\omega = 0.5, f = 2.0, f' = 2.0$ results in a model with good segmentation accuracy.

5. SEGMENTATION EXPERIMENTS

In this section we report the query segmentation results obtained by our model and other baselines on two datasets. One is from a standard dataset established by previous research, and the other is constructed by ourselves. We also conduct extensive analysis on several aspects of the results.

5.1 Data Preparation and Evaluation Metrics

We use two sets of queries for evaluating the query segmentation models. The first set (Set 1) is a standard query segmentation dataset established by Bergsma and Wang [3], which is also applied in [19]. In this dataset, annotator A, B, and C independently segmented 500 queries which are sampled from the AOL 2006 query log. Among these 500 human queries, the 220 where the 3 judges agree are called the "Intersection" set.

The above segmentation dataset is focused on noun queries. But in this work we are also interested in web queries. Therefore we prepare another set of 1,000 queries sampled from the search log of a major commercial search engine, which we name the 1000-query dataset. We invite three domain experts to segment the queries independently, employing the same evaluation metrics as Set 1. Besides expert annotations, this dataset also has clickthrough information and relevance judgments for the top documents, which is used by subsequent experiments when comparing retrieval models.

To measure the segmentation effectiveness, we report results on three evaluation metrics. (1) Query accuracy: the percentage of queries for which the predicted segmentation matches the gold standard completely; (2) Classification accuracy: the ratio of correctly predicted boundaries in between every two consecutive words; (3) Segment accuracy: how well the predicted segments match the gold standard under the information retrieval measures of precision, recall, and F-score.

As baseline, we include the three models in [19]: Mutual information, EM + corpus (query log), and EM + corpus + Wikipedia. We also include a method using Google Web N-gram [6] and a simple model with MS Web N-gram, as defined in Section 4.2. Our model + clickthrough and our model + clickthrough + MS Web N-gram are included in the comparison. The parameters of our segmentation model is trained on a large set of search log containing about 20 millions query-clicked document pairs.

5.2 Query Segmentation Results

Table 2 shows the results of our model as well as the baseline models on the standard dataset. Columns 3 to 5 represent models without using external data source (basic models), while columns 6 to 9 are models utilizing large external sources, such as Wikipedia and web-scale n-gram (extended models). Among the basic models, our model performs the best according to annotator A, C and the intersection of these annotators. These results are significantly better than the corresponding results by the EM + corpus model in [19]. For the result based on annotator B, our model is comparable to that of [19] (0.571 vs 0.573 on segment F score). For the extended models, simple model + MS Web N-gram performs well, similar to the results for simple model with Google Web N-gram as reported in [6]. It indicates the positive impact of n-gram statistics on query segmentation. However, our model, as well as EM model + Wikipedia in [19] outperforms the simple models consistently in all annotators' judgments; and our extended model performs better than that of [19]. For example, in the intersection judgments, the F score of our model is 0.779, while model in [19] is 0.774. Compared to the simple model + MS Web N-gram, whose intersection F score is 0.728, our model achieves a 7.0% gain on the same measure. It suggests the

Table 3: Results on the 1000-query Dataset

Annotator	Measure	Our Model	Simple Model + MS Web N-gram
A	query accuracy	0.386	0.316
	classify accuracy	0.631	0.538
	segment precision	0.434	0.368
	segment recall	0.540	0.552
	segment F	**0.481**	0.441
B	query accuracy	0.447	0.403
	classify accuracy	0.690	0.619
	segment precision	0.533	0.476
	segment recall	0.602	0.648
	segment F	**0.565**	0.549
C	query accuracy	0.472	0.545
	classify accuracy	0.703	0.749
	segment precision	0.670	0.693
	segment recall	0.582	0.730
	segment F	0.623	**0.713**
Intersection	query accuracy	0.624	0.567
	classify accuracy	0.761	0.642
	segment precision	0.372	0.301
	segment recall	0.405	0.395
	segment F	**0.388**	0.342

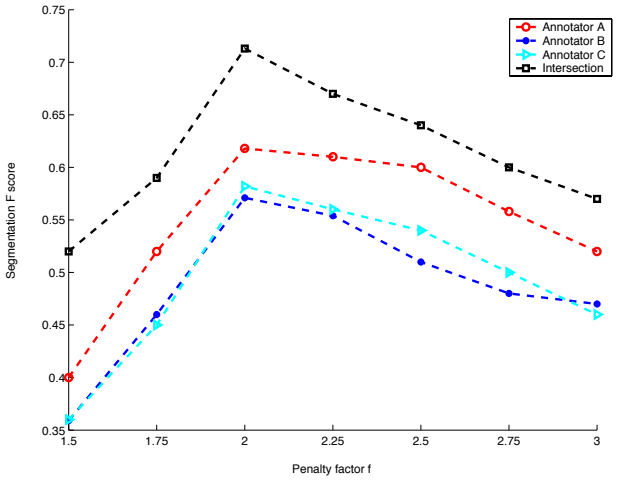

Figure 3: Query Segmentation Performance with Respect to Penalty Factor

effectiveness of our model and the benefit from combining additional large scale N-gram statistics.

5.3 Results on the 1000-query Dataset

We compare our query segmentation model with the simple model + MS Web N-gram on the 1000-query dataset. Table 3 shows the segmentation results on the this set. Although the simple segmentation model with web n-gram works very well in the standard dataset, it performs inferior to our model in the 1000-query dataset. In 2 out of 3 annotator judgments, our model outperforms the simple model. And in the intersection judgments our model also works better than the simple model by 4.6%. Since this dataset is sampled from a set of web search queries, results in this experiment indicate that our model fits web search queries, whose characteristics are different from noun queries, better.

5.4 Effect of the Penalty Factor

The factor f in Equation (4), which controls how much penalty is given to a segment of length $|S_m|$, is important to the our proposed model. We now investigate how the segmentation result changes according to different values of f. For this purpose, we re-run our model (without web n-gram) on the standard dataset with f ranging from 1.5 to 3.0 in steps of 0.25. Figure 3 summarizes the results. There are common trends across annotator A, B, C and their intersection. The F score increases when f increases from 1.5 to 2.0, and decreases afterwards. It suggests that too little penalty (small f) favors long segments and hurts segmentation accuracy, while too much penalty (big f) negatively impacts on the results since it favors segments with very short length. It also indicates that a moderate penalty at $f = 2.0$ is a reasonable choice for the proposed model.

6. INTEGRATED LANGUAGE MODEL

In this section we will introduce the proposed integrated language model with query segmentation (QSLM). We first motivate QSLM with an oracle experiment, then describe the derivation of QSLM, and finally conduct extensive experiments on a large scale web search dataset.

6.1 Oracle Ranker

To motivate the formulation of QSLM, we have carried out an intuitive and interesting experiment. Given an oracle ranker, we let the ranker choose the bigram or unigram language model for each query, whichever gives a better NDCG score. Table 4 lists the result of the oracle ranker compared to other models. As such a simple oracle performs significantly better than either the bigram or unigram language models, it suggests that it may be possible to improve the search ranking if one can successfully emulate the behavior of the Oracle – to accurately predict when to use a unigram model and when to use a bigram model. We will show that query segmentation can help achieve a similar effect.

Table 4: Oracle Ranker

Method	NDCG@1	NDCG@3	NDCG@10
BM25	0.3108	0.3358	0.3986
Unigram LM	0.3117	0.3366	0.3999
Bigram LM	0.3141	0.3380	0.3999
Oracle Ranker	0.3471	0.3628	0.4186

6.2 Integrated Language Model

Given a model for computing the probability of a segmentation S for a query Q, we can exploit this information and develop a new retrieval model incorporating the query segmentation structure. Note that the retrieval model proposed here is independent of the query segmentation technique. We start by formulating the integrated language model with query segmentation based on the probabilistic ranking principle [15]. Specifically, we can rewrite the probability that a document is relevant to a query as follows:

Table 2: Segmentation Performance on the Standard Dataset

Annotator	Measure	MI	[19] EM + Corpus	Our Model	Simple Model + MS Web N-gram	[6] Simple Model + Google Web N-gram	[19] EM + Corpus + Wiki	Our Model + MS Web N-gram
A	query accuracy	0.274	0.414	0.440	0.482	0.536	0.526	0.540
	classify accuracy	0.693	0.762	0.776	0.782	0.807	0.810	0.803
	segment precision	0.469	0.562	0.598	0.645	0.665	0.657	0.669
	segment recall	0.534	0.555	0.639	0.602	0.708	0.657	0.713
	segment F	0.499	0.558	**0.618**	0.622	0.686	0.657	**0.690**
B	query accuracy	0.244	0.440	0.410	0.466	0.380	0.494	0.485
	classify accuracy	0.634	0.774	0.750	0.777	0.752	0.802	0.776
	segment precision	0.408	0.568	0.521	0.568	0.519	0.623	0.591
	segment recall	0.472	0.578	0.631	0.601	0.626	0.640	0.650
	segment F	0.438	**0.573**	0.571	0.584	0.568	**0.631**	0.619
C	query accuracy	0.264	0.416	0.402	0.460	0.454	0.494	0.465
	classify accuracy	0.666	0.759	0.756	0.772	0.772	0.796	0.803
	segment precision	0.451	0.558	0.548	0.597	0.581	0.634	0.624
	segment recall	0.519	0.561	0.619	0.590	0.653	0.642	0.655
	segment F	0.483	0.559	**0.582**	0.594	0.615	0.638	**0.639**
Intersection	query accuracy	0.343	0.528	0.586	0.636	0.627	0.671	0.682
	classify accuracy	0.728	0.815	0.842	0.847	0.851	0.871	0.855
	segment precision	0.510	0.640	0.681	0.736	0.718	0.767	0.770
	segment recall	0.550	0.650	0.747	0.721	0.778	0.782	0.788
	segment F	0.530	0.645	**0.713**	0.728	0.746	0.774	**0.779**

$$
\begin{aligned}
&P(R=1|Q,D)\\
&=\sum_B P(B|Q,D)P(R=1|B,Q,D)\\
&=\sum_B P(B|Q,D)\frac{P(Q|B,D,R=1)P(R=1|B,D)}{\sum_{r=\{0,1\}}P(Q|B,D,R=r)P(R=r|B,D)}\\
&=\sum_B P(B|Q,D)\frac{\frac{P(Q|B,D,R=1)}{P(Q|B,D,R=0)}}{\frac{P(Q|B,D,R=1)}{P(Q|B,D,R=0)}+\frac{P(R=0|B,D)}{P(R=1|B,D)}}\\
&\equiv\sum_B P(B|Q,D)\frac{a}{a+b}
\end{aligned}
$$

where:

$$a=\frac{P(Q|B,D,R=1)}{P(Q|B,D,R=0)},\qquad b=\frac{P(R=0|B,D)}{P(R=1|B,D)}.$$

As the query segmentation is performed independently of the document, $P(B|Q,D)=P(B|Q)$. Furthermore, when a document is irrelevant, we can approximate the query as being generated from the background corpus statistics, independent of the document:

$$a\approx\frac{P(Q|B,D,R=1)}{P(Q|B,\theta_C)}$$

Finally, as the relevance of a document is independent of the segmentation partition without knowing the query, we will assume that all document has an equal probability of being relevant. Thus, we can approximate b as the average ratio of irrelevant to relevant documents over a set of queries.

$$b\approx\frac{P(R=0)}{P(R=1)}\approx\sum_Q\frac{|\text{Irrelevant}(Q)|}{|\text{Relevant}(Q)|}$$

In this work, we apply a language model approach to estimate the the probability ratio a:

$$
\begin{aligned}
a&\approx\frac{P(Q|B,D,R=1)}{P(Q|B,\theta_C)}\\
&=\prod_{m=1}^{M}\frac{P(S_m|D,R=1)}{P(S_m|\theta_C)}\\
&=\prod_{m=1}^{M}\frac{P(w_{k_m},...,w_{k_{m+1}-1}|\theta_D)}{P(w_{k_m},...,w_{k_{m+1}-1}|\theta_C)}\\
&=\prod_{m=1}^{M}\prod_{i=k_m}^{k_{m+1}-1}\frac{P(w_i|w_{i-1},\theta_D)}{P(w_i|w_{i-1},\theta_C)}
\end{aligned}
$$

For irrelevant documents, the query segments are generated from the an n-gram language model trained from the background corpus. For relevant documents, the query segments are modeled using a smoothed bigram model trained from the document, interpolated with the background corpus. Specifically:

$$P(w_i|w_{i-1},\theta_C)=\frac{f_{w_{i-1}w_i,C}+\mu_C\frac{f_{w_i,C}}{|C|}}{f_{w_{i-1},C}+\mu_C},\qquad(13)$$

$$P_{bi}(w_i|w_{i-1},\theta_D)=\frac{f_{w_{i-1}w_i,D}+\mu_D\frac{f_{w_i,D}}{|D|}}{f_{w_{i-1},D}+\mu_D},\qquad(14)$$

$$P(w_i|w_{i-1},\theta_D)=(1-\lambda)P_{bi}(w_i|w_{i-1},\theta_D)+\lambda P(w_i|w_{i-1},\theta_C)\qquad(15)$$

7. RETRIEVAL EXPERIMENTS

In this section we conduct a set of experiments for the QSLM model on the web search task. The main reason why we did not carry out experiments on the TREC datasets is due to the lack of clickthrough data for TREC queries, which is important to our study. In the following sections, we invest several variants of the model and discuss the choice in model parameters.

7.1 Evaluation Metrics and Baselines

We evaluate the retrieval models on a large-scale real world dataset, containing 12,064 English queries sampled from the query log of a major commercial search engine. On average, each query is associated with 74 web documents, with each query-document pair manually assigned a relevance label on a 5-level scale: 0 means that the document D is detrimental to query Q, 4 means that the document D is most relevant to Q. For comparison, we include 3 baseline models in the results: BM25 [16], unigram LM with Dirichlet smoothing [25], and bigram LM as specified in Equation (15). In order to obtain the optimal parameters in our model as well as in the baselines, we divide the whole dataset evenly into a training set and a test set, each containing 6,032 queries, and estimate the parameters from the training set using grid search, as proposed in [21]. The optimal parameters of the models are reported in Table 5. Finally we also list the simple oracle results as reference. The performance of all the retrieval models is measured by mean normalized discounted cumulative gain (NDCG) [7] at

truncation levels 1, 3, and 10. We list the dataset statistics in Figure 4 and report detailed results of the retrieval models in Table 6. In Table 6, we report the results by query

Table 5: Optimal values of parameters

Model	NDCG
Unigram LM	$\mu = 2.702$
Bigram LM	$\mu_C = 425026, \mu_D = 0.51, \lambda = 0.681$
QSLM	$\mu_C = 500213, \mu_D = 0.50, \lambda = 0.90, b = 720$

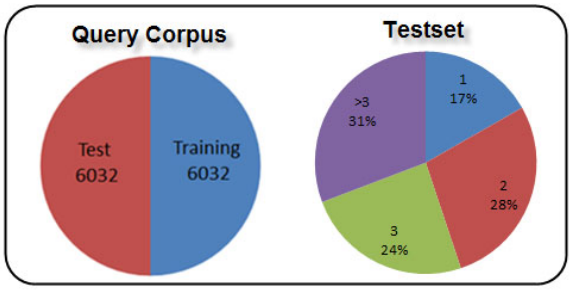

Figure 4: Query Distribution in the Datasets

length. For short queries, there are few variations in the segmentation. Thus, there is little room for improvement by exploiting segmentation information. However, the effect of query segmentation is more pronounced when the query contains 4 or more words, which we consider as a long query. In this case, the NDCGs of BM25 and unigram LM are similar, both outperformed by the bigram LM. However, QSLM's performance (0.3419, 0.3539 and 0.4040) is significantly better than all other models at all levels of NDCG truncation. In fact, we have conducted a paired t-test between QSLM and the other models. At confidence level $\alpha = 0.01$, the difference between QSLM and BM25/unigram LM at all three levels of NDCG truncation is statistically significant. The difference between QSLM and bigram LM at both NDCG@1 and NDCG@3 are significant.

Table 6: Results of IR Models on Web Search

Length	#Queries	Model	NDCG@1	NDCG@3	NDCG@10
1	1012	BM25	0.2515	0.2773	0.3496
1	1012	Unigram LM	0.2515	0.2767	0.3497
1	1012	Bigram LM	0.2462	0.2737	0.3452
1	1012	QSLM	0.2462	0.2737	0.3452
2	1694	BM25	0.3125	0.3391	0.4068
2	1694	Unigram LM	0.3131	0.3393	0.4076
2	1694	Bigram LM	0.3132	0.3392	0.4074
2	1694	QSLM	**0.3169**	**0.3404**	**0.4078**
2	1694	Oracle Ranker	0.3488	0.3656	0.4266
3	1471	BM25	0.3273	0.3603	0.4226
3	1471	Unigram LM	0.3293	0.3607	0.4242
3	1471	Bigram LM	0.3322	0.3617	0.4244
3	1471	QSLM	**0.3332**	**0.3619**	**0.4251**
3	1471	Oracle Ranker	0.3657	0.3877	0.4423
>3	1855	BM25	0.3287	0.3454	0.3988
>3	1855	Unigram LM	0.3294	0.3476	0.4009
>3	1855	Bigram LM	0.3354	0.3500	0.4009
>3	1855	QSLM	**0.3419**	**0.3539**	**0.4040**
>3	1855	Oracle Ranker	0.3651	0.3752	0.4222

7.2 Alternative Choice of the Bigram Model

As the choice of smoothing methods is critical to language modeling for information retrieval [25], we also test QSLM with an alternative form of bigram model (2-stage smoothed bigram) proposed in [8]:

$$P(q_l|q_{l-1},\theta_D) = (1-\lambda_1)[(1-\lambda_2)\frac{f_{q_l,D}+\mu_1/|V|}{|D|+\mu_1} + \lambda_2\frac{f_{q_{l-1}q_l,D}+\mu_2/|V|^2}{f_{q_{l-1},D}+\mu_2}] + \lambda_1[(1-\lambda_3)\frac{f_{q_l,C}+\mu_3/|V|}{|C|+\mu_3} + \lambda_3\frac{f_{q_{l-1}q_l,C}+\mu_4/|V|^2}{f_{q_{l-1},C}+\mu_4}] \quad (16)$$

$$P(q_l|q_{l-1},\theta_C) = (1-\lambda_3)\frac{f_{q_l,C}+\mu_3/|V|}{|C|+\mu_3} + \lambda_3\frac{f_{q_{l-1}q_l,C}+\mu_4/|V|^2}{f_{q_{l-1},C}+\mu_4} \quad (17)$$

Results in Table 7 show that QSLM with 2-stage smoothed bigram performs slightly better than the original form of smoothed bigram (see Equation (15)) on NDCG3 and NDCG10. However, this smoothed bigram LM has a larger set of parameters to tune than the original bigram LM, from 3 in the original to 7. Therefore, the original form of smoothed bigram model might be more generalizable to other scenarios.

Table 7: QLSM with Difference Bigram Models

Model	NDCG@1	NDCG@3	NDCG@10
QSLM - Bigram Model 1	**0.3419**	0.3539	0.4040
QSLM - Bigram Model 2	0.3404	**0.3549**	**0.4045**

Note: tests are made on 1855 queries of length greater than 3 in the test set.

7.3 Effect of Number of Segmentations

The effectiveness of a retrieval model combined with query segmentation relies on the ability to properly quantify the uncertainty of a segmentation. Ambiguous queries tend to have several segmentations with equal likelihood. So it's necessary to explore how the retrieval model performance changes with respect to the number of segmentations. In this experiment, we run QSLM on long queries from the web search test set (1855 queries) and vary the maximum number of query segmentations from 1 to 10. All segmentations are sorted by their probability. Figure 5 shows the performance trends at NDCG@1, NDCG@3 and NDCG@10. At all three levels of NDCG, it follows a common trend: the results do improve as the number of segmentations increases. In general, the NDCG scores reach a max when the number of segmentations reaches 3 or 4. This demonstrates that we should consider the segmentation probability and incorporate more than one candidate segmentations into the retrieval model. It also illustrates that we can achieve a reasonable result by considering only the top few segmentations.

7.4 Effect of the Penalty Factor on Retrieval

As discussed in Section 5.4, the penalty factor f is important to query segmentation. Here we investigate how

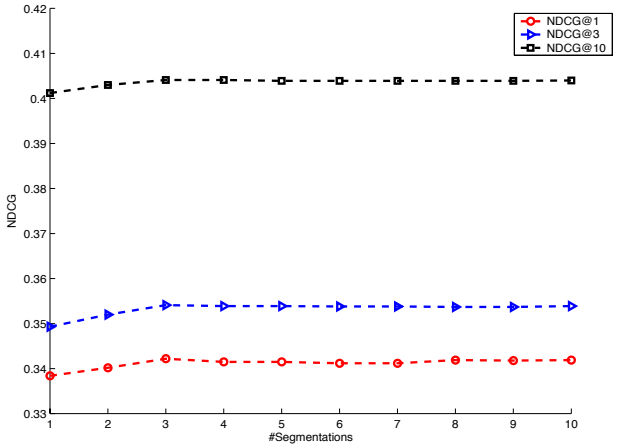

Figure 5: Retrieval Performance with Respect to #Segmentations

this factor impacts the retrieval performance. Towards this goal, we run QSLM on the long queries from the web search test set (1855 queries) with f ranging from 1.5 to 3.5 in steps of 0.25. Figure 6 indicates a very different result from query segmentation: for our retrieval model with query segmentation, the result is not sensitive to the change in f. For example, for NDCG@1, the performance of QSLM increases only slightly from $f = 1.5$ to $f = 2.0$, and decreases slowly afterwards. For NDCG@3 and NDCG@10, the differences in NDCG scores at difference values of f are negligible. The results suggest that our integrated retrieval model is robust to the choice of the penalty factor f.

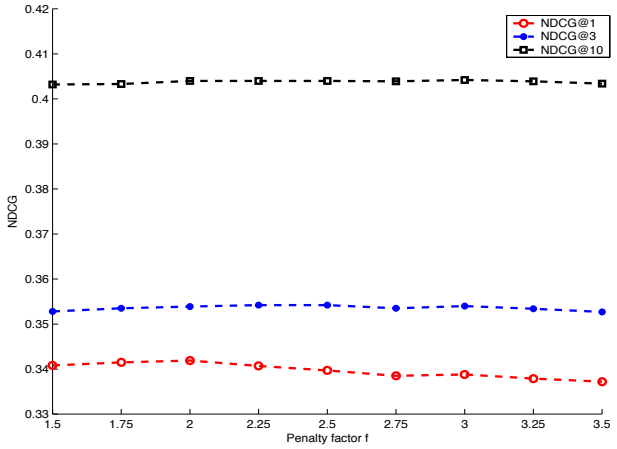

Figure 6: Retrieval Performance with Respect to Penalty Factor

7.5 Comparing to Other Query Segmentation Methods

The formulation of the QSLM model is not constrained by our query segmentation model. Theoretically it can work with query segmentations computed by any other model. It is thus interesting to know whether different query segmentation models will lead to different retrieval results. As mentioned in Section 5.3, we have constructed a 1000-query dataset with expert-labeled segmentations. More importantly, this dataset is randomly sampled from our web search dataset of 12,064 queries. Therefore, there are relevance judgments on each query. In addition to the baselines BM25, unigram LM and bigram LM, we report the results of 3 other models in this experiments: QSLM with query segmentations labeled by three human experts (QSLM^a), QSLM with segmentations computed by simple model + MS Web N-gram (QSLM^b), and QSLM with segmentations computed by our segmentation model with clickthrough only (QSLM^c). Table 8 summarizes the results of these models on 382 long queries. In general, the QSLM models work better than the 3 baselines, no matter what query segmentation model is used. However, the results demonstrate that there are noticeable differences with respect to different segmentation models. Although the simple segmentation model with web n-gram works very well for query segmentation, it is inferior to using human-labeled query segmentations. Meanwhile, QSLM, when coupled with our query segmentation model, works better than the other variants of QSLMs at all NDCG truncation levels. We believe the superior performance of our model is attributed to the appropriate modeling of relevance information in clickthroughs. Such relevance information is embedded in the query segmentations in the users' preferences and subsequently exploited by the integrated language model.

Table 8: IR Models on 1000-query Dataset

Model	NDCG@1	NDCG@3	NDCG@10
BM25	0.3410	0.3525	0.4003
Unigram LM	0.3466	0.3624	0.4057
Bigram LM	0.3626	0.3643	0.4025
QSLM^a	0.3649	0.3656	0.4034
QSLM^b	0.3600	0.3619	0.4020
QSLM^c	**0.3700**	**0.3684**	**0.4058**
Oracle Ranker	0.3929	0.3934	0.4285

Note: tests are made on 382 queries of length greater than 3.

8. CONCLUSIONS

In this paper we propose a novel unsupervised query segmentation model by jointly modeling the query-clicked documents from the search log. Experimental results on two datasets confirm the effectiveness of our model. Furthermore, we develop a unified language model with query segmentation to improve the search ranking. The implicit relevance information in the clickthrough data is the bridge between our query segmentation model and QSLM. Thorough experiments on a large-scale web search dataset show that search relevance can be improved by leveraging the query segmentations. As there is still a large gap in retrieval performance between the oracle ranker and the QSLM model, we plan to further refine the model to reduce gap in the future. Specifically, we would like to explore the use of QSLM as features to other advanced retrieval models [12].

9. ACKNOWLEGEMENTS

This paper is based upon work supported in part by the National Science Foundation under grant CNS-0834709.

10. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26, New York, NY, USA, 2006. ACM Press.

[2] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[3] S. Bergsma and Q. I. Wang. Learning noun phrase query segmentation. In *In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 819–826.

[4] G. Cao, J.-Y. Nie, and J. Bai. Integrating word relationships into language models. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 298–305, New York, NY, USA, 2005.

[5] J. Gao, J.-Y. Nie, G. Wu, and G. Cao. Dependence language model for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 170–177, New York, NY, USA, 2004. ACM.

[6] M. Hagen, M. Potthast, B. Stein, and C. Braeutigam. The power of naive query segmentation. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 797–798, New York, NY, USA, 2010.

[7] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20:422–446, October 2002.

[8] S. Javanmardi, J. Gao, and K. Wang. Optimizing two stage bigram language models for ir. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1125–1126, New York, NY, USA.

[9] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM Press.

[10] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 387–396, New York, NY, USA, 2006.

[11] T. M. K. M. Risvik and P. Boros. Query segmentation for web search. In *Proceeding of the 12th international conference on World Wide Web*, WWW '03, 2003.

[12] D. Metzler and W. B. Croft. 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*, SIGIR '05, pages 472–479, New York, NY, USA, 2005.

[13] G. Mishne and M. de Rijke. Boosting Web Retrieval through Query Operations. In *In Proc. 27th European Conference on Information Retrieval (ECIR '05)*, pages 502–516, 2005.

[14] F. Peng and D. Schuurmans. Self-supervised chinese word segmentation. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*, IDA '01, pages 238–247, London, UK, 2001. Springer-Verlag.

[15] S. E. Robertson. *The probability ranking principle in IR*, pages 281–286. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[16] S. E. Robertson and S. Walker. 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*, SIGIR '94, pages 232–241, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[17] X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 824–831, New York, NY, USA, 2005. ACM.

[18] M. Srikanth and R. Srihari. Biterm language models for document retrieval. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 425–426, New York, NY, USA, 2002.

[19] B. Tan and F. Peng. Unsupervised query segmentation using generative language models and wikipedia. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 347–356, New York, NY, USA, 2008. ACM.

[20] T. Tao and C. Zhai. 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*, SIGIR '07, pages 295–302, New York, NY, USA, 2007.

[21] M. Taylor, H. Zaragoza, N. Craswell, S. Robertson, and C. Burges. Optimisation methods for ranking functions with multiple parameters. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, CIKM '06, pages 585–593, New York, NY, USA, 2006. ACM.

[22] K. Wang, C. Thrasher, E. Viegas, X. Li, and B.-j. P. Hsu. An overview of microsoft web n-gram corpus and applications. In *Proceedings of the NAACL HLT 2010 Demonstration Session*, HLT-DEMO '10, pages 45–48, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[23] X. Yu and H. Shi. Query segmentation using conditional random fields. In *Proceedings of the First International Workshop on Keyword Search on Structured Data*, KEYS '09, pages 21–26, New York, NY, USA, 2009. ACM.

[24] C. Zhai. Fast statistical parsing of noun phrases for document indexing. In *Proceedings of the fifth conference on Applied natural language processing*, pages 312–319, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[25] C. Zhai and J. Lafferty. 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*, SIGIR '01, pages 334–342, New York, NY, USA, 2001. ACM.