Modeling Reformulation Using Query Distributions

XIAOBING XUE and W. BRUCE CROFT, University of Massachusetts, Amherst

Query reformulation modifies the original query with the aim of better matching the vocabulary of the relevant documents, and consequently improving ranking effectiveness. Previous models typically generate words and phrases related to the original query, but do not consider how these words and phrases would fit together in actual queries. In this article, a novel framework is proposed that models reformulation as a distribution of actual queries, where each query is a variation of the original query. This approach considers an actual query as the basic unit and thus captures important query-level dependencies between words and phrases. An implementation of this framework that only uses publicly available resources is proposed, which makes fair comparisons with other methods using TREC collections possible. Specifically, this implementation consists of a query generation step that analyzes the passages containing query words to generate reformulated queries and a probability estimation step that learns a distribution for reformulated queries by optimizing the retrieval performance. Experiments on TREC collections show that the proposed model can significantly outperform previous reformulation models.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Query reformulation, query substitution, query segmentation, passage analysis, information retrieval

ACM Reference Format:
Xue, X. and Croft, W. B. 2013. Modeling reformulation using query distributions. ACM Trans. Inf. Syst. 31, 2, Article 6 (May 2013), 34 pages.
DOI: http://dx.doi.org/10.1145/2457465.2457466

1. INTRODUCTION

In a typical search scenario, users pose keyword queries to express their information needs. Due to vocabulary mismatch and ambiguity, it is sometimes difficult to retrieve relevant documents using the original query. In response, techniques for reformulating the original query to improve retrieval performance have been developed. In this article, *query reformulation* is defined as a process of modifying or rewriting the original query to better match the vocabulary of relevant documents.

Many previous models of query reformulation have focused on generating related words and phrases to expand the original query. For example, the relevance model

This work was supported in part by the Center for Intelligent Information Retrieval and in part by ARRA NSF IIS-9014442. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.
Authors' addresses: X. Xue, Computer Science Department, University of Massachusetts, Amherst, MA; email: xuexb@cs.umass.edu; W. B. Croft, Computer Science Department, University of Massachusetts, Amherst, MA.

DOI: http://dx.doi.org/10.1145/2457465.2457466

approach [Lavrenko and Croft 2001; Zhai and Lafferty 2001a] adds new words to the original query, the sequential dependency model [Metzler and Croft 2005] adds phrase structure, and the latent concept expansion model [Metzler and Croft 2007] adds new word proximity features and words. These types of model do not, however, consider how the new words and phrases can be used with the original query words to form actual user queries that are variations of the original query. Instead, these new words and phrases are typically added as a large, possibly weighted, "bag of words" style to the original query, which assumes that words and phrases are independent to each other. Since an actual query is formulated by a user, it is semantically coherent and involves a specific choice of words and phrases. By ignoring how words and phrases are used together in actual queries, important query-level dependencies can be missed.

For example, given the original query "oil industry history," our implementation of the relevance model generates a query representation as "0.44 industry, 0.28 oil, 0.08 petroleum, 0.08 gas, 0.08 county, 0.04 history, . . . ".[1] This representation includes both the original query words and new words and the score in front of a word can be considered as the importance of this word. Documents may contain different combinations of words from this query representation. For instance, a word combination "oil industry county" is considered more important than another one "oil industry history," since the importance of "county" is higher than "history" according to this representation. Thus, a document containing "oil industry county" is favored compared with a document containing "oil industry history." However, "oil industry history" is an actual query (also the original query), while "oil industry county" is just some random combination of words. Also, the retrieval performance of the former is much better than the latter. Therefore, the query-level dependencies between words and phrases imposed by actual user queries are important for retrieval.

Other research on Web query reformulation, on the other hand, has tended to focus on generating a single new query (e.g., Bergsma and Wang [2007], Jones et al. [2006]) by applying a specific reformulation operation. This new query is an actual user query. Different operations have been studied. *Query segmentation* [Bergsma and Wang 2007] tries to detect underlying concepts in keyword queries and annotate those concepts as phrases, which adds phrase structure into the original query. For example, given the query "oil industry history," query segmentation techniques may detect "oil industry" as a concept and annotate it as a phrase in the new query "(oil industry) history." *Query substitution* [Jones et al. 2006] tries to change some words of the original query to bridge the vocabulary mismatch. For example, the query "oil industry history" could be changed to "petroleum industry history," since some relevant documents may contain "petroleum industry" instead of "oil industry." Although these operations all generate actual queries, little work considers combining them from a unified perspective, thus important information about alternative query reformulations is not captured.

Furthermore, most of previous research considers query reformulation as a query processing step which is independent from the following retrieval step. For example, the Relevance Model [Lavrenko and Croft 2001] generates the expanded query representation based on the "relevance" of each word. Some Web reformulation techniques [Bergsma and Wang 2007; Jones et al. 2006] train their models according to the judgments of human annotators, who are asked to decide whether a reformulated query is "good" or not according to their knowledge. The relevance of words and the goodness of reformulated queries are both potentially important indicators for the retrieval performance, but they may not necessarily lead to good retrieval performance, especially considering that different retrieval models can have widely different performance for a given set of words or queries. Therefore, it is essential to jointly consider

[1]The details of the relevance model will be described later.

the reformulation model and the retrieval model. In other words, the reformulation model should be trained by directly optimizing the performance of the retrieval model that will be used. There has, however, been little prior research in this direction.

In this article, we propose a novel framework where the original query is transformed into a distribution of reformulated queries. A reformulated query is generated by applying different operations including adding or replacing query words, detecting phrase structures, and so on. First, since the reformulated query is an actual query that involves a particular choice of words and phrases, this framework captures important query-level dependencies between words and phrases. Second, this framework naturally combines query segmentation, query substitution and other possible reformulation operations, where all these operations are considered as methods for generating reformulated queries. In other words, a reformulated query is the output of applying single or multiple reformulation operations. The probabilities of alternative reformulated queries can then be estimated within the same framework. Third, the probabilities assigned to each reformulated query are estimated by directly optimizing the retrieval performance on the training set. In this way, the reformulation model is adapted to the retrieval model that will be used.

The idea of transforming the original query into a distribution of actual reformulated queries is motivated by the availability of large scale query logs. These query logs record actual user queries, which makes it possible to capture the important query-level dependencies between words and phrases. Also, several important features can be extracted from query logs. For example, the frequency of a query observed provides a reasonable measure for its popularity and the number of clicked documents after posing a query is related to this query's quality. Previous reformulation models cannot fully exploit these query-level features, especially for "bag of words" style models, since they do not explicitly model a query. Thus, a reformulation framework that considers a query as the basic unit is required.

In this article, we first explore the relationships between the proposed framework and previous reformulation models. Following that, we consider the implementation of the proposed framework. Although using query logs seems a natural choice and is likely to demonstrate the potential of the query distribution framework, large-scale query logs are proprietary resources that are difficult to obtain in an academic environment. Thus, we propose an implementation using only publicly available resources. These resources help us simulate the query-level dependencies imposed by actual user queries. This implementation, on one hand, is useful for some important search applications such as legal search and forum search, where large scale query logs are not available as in Web search. On the other hand, even if in Web search, this implementation could be used as a complement to the query log–based implementation. Furthermore, since only publicly available resources are used, this approach allows us to do fair comparisons with other methods on TREC collections. Dang and Croft [2010] showed that reformulation techniques that worked well on ill-formed Web queries do not significantly improve well-formed TREC queries, especially for title queries. Thus, another motivation of this article is to design an effective query reformulation model for TREC queries.

Our implementation consists of two major components, reformulated query generation and probability estimation.

First, a set of reformulated queries is generated by using a passage analysis technique on the target corpus. The general idea of this technique is based on the observation that passages containing all query words or most of the query words provide a good source of information for query segmentation and substitution. Specifically, passages with all query words provide information about the common ways that people split the original query into different concepts. For example, given the aforementioned query "oil industry history," the analysis of passages containing all these three words in the

Gov2 collection[2] shows that the original query is split as "(oil industry)(history)" in 51 passages. Similarly, passages only containing some query words can indicate possible ways of substituting missing words. For example, the analysis of passages containing "industry history" on Gov2 shows that "petroleum industry history" appears in 46 passages, which indicates that "petroleum industry history" is a potential substitution for the original query. Note that, in this article, we focus on title queries from TREC collections. Since most title queries are short with no more than five words, it is reasonable to assume that there are some passages containing all query words, especially for large collections.

Second, a probability is estimated for each reformulated query, thereby creating a query distribution. Several important features are extracted to characterize the reformulated query as a whole. Examples of these features include the simulated probability of observing this reformulated query in query logs, the number of passages that contain this query in the target corpus and the type of reformulation operations applied to generate this reformulated query. Based on these features, a probability estimation approach is proposed to obtain the reformulated query distribution that optimizes the retrieval performance on the training set. The general idea of this approach is to transform query features into a set of retrieval features and then optimize the retrieval performance using learning to rank techniques. In addition, since the original query also belongs to the query distribution, this approach provides a principled way to combine the original query with other reformulated queries instead of tuning combination parameters.

There are three major contributions of this article: first, a novel framework for modeling query reformulation is proposed, where the original query is reformulated as a distribution of queries; second, an implementation of this framework using only publicly available resources is described, which consists of a passage analysis technique for generating reformulated queries and a probability estimation approach for learning the query distribution; third, the proposed framework is compared with previous models theoretically and empirically, where we further validate its advantages over previous models using experiments on TREC collections.

The rest of the article is organized as follows: Section 2 introduces background about previous reformulation models, Section 3 proposes our framework and compares it with previous models, Section 4 describes the passage analysis technique for generating reformulated queries, Section 5 introduces the probability estimation approach, Section 6 reports the experimental results, Section 7 reviews the related work, Section 8 makes some discussions, and Section 9 concludes.

2. BACKGROUND

A *Collection* is a set of documents, which is denoted as $C = \{D_i\}_{i=1}^{|C|}$. Here, D_i denotes a document. $|C|$ denotes the size of the set C.

A *Passage* is defined as a sequence of words. In this article, we consider a passage to be a non-overlapped window with a fixed number of words. *Passage Analysis* is the technique that extracts the required information from passages containing all or most query words.

The *Original Query* is the query posed by the user, which is denoted as $Q = q_1 q_2 \dots q_l$. Here, q_i is a query word and l is the length of the query. This article focuses on short queries, where l is usually not bigger than five. Using our previous example, the original query is "oil industry history."

[2]Gov2 is a TREC collection used for ad-hoc retrieval.

Query Reformulation is the process of modifying the original query in order to retrieve more relevant documents. A *Reformulation Model* is a model that transforms the original query Q into some new representation. θ_Q denotes the parameters of a reformulation model. Previous reformulation models can be divided into two categories.

2.1. Concept Distribution

The first category of previous models is a distribution of query concepts, which is denoted as **CDist**. Note that a query concept has a broad definition in this article. Besides the word and phrase in the original query, it also includes some proximity feature [Metzler and Croft 2005] and latent word [Lavrenko and Croft 2001; Zhai and Lafferty 2001a] and phrase [Metzler and Croft 2007]. Formally, this category of models are defined as follows.

Given a vocabulary of concepts $S_C = \{c\}$, where c is a concept, CDist transforms the original query into a categorial distribution over S_C. The parameters of this distribution $\theta_Q = \{P(c|R)\}$ consist of the probabilities of generating concepts c from the underlying information need or relevance model R. The sum of the generation probabilities of all concepts is equal to one, that is, $\sum_{c \in S_C} P(c|R) = 1$. CDist can be represented as a set of tuples $\{(P(c|R) \;\; c)\}$, where each tuple $(P(c|R) \;\; c)$ denotes c and its corresponding generation probability $P(c|R)$.

Now we consider how to use this concept distribution for retrieval.

Lafferty and Zhai [2001] proposed the KL-divergence retrieval model as a special case of the more general risk minimization framework for information retrieval. Zhai and Lafferty [2001a] used it for pseudo-relevance feedback, where the retrieval score of a document is calculated as the negative KL divergence between the unigram query language model θ_Q and the unigram document language model θ_D, which is calculated in Equation (1).

$$score(Q, D) = -KL(\theta_Q, \theta_D) \stackrel{rank}{=} \sum_w P(w|R) \log P(w|D), \tag{1}$$

where $P(w|R)$ is the probability of generating w from the relevance model R and $P(w|D)$ is the probability of generating w from a document D. Different models were developed to estimate $P(w|R)$ such as the model-based feedback [Zhai and Lafferty 2001a] and the relevance model [Lavrenko and Croft 2001]. $P(w|D)$ is estimated using the language modeling approach as shown in Equation (2).

$$P(w|D) = \lambda P_{ml}(w|D) + (1 - \lambda) P(w|C), \tag{2}$$

where $P(w|D)$ is estimated by a mixture of the maximum likelihood estimation $P_{ml}(w|D)$ and the background model $P(w|C)$ [Zhai and Lafferty 2001b]. λ is the mixture parameter.

Equation (1) calculates the negative cross entropy between θ_R and θ_D where the event space is a set of words $\{w\}$. The unigram query language model in Equation (1) is a special case of the more general concept distribution, where a concept c is limited to a word w. Thus, the retrieval score of a document D using the concept distribution is calculated in Equation (3), which calculates the negative entropy between θ_R and θ_D in the space of words $\{c\}$.

$$score(Q, D) = \sum_{c \in S_C} P(c|R) \log P(c|D). \tag{3}$$

Next, we describe two typical models from this category, the relevance model and the sequential dependence model.

The *Relevance Model (RM)* [Lavrenko and Croft 2001] limits a concept c to a word w including both the original query words and the new words that are related to the original query. Thus, a concept distribution degenerates to a word distribution, that is, $\theta_Q = \{P(w|R)\}$. In the relevance model, $P(w|R)$ is estimated by mixing the language models of the top ranked documents, which is shown in Equation (4).

$$P(w|R) \approx P(w|Q) = \frac{\sum_D P(w|D)P(Q|D)P(D)}{P(Q)}, \tag{4}$$

where $P(Q|D)$ is the probability of generating Q from the language model induced from a document D and $P(D)$ is the prior probability of D. $P(Q)$ serves as a normalizer.

For example, using the relevance model, the original query "oil industry history" is reformulated as a word distribution "(0.44 industry), (0.28 oil), (0.08 petroleum), (0.08 gas), (0.08 county), (0.04 history), . . . ", which includes not only words from the original query such as "oil," "industry," and "history," but also new words like "gas" and "petroleum."

The *Sequential Dependency Model* (SDM) [Metzler and Croft 2005] considers three types of concepts, query word (t), ordered bigram (o) and unordered bigram (u). The ordered bigram considers the exact match of a bigram and thus captures the information of phrase. The unordered bigram considers matching the two words of a bigram within a word window and thus considers the word proximity. For example, there are two bigrams in the query "oil industry history," that is, "oil industry" and "industry history", and each bigram has its corresponding ordered bigram and unorder bigram. "#1(oil industry)" and "#uw10(oil industry)" denote an ordered bigram and an unordered bigram using a ten word window, respectively. "#1" is a phrase operator and "#uw10" is a word proximity operator in the Indri query language [Metzler and Croft 2004].

The retrieval score of a document D using SDM is calculated in Equation (5).

$$\begin{aligned} score(Q, D) \; = \; & \sum_{t \in T(Q)} \frac{\lambda_T}{|T(Q)|} \log P(t|D) + \sum_{o \in O(Q)} \frac{\lambda_O}{|O(Q)|} \log P(o|D) \\ & + \sum_{u \in U(Q)} \frac{\lambda_U}{|U(Q)|} \log P(u|D), \end{aligned} \tag{5}$$

where $T(Q)$ denotes a set of query words in Q and $|T(Q)|$ is the size of $T(Q)$. $O(Q)$ denotes a set of ordered bigrams and $U(Q)$ denotes a set of unordered bigrams. λ_T, λ_O and λ_U are parameters that represent the weights corresponding to $T(Q)$, $O(Q)$ and $U(Q)$, respectively. Comparing Equation (5) with Equation (3), it is not difficult to observe that $P(t|R)$, $P(o|R)$ and $P(u|R)$ are estimated as $\frac{\lambda_T}{|T(Q)|}$, $\frac{\lambda_O}{|O(Q)|}$ and $\frac{\lambda_U}{|U(Q)|}$, respectively. In other words, each word t will receive the same probability and it is true for the ordered bigram o and the unordered bigram u.

For example, SDM reformulates the original query "oil industry history" as "(0.28 oil), (0.28 industry), (0.28 history), (0.08 (oil industry)), (0.08 (industry history))."[3] Besides the words from the original query, this representation also includes bigrams such as "oil industry" and "industry history," which are denoted as "(oil industry)" and "(industry history)."

2.2. Single Reformulated Query

The second category of previous models is a single reformulated query generated by applying a specific reformulation operation, which is denoted as SRQ.

[3]For simplicity, we don't differentiate the ordered bigram and the unordered bigram, but the following analysis is still valid if this differentiation is considered.

Formally, $Q_r^\star(Oper)$ denotes the new query generated after applying the reformulation operation *Oper* to the original query Q. $Q_r^\star(Oper)$ is simplified to $Q_r^\star$ if the operation *Oper* is not explicitly mentioned.

When $Q_r^\star$ is used for retrieval, the score of document D is calculated by $P(Q_r^\star|D)$, that is, the probability of generating $Q_r^\star$ from document D. The estimation of $P(Q_r^\star|D)$ depends on implementations.

Next, we describe two reformulation operations, query segmentation and query substitution.

Query Segmentation (SEG) [Bergsma and Wang 2007] is the operation of grouping query words into phrases. Given the original query $Q = q_1 q_2 \dots q_l$, the segmented query is denoted as $p_1 p_2 \dots p_m$. Here, p_i is a phrase, which groups some original query words $q_{j+1} \dots q_{j+k}$ together. j indicates the index of the original query and k is the length of the phrase p_i. If k is equal to one, p_i is a single word. For example, "(oil industry)(history)" is a segmentation of the original query.

Query Substitution (SUB) [Jones et al. 2006] is the operation of replacing some original query words with new ones. Given the original query $Q = (q_1 \dots q_{i+1} \dots q_{i+s} \dots q_l)$, the substituted query is denoted as $q_1 \dots q'_1 \dots q'_t \dots q_l$, where the original words $q_{i+1} \dots q_{i+s}$ are replaced with $q'_1 \dots q'_t$. Here, s is the number of the original query words to be replaced and t is the number of new query words. For example, "petroleum industry history" replaces "oil industry" with "petroleum industry." s and t are not necessarily equal so substitution can expand the original query. For example, "oil and gas industry history" substitutes "oil industry" with "oil and gas industry."

3. QUERY REFORMULATION AS A DISTRIBUTION OF QUERIES

In this section, the *Query Distribution* (QDist) framework is proposed and compared with previous reformulation models.

3.1. Framework

Formally, we first generate a set of reformulated queries $S_{Q_r} = \{Q_r\}$, where Q_r is an actual query that involves a particular choice of words and phrases. Q_r is the output of applying single or multiple reformulation operations. Then, the original query Q is transformed into a categorial distribution over S_{Q_r}. The parameters of this distribution $\theta_Q = \{P(Q_r|R)\}$ include the probabilities of generating a reformulated query Q_r from the underlying information need or the relevance model R. This distribution can be displayed as a set of tuples $\{(P(Q_r|R) \;\; Q_r)\}$, where each tuple $(P(Q_r|R) \;\; Q_r)$ denotes Q_r and its corresponding generation probability $P(Q_r|R)$. Note that the original query Q also belongs to S_{Q_r}, which can be considered as a special reformulated query without applying any reformulation operation. The framework itself does not specify how to generate reformulated queries or how to estimate the probability for each reformulated query. Different strategies can be adopted based on implementations.

CDist and QDist are two distributions based on different random variables. CDist considers a concept c as the random variable, while QDist considers an actual reformulated query Q_r as the random variable. When QDist is used for retrieval, it is reasonable to borrow the idea of Equation (3) and replace c with Q_r. Thus, the retrieval score of a document D using QDist is shown in Equation (6), which is a negative cross entropy between θ_R and θ_D where the event space is a set of reformulated queries $\{Q_r\}$.

$$score(Q, D) = \sum_{Q_r \in S_{Q_r}} P(Q_r|R) \log P(Q_r|D), \tag{6}$$

where $P(Q_r|D)$ is the probability of generating Q_r from the document D. $\log P(Q_r|D)$ can be considered as the retrieval score of a document D using the query Q_r. Note that

Table I.
Different query representations for the original query "oil industry history".

Model	Output
Concept Distribution (CDist)	
RM	(0.44 industry), (0.28 oil), (0.08 petroleum), (0.08 gas), (0.08 county), (0.04 history), . . .
SDM	(0.28 oil), (0.28 industry), (0.28 history), (0.08 (oil industry)), (0.08 (industry history))
Single Reformulated Query (SRQ)	
SEG	(oil industry)(history)
SUB	petroleum industry history
Query Distribution (QDist)	
(0.78 oil industry history), (0.08 (oil industry)(history)), (0.05 (petroleum industry)(history)), (0.05 (oil)(industrialized)(history)), (0.04 (oil and gas industry)(history)) . . .	

the calculation of $\log P(Q_r|D)$ depends on implementation. Thus, Equation (6) shows that the retrieval score using QDist is a weighted combination of the retrieval scores using each reformulated query Q_r in S_{Q_r}, where the weight is $P(Q_r|R)$.

For example, given the original query "oil industry history", we first generate a set of reformulated queries "(oil industry)(history), (petroleum industry)(history), (oil and gas industry)(history), (oil)(industrialized)(history) . . . ." Here, a reformulated query is generated by first applying query substitution and then applying query segmentation. For example, the original query first has a substitution to produce "petroleum industry history" and then it is segmented to produce "(petroleum industry)(history)." Then we estimate the probability for each reformulated query and the original query. Clearly, different reformulation operations can be naturally combined within this framework. The final representation generated by QDist is displayed in Table I, where the representations generated by previous models are also shown for comparison.

3.2. Comparison of Query Representations

We first compare CDist with QDist. CDist augments the original query with a set of concepts including words, phrases and some proximity features but does not consider how to fit them together to form actual queries. In contrast, QDist augments the original query with a set of actual reformulated queries, which captures the important query-level dependencies between words and phrases. Two examples are provided to show why CDist cannot capture the dependencies imposed by actual queries.

In the first example, given the original query "oil industry history" (AP = 5.9)[4] and one of its potential reformulated queries "oil industry county" (AP = 2.5), we consider the generation probabilities of these two queries using the RM representation (see Table I).

$$P(\text{oil industry history}) = P(\text{oil}) \times P(\text{industry}) \times P(\text{history}) = 0.28 \times 0.44 \times 0.04$$
$$P(\text{oil industry county}) = P(\text{oil}) \times P(\text{industry}) \times P(\text{county}) = 0.28 \times 0.44 \times 0.08.$$

According to RM, the probability assigned to "oil industry county" is even higher than the original query "oil industry history," since RM assumes that the words in a query are independent of each other. However, "oil industry county" is not an actual query and cannot lead to good retrieval performance. Instead, QDist can produce the correct

[4]AP denotes the average precision.

generation probability for these two queries by using features such as the frequencies of being observed in query logs, where the frequency of "oil industry history" is much higher than "oil industry county."

Furthermore, the query-level dependencies imposed by actual queries are also helpful for SDM, even though SDM already uses phrases and proximity features to capture some local dependencies. In the second example, two reformulated queries with different effectiveness, that is, "(oil industry) history" (AP = 4.7) and "(oil industry) industry" (AP = 2.0), receive the same generation probability, using the SDM query representation (Table I).

$$\begin{aligned} P(\text{(oil industry) history}) &= P(\text{(oil industry)}) \times P(\text{history}) = 0.08 \times 0.28 \\ P(\text{(oil industry) industry}) &= P(\text{(oil industry)}) \times P(\text{industry}) = 0.08 \times 0.28. \end{aligned}$$

Without considering whether "history" or "industry" is more likely to be used with the phrase "(oil industry)" to form actual queries, SDM cannot discriminate these two queries. On the other hand, since "(oil industry) history" as an actual query has a higher frequency than "(oil industry) industry" in query logs, QDist will assign higher probability to this one.

Second, we compare SRQ with QDist. SRQ and QDist both consider an actual query as the basic unit. However, SRQ only uses a single reformulated query generated by applying a specific operation, while QDist generates a variety of reformulated queries where each is the output of applying single or multiple operations. Thus, QDist is more general than SRQ and takes alterative reformulated queries into consideration.

3.3. Comparison of Retrieval Scores

We first compare CDist with QDist. We assume QDist use the unigram language model as the retrieval model, that is, $P(Q|D) = \prod_{w \in Q} P(w|D)$. Then, Equation (6) can be rewritten as follows.

$$\begin{aligned} score(Q, D) &= \sum_{Q_r} P(Q_r|R) \log P(Q_r|D) \\ &= \sum_{Q_r} P(Q_r|R) \log \prod_{w \in Q_r} P(w|D) \\ &= \sum_{Q_r} P(Q_r|R) \sum_{w \in Q_r} \log P(w|D) \qquad (7) \\ &= \sum_{w} \log P(w|D) \sum_{Q_r : w \in Q_r} P(Q_r|R). \qquad (8) \end{aligned}$$

We reorganize Equation (7) to obtain Equation (8) by moving the same word w together. In Equation (8), "$Q_r : w \in Q_r$" denotes a set of reformulated queries containing w. $\sum_{Q_r : w \in Q_r} P(Q_r|R)$ denotes the sum of the generation probabilities of the reformulated queries that contain w. Equation (8) shows that, using the unigram language model as the retrieval model, QDist can also be represented as the weighted combination of the retrieval scores using each word w. However, the weight assigned to w is $\sum_{Q_r \in \{Q_r | w \in Q_r\}} P(Q_r|R)$ instead of $P(w|Q)$. In this way, the probability assigned to an actual query $P(Q_r|R)$ is incorporated into the retrieval model.

Note that, although the given analysis uses the unigram language model as the retrieval model, the basic idea still holds when the retrieval model uses other concepts such as phrases and word proximity features.

Second, we compare SRQ with QDist. The retrieval score of SRQ, that is, $\log P(Q^\star|D)$, can be obtained by using a special query distribution that assigns all probability to $Q_r^\star$

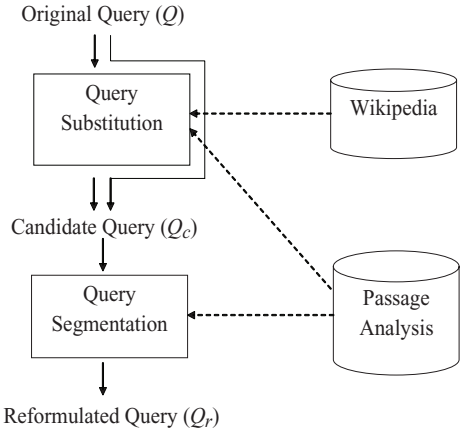

Fig. 1. The framework for generating reformulated queries.

and assigns zero probability to the other reformulated queries in S_{Q_r}. In this way, Equation (6) can be written as follows:

$$\begin{aligned} score(Q, D) &= \sum_{Q_r} P(Q_r|R) \log P(Q_r|D) \\ &= \sum_{Q_r \neq Q_r^\star} 0 \cdot \log P(Q_r|D) + 1 \cdot \log P(Q_r^\star|D) \\ &= \log P(Q_r^\star|D) \end{aligned} \tag{9}$$

Thus, it is clear that QDist consider alternative reformulated queries for retrieval instead of a single query $Q_r^\star$.

4. GENERATING REFORMULATED QUERIES

In this section, we will describe how to generate reformulated queries by analyzing passages extracted from the target corpus. This implementation can be divided into two steps: first, the original query (Q) is reformulated by substitution through passage analysis and with the help of Wikipedia; second, the candidate queries (Q_c) consisting of the original query and the queries with substitutions are segmented based on passage analysis to generate the reformulated queries (Q_r) . The framework is illustrated in Figure 1. Note that, since the large scale query logs are not available in academic environment, we use the reformulated queries generated by the passage analysis technique as a simulation. Using this technique, the important query-level dependencies between words and phrases imposed by actual queries are simulated using passage-level evidence.

4.1. Generating Query Substitutions

Three different methods are considered to find query substitutions. The first two methods are based on passage analysis, and the last uses information from Wikipedia.

4.1.1. Morphologically Similar Words. Using morphologically similar words is a reliable way to substitute the original query words using appropriate morphological variants. In this article, the morphologically similar words are chosen by considering other query words. Specifically, given the original query $Q = (q_1 \dots q_i \dots q_l)$, for each query word q_i, we extracted all passages containing all query words except q_i. For each extracted

passage, if we find a word q_i' which is morphologically similar to q_i, q_i' will be considered as a substitution of q_i and a candidate query is generated as $Q_c = (q_1 \dots q_i' \dots q_l)$. Note that a morphologically similar word is considered as a substitution only when it appears in passages containing all other query words, which further improves the quality of the substitution word.

For example, given the original query "oil industry history," we consider substitutions for each query word. For "history," passages containing both "oil" and "industry" are first extracted. Then, we consider all words in these passage except the query words as potential substitutions. Since some extracted passages contain the word "historical," which is similar to "history," "historical" is used to substitute for "history" and "oil industry historical" is generated as a candidate query. The same process will also be applied to "industry" and "oil," and "industrialized" is found as a substitution for "industry."

A simple method is used to decide whether two words are morphologically similar to each other. Given two words w_i and w_j, the Porter Stemmer [Porter 1980] is used to get their root forms $stem_i$ and $stem_j$. If $stem_i$ is equal to $stem_j$ or w_i starts with $stem_j$, or w_j starts with $stem_i$, w_i and w_j are considered as morphologically similar, otherwise they are not similar.

4.1.2. Pattern-Based Method. The pattern-based method is another way to find query substitutions. Several types of patterns are derived from the original query and these patterns are then used to match qualified passages to find query substitution. Here, two types of patterns are considered, adding-word patterns and changing-word patterns, which will be described in turn.

Adding-word patterns are used to find substitutions which replace a bigram of the original query with an n-gram that adds some words in the middle of the query bigram. Specifically, given the original query $Q = (q_1 \dots q_i q_{i+1} \dots q_l)$, we consider substituting any $q_i q_{i+1}$ with $q_i w_1 \dots w_s q_{i+1}$. $w_1 \dots w_s$ are the added words and s is the number of added words. Here, we only consider adding one or two words. First, for each bigram $q_i q_{i+1}$, a pattern is designated as $q_i \star q_{i+1}$, where $\star$ denotes any word or any two words. Then, passages containing all query words $q_1 \dots q_l$ are extracted. For each extracted passage, if the designated pattern $q_i \star q_{i+1}$ matches this passage, $q_i w q_{i+1}$ is collected as a substitution of $q_i q_{i+1}$. Here, w denotes the added word/words. Thus, a candidate query is generated as $q_1 \dots q_i w q_{i+1} \dots q_l$.

For example, given the original query "oil industry history", two patterns are derived as "oil $\star$ industry" and "industry $\star$ history." Then, passages containing all three query words are extracted. "oil $\star$ industry" matches the expression "oil and gas industry" in some passages, thus "oil and gas industry" is considered as substitution for "oil industry" and a candidate query "oil and gas industry history" is generated.

Changing-word patterns are used to find substitutions that replace a trigram of the original query with a new trigram where the middle word is different. Specifically, given the original query $Q = (q_1 \dots q_i q_{i+1} q_{i+2} \dots q_l)$, we consider substituting $q_i q_{i+1} q_{i+2}$ with $q_i w q_{i+2}$, where w is a different word. First, for each trigram $q_i q_{i+1} q_{i+2}$ of the original query, a pattern is designed as $q_i \star q_{i+2}$. Second, passages containing all query words except q_{i+1} are extracted. For each extracted passage, if the pattern $q_i \star q_{i+2}$ matches this passage, $q_i w q_{i+2}$ is collected as a substitution for $q_i q_{i+1} q_{i+2}$ and a candidate query substitution is generated as $q_1 \dots q_i w q_{i+2} \dots q_l$.

For example, given the original query "oil industry history," a pattern is designed as "oil $\star$ history". Then, all passages containing "oil" and "history" but not "industry" are extracted. For some passages, "oil $\star$ history" matches expressions "oil spill history," thus "oil spill history" is considered as a substitution of "oil industry history."

4.1.3. Wikipedia Redirect Page. Some query substitutions are difficult to obtain relying only on corpus information; thus, the third method uses Wikipedia as an external

resource. A redirect page in Wikipedia is designed to send the user to the article with an alternative title.[5] For example, if the user searches for "UK", a redirect page will send the user to the article with title "United Kingdom," since "UK" is the abbreviation of "United Kingdom." Other reasons for maintaining redirect pages include alternative names ("Edison Arantes do Nascimento" redirects to "Pelé"), less or more specific forms of names ("Hitler" redirects to "Adolf Hitler"), alternative spellings or punctuation ("Colour" redirects to "Color"), likely misspellings ("Massachusets" to "Massachusetts"), plurals ("Greenhouse gases" redirects to "Greenhouse gas"), related words ("Symbiont" redirects to "Symbiosis") and so on. Clearly, the redirect page is a valuable resource for query substitution and since they are created by people, the quality is generally good.

All redirect pages are organized as a set of tuples $\{(p_{src}, p_{tar})\}$, where the phrase p_{src} is redirected to the phrase p_{tar}. Given the original query $Q = (q_1 \dots q_{i+1} \dots q_{i+n} \dots q_l)$, all n-grams ($n > 1$) are extracted. For each n-gram $q_{i+1} \dots q_{i+n}$, if it matches p_{src} or p_{tar} in any tuple, the corresponding p_{tar} or p_{src} will be collected as a substitution. Then, a candidate query substitution is generated as $(q_1 \dots p_{tar} \dots q_n)$ or $(q_1 \dots p_{src} \dots q_n)$.

For example, for the original query "oil industry history", we extract n-grams "oil industry", "industry history" and "oil industry history". Each of these n-grams is used to match the redirect tuples. The n-gram "oil industry" matches a tuple ("oil industry", "petroleum industry"), thus "petroleum industry" is used as a substitution of "oil industry" and a candidate query "petroleum industry history" is generated.

4.2. Generating Query Segmentations

In this step, phrase structures are detected using passage analysis for all candidate queries including both the original query and query substitutions. The basic idea can be described as follows. Given a candidate query, passages containing all query words are extracted. Then, each extracted passage tells us one way to segment the candidate query. After analyzing all extracted passages, the most frequent ways of segmenting the candidate query can be determined.

The details of the algorithm are provided in Figure 2. Generally, the function *DetectSegmention* returns how the input query is segmented in the given passage. Then, the algorithm records all returned segmentations and their associated passage id (psg_{id}) and document id (doc_{id}), which can be used to rank different segmentations.

As an example to show how the function *DetectSegmentation* works, considering the input query "oil and gas industry history," which is a query substitution. Given the passage "... shape the history of the oil and gas industry in Oklahoma during the early days of the Oklahoma Oil boom ...," S = {"history," "oil and gas industry," "oil"} after Step 2. Since "oil" is a substring of "oil and gas industry", "oil" is removed from S in Step 3 and S becomes {"history", "oil and gas industry"}. According to S, a segmentation is formed as "(oil and gas industry)(history)."

4.3. Examples

We have discussed how to generate a set of reformulated queries by applying the substitution and segmentation operations. More examples of the reformulated queries generated are shown in Table II. "Orig" denotes the segmentation of the original query without substitution. "Wiki" denotes the reformulated queries generated from the Wikipedia redirect page. "Morph" denotes the reformulated queries generated from morphologically similar words. "Pat-add" denotes the method of using adding-word patterns and "Pat-chg" denotes the method of using changing-word patterns. Note that

[5]The definition of redirect pages and the following examples can be found at http://en.wikipedia.org/wiki/Redirects_on_wikipedia.

ALGORITHM: Generating Reformulated Query
INPUT: candidate query $Q_c = (q_1 q_2 ... q_m)$, corpus C
OUTPUT: a set of reformulated queries $R = \{(Q_r, Stat)\}$, where Q_r is one way to segment Q_c and $Stat = \{(psg_{id}, doc_{id})\}$ records from which passages (psg_{id}) and documents (doc_{id}), Q_r is detected.
PROCESS:

(1) select passages containing $q_1 q_2 ... q_m$ from C.
(2) for each selected passage psg
- get (psg_{id}, doc_{id}), the passage id and corresponding document id of psg.
- $Q_r = DetectSegments(Q_c, psg)$
- add $(Q_r, (psg_{id}, doc_{id}))$ into R.

FUNCTION: $DetectSegmention$
INPUT: query $Q_c = (q_1 q_2 ... q_m)$, passage $psg = w_1 w_2 ... w_g$, the size of the passage is g
OUTPUT: Q_r
PROCESS:

(1) $S =$Ø, $i = 1$
(2) while $i \leq g$
- search the longest string $str = w_i w_{i+1} ... w_{i+s}$ that starts with w_i and matches a substring of $q_1 q_2 ... q_m$.
- if str is found
  $S \leftarrow str$, $i = i + s + 1$
  else
  $i = i + 1$

(3) for each str in S
- if str is a substring of another string in S
  $S = S - str$

(4) According to S, Q_c is segmented to form Q_r.

Fig. 2. Algorithm of generating reformulated queries.

for each substitution method, only one example is displayed, but actually more than one reformulated query would typically be generated.

Table II contains many interesting reformulated queries. "(controlling)(type 2 diabetes)" reformulates "controlling type ii diabetes," "1-up system" reformulates "pyramid scheme," "(low)(leukocyte count)" reformulates "low white blood cell count," "(Enron)(California electricity crisis)" reformulates "Enron California energy crisis," and so on. Sometimes, the reformulated queries generated from different sources happen to be the same. For example, Wiki and Pat-chg both generate "(controlling)(type 2 diabetes)". Also, some sources cannot generate reformulated queries for certain queries (denoted as "n/a" in Table II).

5. ESTIMATING QUERY DISTRIBUTIONS

In this section, we first propose a model to learn the query distribution by optimizing the retrieval performance and then we describe the query features and the retrieval models, respectively.

Table II. Examples of Reformulated Queries

Source	Controlling Type ii Diabetes	Pyramid Scheme
Orig	(controlling)(type ii diabetes)	(pyramid scheme)
Wiki	(controlling)(type 2 diabetes)	(1)(up)(system)
Morph	(control)(type ii diabetes)	n/a
Pat-add	n/a	(pyramid promotional scheme)
Pat-chg	(controlling)(type 2 diabetes)	n/a
source	low white blood cell count	Enron California energy crisis
Orig	(low white blood cell count)	(Enron)(California energy crisis)
Wiki	(low)(leukocyte count)	(Enron)(California electricity crisis)
Morph	(lower)(white blood cell count)	(Enrons)(California energy crisis)
Pat-add	n/a	n/a
Pat-chg	(low red blood cell count)	(Enron)(California electricity crisis)
source	hybrid alternative fuel cars	ban human cloning
Orig	(hybrid)(alternative fuel)(cars)	(ban)(human cloning)
Wiki	(hybrid)(alternative fuel)(vehicle)	(ban)(human)(clone)
Morph	(hybrid)(alternative fueled)(cars)	(bans human cloning)
Pat-add	n/a	(ban on human cloning)
Pat-chg	n/a	(ban reproductive cloning)

5.1. Model

The probability for each reformulated query $P(Q_r|R)$ is approximated by its weight $w(Q_r)$. $w(Q_r)$ is calculated as a linear combination of their query feature values as shown in Equation (10).

$$P(Q_r|R) \propto w(Q_r) = \sum_k \lambda_k f_k(Q_r), \tag{10}$$

where $f_k(Q_r)$ denotes the query feature value extracted to characterize a reformulated query Q_r and λ_k is the parameter corresponding to f_k. Representing the weight of query as a linear combination of query feature values has been used in previous work [Bendersky et al. 2010; Wang et al. 2010]. Finally, $w(Q_r)$ is normalized to obtain $P(Q_r|R)$.

Given Equation (10), we need to learn the parameters λ_k for each query feature f_k. As discussed in introduction, we jointly consider the query reformulation model and the retrieval model. Therefore, the parameters λ_k are learned by directly optimizing the retrieval performance of a retrieval model M. In this way, the learned query distributions can be adaptive to different retrieval models. The general idea of the learning strategy is to transform each query feature into the corresponding retrieval feature, where these two types of features share the same parameters. Then, using learning to rank techniques, the parameters for the retrieval features can be learned by optimizing the retrieval performance. Since the query features share the same parameters as the retrieval features, the learned parameters can be used in Equation (10) to estimate $P(Q_r|R)$.

Specifically, we add Equation (10) into Equation (6) as follows:

$$\begin{aligned} score(Q_r, D) &= \sum_{Q_r} P(Q_r|R) \log P(Q_r|D) \\ &= \sum_{Q_r} \sum_k \lambda_k f_k(Q_r) \log P(Q_r|D) \end{aligned}$$

Table III.
An example of reformulated queries with their corresponding feature values and retrieval scores.

Q_r	f_1	f_2	$\log P(Q_r\|D)$
oil industry history	0.5	0.1	−3
petroleum industry history	0.3	0.6	−4
oil and gas industry history	0.2	0.3	−5

$$= \sum_k \lambda_k \sum_{Q_r} f_k(Q_r) \log P(Q_r|D) \tag{11}$$

$$= \sum_k \lambda_k F_k(\{Q_r\}, D) \tag{12}$$

In Equation (12), the retrieval score using the query distribution is rewritten as a linear combination of retrieval features F_k, where F_k and f_k share the same parameter λ_k. Comparing Equation (11) and Equation (12), F_k is calculated as follows.

$$F_k(\{Q_r\}, D) = \sum_{Q_r} f_k(Q_r) \log P(Q_r|D). \tag{13}$$

Equation (13) shows how to transform f_k into F_k. Basically, F_k is the weighted combination of the retrieval scores using all reformulated queries in $S_{Q_r} = \{Q_r\}$, where the weight is the query feature value $f_k(Q_r)$. $\log P(Q_r|D)$ is the retrieval score for Q_r using the indicated retrieval model M.

An example is provided. Table III shows a set of reformulated queries and their corresponding feature values and retrieval scores. Then, the retrieval features can be calculated as follows.

$$F_1 = 0.5 \times (-3) + 0.3 \times (-4) + 0.2 \times (-5) = -3.7$$
$$F_2 = 0.1 \times (-3) + 0.6 \times (-4) + 0.3 \times (-5) = -4.2$$

After transforming the query feature into the retrieval feature, any learning-to-rank approach assuming that the retrieval model is a linear combination of the input retrieval features, as Equation (12) as can be used to learn parameters. In this article, ListNet [Cao et al. 2007] is used to learn the parameters. ListNet optimizes the cross entropy of the permutation probabilities between the predicted ranked list output by the retrieval model and the gold-standard ranked list produced using relevance judgments. Specificially, a variation of ListNet is considered. Instead of using the neural network, a limited-memory version of BFGS [Byrd et al. 1994] is used for optimization due to its efficiency.

Many learning-to-rank approaches do not assume that the retrieval model is a linear combination of input features. In other words, the parameters learned are not one-to-one correspondent to the input features. Thus, it is an interesting research issue to generalize the idea used here to an arbitrary learning-to-rank approach. One possible solution is to measure the performance gain of each feature on the training set using the leave-one-out strategy.

Figures 3 and 4 show the framework of the training and predicting phases, respectively. Note that the parameters learned, $\{\lambda_k\}(M)$, are based on the retrieval model M.

5.2. Features

One of the advantages of QDist comes from capturing the query-level dependencies between words and phrases imposed by actual queries. Thus, features that characterize

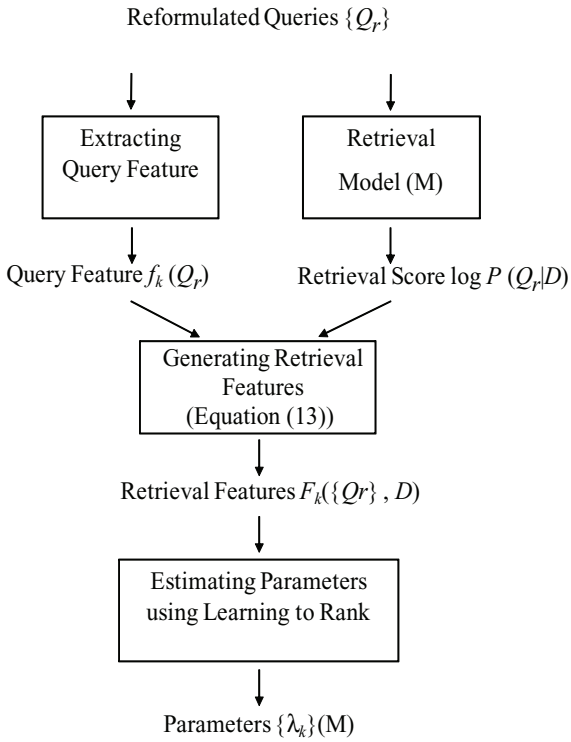

Fig. 3. The framework of the training phase for estimating the query distribution.

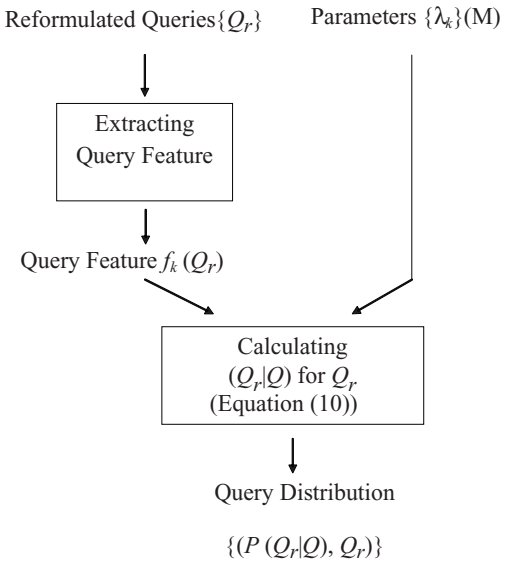

Fig. 4. The framework of the testing phase for estimating the query distribution.

Table IV. Three Types of Features for the Reformulated Query Q_r

PSG Features	
psg$N$-count	count of passages with size N containing Q_r
doc-count	count of documents containing Q_r
NGRAM Features	
qlog	probability estimated from query logs
title	probability estimated from title
body	probability estimated from body
anchor	probability estimated from anchor text
OPER Features	
Orig	whether it is the original query
Sub	whether it is a substituted query
Morph	whether it is a substituted query using Morph
Pat-add	whether it is a substituted query using Pat-add
Pat-chg	whether it is a substituted query using Pat-chg
Wiki	whether it is a substituted query using Wiki
Seg	whether it is a segmented query

the reformulated query Q_r as a whole are essential for estimating $P(Q_r|R)$. Three types of features are considered.

The first type of features (PSG) are information extracted from the target corpus. Specifically, we consider the number of passages that contain the whole reformulated query as a feature, which provides evidence about whether this reformulated query is widely used in the target corpus. Different passage sizes are considered. A smaller passage size indicates stronger dependencies between query terms, but has lower coverage since a lot of reformulated queries can not be observed within a tighter window. On the other hand, using a bigger passage size increases the coverage at the cost of sacrificing some quality. Thus, it is interesting to consider features extracted based on different passage sizes. As an extreme case, we also consider the number of documents where a reformulated query appears.

The second type of features (NGRAM) are based on query logs and the Web corpus. Query logs record the frequencies of the queries used by search engine users, which can be directly used as features of a reformulated query. On the other hand, the Web corpus used by search engine companies provides better coverage than the target corpus, and thus can be used as a complement. Furthermore, different fields of a Web page serve different purposes, thus additional information can be obtained by splitting the frequencies of a reformulated query in the Web corpus according to different fields. Using the Web N-gram Services provided by Microsoft [Huang et al. 2010], this information can be efficiently obtained, where raw frequencies are simulated by the N-gram language model probabilities. Particularly, these probabilities calculated from query logs and different fields of a Web page (body, title and anchor) are provided, respectively.

The third type of features (OPER) indicate the operations applied to the original query to generate the concerned reformulated query. These features correspond to questions such as whether the reformulated query is the original query, or a substituted query, or a segmented query. For a substituted query, we further consider what kind of methods are used (Morphologically Similar Words, Adding Word Patterns, Changing Word Patterns, and Wikipedia Redirect Page from Section 4.1). These types of features help combine different reformulation operations within the same framework.

These three types of features are summarized in Table IV. psg$N$-count can be instantiated to a variety of features by taking different values of the passage size N.

Table V. The Indri Queries for "(petroleum industry)(history)"

document-level:
#combine(#1(petroleum industry) history)
passage-level:
#uw20(#1(petroleum industry) history)

5.3. Retrieval Models

In this section, we first describe two types of retrieval models and then discuss how to combine different retrieval models within the query distribution framework.

A retrieval model M corresponds to estimating $\log P(Q_r|D)$. Two types of retrieval models are described, the document-level model and the passage-level model. The former assumes that words and phrases are independent given a document, while the latter captures some passage evidence.

The document-level model (doc) assumes the query concepts in the reformulated queries are independent given the document. Specially, $P(Q_r|D)$ is estimated in Equation (14).

$$P(Q_r|D) = \prod_{c \in Q_r} P(c|D), \tag{14}$$

where $P(c|D)$ is estimated by the language modeling approach [Ponte and Croft 1998; Zhai and Lafferty 2001b]. The Dirichelet smoothing is used and the details can be found in Section 6.

The passage-level model (psg) prefers documents where the whole query is observed within a passage. Specifically, $P(Q_r|D)$ is estimated in Equation (15).

$$P(Q_r|D) = \frac{\#psgN(Q_r, D)}{\#psgN(D)}, \tag{15}$$

where $\#psgN(Q_r, D)$ denotes the number of passages with size N containing Q_r in document D and $\#psgN(D)$ denotes the total number of passages with size N in document D. Based on the value of N, different passage-level models can be generated.

The Indri query language [Metzler and Croft 2004] provides an implementation of these retrieval models. For example, the Indri queries for a reformulated query "(petroleum industry)(history)" are displayed in Table V by using the document-level model and the passage-level model, respectively. In this query language, the operator "#combine" is an implementation of Equation (14) and the operator "#uw$N$" is an implementation of Equation (15) where N is the passage size. "#1" is an operator for a phrase. After running these Indri queries, the retrieval scores returned by the system are $\log(P(Q_r|D))$, which can be used in Equation (13).

Given a set of reformulated queries $\{Q_r\}$, different query distributions are generated according to the retrieval model M, since the parameters $\{\lambda_k\}(M)$ learned are dependent on M as shown in Figure 3. Table VI shows the query distributions learned using the document-level model *QDist(doc)* and the passage-level model *QDist(psg)*, respectively. The passage size used in this example is 20. Note that in order to explicitly indicate the retrieval model, we use the Indri queries to represent the reformulated queries in the query distribution.

The document-level model retrieves more potentially relevant documents, while the passage-level model retrieves higher quality documents by imposing more restrictions. Considering their different properties, it is reasonable to combine these two types of models. Suppose that there are p retrieval models. One possible solution is to first generate p query distributions respectively and then linearly combine the generated

Table VI. Query Distributions Learned for the Original Query "oil industry history"

QDist(doc)
0.78 #combine(oil industry history), 0.08 #combine(#1(oil industry) history), 0.05 #combine(#1(petroleum industry) history), 0.05 #combine(oil industrialized history), 0.04 #combine(#1(oil and gas industry) history), . . .
QDist(psg)
0.59 #uw20(oil industry history), 0.19 #uw20(#1(oil industry) history), 0.12 #uw20(#1(petroleum industry) history), 0.05 #uw20(oil industrialized history), 0.05 #uw20(#1(oil and gas industry) history), . . .
QDist(doc+psg)
0.804 #combine(oil industry history), 0.054 #combine(#1(oil industry) history), 0.043 #combine(oil industrialized history), 0.042 #combine(#1(petroleum industry) history), 0.032 #combine(#1(oil and gas industry) history), 0.015 #uw20(oil industry history), 0.005 #uw20(#1(oil industry) history), 0.003 #uw20(#1(petroleum industry) history), 0.001 #uw20(oil industrialized history), 0.001 #uw20(#1(oil and gas industry) history), . . .

query distributions. However, it is not easy to decide the combination weights. Instead, an alternative approach is considered in this article. Each query feature f_k is now transformed into p retrieval features $F_k^1 \sim F_k^p$ using different retrieval models. Then, p sets of parameters $\{\lambda_k\}^1 \sim \{\lambda_k\}^p$ are learned. Using different sets of parameters, a reformulated query Q_r is expanded as p Indri queries with different probabilities, where each Indri query corresponds to a retrieval model. Since the parameters for different retrieval models are learned within the same framework, the learning procedure automatically takes care of how to balance the weights assigned to different models. Table VI shows the query distribution that combines the document-level model and the passage-level model *QDist(doc+psg)*.

6. EXPERIMENTS

Three TREC collections, Gov2, ClueWeb (Category B), and Robust04, are used for experiments. The statistics of each collection are summarized in Table VII. Gov2 and ClueWeb are large Web collections with large and varied vocabulary, while Robust04 is a newswire collection that uses a more homogeneous vocabulary. For each collection, two indexes are built using Indri [Metzler and Croft 2004], one not stemmed and the other stemmed with the Porter Stemmer [Porter 1980]. Stemming transforms a word into its root form, which is conducted either during indexing or during query processing. The latter case treats stemming as a part of query reformulation, which has been shown effective for Web search [Peng et al. 2007]. Both cases are considered in this article using two types of indexes. No stopword removal is done during indexing. For each topic, the title part is used as the query.

The query set is split into a training set and a test set. On the training set, the parameters λ_k (see Equation (12)) are learned according to Section 5. On the test set, the learned parameters λ_k are used to generate the query distribution for each test

Table VII. TREC Collections Used in Experiments

Name	Docs	Topics
Gov2	25, 205, 179	701-850
Robust04	528, 155	301-450, 601-700
ClueWeb	50, 220, 423	1-100[6]

Table VIII. Baselines and Their Parameters

Baseline	Parameters
SDM	$\lambda_O = \{0, 0.05, 0.1, 0.15\}$, $\lambda_U = \{0, 0.05, 0.1, 0.15\}$
RM	$fbDocs = \{10, 20, 30, 40, 50\}$, $fbTerms = \{10, 20, 30, 40, 50\}$ $fbOrigWeight = \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$
PRM1,PRM2	$fbDocs = \{10, 20, 30, 40, 50\}$, $fbTerms = \{10, 20, 30, 40, 50\}$ $fbOrigWeight = \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ $\sigma = \{75, 125, 175, 275, 500\}$, $\lambda = \{0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$
SegSVM	$origWeight = \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$
PSG	$psgSize = \{20, 100\}$, $origWeight = \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$
PSGRM	$psgSize = \{20, 100\}$, $fbPsgs = \{10, 20, 30, 40, 50\}$, $fbTerms = \{10, 20, 30, 40, 50\}$ $fbOrigWeight = \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$

query and the performance of using the generated query distribution is reported. Ten-fold cross validation is used in this article.

Two passage sizes are used in this article, that is, 20 and 100, which represent a tight window and a loose window respectively, according to previous work [Xu and Croft 2000]. Passages are first used to generate reformulated queries as described in Section 4. The reformulated queries generated from different passage sizes are put together to form the final set of reformulated queries after removing duplicates. Then, two query features related to the passages are extracted, that is, psg20-count and psg100-count (see Table IV). For the retrieval models, two passage-level retrieval models are considered, that is, "#uw20" and "#uw100" (see Table V).

Two types of query distributions are considered: the first one only uses the document-level model ("#combine", see Table V), which is denoted as *QDist(doc)*; the second one combines both the document-level model ("#combine") and two passage-level models ("#uw20" and "#uw100"), which is denoted as *QDist(doc+psg)*. QDist(doc) captures the dependencies imposed by actual queries during estimating the probability for each reformulated query, but not for the retrieval, since the document-level retrieval model is used. In contrast, QDist(doc+psg) captures the dependencies imposed by actual queries both in estimating the probability and in retrieving documents, since the passage-level model is combined with the document-level model.

Several baselines are compared. QL denotes the query likelihood language model [Ponte and Croft 1998; Zhai and Lafferty 2001b]. SDM denotes the sequential dependence model [Metzler and Croft 2005]. RM represents the relevance model [Lavrenko and Croft 2001]. PRM1 and PRM2 represent the positional relevance model [Lv and Zhai 2010]. SegSVM is a SVM-based query segmentation method [Bendersky et al. 2009], which is trained on a corpus of 500 pre-segmented noun phrases [Bergsma and Wang 2007]. PSG denotes a passage-augmented language model [Liu and Croft 2002] that combines the retrieval score of a document and the retrieval score of the best passage of this document. Also, PSGRM represents a passage-based relevance model [Liu and Croft 2002], where the expanded words are generated from the top ranked passages. The parameters of the given baselines and their value-ranges are summarized in Table VIII. These parameters are tuned using the same ten-fold cross validation as the query distribution models.

In Table VIII, λ_O and λ_U are the weights assigned to the ordered bigrams and the unordered bigrams in SDM. In RM, *fbDocs* is the number of top ranked documents used for pseudo-relevance feedback, *fbTerms* is the number of words for expansion and *fbOrigWeight* is the weight assigned to the original query. In PRM1 and PRM2, σ and λ are the parameters of the positional language model. In SegSVM, *origWeight* is the weight assigned to the original query. In PSG, *psgSize* is the size of the passage. For all the baselines and our methods, we use the language model implemented by *Lemur 4.10*[7], where the Dirichlet smoothing with the default parameter is adopted.

The standard performance measures, mean average precision (MAP) and the normalized discounted cumulative gain at 10 (NDCG10), are used to measure the retrieval performance.[8] The two-tailed t-test is conducted to measure significance with the p-value equal to 0.05.

6.1. Examples

First, we present examples of the query distribution learned by using the document-level retrieval model *QDist(doc)*. Table IX shows the example on the nonstemmed index. The retrieval performance of using the original query and using the query distribution is compared. The retrieval performance of the top ranked reformulated queries in the query distribution is also displayed.

Table IX shows that the learned query distribution obviously outperforms the original query. In the query distribution, the reasonable probability is assigned to the original query. In most cases, the original query receives the highest probability, since it is not safe to deviate from the original query too much. In some cases, the reformulated queries receive higher probabilities than the original one. For example, given the original query "prostate cancer treatments," the reformulated queries "prostate cancer treatment" and "#1(prostate cancer) treatment" receive higher probability and they both outperform the original query, since "treatment" is more likely to be used with "prostate cancer" in actual queries. "kudzu pueraria lobata" is another example, where the reformulated query "kudzu" receives higher probability, since "pueraria lobata" is another and less popular name of "kudzu" and not very useful for retrieval.

Besides the original query, many reasonable and effective reformulated queries can also be observed in the learned query distribution. For example, "#1(cruise ship) damage sea life" and "kyrgyzstan #1(united states) relations" are segmented queries, where much better retrieval performance is achieved by discovering the concepts "cruise ship" and "united states" in original queries. "school mercury exposure" and "bluegrass music festival history" are substituted queries where the original query words "mercury poisoning" and "blue grass" are replaced with "mercury exposure" and "bluegrass," respectively. Significant performance improvement can be observed by using these substituted queries.

Furthermore, we present the query distribution learned by using both the document-level and the passage-level retrieval models *QDist(doc+psg)*. Table X displays the example of this type of query distribution using the non-stemmed index.

Table X shows that besides using the document-level retrieval model, QDist(doc+psg) also applies the passage-level retrieval models to some important reformulated queries such as "#uw100(ephedra deaths)" and "#uw20(ephedra deaths)". These passage-level queries help further improve the performance of QDist(doc).

[7]http://www.lemurproject.org/.

[8]In order to improve readability, we report 100 $\times$ the actual values of the performance measures.

Table IX.
Example of the query distribution learned on the nonstemmed index using the document-level retrieval model. Top ranked reformulated queries (Q_r) are displayed. In the query distribution, the original query (Q) is italicized. Average Precision (AP) is reported as the retrieval performance.

$P(Q_r\|R)$	Reformulated Query (Q_r)	AP
Q: prostate cancer treatments		41.21
$QDist(doc)$		50.23
0.1390	#combine(prostate cancer treatment)	42.51
0.1118	#combine(#1(prostate cancer) treatment)	48.88
0.0920	#combine(*prostate cancer treatments*)	41.21
0.0485	#combine(#1(prostate cancer treatment))	11.50
Q: cruise ship damage sea life		6.75
$QDist(doc)$		25.83
0.1820	#combine(*cruise ship damage sea life*)	6.75
0.0950	#combine(#1(cruise ship) damage sea life)	21.69
0.0544	#combine(cruise ship damage #1(sea life))	9.80
0.0544	#combine(#1(cruise ship) damage #1(sea life))	7.43
Q: kyrgyzstan united states relations		25.88
$QDist(doc)$		39.18
0.1652	#combine(*kyrgyzstan united states relations*)	25.88
0.1128	#combine(kyrgyzstan #1(united states) relations)	36.56
0.0573	#combine(kyrgyzstan united states foreign relations)	14.99
0.0566	#combine(kyrgyzstan us relations)	35.74
Q: school mercury poisoning		10.29
$QDist(doc)$		16.64
0.2161	#combine(*school mercury poisoning*)	10.29
0.1284	#combine(school mercury exposure)	20.26
0.0441	#combine(school #1(mercury exposure))	13.64
0.0335	#combine(schools mercury poisoning)	9.11
Q: pet therapy		2.77
$QDist(doc)$		45.09
0.2753	#combine(*pet therapy*)	2.77
0.1249	#combine(#1(pet therapy))	38.93
0.0613	#combine(animal assisted therapy)	7.49
0.0486	#combine(pet therapeutic)	1.15
Q: blue grass music festival history		16.65
$QDist(doc)$		38.06
0.1953	#combine(*blue grass music festival history*)	16.65
0.1456	#combine(bluegrass music festival history)	50.10
0.0698	#combine(#1(bluegrass music) festival history)	23.90
0.0411	#combine(bluegrass #1(music festival) history)	22.46
Q: kudzu pueraria lobata		44.96
$QDist(doc)$		51.83
0.2778	#combine(kudzu)	52.69
0.1244	#combine(*kudzu pueraria lobata*)	44.96
0.0596	#combine(#1(kudzu pueraria lobata))	22.08
0.0580	#combine(#1(kudzu kudzu))	1.93

Table X.
Example of the query distribution learned on the non-stemmed index using the document-level and the passage-level retrieval models.

$P(Q_r\|R)$	Reformulated Query (Q_r)	AP
Q:	ephedra ma huang deaths	47.42
QDist(doc)		57.07
QDist(doc+psg)		62.58
0.1682	#combine(ephedra deaths)	61.78
0.1434	#combine(*ephedra ma huang deaths*)	47.42
0.0439	#combine(ephedra ma huang death)	47.12
0.0395	#combine(ephedra #1(ma huang) death)	47.41
0.0394	#combine(#1(ephedra sinica ma huang) deaths)	1.49
0.0392	#combine(ephedra sinica ma huang deaths)	28.83
0.0284	#uw100(ephedra deaths)	44.34
0.0276	#combine(#1(ephedra ephedra) deaths)	4.69
0.0233	#combine(#1(ephedra ma huang) death)	2.66
0.0193	#uw20(ephedra deaths)	36.45

6.2. Results

The first experiment is conducted to compare the query distribution model with other reformulation models. SDM, RM, PRM1, PRM2 and PSGRM represent the Concept Distribution (CDist) models. SegSVM generates a single segmented query and combines it with the original query, which can be considered as a variant of the Single Reformulated Query (SRQ) model. The results are provided in Table XI.

Table XI shows that the QDist models outperform both the CDist models (SDM, RM, PRM1, PRM2, PSGRM) and the SRQ model (SegSVM) in most situations, which supports the advantages of modeling reformulation as a distribution of queries instead of a distribution of concepts and a single reformulated query. Specifically, on Gov2, QDist(doc+psg) significantly outperforms all baselines in terms of both MAP and NDCG10. The only exception happens on the Porter-stemmed index, where QDist(doc+psg) also improves SDM and SegSVM on NDCG10, but it is not significant. On ClueWeb, NDCG10, as a more interesting performance measure for Web search, is significantly improved by QDist(doc+psg) over all baselines. For example, on the Porter-stemmed index, the improvements on NDCG10 over PRM1 and PRM2 are 23% and 34%, respectively. Large improvements are also observed on MAP, but they are not significant for some baselines. On Robust04, QDist is comparable with the baselines, where it cannot beat PRM on MAP, but still performs the best on NDCG10. Also, QDist(doc+psg) performs better than QDist(doc).

It is interesting to observe that the behavior of QDist(doc+psg) varies with different collections, although it generally achieves the best or close to the best performance on all collections. The advantages of the query distribution models over the concept distribution models are more obvious on large Web collections such as Gov2 and ClueWeb. It is reasonable, since, as aforementioned, the concept distribution models may assign high probability to some random combination of concepts. It will negatively affect the retrieval performance only when some documents in the collection contain such random combination of concepts. These documents are less likely to be observed in small homogenous newswire collections such as Robust04, while they probably appear in large scale heterogenous Web collections. Also, the query distribution model significantly improves the quality of the top ranked documents (measured by NDCG10) by considering evidence provided by multiple reformulated queries. In other words, the query distribution model favors documents that are recommended by several important

Table XI.
The results of query distribution models. The best performance is bolded. The superscript of each baseline indicates that the query distribution approach is significantly different with the baseline approach.

	Gov2		Robust04		ClueWeb	
	MAP	NDCG10	MAP	NDCG10	MAP	NDCG10
Non-stemmed index						
QL[1]	26.89	40.41	22.73	41.50	17.88	17.75
SDM[2]	28.48	42.37	23.74	43.03	18.83	19.61
SegSVM[3]	27.66	41.44	23.38	42.02	18.30	19.76
RM[4]	28.73	39.46	26.61	41.96	17.84	18.82
PRM1[5]	29.10	41.95	26.80	43.28	17.58	19.40
PRM2[6]	29.95	42.74	**26.99**	43.57	18.28	20.00
PSG[7]	26.89	40.41	22.95	41.81	17.88	17.75
PSGRM[8]	27.26	41.08	24.14	40.36	18.46	18.84
QDist(doc)	31.09 $^{1,2,3,4}_{5,7,8}$	45.36$^{1,2,3,4}_{5,7,8}$	25.76$^{1,2,3}_{7,8}$	**43.85**$^{1,3}_{7,8}$	20.23$^{1,3,4}_{5,7}$	23.76$^{1,2,3,4}_{5,7,8}$
QDist(doc+psg)	**32.02** $^{1,2,3,4}_{5,6,7,8}$	**46.27**$^{1,2,3,4}_{5,6,7,8}$	26.16$^{1,2,3}_{7,8}$	43.84^{1,7,8}	**20.28**^{1,3,7}	**25.22**$^{1,2,3,4}_{5,6,7,8}$
Porter-stemmed index						
QL[1]	29.27	40.68	24.98	42.08	17.70	16.24
SDM[2]	32.41	44.81	26.78	44.53	19.73	18.17
SegSVM[3]	31.42	43.54	26.49	44.93	18.87	17.74
RM[4]	31.52	39.54	27.95	42.40	17.31	15.88
PRM1[5]	31.35	40.67	**28.57**	44.26	17.33	17.51
PRM2[6]	31.52	39.95	27.94	43.25	17.74	16.05
PSG[7]	29.50	41.66	25.47	42.41	17.49	16.80
PSGRM[8]	29.73	40.96	27.04	41.12	18.10	16.46
QDist(doc)	33.31 $^{1,3,4,5}_{6,7,8}$	46.07 $^{1,4,5}_{6,7,8}$	27.48 ^{1,7}	45.13 $^{1,4}_{7,8}$	19.53	19.96$^{1,3,4}_{6,7,8}$
QDist(doc+psg)	**33.98**$^{1,2,3,4}_{5,6,7,8}$	**46.08**$^{1,4,5}_{6,7,8}$	27.88 ^{1,2,7}	**45.20** $^{1,4}_{7,8}$	**20.13**$^{1,4,5}_{6,7}$	**21.56**$^{1,2,3,4}_{5,6,7,8}$

reformulated queries instead of just the original query, which is especially helpful in "noisy" collections such as ClueWeb.

Another observation is about the nonstemmed index and the Porter-stemmed index. In general, QDist provides more improvements over baselines using the nonstemmed index than using the Porter-stemmed one. It is not difficult to understand, since some of the effect of using the reformulated queries, especially morphologically similar queries is provided by the Porter stemmer. However, using the Porter-stemmed index is not always better than using the non-stemmed index, especially for large Web collections. For example, the best MAP and NDCG10 on ClueWeb and the best NDCG10 on Gov2 are achieved using the nonstemmed index and considering the stemming as part of the query reformulation. This observation is consistent with Peng et al. [2007].

The second experiment is conducted to further analyze the query distribution model. Specifically, we consider two additional baselines. The first baseline is to use the best reformulated query (except the original one) in the learned query distribution, which is denoted as "single." Compared with Seg-SVM, that is, the variant SRQ method used in the previous experiment, "single" is an exact SRQ method and it uses the same set of reformulated queries as QDist. Thus, the comparison between "single" and QDist will focus on the effect of using the query distribution or a single best reformulated query.

Table XII. Further Analysis of Query Distribution. QDist Denotes QDist(doc)

	Gov2		Robust04		ClueWeb	
	MAP	NDCG10	MAP	NDCG10	MAP	NDCG10
Non-stemmed index						
QL	26.89	40.41	22.73	41.50	17.88	17.75
SDM	28.48	42.37	23.74	43.03	18.83	19.61
RM	28.73	39.46	26.61	41.96	17.84	18.82
single	23.26	35.83	19.37	36.34	13.29	17.85
equal	28.21	43.53	24.62	42.32	17.40	22.71
QDist	31.09	45.36	25.76	43.85	20.23	23.76
Porter-stemmed index						
QL	29.27	40.68	24.98	42.08	17.70	16.24
SDM	32.41	44.81	26.78	44.53	19.73	18.17
RM	31.52	39.54	27.95	42.40	17.31	15.88
single	25.88	37.21	22.33	39.22	15.11	17.57
equal	31.31	44.87	26.55	44.02	16.97	21.09
QDist	33.31	46.07	27.48	45.13	19.53	19.96

The second baseline assigns equal probabilities to all reformulated queries in the distribution, which is denoted as "equal." The comparison between "equal" and QDist helps show the effect of the probability estimation method used by QDist. Table XII shows the results, where QDist(doc) is used as the query distribution method and QL, SDM and RM are used as references.

Table XII shows that as with Seg-SVM, "single" is worse than QDist, which clearly indicates that using the whole query distribution is much better than using the single reformulated query. "single" is even worse than QL, which supports the observations of previous research [Bendersky and Croft 2008] that it is important to combine the original query and the reformulated query to achieve good performance on TREC collections. "equal" is better than QL, comparable with SDM and RM, but it is still worse than QDist. This shows that the query probability estimation method proposed in this article is effective. In addition, it is expensive to use the "equal" method, since it has to use all reformulated queries generated for retrieval. When the number of reformulated queries is big, this causes considerable computational cost. In contrast, QDist assigns reasonable probabilities to reformulated queries, thus it is easy to pick the top ranked reformulated queries for retrieval in practice. The effect of the number of reformulated queries chosen is explored in the fourth experiment.

In the third experiment, we further test the performance of the query distribution model when it is trained and tested on different collections. This experiment helps explore the stability of the query distribution model when the test collection is different from the training collection. The results are shown in Table XIII.

Table XIII shows that QDist(doc) that is trained and tested on different collections performs as well as the model that is trained and tested on the same collection and sometimes the former even outperforms the latter. For example, on the Gov2 collection, QDist(doc) trained on ClueWeb performs better than the model directly trained on this collection.

In the fourth experiment, the effect of using the top ranked reformulated queries in the learned query distribution is shown in Figure 5. QDist(doc) is used to represent the query distribution method. The results are reported using the nonstemmed index. Similar results are observed by using the Porter-stemmed index.

Figure 5 shows that, on all collections, the performance of using the top ten reformulated queries is already very close to the performance of using the whole distribution.

Table XIII. The Performance of QDist(doc) when it is Trained and Tested on Different Collections

	Gov2		Robust04		ClueWeb	
	MAP	NDCG10	MAP	NDCG10	MAP	NDCG10
Non-stemmed index						
QL	26.89	40.41	22.73	41.50	17.88	17.75
SDM	28.48	42.37	23.74	43.03	18.83	19.61
RM	28.73	39.46	26.61	41.96	17.84	18.82
QDist(doc)						
trained on Gov2	31.09	45.36	25.79	43.50	19.08	24.74
trained on Robust04	30.39	44.01	25.76	43.85	19.32	24.70
trained on ClueWeb	31.83	46.33	25.37	43.30	20.23	23.76
Porter-stemmed index						
QL	29.27	40.68	24.98	42.08	17.70	16.24
SDM	32.41	44.81	26.78	44.53	19.73	18.17
RM	31.52	39.54	27.95	42.40	17.31	15.88
QDist(doc)						
trained on Gov2	33.31	46.07	27.06	44.23	19.32	21.68
trained on Robust04	32.36	44.92	27.48	45.13	19.12	20.79
trained on ClueWeb	33.73	46.56	26.73	44.00	19.53	19.96

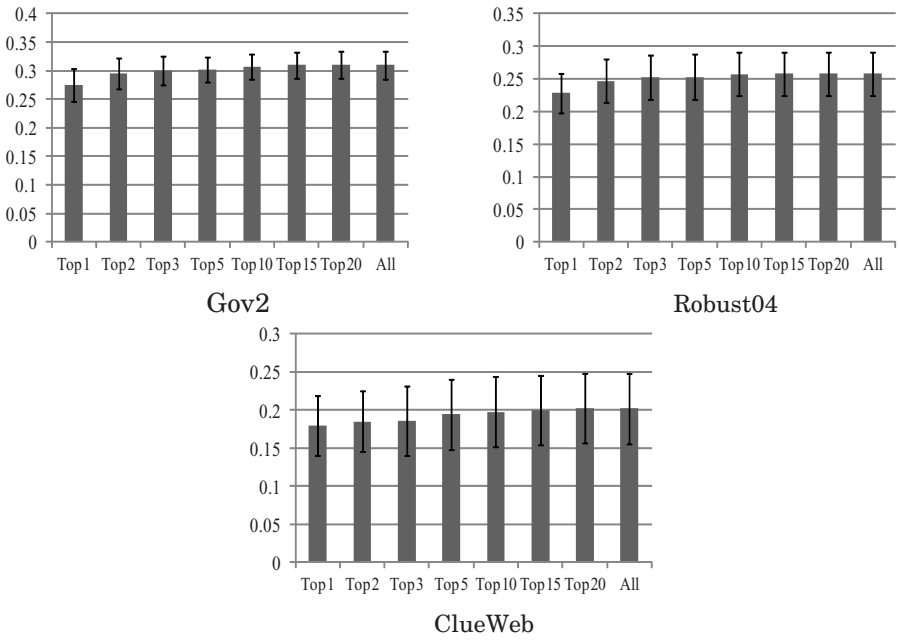

Fig. 5. The effect of the number of reformulated queries. x-axis is the number of top ranked reformulated queries and y-axis is MAP. The 95% confidence interval of the ten-fold cross validation is also provided.

Table XIV.
The performance of top three queries in QDist(doc) using different types of features. p denotes PSG, n denotes NGRAM and o denotes OPER. N10 denotes NDCG10.

	Gov2		Robust04		ClueWeb	
	MAP	NDCG10	MAP	NDCG10	MAP	NDCG10
Non-stemmed index						
p	**30.36**	43.35	24.52	42.42	19.18	20.21
n	29.45	42.59	24.93	43.49	19.40	21.08
o	27.86	41.12	24.67	42.83	18.81	21.27
p+n	30.29	**43.45**	25.16	**43.88**	**19.57**	**22.04**
p+o	30.14	43.37	24.61	42.62	18.43	21.69
n+o	28.63	41.83	25.22	43.69	18.60	20.81
all	30.06	42.69	**25.23**	43.83	18.62	21.54
Porter-stemmed index						
p	31.94	43.01	25.90	43.01	18.60	18.09
n	31.93	42.72	26.64	44.36	19.14	18.76
o	30.98	42.74	26.75	44.57	19.11	18.48
p+n	32.02	42.70	26.73	44.35	19.07	**19.23**
p+o	32.06	**43.76**	26.59	44.17	18.86	18.48
n+o	31.36	42.84	27.00	**44.70**	**19.19**	18.44
all	**32.10**	43.00	**27.01**	44.58	18.91	18.76

The fifth experiment explores the effect of different types of features (PSG, NGRAM, OPER, see Table IV). We are particularly interested in their effects on picking the top ranked queries. The performance of the top three reformulated queries in QDist(doc) using different types of features is reported in Table XIV. The best performance is bolded.

Table XIV shows that the best feature type is not consistent over collections. Generally, combining different types of features together is more effective than using a single type of feature. In addition to the top three reformulated queries, we also explore other numbers of top ranked reformulated queries. In general, when more reformulated queries are used, the performance of using different feature sets becomes more similar.

6.3. Failure Analysis

Although the query distribution model generally performs well, some failures are also observed. Table XV shows three typical examples where the query distribution model does not perform well. For the query "pyramid scheme," several reformulated queries are generated using the Wikipedia redirect pages such as "1-up system," "women helping women," and "peoples program." Although these reformulated queries seem reasonable, they do not help for retrieval, sometimes even decrease the performance. For the query "church arson," the segmented queries "#1(church arson)" and "#1(national church arson)" treat the whole query as a phrase. The strict restrictions of these queries decrease the retrieval performance. For the query "hubble telescope repairs", the reformulated queries "#1(hubble space telescope) repairs" and "hubble space telescope repairs" add a new word "space", but this query expansion does not help improve the original query.

7. RELATED WORK

Query reformulation has been an important topic in the information retrieval area for a long time. Recently, it has attracted much attention in Web search, since it significantly improves Web users' search experience. However, to the best of our knowledge, there is

Table XV.
Examples where the query distribution model does not perform well. Top ranked reformulated queries (Q_r) are displayed. In the query distribution, the original query (Q) is italicized. Average Precision (AP) is reported as the retrieval performance.

$P(Q_r\|R)$	Reformulated Query (Q_r)	AP
Q: pyramid scheme		53.94
$QDist(doc)$		52.10
0.2676	#combine(*pyramid scheme*)	53.94
0.0799	#combine(1 up system)	0
0.0748	#combine(women helping women)	0
0.0688	#combine(peoples program)	0
Q: church arson		47.19
$QDist(doc)$		32.64
0.2546	#combine(*church arson*)	47.19
0.1093	#1(church arson)	19.57
0.0457	#1(national church arson)	5.91
0.0446	#combine(national arson church)	41.12
Q: hubble telescope repairs		38.17
$QDist(doc)$		26.49
0.1465	#combine(hubble telescope repair)	42.08
0.1434	#combine(*hubble telescope repairs*)	38.17
0.1064	#combine(#1(hubble space telescope) repairs)	24.03
0.1064	#combine(hubble space telescope repairs)	37.66

little work that models reformulation as a query distribution, where the probabilities of queries are learned by directly optimizing the retrieval performance. This article significantly extends our previous work [Xue and Croft 2010; Xue et al. 2010; Xue and Croft 2011], where Xue and Croft [2010] introduced the concept of query distribution, Xue et al. [2010] proposed generating reformulated queries using the passage analysis technique and Xue and Croft [2011] briefly mentioned how to transform the query features into the retrieval features and learn the parameters using the learning-to-rank techniques.

In this section, we will first review previous research in the information retrieval area and then introduce some recent work in Web search. Previous work on learning to rank is also mentioned. Finally, we describe some closely related work.

7.1. Query Reformulation in Information Retrieval

In the information retrieval area, the studies on query reformulation can be roughly classified as adding new words to the original query and extracting phrases or proximity features from the original query.

Adding new words to the original query is usually known as Query Expansion. Query expansion and term reweighting can be tracked back to some very early work for the vector space model [Rocchio 1971; Ide 1971] in the 1970s. This early work required relevance feedback from the user. The following studies attempted to expand the original query automatically. Crouch and Yang [1992] and Qiu and Frei [1993] built a thesaurus from the collection and used it to add words related to the original query words. Since the relationships between words are extracted from the whole corpus, this type of technique is often referred to Global Analysis. Xu and Croft [2000] noticed that the expansion words were more helpful if they were related to the whole query context instead of a single query word. They proposed the Local Context Analysis model where the expansion words were extracted from the top ranked documents

after running the original query. This type of Local Analysis technique is also called Pseudo Relevance Feedback. Lavrenko and Croft [2001] and Zhai and Lafferty [2001a] revisited pseudo-relevance feedback in the language modeling framework [Ponte and Croft 1998; Zhai and Lafferty 2001b]. Lavrenko and Croft [2001] proposed a framework for estimating the relevance model, that is, the language model of the relevant class, and the estimated relevance model can be naturally used for query expansion. Zhai and Lafferty [2001a] assumed the language models of the top ranked documents were mixtures of the relevance model and the background model and estimated the relevance model using Expectation-Maximization algorithm. Metzler and Croft [2007] proposed a Latent Concept Expansion model that added both words and phrases to the original query. Recently, Cao et al. [2008] noticed that many expansion terms returned by the pseudo relevance feedback model were actually unrelated to or even harmful for the original query. According to those observations, they proposed a classifier to help decide the good expansion terms. Xu et al. [2009] proposed a query dependent pseudo relevance feedback model that exploited the information from Wikipedia in different ways according to the query types. Lang et al. [2010] proposed the Hierarchical Markov random fields (HMRFs) to improve the Latent Concept Expansion model [Metzler and Croft 2007] by considering the hierarchical structure within documents. Lv and Zhai [2010] proposed the Positional Relevance Model to capture the term position and proximity in feedback documents.

Metzer and Croft [2005] detected phrases from the original query and proposed a sequential dependency model to combine the evidence from query words and query phrases. Bendersky et al. [2010] extended the sequential dependency model by learning different weights for words and phrases. Svore et al. [2010] studied the use of proximity and phrase information to improve Web search accuracy.

In general, previous research added different query terms such as words, phrases and proximity features to the original query, but did not consider how these words and phrases would fit together to form actual queries. In the proposed query distribution framework, a reformulated query is considered as the basic unit, where the important dependencies imposed by actual queries are explicitly modeled.

7.2. Query Reformulation in Web Search

Recently, query reformulation has attracted more attention from the Web search area, since it significantly improves Web users' search experiences. Different reformulation operations have been studied such as query segmentation, query substitution and query deletion.

Bergsma and Wang's work [2007] is among the earliest on query segmentation on the Web. They trained a SVM classifier to make a decision for each segmentation position using several types of features. Tan and Peng [2008] proposed an unsupervised query segmentation method using a concept-based language model. The parameters of the language model were estimated by the EM algorithm conducted on the partial corpus specific to the query.

Jones et al. [2006] first provided a clear definition for query substitution on the Web. In their work, a substitution pair is generated from two successive queries that share some common parts. Then, a linear regression classifier is trained to decide the quality of each pair. Wang and Zhai [2008] mined the query log to find potential query term substitution and addition patterns. Their basic idea is that similar words would have similar neighborhood words in query logs. Dang and Croft [2010] reimplemented and tested Wang and Zhai's method for TREC collections. The results showed this method does not work well for the well-formed TREC queries. They also showed that anchor text can be used as a good substitute for real query logs.

Jones et al. [2003] proposed a model to predict which word should be removed from the original query in order to increase the query coverage in the Web search.

Peng et al. [2007] proposed a context-sensitive stemming method for Web queries, where query words are stemmed based on the analysis of their context. Stemming can be considered as a special case of query substitution, since it replaces original query words with their roots.

Guo et al. [2008] proposed a CRF-based model for query refinement, which combined several tasks like spelling correction, stemming and phrase detection. Within their model, different tasks can be solved simultaneously instead of sequentially. Their model mainly focused on morphological changes of the query words such as spelling correction and stemming, but did not consider query substitution.

Few researchers have considered different query reformulation operations from a unified view and studied their effect on improving retrieval performance. In the proposed query distribution framework, a reformulated query is the output of applying a single or multiple operations. Thus, the properties of the reformulated queries generated by different operations can be estimated within the same framework. Moreover, the retrieval model is an integrated part of the proposed framework.

7.3. Learning to Rank

Learning to Rank techniques have attracted considerable attention recently. The basic idea of this approach is to describe each query-document pair with a set of features and then construct a training set using previously seen queries and their relevance judgments. A ranking model is learned using machine learning techniques. The learned model can then be used to rank documents for the unseen queries. Several learning to rank methods have been proposed, such as RankSVM [Herbrich et al. 2000], RankBoost Freund et al. [2003] and RankNet [Burges et al. 2005]. These methods optimize the ranking loss for a pair of documents, thus they are called as the pairwise methods. Cao et al. [2007] extended the pairwise methods to the listwise methods that considered a list of objects as the "instance" of the ranking model and designed a neural network based method called ListNet. AdaRank [Xu and Li 2007] was another listwise method that used boosting for learning and had the property of directly optimizing the retrieval performance. Qin et al. [2008] further considered the relationships between documents and proposed a method to learn the retrieval scores of a list of documents simultaneously.

Previous research on learning to rank focuses on how to learn good parameters for a set of retrieval features in order to optimize the retrieval performance. In the query distribution framework, we study how to estimate parameters for query features. We transform each query feature into a corresponding retrieval feature and estimate the parameters using the state-of-the-art learning to rank techniques.

7.4. Some Recent Work

Collins-Thompson [2007, 2008] studied the uncertainty in information retrieval. In Collins-Thompson [2008], a Monte Carlo integration framework is proposed to map the original query as a distribution of query variants, where the retrieval score is also a weighted sum over these query variants. In Collins-Thompson and Callan [2007], each query variant is represented as an aggregated feedback language model, which characterizes a distribution of feedback language models obtained by resampling the top ranked documents. Although our work is conceptually similar to Collins-Thompson's work, several differences exist. First, since each query variant in his work is represented as a feedback language model, the final query representation after combining all query variants is still a "bag-of-words" style that falls into the category of the concept distribution discussed in this article. Second, the query variants generated in his

work are not actual user queries, considering the query resampling methods used in his work. In contrast, an actual user query is considered as the basic unit in our query distribution model. Third, we estimate the probability of each reformulated query by directly optimizing the retrieval performance.

Soskin et al. [2009] explored integrating multiple relevance models using the "faithfulness" score that measures to what extend the original information need is satisfied by a relevance model. Lv et al. [2011] proposed a boosting approach for pseudo-relevance feedback that directly optimizes a loss function considering both effectiveness and robustness. Similar to Collins-Thompson and Callan [2007], these approaches can be considered as a combination of several feedback language models, thus the final query representation is still a "bag of words."

Sheldon et al. [2011] studied merging the search results of query reformulations. They proposed a supervised merging algorithm called λ-merge to directly optimize a retrieval metric. Our work, independently developed in the same period of their work, shared several similar insights. However, we focus on developing a better query representation, while they consider merging the search results. Furthermore, our implementation uses only publicly available resources, while their model uses proprietary resources from a commercial search engine.

Cummins et al. [2011] analyzed the query space consisting of queries generated from TREC topic descriptions and used the query quality predictors as an estimation of the query generation probability.

8. DISCUSSION

The most interesting part of this article is providing a new perspective for query reformulation, which models the underlying information need as a distribution of actual user queries. However, since large scale query logs are not available on TREC collections, we have to develop some ways to simulate the effect of actual queries. Thus, it is better to consider the current passage-analysis based implementation as a validation of the query distribution idea using the available resources, than another query expansion or reformulation model, among many others, using the passage information. It is interesting to further explore the differences between a reformulated query generated using the passage analysis and an actual query in future work.

The efficiency of the current passage analysis technique is almost comparable with the pseudorelevance feedback approach. Specifically, instead of just constructing the language models for the top ranked documents in pseudorelevance feedback, we further analyze the passages of the top ranked documents to generate query substitutions and segmentations. The passage analysis technique only scans the top ranked documents once, which is the same as pseudo-relevance feedback.

For retrieval, instead of submitting multiple reformulated queries to the search system, the query distribution model, especially QDist(doc), can be optimized according to Equation (8). Since the reformulated queries share many common words and phrases, we only calculate the retrieval scores of these words and phrase once and reuse them for different queries.

9. CONCLUSION

In order to capture the dependencies of query words and phrases, a novel reformulation framework is proposed in this article, where the original query is reformulated as a distribution of queries. Experiments on TREC collections show that this model significantly improves retrieval performance on both Web corpora and a newswire corpus.

Currently, we study the dependencies within a reformulated query. How to capture the dependencies between reformulated queries will be an important future issue. Also, this article focuses on short title queries. It is promising to apply the query distribution

model for verbose queries. Furthermore, connecting the probability estimation for reformulated queries with relative query performance prediction is also an interesting future problem.

REFERENCES

Bendersky, M. and Croft, W. B. 2008. Discovering key concepts in verbose queries. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*. 491–498.

Bendersky, M., Metzler, D., and Croft, W. B. 2010. Learning concept importance using a weighted dependence model. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM '10)*. 31–40.

Bendersky, M., Smith, D. A., and Croft, W. B. 2009. Two-stage query segmentation for information retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09)*. 810–811.

Bergsma, S. and Wang, Q. I. 2007. Learning noun phrase query segmentation. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL '07)*. 819–826.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. 2005. Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML '05)*. 89–96.

Byrd, R. H., Nocedal, J., and Schnabel, R. B. 1994. Rrepresentations of quasi-newton matrices and their use in limited memory methods. *Math. Program. 63*, 2, 129–156.

Cao, G., Nie, J. Y., Gao, J., and Robertson, S. 2008. Selecting good expansion terms for pseudorelevance feedback. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*. 243–250.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the International Conference on Machine Learning (ICML '07)*. ACM, 129–136.

Collins-Thompson, K. 2008. Robust model estimation methods for information retrieval. Ph.D. thesis, Carnegie Mellon University.

Collins-Thompson, K. and Callan, J. 2007. Estimation and use of uncertainty in pseudo-relevance feedback. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*. 303–310.

Crouch, C. J. and Yang, B. 1992. Experiments in automatic statistical thesaurus construction. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '92)*. 77–88.

Cummins, R., Lalmas, M., Oriordan, C., and Jose, J. 2011. Navigating the user query space. In *Proceedings of the 18th International Conference on String Processing and Information Retrieval*. Springer, 380–385.

Dang, V. and Croft, W. B. 2010. Query reformulation using anchor text. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM '10)*. 41–50.

Freund, Y., Iyer, R. D., Schapire, R. E., and Singer, Y. 2003. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res. 4*, 933–969.

Guo, J., Xu, G., Li, H., and Cheng, X. 2008. A unified and discriminative model for query refinement. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*. 379–386.

Herbrich, R., Graepel, T., and Obermayer, K. 2000. Large Margin Rank Boundaries for Ordinal Regression. MIT Press, Cambridge, MA.

Huang, J., Gao, J., Miao, J., Li, X., Wang, K., Behr, F., and Giles, C. L. 2010. Exploring web scale language models for search query processing. In *Proceedings of the International Conference on World Wide Web (WWW '10)*. ACM, 451–460.

Ide, E. 1971. New experiments in relevance feedback. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, Prentice Hall, Englewood Cliffs, NJ.

Jones, R. and Fain, D. C. 2003. Query word deletion prediction. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '03)*. 435–436.

Jones, R., Rey, B., Madani, O., and Greiner, W. 2006. Generating query substitutions. In *Proceedings of the International Conference on World Wide Web (WWW '06)*. 387–396.

Lafferty, J. and Zhai, C. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the International Conference on Machine Learning (ICML '01)*. 111–119.

LANG, H., METZLER, D., WANG, B., AND LI, J.-T. 2010. Improved latent concept expansion using hierarchical Markov random fields. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM '10)*. 249–258.

LAVRENKO, V. AND CROFT, W. B. 2001. Relevance based language models. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '01)*. 120–127.

LIU, X. AND CROFT, W. B. 2002. Passage retrieval based on language models. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM '02)*. 375–382.

LV, Y. AND ZHAI, C. 2010. Positional relevance model for pseudo-relevance feedback. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*. 579–586.

LV, Y., ZHAI, C., AND CHEN, W. 2011. A boosting approach to improving pseudo-relevance feedback. In *Proceedings of the International Conference on Machine Learning (ICML '11)*. ACM, 165–174.

METZLER, D. AND CROFT, W. B. 2004. Combining the language model and inference network approaches to retrieval. *Inf. Process. Manage. 40*, 5, 735–750.

METZLER, D. AND CROFT, W. B. 2005. A Markov random field model for term dependencies. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '05)*. 472–479.

METZLER, D. AND CROFT, W. B. 2007. Latent concept expansion using Markov random fields. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*. 311–318.

PENG, F., AHMED, N., LI, X., AND LU, Y. 2007. Context sensitive stemming for web search. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*. 639–646.

PONTE, J. M. AND CROFT, W. B. 1998. A language modeling approach to information retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*. 275–281.

PORTER, M. F. 1980. An algorithm for suffix stripping. *Program 14*, 3, 130–137.

QIN, T., LIU, T.-Y., ZHANG, X.-D., WANG, D.-S., XIONG,W.-Y., AND LI, H. 2008. Learning to rank relational objects and its application to web search. In *Proceedings of the International Conference on World Wide Web (WWW '08)*. 407–416.

QIU, Y. AND FREI, H. P. 1993. Concept based query expansion. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '93)*. 160–169.

ROCCHIO, J. J. 1971. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, Prentice Hall, Englewood Cliffs, NJ.

SHELDON, D., SHOKOUHI, M., SZUMMER, M., AND CRASWELL, N. 2011. Lambdamerge: merging the results of query reformulations. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM '11)*. 795–804.

SOSKIN, N., KURLAND, O., AND DOMSHLAK, C. 2009. Navigating in the dark: modeling uncertainty in ad hoc retrieval using multiple relevance models. In *Proceedings of the International Conference on the Theory of Information Retrieval (ICTIR '09)*. 79–91.

SVORE, K. M., KANANI, P. H., AND KHAN, N. 2010. How good is a span of terms?: Exploiting proximity to improve web retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*. 155–161.

TAN, B. AND PENG, F. 2008. Unsupervised query segmentation using generative language models and Wikipedia. In *Proceedings of the International Conference on World Wide Web (WWW '08)*. 347–356.

WANG, L., LIN, J., AND METZLER, D. 2010. Learning to efficiently rank. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*. 138–145.

WANG, X. AND ZHAI, C. 2008. Mining term association patterns from search logs for effective query reformulation. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM '08)*. 479–488.

XU, J. AND CROFT, W. B. 2000. Improving the effectiveness of information retrieval with local context analysis. *ACM Trans. Inf. Syst. 18*, 1, 79–112.

XU, J. AND LI, H. 2007. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*. 391–398.

XU, Y., JONES, G. J., AND WANG, B. 2009. Query dependent pseudo-relevance feedback based onWikipedia.In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09)*. 59–66.

XUE, X. AND CROFT, W. B. 2010. Representing queries as distributions. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10) Workshop on Query Representation and Understanding*. 9–12.

XUE, X. AND CROFT, W. B. 2011. Modeling subset distributions for verbose queries. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '11)*. 1133–1134

XUE, X., CROFT, W. B., AND SMITH, D. A. 2010. Modeling reformulation using passage analysis. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM '10)*. 1497–1500.

ZHAI, C. AND LAFFERTY, J. 2001a. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM '01)*. 403–410.

ZHAI, C. AND LAFFERTY, J. 2001b. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '01)*. 334–342.

Received November 2011; revised June 2012; accepted November 2012