Modeling Reformulation Using Query Distributions

ABSTRACT

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 new queries. In this paper, a novel framework is proposed that models reformulation as a distribution of queries, where each query is a variation of the original query. This approach considers a query as a basic unit and can capture important dependencies between words and phrases in the query. Previous reformulation models are special cases of the proposed framework by making certain assumptions. An implementation of this framework 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

Keywords

Query Reformulation, Query Substitution, Query Segmentation, Passage Analysis, Information Retrieval

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 paper, *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 approach [14] adds new words to the original query, the sequential dependency model [17] adds phrase structure, and the latent concept expansion model [18] adds new term proximity features and words. These types of model do not, however, consider how the new words and phrases can be used together to form queries that are variations of the original query. Instead, the new terms are typically added as a large, possibly weighted, "bag of words" to the original query. By ignoring how these words and phrases are used together in actual queries, important dependencies can be missed.

Other research on web query reformulation, on the other hand, has tended to focus on generating a single new query (e.g. [4][13]) by applying a specific reformulation operation. Different operations have been studied. *Query segmentation* [4] 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* [13] 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". However, little work considers combining these operations from a unified perspective, thus important information about alternative query reformulations is not captured.

In this paper, 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. Since the reformulated query that involves a particular choice of words and phrases is explicitly modeled, this framework captures dependencies between those query components. On the other hand, 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. Furthermore, this framework potentially provides a better way of modeling users' reformulation behavior. During a search session, the user will pose a series of reformulated queries if the original one doesn't work well, thus modeling reformulation as a distribution of related queries instead of a large bag of related words seems more natural.

The implementation of the proposed framework can be divided into two major steps. First, a set of reformulated queries is generated by applying a series of reformulation operations on the original query. Second, the probability is estimated for each reformulated query, thereby creating a query distribution.

For the first step, a passage analysis technique on the target corpus is used to generate reformulated queries. Our previous research [omitted for anonymous review] has shown the effectiveness of this technique. The general idea is based on the observation that passages containing all query words or most of the query words provide a good source of information for reformulating queries. Specifically, passages with all query words provide information about the common ways that people split the original query into different concepts. Similarly, passages only containing some query words indicate possible ways of substituting those missing query words.

For the second step, several important features are extracted to characterize the reformulated query as a whole. Those features are not available in previous reformulation models, especially for "bag of words" models, since they fail to model a reformulated query as a whole. Examples of those features include the chance of observing a reformulated query in the target corpus and in an external web corpus, the probability of observing this reformulated query in query logs, and the operations used 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. 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 paper: 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 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 prove that previous models can be considered as special cases of the proposed framework and we further validate its advantages over previous models using experiments on TREC collections.

The rest of paper is organized as follows: we first introduce the background and then propose our reformulation framework. Following that, we describe a passage analysis technique and a probability estimation approach for implementation. Experimental results are then reported. Finally, we review the related work and conclude the paper.

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 paper, we consider a passage to be a non-overlapped window with fixed word size. **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 $Q = q_1q_2...q_l$. Here, q_i is a query word and l is the length of the query. This paper focuses on short queries, where l is usually not bigger than five. Using our previous example, the original query is "oil industry history".

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. Previous reformulation models can be divided into two categories.

2.1 Distribution of Terms

The first category of previous models is a distribution of terms, which is denoted as **DOT**. Besides the original query words, the terms in this distribution include query phrases [17], new words [14][18], and new phrases [18]. Formally, this category of models is defined as follows.

Given a vocabulary of terms $V_T = \{t_i\}_{i=1}^{|V_T|}$, where t_i is a term, **DOT** transforms the original query into a distribution over V_T, i.e. $\mathbf{P}_T = \{(P(t_i|Q) \;\; t_i)\}_{i=1}^{|V_T|}$, where $P(t_i|Q)$ is the probability assigned to t_i.

When $\mathbf{P}_T$ is used for retrieval, the retrieval score of a document D is calculated in Eq. 1.

$$P(\mathbf{P}_T|D) = \prod_{i=1}^{|V_T|} P(t_i|D)^{P(t_i|Q)} \tag{1}$$

where $P(t_i|D)$ is the probability of generating t_i from the document D.

Next, we describe two typical models of this category.

The **Relevance Model (RM)** [14] limits t_i to the original query words and the new words that are related to the original query. In the relevance model, $P(t_i|Q)$ is estimated by mixing the language models of the top ranked documents. For example, the original query "oil industry history" could be reformulated as "(0.44 industry), (0.28 oil), (0.08 petroleum), (0.08 gas), (0.08 country), (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)** [17] limits t_i to the original query words and the bigrams extracted from the original query Q[1]. In the sequential dependency model, $P(t_i|Q)$ is estimated based on the type of t_i, i.e. word or bigram. In other words, all words have the same probability value and this is also true for bigrams. For example, the original query "oil industry history" could be reformulated as "(0.28 oil), (0.28 industry), (0.28 history), (0.08 (oil industry)), (0.08 (industry history))". Besides the words from the original query, this also includes bigrams such as "oil industry" and "industry history".

[1] The sequential dependency model considers two types of bigrams, ordered and unordered.

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**.

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 $P(Q_r^{\star}|D)$, i.e. 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 (SEG) [4] is the operation of grouping query words into phrases. Given the original query $Q = q_1q_2...q_l$, the segmented query is denoted as $p_1p_2...p_m$. Here, p_i is a phrase, which groups some original query words $q_{j+1}...q_{j+k}$ 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) [13] is the operation of replacing some original query words with new ones. Given the original query $Q = (q_1...q_{i+1}...q_{i+s}...q_l)$, the substituted query is denoted as $q_1...q'_1...q'_t...q_l$, where the original words $q_{i+1}...q_{i+s}$ are replaced with $q'_1...q'_t$. 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, a **Distribution Of Queries (DOQ)** framework is proposed and compared with previous reformulation models.

3.1 Framework

Formally, we first generate a set of reformulated queries $V_{Q_r} = \{Q_{r_i}\}_{i=1}^{|V_{Q_r}|}$, where Q_{r_i} is a reformulated query. Q_{r_i} is the output of applying single or multiple reformulation operations. Then, the original query Q is transformed into a distribution over V_{Q_r}, i.e. $\mathbf{P}_{Q_r} = \{(P(Q_{r_i}|Q)\ \ Q_{r_i})\}_{i=1}^{|V_{Q_r}|}$. $P(Q_{r_i}|Q)$ is the probability corresponding to Q_{r_i}. Note that the original query Q also belongs to V_{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 and how to estimate the probability for each reformulated query. Different strategies can be adopted based on implementations.

Given this query distribution based representation, i.e. $\mathbf{P}_{Q_r} = \{(P(Q_{r_i}|Q)\ \ Q_{r_i})\}_{i=1}^{|V_{Q_r}|}$, the retrieval score of a document is calculated in Eq. 2.

$$P(\mathbf{P}_{Q_r}|D) = \prod_{i=1}^{|V_{Q_r}|} P(Q_{r_i}|D)^{P(Q_{r_i}|Q)} \tag{2}$$

where $P(Q_{r_i}|D)$ is the probability of generating Q_{r_i} from the document D.

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 is first substituted as "petroleum industry history" and then it is segmented as "(petroleum industry)(history)". Then, we estimate the probability for each reformulated query and the original query. Clearly, different reformulation operations are naturally combined within this framework. The final representation generated by DOQ is displayed in Table 1, where the representations generated by previous models are also shown for comparison.

Table 1: Different query representations for the original query "oil industry history"

Model	Output
Distribution Of Terms (DOT)	
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
Distribution Of Queries (DOQ)	
(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))...	

3.2 Comparison of Query Representations

We first compare DOT with DOQ. DOT augments the original query with a bag of new terms but does not consider how to fit these terms together to form actual queries. In contrast, DOQ augments the original query with a set of new queries, which captures the important dependencies between terms. This difference is reflected on the new terms added by these two representations, either directly or through adding queries that contain the new terms. DOT adds a new term t according to its own relationship with the original query Q (i.e. $P(t|Q)$), while DOQ adds a new term according to the relationships between the query containing this term Q_r and the original query Q (i.e. $P(Q_r|Q)$), where considering Q_r as a whole captures dependencies between terms in Q_r. As shown in Table 1, RM (a representative of DOT) assigns high probability for "county" while DOQ does not, since "county" frequently cooccurs with the original query but it is not usually found in reformulated queries. On the other hand, DOQ provides high probability for "industrialized" while RM does not, since "industrialized" can be used in queries such as "(oil)(industrialized)(history)" but it rarely cooccurs with the original query.

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

3.3 Comparison of Retrieval Scores

In this subsection, we further compare the retrieval scores of different models. We show that DOT and SRQ are both special cases of DOQ under certain assumptions.

Claim 1. *DOQ becomes DOT given two assumptions:*
1. the estimation of $P(Q_{r_i}|Q)$ assumes that the terms t in a reformulated query Q_{r_i} are independent given the original query Q, i.e.

$$P(Q_{r_i}|Q) = \prod_{t \in Q_{r_i}} P(t|Q) \prod_{t' \notin Q_{r_i}} (1 - P(t'|Q))$$

2. the estimation of $P(Q_{r_i}|D)$ assumes that the terms t in a reformulated query Q_{r_i} are independent given a document D, i.e. $P(Q_{r_i}|D) = \prod_{t \in Q_{r_i}} P(t|D)$.

Note that in the first assumption we consider not only the terms t appearing in Q_{r_i} but also the terms t' not in Q_{r_i} similar to previous work [21].

We provide a brief proof for Claim 1. Given the second assumption, Eq. 2 can be written as follows:

$$\begin{aligned} P(\mathbf{P}_{Q_r}|D) &= \prod_{i=1}^{|V_{Q_r}|} P(Q_{r_i}|D)^{P(Q_{r_i}|Q)} \\ &= \prod_{i=1}^{|V_{Q_r}|} (\prod_{t \in Q_{r_i}} P(t|D))^{P(Q_{r_i}|Q)} \qquad (3) \\ &= \prod_{j=1}^{|V_T|} P(t_j|D)^{\sum_{Q_r \in \{Q_r|t_j \in Q_r\}} P(Q_r|Q)} \qquad (4) \end{aligned}$$

Eq. 3 is obtained by directly using the second assumption. Since it is reasonable to assume each Q_{r_i} in V_{Q_r} only contains terms from the vocabulary V_T of DOT, we can reorganize Eq. 3 to obtain Eq. 4 by merging the same term t_j together. In Eq. 4, $\{Q_r|t_j \in Q_r\}$ denotes the set of reformulated queries containing t_j. Furthermore, using the first assumption, we can obtain Claim 2.

Claim 2. *$\sum_{Q_r \in \{Q_r|t_j \in Q_r\}} P(Q_r|Q) = P(t_j|Q)$, given $P(Q_r|Q) = \prod_{t \in Q_r} P(t|Q) \prod_{t' \notin Q_r} (1 - P(t'|Q))$.*

The proof of Claim 2 can be found in the Appendix. After applying Claim 2 to Eq. 4, we obtain Eq. 5, which finishes the proof of Claim 1.

$$P(\mathbf{P}_{Q_r}|D) = \prod_{j=1}^{|V_T|} P(t_j|D)^{P(t_j|Q)} = P(\mathbf{P}_T|D) \qquad (5)$$

Note that the two assumptions in Claim 1 indicate the advantages of DOQ over DOT, where DOQ can consider dependencies between query terms when it estimates the query distribution (Assumption 1) and when it retrieves documents (Assumption 2).

Second, we will show SRQ is also a special case of DOQ.

Claim 3. *DOQ becomes SRQ given one assumption:*
1. the query distribution assigns all probability to $Q_r^\star$ and assigns zero probability to other reformulated queries in V_{Q_r}

Given the assumption of Claim 3, Eq. 2 can be written as follows:

$$\begin{aligned} P(\mathbf{P}_{Q_r}|D) &= \prod_{i=1}^{|V_{Q_r}|} P(Q_{r_i}|D)^{P(Q_{r_i}|Q)} \\ &= (\prod_{Q_{r_i} \neq Q_r^\star} P(Q_{r_i}|D)^0) \cdot P(Q_r^\star|D)^1 \\ &= P(Q_r^\star|D) \qquad (6) \end{aligned}$$

Eq. 6 finishes the proof of Claim 3.

4. GENERATING REFORMULATED QUERIES

In this section, we will briefly describe how to generate reformulated queries through analyzing passages extracted from the target corpus. More details can be found in our previous work [omitted for anonymous review]. Following that, we will describe how to further incorporate the retrieval model to generate operational search queries.

4.1 Query Generation with Passage Analysis

First, the original query is substituted to generate candidate queries. In order to replace some query words, passages containing the rest of the query are extracted. Three different methods are developed to analyze these passages which provide possible ways of query substitution. **Morphologically Similar Words (Morph)** are a reliable way to substitute the original query words using appropriate morphological variants. For example, "industrialized" is morphologically similar to "industry" and also "oil industrialized history" is observed in many extracted passages, thus "oil industrialized history" is considered as a substitution of "oil industry history". 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. Two types of patterns are considered here, **Adding-Word Patterns (Pat-add)** and **Changing-Word Patterns (Pat-chg)**. For example, "oil ⋆ industry history" is an adding-word pattern extracted from the original query "oil industry history" and the substitution "oil and gas industry history" can be found after applying this pattern to those extracted passages. Similarly, "oil ⋆ history" is a changing-word pattern and another substitution "oil spill history" can be obtained through using this pattern. Some query substitutions are difficult to obtain only relying on corpus information, thus the third method uses **Wikipedia Redirect Page (Wiki)**, which is designed to maintain alternative expressions for rediction[2]. For example, "oil industry" and "petroleum industry" form an alternative pair and "petroleum industry history" has been observed in many extracted passages, thus it is considered as a substitution.

Second, the candidate queries generated from the previous step are further segmented to generate the final reformulated queries. 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. For example, the candidate query is "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...", we obtain a set of query components {"history", "oil and gas industry", "oil"}. Since "oil" is a substring of "oil and gas industry", "oil" is removed from this set. Finally, a segmentation is recovered from this set as "(oil and gas industry)(history)".

More examples of the reformulated queries generated are shown in Table 2. "Orig" denotes the segmentation of the original query without substitution.

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

Table 2: Examples of reformulated queries.

	embryonic stem cells	nuclear reactor types	increase mass transit	abandoned mine reclamation
Orig	(embryonic stem cells)	(nuclear reactor)(types)	(increase)(mass transit)	(abandoned mine reclamation)
Wiki	(embryonic stem cell)	(nuclear reactor)(technology)(types)	(increase)(public transport)	(mining)(reclamation)
Morph	(embryos)(stem cells)	(nuclear reactor)(type)	(increased)(mass transit)	(abandoned mines)(reclamation)
Pat-add	(embryonic stem es cells)	(nuclear power reactor)(types)	n/a	(abandoned mine land reclamation)
Pat-chg	(embryonic germ cells)	(nuclear fuel types)	(increase public transit)	(abandoned land reclamation)

4.2 Indicating Retrieval Models

In the previous subsection, we generated a set of *conceptual* reformulated queries ($V^c_{Q_r}$) that are understandable to users, but what retrieval models should be applied to those queries are still unclear. In this subsection we produce *operational* reformulated queries by explicitly indicating the retrieval models used. For example, given the *conceptual* reformulated query "(petroleum industry)(history)", users can easily understand its meaning, i.e. treating "petroleum industry" as a phrase and "history" as a word, but it is not clear how search engines should implement this query.

Indicating the retrieval model corresponds to how to estimate $P(Q_r|D)$. Recall that the second assumption of Claim 1 indicates that the proposed Distribution Of Queries (DOQ) model has the ability of capturing the dependencies of queries terms during the retrieval step. Thus, two types of retrieval models are considered: the document-level model and the passage-level model.

The document-level model assumes the query terms in the reformulated queries are independent given the document. Specially, $P(Q_r|D)$ is estimated in Eq. 7.

$$P(Q_r|D) = \prod_{t \in Q_r} P(t|D) \tag{7}$$

where $P(t|D)$ is estimated by the language modeling approach [21][28].

The passage-level model considers the dependencies between query terms in a reformulated query by preferring documents where the whole query is observed within a passage. Specifically, $P(Q_r|D)$ is estimated in Eq. 8.

$$P(Q_r|D) = \frac{\#psgN(Q_r, D)}{\#psgN(D)} \tag{8}$$

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. Note that $\#psgN(Q_r, D)$ can be easily collected in the previous step where the conceptual reformulated queries are generated. This maximal likelihood estimation (Eq. 8) can be smoothed with the background model. Based on the value of N, different passage-level models can be generated.

The Indri query language [16] provides an implementation of these retrieval models. For example, using the Indri query language, the operational queries for a conceptual query "(petroleum industry)(history)" are displayed in Table 3 by using the document-level model and the passage-level model, respectively. In this query language, the operator "#combine" is an implementation of Eq. 7 and the operator "#uw$N$" is an implementation of Eq. 8 where N is the passage size. "#1" is an operator for a phrase.

Based on the choice of retrieval models, the set of operational reformulated queries are different. $V^d_{Q_r}$ and $V^p_{Q_r}$ denote the set of operational queries using the document-level model and the passage-level model, respectively. Table 4 shows $V^d_{Q_r}$ and $V^p_{Q_r}$ for "oil industry history". The corresponding conceptual reformulated queries ($V^c_{Q_r}$) can be found in Table 1. The final set of reformulated queries V_{Q_r} could be $V^d_{Q_r}$, or $V^p_{Q_r}$, or the union of both according to the retrieval models used. The union of both is used when the document-level and the passage-level models are combined.

Table 3: The Indri queries for "(petroleum industry)(history)"

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

Table 4: The set of operational reformulated queries for the original query "oil industry history"

document-level model ($V^d_{Q_r}$)
#combine(oil industry history), #combine(#1(oil industry) history), #combine(#1(petroleum industry) history), #combine(#1(oil and gas industry) history), #combine(oil industrialized history),...
passage-level model ($V^p_{Q_r}$)
#uw20(oil industry history), #uw20(#1(oil industry) history), #uw20(#1(petroleum industry) history), #uw20(#1(oil and gas industry) history), #uw20(oil industrialized history),...

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 features used to characterize each reformulated query.

5.1 Model

In order to estimate the probability for each reformulated query, we assume it is a linear combination of their feature values as shown in Eq. 9.

$$P(Q_r|Q) = \sum_k \lambda_k f_k(Q_r) \tag{9}$$

where $f_k(Q_r)$ denotes the feature value extracted to characterize a reformulated query Q_r and λ_k is the parameter corresponding to f_k. Similar assumptions have been made in previous work [2, 24].

Using Eq. 9, Eq. 2 can be rewritten as follows:

$$\begin{aligned} & \log(P(\mathbf{P}_{Q_r}|D)) \\ = & \sum_{i=1}^{|V_{Q_r}|} P(Q_{r_i}|Q) \log(P(Q_{r_i}|D)) \\ = & \sum_{i=1}^{|V_{Q_r}|} \sum_k \lambda_k f_k(Q_{r_i}) \log(P(Q_{r_i}|D)) \\ = & \sum_k \lambda_k \sum_{i=1}^{|V_{Q_r}|} f_k(Q_{r_i}) \log(P(Q_{r_i}|D)) \\ = & \sum_k \lambda_k F_k(V_{Q_r}, D) \end{aligned} \tag{10}$$

Table 5: 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

where $F_k(V_{Q_r}, D) = \sum_{i=1}^{|V_{Q_r}|} f_k(Q_{r_i}) \log(P(Q_{r_i}|D))$ is the combination of the retrieval scores using each reformulated query weighted by the feature f_k. $F_k(V_{Q_r}, D)$ can be considered as the retrieval feature extracted from the document D and a set of reformulated queries V_{Q_r}. Each reformulated query feature f_k corresponds to a retrieval feature F_k and they share the same parameter λ_k. Therefore, the parameters λ_k can be learned by optimizing the performance of the retrieval model (Eq. 10), which is a linear combination of retrieval features F_k.

A learning to rank approach (ListNet [8]) is used to learn the parameters, since it considers the same form of retrieval models as Eq. 10. Specifically, the parameters are learned by optimizing the cross entropy between the estimated document distribution ($P(D|P_{Q_r})$) and the underlying document distribution ($P^\star(D|Q)$), where $P(D|P_{Q_r})$ is estimated with the help of Eq. 10 and $P^\star(D|Q)$ is estimated based on the relevance judgments of Q.

In this paper, a variation of ListNet is considered. Instead of using the neural network, a limited-memory version of BFGS [5] is used for optimization due to its efficiency.

5.2 Features

Recall that the first assumption of Claim 1 indicates that the proposed DOQ model has the advantage of considering dependencies between query terms during the estimation of $P(Q_r|Q)$. Thus, features that characterize the reformulated query Q_r as a whole are essential for estimating $P(Q_r|Q)$. Three types of features are considered here.

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. We also extend the passage size to the document length.

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. Thanks to the Web N-gram Services provided by Microsoft [12], the above 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). This type of features help combine different reformulation operations within the same framework.

The above three types of features are summarized in Table 5. psg$N$-count can be instantiated to a variety of features by taking different values of N.

Note that the features discussed in Table 5 are extracted based on the conceptual queries. When document-level and passage-level retrieval models are both used, differentiating $V^d_{Q_r}$ and $V^p_{Q_r}$ becomes a problem, since they are generated from the same set of conceptual queries and thus share the same feature values. The solution is to learn different sets of parameters (λ_k) for each retrieval model. In Eq. 10, λ_k corresponds to the retrieval feature $F_k(V_{Q_r}, D) = \sum_{i=1}^{|V_{Q_r}|} f_k(Q_{r_i}) \log(P(Q_{r_i}|D))$. $F_k(V_{Q_r}, D)$ consists of two parts, i.e. $f_k(Q_{r_i})$ and $\log(P(Q_{r_i}|D))$. Queries in $V^d_{Q_r}$ and $V^c_{Q_r}$ will have the same $f_k(Q_{r_i})$ since they are generated from the same conceptual query, but they have different $\log(P(Q_{r_i}|D)$ values since different retrieval models are used. Therefore, $F_k(V_{Q_r}, D)$ is different for $V^d_{Q_r}$ and $V^p_{Q_r}$. Based on different F_k, different sets of λ_k can be learned for $V^d_{Q_r}$ and $V^p_{Q_r}$.

6. EXPERIMENTS

Two TREC collections (Gov2 and Robust04) are used for experiments. The statistics of each collection are summarized in Table 6. These two collections have different properties. Gov2 is a large web collection with abundant variations of expressions, while Robust04 is a newswire collection using consistent vocabularies.

For each collection, two indexes are built, one not stemmed and the other stemmed with the Porter Stemmer[22]. Stemming is a process of transforming words to their root forms, which can be regarded as a specifical form of query reformulation conducted during the indexing phase. These two indexes help explore the effect of the proposed reformula-

Table 6: TREC collections used in experiments

Name	Docs	Topics
Gov2	25,205,179	701-850
Robust04	528,155	301-450,601-700

Table 7: Example of the query distribution learned on the non-stemmed 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.

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

tion model when some basic reformulation operation (i.e., stemming) is applied or not. No stopword removal is done during indexing. For each topic, the title part is used as the query. The reformulated queries are generated from the non-stemmed index. The query distribution is learned for each type of index respectively and then the retrieval performance of using the learned query distribution on the corresponding index is reported.

Two passage sizes are used in this paper, i.e., 20 and 100, according to [26], which represent a tight window and a loose window, respectively. Correspondingly, two passage-level retrieval models are developed to generate reformulated queries, i.e. "#uw20" and "#uw100" (see Table 3). Similarly, two features, i.e. psg20-count and psg100-count (see Table 5), are extracted to describe a reformulated query.

Two types of query distributions are particularly considered: the first one only contains the reformulated queries using the document-level model ("#combine", see Table 3), which is denoted as **QDist(doc)**; the second one combines the reformulated queries using both the document-level model ("#combine") and the passage-level models ("#uw20" and "#uw100"), which is denoted as **QDist(doc+psg)**.

Several baselines are compared. QL denotes the query likelihood language model [21, 28]. SDM denotes the sequential dependence model [17]. RM denotes the relevance model [14]. The parameters of RM are set according to [27]. Seg-SVM denotes a SVM-based query segmentation method [3], which is trained on a corpus of 500 pre-segmented noun phrases [4]. QL-psg denotes a passage-augmented language model [15] which combines the retrieval score of a document and the retrieval score of the best passage of this document.

The standard performance measures, mean average precision (MAP) and the normalized discounted cumulative gain at 10 (NDCG10), are used to measure the retrieval performance[3]. The two-tailed t-test measures significance.

6.1 Examples

First, we present examples of the query distribution learned by using the document-level retrieval model **QDist(doc)**. Table 7 shows the example on the non-stemmed 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 8: Example of the query distribution learned on the non-stemmed index using the document-level and the passage-level retrieval models.

$P(Q_r\|Q)$ 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

Table 7 shows that the learned query distribution obviously outperforms the original query. In the query distribution, the appropriate probability is assigned to the original query. In most cases, the original query receives the highest probability, since it is safe not 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

[3] In this paper, we report MAP×100 and NDCG10×100.

Table 9: The results of different reformulation models. The best performance is bolded. * denotes significantly different with baselines.

	Gov2				Robust04			
	nonstem		pstem		nonstem		pstem	
	MAP	NDCG10	MAP	NDCG10	MAP	NDCG10	MAP	NDCG10
QL	26.89	40.41	29.27	40.68	22.73	41.50	24.98	42.08
SDM	28.29	41.59	32.40	44.80	23.76	42.90	26.78	44.53
RM	28.72	41.30	31.07	40.64	24.82	42.49	26.71	42.42
Seg-SVM	26.36	38.84	28.90	39.90	22.73	40.80	25.12	42.34
QL-psg	26.52	41.30	29.25	41.02	22.99	41.67	25.59	42.42
QDist(doc)	31.09	45.36	33.31	46.07	25.76	**43.85**	27.48	45.13
vs. QL	*15.6%**	*12.2%**	*13.8%**	*13.2%**	*13.3%**	*5.7%**	*10.0%**	*7.2%**
vs. SDM	*9.9%**	*9.1%**	*2.8%*	*2.8%*	*8.4%**	*2.2%*	*2.6%*	*1.3%*
vs. RM	*8.3%**	*9.8%**	*7.2%**	*13.4%**	*3.8%*	*3.2%*	*2.9%*	*6.4%**
QDist(doc+psg)	**32.02**	**46.27**	**33.98**	**46.08**	**26.16**	43.84	**27.88**	**45.20**
vs. QL	*19.1%**	*14.5%**	*16.1%**	*13.3%**	*15.1%**	*5.6%**	*11.6%**	*7.4%**
vs. SDM	*13.2%**	*11.3%**	*4.9%**	*2.9%*	*10.1%**	*2.2%*	*4.1%**	*1.5%*
vs. RM	*11.5%**	*12.0%**	*9.4%**	*13.4%**	*5.4%**	*3.2%*	*4.4%**	*6.6%**

"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 8 displays the example of this type of query distribution using the non-stemmed index.

Table 8 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).

6.2 Results

The first experiment is conducted to compare the query distribution model with other reformulation models. The Relevance Model (RM) and the Sequential Dependency Model (SDM) represent the Distribution Of Terms (DOT) models. Seg-SVM 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 performance of the language model augmented by the passage information (QL-psg) is also reported. The passage sizes (20 and 100) as used in the query distribution model are considered in QL-psg respectively and better performance is reported for QL-psg. The results are provided in Table 9.

Table 9 shows that the query distribution models outperform both the DOT models (SDM and RM) and the SRQ model (Seg-SVM), which supports the advantages of modeling reformulation as a distribution of queries instead of a distribution of terms and a single reformulated query. When the passage-level retrieval models are incorporated, QDist(doc+psg) further improves QDist(doc). Particularly, QDist(doc+psg) performs significantly better than both SDM and RM for MAP, which are the state-of-the-art reformulation techniques. QL-psg is also worse than QDist methods, which shows that the benefits of QDist come from modeling the original query as a query distribution instead of simply using the passage information.

Table 10: Further analysis of query distribution. N10 denotes NDCG10. QDist denotes QDist(doc).

	Gov2				Robust04			
	nonstem		pstem		nonstem		pstem	
	MAP	N10	MAP	N10	MAP	N10	MAP	N10
QL	26.89	40.41	29.27	40.68	22.73	41.50	24.98	42.08
SDM	28.29	41.59	32.40	44.80	23.76	42.90	26.78	44.53
RM	28.72	41.30	31.07	40.64	24.82	42.49	26.71	42.42
single	23.26	35.83	25.88	37.21	19.37	36.34	22.33	39.22
equal	28.21	43.53	31.31	44.87	24.62	42.32	26.55	44.02
QDist	31.09	45.36	33.31	46.07	25.76	43.85	27.48	45.13

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, i.e. 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. The second baseline is to assign 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 10 shows the results, where QDist(doc) is used as the query distribution method and QL, SDM and RM are used as references.

Table 10 shows that as Seg-SVM, "single" is also 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 [1] 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 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, it causes considerable computational cost. In contrast, QDist assigns appropriate probabilities to reformulated queries, thus it is easy to pick the top ranked reformulated queries for retrieval

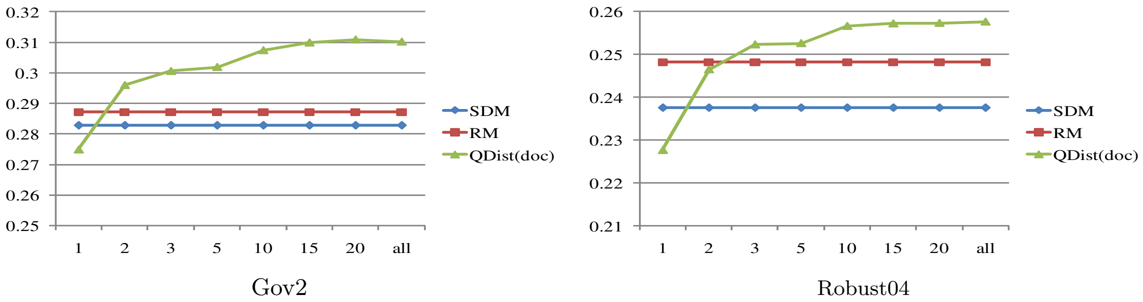

Figure 1: The effect of the number of reformulated queries. Non-stemmed index is used. x-axis is the number of top ranked reformulated queries and y-axis is MAP.

Table 11: 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			
	nonstem		pstem		nonstem		pstem	
	MAP	N10	MAP	N10	MAP	N10	MAP	N10
p	**30.36**	43.35	31.94	43.01	24.52	42.42	25.90	43.01
n	29.45	42.59	31.93	42.72	24.93	43.49	26.64	44.36
o	27.86	41.12	30.98	42.74	24.67	42.83	26.75	44.57
p+n	30.29	**43.45**	32.02	42.70	25.16	**43.88**	26.73	44.35
p+o	30.14	43.37	32.06	**43.76**	24.61	42.62	26.59	44.17
n+o	28.63	41.83	31.36	42.84	25.22	43.69	27.00	**44.70**
all	30.06	42.69	**32.10**	43.00	**25.23**	43.83	**27.01**	44.58

in practice. The effect of the number of reformulated queries chosen is explored in the third experiment.

Fig. 1 displays the effect of using the top ranked reformulated queries in the learned query distribution. On both collections, only using the top three reformulated queries can outperform both RM and SDM. The performance of using the top 10 reformulated queries is already very close to the performance of using the whole distribution. Therefore, QDist can be easily used in practice. Similar results are observed on the Porter-Stemmed index.

The fourth experiment explores the effect of different types of features (PSG, NGRAM, OPER, see Table 5). 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 11. The best performance is bolded.

Table 11 shows that better performance can be achieved when different types of features are combined together. Besides the top three reformulated queries, we also explore other numbers of top ranked reformulated queries. Generally, when more reformulated queries are used, the performance from using different feature sets becomes closer.

7. RELATED WORK

The Relevance Model [14] is representative of a series of reformulation models [18][7][26] that expand the original query with new words. The Sequential Dependency Model [17] is an example of models [19] that augment the query representation with phrases. As discussed, these models focus on adding new query components, words or phrases, but do not consider how to use those components to construct queries.

Many models have been proposed for specific reformulation operations such as segmentation and substitution.

Bergsma and Wang's work [4] is among the earliest on query segmentation. They trained a SVM classifier to make a decision for each segmentation position using several types of features. Tan and Peng [23] 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. Bendersky et al [3] proposed a two-stage query segmentation method.

Jones et al [13] first provided a clear definition for query substitution. 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 [25] 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 [10] re-implemented 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.

Peng et al [20] 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 [11] 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 query segmentation and query substitution from a unified view and reported experimental results on TREC collections.

Collins-Thompson and Callan [9] represented the original query as a distribution of feedback language models. The feedback language model is a distribution of words including original query words and new words and can be estimated similar to Relevance Model. Thus, their method still belongs to the category of Distribution Of Terms.

Our previous work [omitted for anonymous review] proposed improving verbose queries using subset distribution. This is a special case of the reformulation framework proposed in this paper, where only subset queries are considered in the query distribution.

Passage level evidence has been widely used for information retrieval. Callan [6] explored the effect of different types of passages with INQUERY. Liu and Croft [15] studied passage retrieval within the language modeling framework. Xu and Croft [26] proposed Local Context Analysis, a query expansion method, that analyzed passages. However, few researchers have considered using passage level evidence for query reformulation.

8. CONCLUSION

In order to capture the dependencies of query words and phrases, a novel reformulation framework is proposed in this paper, where the original query is reformulated as a distribution of queries instead of a bag of components. Experiments on TREC collections show that this model significantly improves retrieval performance on both the web corpus and the newswire corpus. Currently, we focus on modeling the dependencies within a reformulated query. How to capture the dependencies between reformulated queries will be an interesting future issue.

Acknowledgments

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 the author's and do not necessarily reflect those of the sponsor.

9. REFERENCES

[1] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *SIGIR08*, pages 491–498, Singapore, 2008.

[2] M. Bendersky, D. Metzler, and W. B. Croft. Learning concept importance using a weighted dependence model. In *WSDM10*, pages 31–40, New York City, NY, 2010.

[3] M. Bendersky, D. A. Smith, and W. B. Croft. Two-stage query segmentation for information retrieval. In *SIGIR09*, pages 810–811, Boston, MA, 2009.

[4] S. Bergsma and Q. I. Wang. Learning noun phrase query segmentation. In *EMNLP-CoNLL07*, pages 819–826, Prague, 2007.

[5] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Rrepresentations of quasi-newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(2):129–156, 1994.

[6] J. P. Callan. Passage-level evidence in document retrieval. In *SIGIR94*, pages 302–310, Dublin,Ireland, 1994.

[7] G. Cao, J. Y. Nie, J. Gao, and S. Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *SIGIR08*, pages 243–250, Singapore, 2008.

[8] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML07*, pages 129–136, New York, NY, USA, 2007. ACM.

[9] K. Collins-Thompson and J. Callan. Estimating and use of uncertainty in pseudo-relevance feedback. In *SIGIR07*, pages 303–310, Amsterdan, Netherland, 2007.

[10] V. Dang and W. B. Croft. Query reformulation using anchor text. In *WSDM10*, pages 41–50, New York, NY, 2010.

[11] J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. In *SIGIR08*, pages 379–386, Singapore, 2008.

[12] J. Huang, J. Gao, J. Miao, X. Li, K. Wang, F. Behr, and C. L. Giles. Exploring web scale language models for search query processing. In *WWW10*, pages 451–460, New York, NY, USA, 2010. ACM.

[13] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW06*, pages 387–396, Ediburgh, Scotland, 2006.

[14] V. Lavrenko and W. B. Croft. Relevance based language models. In *SIGIR01*, pages 120–127, New Orleans, LA, 2001.

[15] X. Liu and W. B. Croft. Passage retrieval based on language models. In *CIKM02*, pages 375–382, McLean, VA, 2002.

[16] D. Metzler and W. B. Croft. Combining the language model and inference network approaches to retrieval. *Information Processing and Management*, 40(5):735–750, 2004.

[17] D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *SIGIR05*, pages 472–479, Salvador,Brazil, 2005.

[18] D. Metzler and W. B. Croft. Latent concept expansion using markov random fields. In *SIGIR07*, pages 311–318, Amsterdam, the Netherlands, 2007.

[19] G. Mishne and M. de Rijke. Boosting web retrieval through query operations. In *ECIR05*, pages 502–516, Spain, 2005.

[20] F. Peng, N. Ahmed, X. Li, and Y. Lu. Context sensitive stemming for web search. In *SIGIR07*, pages 639–646, Amsterdam, the Netherlands, 2007.

[21] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR98*, pages 275–281, Melbourne, Australia, 1998.

[22] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[23] B. Tan and F. Peng. Unsupervised query segmentation using generative language models and Wikipedia. In *WWW08*, pages 347–356, Beijing,China, 2008.

[24] L. Wang, J. Lin, and D. Metzler. Learning to efficiently rank. In *SIGIR10*, pages 138–145, Geneva, Switzerland, 2010.

[25] X. Wang and C. Zhai. Mining term association patterns from search logs for effective query reformulation. In *CIKM08*, pages 479–488, Napa Valley, CA, 2008.

[26] J. Xu and W. B. Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems*, 18(1):79–112, 2000.

[27] Y. Xu, G. J. Jones, and B. Wang. Query dependent pseudo-relevance feedback based on Wikipedia. In *SIGIR09*, pages 59–66, Boston, MA, 2009.

[28] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR01*, pages 334–342, New Orleans, LA, 2001.

Appendix

Claim 2. $\sum_{Q_r \in \{Q_r | t_j \in Q_r\}} P(Q_r|Q) = P(t_j|Q)$, given the condition $P(Q_r|Q) = \prod_{t \in Q_r} P(t|Q) \prod_{t' \notin Q_r} (1 - P(t'|Q))$.

Proof. We use the binary variable b_i to denote whether the term t_i appears in Q_r. $b_i = 1$, when t_i is in Q_r and $b_i = 0$ otherwise. Thus, Q_r can be represented as a series of variables $(b_1, b_2, ..., b_{|V_T|})$ that correspondes to each term in V_T. We assume $P(b_i|Q)$ follows a multi-Bernoulli distribution [21][4], i.e. $P(b_i = 0|Q) + P(b_i = 1|Q) = 1$. Then, $P(b_i = 1|Q) = P(t_i|Q)$ and $P(b_i = 0|Q) = 1 - P(t_i|Q)$. Therefore, the condition can be rewritten as $P(Q_r|Q) =$

[4] Claim 2 also holds if multinomial distribution is assumed. The proof is omitted due to space limitation.

$P(b_1...b_j...b_{|V_T|}|Q) = \prod_{k=1}^{|V_T|} P(b_k|Q)$.

$$
\begin{aligned}
& \sum_{Q_r \in \{Q_r | t_j \in Q_r\}} P(Q_r|Q) \\
= & \sum_{b_1 \in \{0,1\}} ... \sum_{b_j = 1} ... \sum_{b_{|V_T|} \in \{0,1\}} P(b_1...b_j...b_{|V_T|}|Q) \\
= & \sum_{b_1 \in \{0,1\}} ... \sum_{b_j = 1} ... \sum_{b_{|V_T|} \in \{0,1\}} \prod_{k=1}^{|V_T|} P(b_k|Q) \\
= & P(b_j = 1|Q) \sum_{b_1 \in \{0,1\}} ... \sum_{b_{|V_T|} \in \{0,1\}} \prod_{k \neq j} P(b_k|Q) \\
= & P(b_j = 1|Q) \prod_{k \neq j} [P(b_k = 0|Q) + P(b_k = 1|Q)] \\
= & P(b_j = 1|Q) \\
= & P(t_j|Q)
\end{aligned}
$$

□