Improving Verbose Queries using Subset Distribution

Xiaobing Xue Samuel Huston W. Bruce Croft
Center for Intelligent Information Retrieval
Computer Science Department
University of Massachusetts, Amherst, MA, 01003, USA
{xuexb,sjh,croft}@cs.umass.edu

ABSTRACT

Dealing with verbose (or long) queries poses a new challenge for information retrieval. Selecting a subset of the original query (a "sub-query") has been shown to be an effective method for improving these queries. In this paper, the distribution of sub-queries ("subset distribution") is formally modeled within a well-grounded framework. Specifically, sub-query selection is considered as a sequential labeling problem, where each query word in a verbose query is assigned a label of "keep" or "don't keep". A novel Conditional Random Field model is proposed to generate the distribution of sub-queries. This model captures the local and global dependencies between query words and directly optimizes the expected retrieval performance on a training set. The experiments, based on different retrieval models and performance measures, show that the proposed model can generate high-quality sub-query distributions and can significantly outperform state-of-the-art techniques.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Performance

Keywords

Sub-query selection, verbose query, Conditional Random Field, Information Retrieval

1. INTRODUCTION

The use of verbose (or long) queries helps users to express their information need naturally and saves efforts in choosing keywords. Unfortunately, previous work [1, 11] has shown that current search engines cannot handle verbose queries well. It seems that the additional information provided in verbose queries is more likely to confuse current search engines rather than help them. Thus, dealing with verbose queries poses a new challenge for information retrieval.

Previous work on verbose queries has been in two main directions, selecting a subset of the verbose query (or sub-query) and weighting query words in the verbose query. For example, Bendersky and Croft [1] learned how to discover the key concepts of a verbose query and Kumaran and Carvalho [11] studied how to automatically reduce a verbose query to a shorter and more effective subset. Examples of the research on weighting include Lease et al [14, 13] who trained a regression model to weight all query words of a verbose query and Bendersky et al [3] who proposed a way to uniformly learn the importance of concepts underlying the verbose query. Both directions have shown promise in improving the retrieval performance of verbose queries. In this work, we focus on selecting subsets, since it simulates a common behavior of people when they deal with verbose queries.

The sub-query selection problem is defined as selecting a subset of the original verbose query. In other words, the problem is to assign a label "keep" or "don't keep" for each query word. Here, we focus on the impact on this labeling of the different types of relations that can exist between query words. For example, some query words may form a noun phrase such as "Spanish Civil War" and others may describe named entities such as "Chesapeake Bay Maryland". A query word may serve as the subject of another query word such as "Mississippi River" being the subject of "flood" when they are used in the query "How frequently does the Mississippi River flood its banks?". Generally, these relations between query words imply the relations between the labels assigned to query words. For example, since "Spanish Civil War" is a noun phrase, these three words tend to have the same labels, either keeping all of them or dropping all of them. This is also true for other relations. Therefore, it is reasonable to model sub-query selection as a sequential labeling problem instead of a classification problem that determines the labels of query words independently. Moreover, instead of selecting the best sub-query, it is more general to model a distribution over the space of all possible sub-queries. This means that the probabilities of observing each sub-query in this distribution can be used as weights when we need to combine different sub-queries.

A Conditional Random Field (CRF) is a well known graphical model designed for sequential labeling problems and provides a unified framework to capture features extracted from the input sequence, especially those features modeling the

CIKM'10,

underlying dependencies. Furthermore, a CRF provides a formally well-founded framework to model the distribution of label sequences. Therefore, based on the above analysis, a CRF model should be a good method for describing the distribution of sub-queries.

However, some unique properties of sub-query selection makes the direct application of CRFs difficult. A conventional CRF is designed to optimize labeling accuracy based on a training set of input sequences and their corresponding gold-standard label sequences. When selecting subset queries, however, it is unclear how to define the gold-standard subset of query words given the input verbose query. A straightforward way is to select the sub-query with the best retrieval performance. However, the sub-query with the best performance may overfit to the collection and thus may not necessarily be the best choice. For example, the sub-query "jockey weight horse" (MAP[1]=77.1) has the best performance given the original query "What are the limits and regulations concerning jockey weight in horse racing?". However, another sub-query "jockey weight horse racing" (MAP=74.2) is clearly a better choice when being used to learn rules that can generalize to unseen queries. In this paper, a novel CRF model called *CRF-perf* is proposed, where instead of selecting the gold-standard sub-query, all sub-queries are used with their associated retrieval performance. Compared with the conventional CRF that optimizes label accuracy, CRF-perf is able to directly optimize the expected retrieval performance over all sub-queries. In this way, CRF-perf can discover the common rules shared by those high performing sub-queries. Those rules are thus likely to apply to unseen queries. Also, CRF-perf provides a general learning framework, which can be adapted to different retrieval models and performance measures.

Several types of features are used in this paper to train CRF-perf, which captures the local and global dependencies between query words. We also propose some retrieval models to incorporate the distribution of sub-queries generated by CRF-perf. Experiments on TREC collections show that CRF-perf can generate high-quality sub-query distributions and can significantly outperform state-of-the-art techniques.

The rest of this paper is organized as follows: Section 2 briefly reviews related work; Section 3 formally defines the notations and problems; Section 4 describes the proposed CRF-perf model in detail; Section 5 considers the retrieval model using sub-query distribution; Section 6 reports the experimental results and Section 7 concludes.

2. RELATED WORK

As mentioned above, previous work on processing verbose queries generally falls into two areas, selecting a subset of the verbose query and weighting all query words of the verbose query.

Kumaran and Allan [10] studied reducing the verbose query into a subset through human interaction. The sub-queries with high mutual information scores are displayed to the user. Their experiments showed that the user can select good sub-queries using snippet information. Bendersky and Croft [1] proposed a method to find key concepts from a verbose query using different types of features. A key concept can be considered as a special subset query, which helps to improve the retrieval performance when combined with the verbose query. Recently, Kumaran and Carvalho [11] used Ranking SVM to learn to how to automatically select sub-queries using several query quality predictors. Ranking SVM learns to rank sub-queries based on their retrieval performance, but, in contrast to our model, it cannot directly optimize the retrieval performance of sub-queries. Also, Kumaran and Carvalho [11] only used the top sub-query for retrieval. In this paper, we consider several types of retrieval models using sub-queries.

Lease et al [14] improved verbose queries by weighting query terms, which assigned more weight to important words and less weight to unimportant ones. They trained a regression model to learn how to map the secondary features to the optimal weights of query words. Lease [13] further incorporated their regression model into the framework of the Dependence Model [19] and observed significant performance improvement. Bendersky et al [3] proposed a unified framework to measure the importance of concepts underlying the verbose queries and extended the conventional Dependence Model to its weighted version. In this paper, we focus on selecting sub-queries instead of weighting query words.

A Conditional Random Field was first proposed by Lafferty et al [12] for segmenting and labeling sequence data. It has been successfully applied to many applications such as shallow parsing [26], named-entity recognition [17], identifying protein names [25] and so on. Guo et al. [4] proposed a CRF-based model for query refinement, which solved several tasks such as spelling correction, word splitting and word stemming within the same framework. Li et al [15] used CRFs for query tagging, which assigns each query term a pre-defined category. Qin et al [22] proposed a continuous CRF to capture the dependencies between documents and assign the retrieval scores for a set of documents simultaneously instead of one by one. In the work described here, a performance-based CRF model is proposed to solve the sub-query selection problem, which can directly optimize the expected retrieval performance of sub-queries.

3. PROBLEM DEFINITION

In this section, we define the problems we are addressing and introduce the notations will be used in the rest of the paper.

Sub-query Selection is the problem of selecting a subset of query words from the original verbose query. The original verbose query Q can be represented as a sequence of words $\mathbf{x} = \{x_1 x_2 ... x_n\}$. x_i is the query word and n is the length of the query. $\mathbf{y} = \{y_1 y_2 ... y_n\}$ denotes a sequence of labels, where y_i takes the value of 1 or 0, corresponding to "keep" or "don't keep" x_i, respectively. Thus, $\mathbf{y}$ represents a selection of query words. Given $\mathbf{x}$ and $\mathbf{y}$, a sub-query Q_s can be generated. Sometimes, $\mathbf{y}$ is also used to denote a sub-query.

Learning to Generate Sub-query Distribution is the problem of learning a model from a training set of queries and using the learned model to generate the sub-query distribution for unseen queries. The training set is denoted as $Train = \{\mathbf{x}, \{\mathbf{y}, m(\mathbf{y}, M)\}\}$, which consists of the original query $\mathbf{x}$ and its corresponding sub-query set $\{\mathbf{y}, m(\mathbf{y}, M)\}$. The sub-query set consists of all sub-queries $\mathbf{y}$ and their corresponding retrieval performance $m(\mathbf{y}, M)$. The function m could be any performance measure used in information retrieval and M is the retrieval model used. $m(\mathbf{y}, M)$ is often abbreviated as $m(\mathbf{y})$, when M is not explicitly claimed. Given the training set $Train$, the model parameters θ are

[1]MAP denotes the mean average precision.

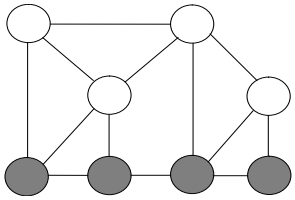


Figure 1: Example of general CRF

learned. Then, for an unseen query $\mathbf{x}'$, a probabilistic distribution over its sub-queries is predicted as $P(\mathbf{y}'|\mathbf{x}', \theta)$. This distribution can then be used to select the best sub-query or used in other applications.

4. PERFORMANCE BASED CONDITIONAL RANDOM FIELD

In this section, we first introduce the conventional CRF model, then describe the proposed performance-based Conditional Random Field (CRF-perf) model and finally provide information about the features used.

4.1 Conditional Random Field

A Conditional Random Field (CRF) is a graphical model designed to directly model the conditional probability $P(\mathbf{y}|\mathbf{x})$. Compared with a generative model, a CRF avoids modeling the joint probability $P(\mathbf{y}, \mathbf{x})$, which is actually unnecessary for a sequential labeling problem. Thus, CRF doesn't need to make unreasonable independence assumptions over the input sequence and can handle a large number of correlated features. The graphical model of a general CRF is depicted in Fig. 1 as an example, where the shaded nodes denote the input variables x_i and the non-shaded nodes denote the output variables y_i. The relations between different nodes are represented by the graph structure.

In a CRF, the conditional probability $P(\mathbf{y}|\mathbf{x})$ is calculated as follows:

$$P(\mathbf{y}|\mathbf{x}) = \frac{\exp(\sum_{k=1}^{K} \lambda_k f_k(\mathbf{x}, \mathbf{y}))}{Z(\mathbf{x})} \tag{1}$$

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \exp(\sum_{k=1}^{K} \lambda_k f_k(\mathbf{x}, \mathbf{y})) \tag{2}$$

where f_k represent the features, which are extracted based on the input sequence $\mathbf{x}$ and the label sequence $\mathbf{y}$. λ_k is the weight of the feature f_k. $\theta = \{\lambda_k\}$ denotes the parameters of the model. K is the number of features. $Z(x)$ is a normalizer, which guarantees $\sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = 1$.

The training set used by the conventional CRF can be denoted as $\{\mathbf{x}, \mathbf{y}^\star\}$, which consists of a set of input sequences $\mathbf{x}$ and their gold-standard label sequences $\mathbf{y}^\star$. The conditional probabilities of the training set can be calculated according to Eq. 3.

$$P(\theta) = \prod_{\mathbf{x}} P(\mathbf{y}^\star|\mathbf{x}) \tag{3}$$

The log-likelihood of $P(\theta)$ can be calculated according to Eq. 4.

$$l(\theta) = \sum_{\mathbf{x}} \sum_{k=1}^{K} \lambda_k f_k(\mathbf{x}, \mathbf{y}^\star) - \sum_{\mathbf{x}} \log Z(\mathbf{x}) - \sum_{k=1}^{K} \frac{\lambda_k^2}{2\delta^2} \tag{4}$$

The last term of Eq. 4 is served as a regularizer which penalizes values of θ that are too large. δ^2 is a parameter set by the user.

The parameters $\theta = \{\lambda_k\}$ can be learned by maximizing the log likelihood. Its partial derivatives can be calculated by Eq. 5.

$$\frac{\partial \log(\theta)}{\partial \lambda_k} = \sum_{\mathbf{x}} f_k(\mathbf{x}, \mathbf{y}^\star) - \sum_{\mathbf{x}} \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) f_k(\mathbf{x}, \mathbf{y}) - \sum_{k=1}^{K} \frac{\lambda_k}{\delta^2} \tag{5}$$

The first term of Eq. 5 is the empirical value of f_k given the gold standard-label $\mathbf{y}^\star$, the second term is the expected value of f_k under the distribution $P(\mathbf{y}|\mathbf{x})$ and the last term is a regularizer. Thus, without considering the regularizer, setting Eq. 5 to zero means making the expected value match the empirical value.

4.2 CRF-perf

As mentioned previously, it is not easy to select the gold-standard sub-queries for our problem, which makes the direct application of CRFs difficult. Therefore, a performance-based based Conditional Random Field (CRF-perf) is proposed to solve this problem, which uses all sub-queries and their associated retrieval performance. Using the notations introduced in Section 3, the objective function of CRF-perf on the training set is defined by Eq. 6.

$$\text{GM-perf}(\theta) = (\prod_{\mathbf{x}} \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) m(\mathbf{y}))^{\frac{1}{T}} \tag{6}$$

Here, T is the number of $\mathbf{x}$ in the training set $Train$. $\sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) m(\mathbf{y})$ is the expected retrieval performance over the distribution of label sequences for a given $\mathbf{x}$. Compared with Eq. 3, which optimizes the probability of observing the gold-standard label sequence $\mathbf{y}^\star$, Eq. 6 directly optimizes the Geometric Mean of the expected retrieval performance of each $\mathbf{x}$ in the training set. For example, if $m(y)$ measures average precision (AP), CRF-perf optimize the geometric mean average precision (GMAP) on the training set, which is a widely used performance measure in information retrieval. According to Robertson [23], GMAP is sensitive to improvements on the difficult queries.

Since T is a constant value after the training set is provided, optimizing Eq. 6 is equivalent to optimizing Eq. 7.

$$\text{GM-perf}'(\theta) = \prod_{\mathbf{x}} \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) m(\mathbf{y}) \tag{7}$$

The corresponding log-likelihood expression of Eq. 7 is shown in Eq. 8.

$$\begin{aligned} l(\theta) \quad = \quad & \sum_{\mathbf{x}} \log \sum_{\mathbf{y}} \exp(\sum_{k=1}^{K} \lambda_k f_k(\mathbf{x}, \mathbf{y})) m(\mathbf{y}) \\ & - \sum_{\mathbf{x}} \log Z(\mathbf{x}) - \sum_{k=1}^{K} \frac{\lambda_k^2}{2\delta^2} \end{aligned} \tag{8}$$

From this, we can show that the partial derivatives of λ_k

can be calculated by Eq. 9.

$$\frac{\partial \log(C)}{\partial \lambda_k} = \sum_{\mathbf{x}} \sum_{\mathbf{y}} P_m(\mathbf{y}|\mathbf{x}) f_k(\mathbf{x}, \mathbf{y}) - \sum_{\mathbf{x}} \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) f_k(\mathbf{x}, \mathbf{y}) - \sum_{k=1}^{K} \frac{\lambda_k}{\delta^2} \quad (9)$$

$P_m(\mathbf{y}|\mathbf{x})$ is a distribution weighted by $m(\mathbf{y})$, which is shown as follows.

$$P_m(\mathbf{y}|\mathbf{x}) = \frac{\exp(\sum_{k=1}^{K} \lambda_k f_k(\mathbf{x}, \mathbf{y})) m(\mathbf{y})}{Z_m(\mathbf{x})} \quad (10)$$

$$Z_m(\mathbf{x}) = \sum_{\mathbf{y}} \exp(\sum_{k=1}^{K} \lambda_k f_k(\mathbf{x}, \mathbf{y})) m(\mathbf{y}) \quad (11)$$

In Eq. 9, the first term is the expected value of f_k weighted by the retrieval performance and the second term is the expected value of f_k without weighting. Setting Eq. 9 to zero means making the distribution of sub-queries match the distribution of their retrieval performance.

Note that if we set $m(\mathbf{y}) = 1$ for the case $\mathbf{y} = \mathbf{y}^\star$ and set $m(\mathbf{y}) = 0$ for other cases, CRF-perf becomes the conventional CRF model. Therefore, the conventional CRF model is a special case of CRF-perf.

The inference problems of the CRF model involve how to marginalize over the label sequence during the training phase and how to pick up the best label sequence during the predicting phase. For some special graph structures such as linear-chain CRF [12], the inference problem can be solved with dynamic programming techniques. For general graph structures, more complex inference techniques are required such as Contrastive Divergence [6] and Loopy Belief Propagation [28].

In our situation, some properties of the sub-query selection makes the inference problem easier. According to the study made by Bendersky and Croft [2] on a search log, 90.3% queries have a length of no more than 4 words, 9.6% queries have a length between 5 to 12 words and only 0.1% queries have a length more than twelve. In addition, they noticed that there are a considerable number of seemingly bot-generated queries among those 0.1% of queries. Therefore, it is reasonable to assume most verbose queries used by web users are no more than 12 words. Then, the space of the label sequence is at the scale of $2^{12} = 4096$, which is computationally tractable for simple iterations over the whole space.

4.3 Features

Three types of features are used to capture different levels of dependencies underlying the input sequence. Some of the features used here are the same as the features used by Bendersky and Croft [3] and by Kumaran and Carvalho [11].

Independency Features characterize a single query word. The general feature function $f_k(\mathbf{x}, \mathbf{y})$ can be specialized as $f_k(x_i, y_i)$. This type of feature includes the standard term frequency and document frequency in the target corpus, the frequency of the query word observed in external resources such as Google Ngram, Wikipedia and some commercial query logs. Besides statistical information, they also include some syntactic features such as POS-tags.

Local Dependency Features capture the dependencies between query words. Since they characterize *some* query words but not all, they are called *Local* Dependency Features. The general feature function $f_k(\mathbf{x}, \mathbf{y})$ can be specialized as $f_k(x_i x_j x_k ..., y_i y_j y_k ...)$. This type of feature includes bigrams, noun phrases, the dependency relations returned by a dependency parser [16] and named entities. All statistical features used for single words can also be applied to bigrams. Noun phrases have been shown to be effective when they are combined with the original query [1]. A dependency parser [16] can return different types of relations between two query words. For example, given the sentence "How frequently does the Mississippi River flood its banks?", "river" and "flood" satisfies the subject relation, since "river" is the subject of "flood". Also, "flood" and "banks" have the object relation, since "banks" is the object of "flood". It helps sometimes to include words satisfying a certain relation in a sub-query. Named entities including names of people, locations and organizations are usually important and may be included in sub-queries.

Global Dependency Features describe properties of the selected sub-query, which is generated from *all* query words ($\mathbf{x}$) and their labels ($\mathbf{y}$). Therefore, they are called *Global* Dependency Features. The feature function used here is $f_k(\mathbf{x}, \mathbf{y})$. As indicated by Kumaran and Carvalho [11], query quality predictors are good features to characterize sub-queries. Here, several types of query quality predictors have been used, such as Mutual Information [10], Query Scope [5], Query Clarity [24] and so on. Passage-level evidence can also be used to describe the sub-query. If a sub-query appears frequently within a passage, it is very likely that query words of this sub-query are closely related. Thus, the number of passages containing the sub-query selected can be used as a feature. The parsing tree [9] of the input

Table 1: Three types of features

Independency Features	
uTF	unigram term frequency
uDF	unigram document frequency
uNGram	unigram count in Google nGram
uWiki	unigram count of matching Wiki titles
uMSNLog	unigram count in MSN query logs
uPosTag	unigram pos-tag="NN", "VB", "JJ"
Local Dependency Features	
bTF	bigram term frequency
bDF	bigram document frequency
bNGram	bigram count in Google nGram
bWiki	bigram count of matching Wiki titles
bMSNLog	bigram count in MSN query logs
np	noun phrases
dep-obj[16]	the object relation
dep-subj[16]	the subject relation
dep-nn[16]	the noun compound modifier relation
PER	person names
LOC	location names
ORG	organization names
Global Dependency Features	
MI[10]	mutual information
SQLen[11]	sub-query length
QS[5]	query scope
QC[24]	query clarity score
SOQ[11]	similarity to original query
psg	count of passages containing sub-query
h-pnode	height of the parent node covering sub-query

query also provides valuable information. It is interesting to consider whether query words of the selected sub-query concentrate on a small part of the parsing tree or spread over the whole tree. This property is partly measured by the height of the direct parent node that covers the sub-query in the parsing tree.

All features used are summarized in Table 1.

5. RETRIEVAL MODELS WITH SUBSET QUERIES

As mentioned previously, CRF-perf provides a general learning framework for different retrieval models and performance measures, which is indicated by $m(\mathbf{y}, M)$ where m is the performance measure function and M is the retrieval model. CRF-perf(M,m) is used to denote the CRF-perf model trained based on the retrieval model M and the performance measure m. Four types of retrieval models using sub-queries are proposed in this paper. Other types of retrieval models can be easily incorporated within this framework. Q_s denotes a sub-query and Q denotes the original verbose query.

SubQL denotes the query likelihood model using the sub-query. The score of a document can be calculated as follows:

$$score_{QL}(D, Q_s) = \sum_{t \in T(Q_s)} \log(P(t|D)) \quad (12)$$

where $T(Q_s)$ represents a set of query terms of Q_s. $P(t|D)$ is the probability of generating a term t from a document D, which is estimated using the Language Modeling approach [20] with Dirchlet Smoothing [29].

SubDM denotes the sequential dependency model [19] using the sub-query. The score of a document can be calculated as follows:

$$score_{DM}(D, Q_s) = \lambda_T \sum_{t \in T(Q_s)} \log(P(t|D)) + \lambda_O \sum_{o \in O(Q_s)} \log(P(o|D)) + \lambda_U \sum_{u \in U(Q_s)} \log(P(u|D)) \quad (13)$$

where $T(Q_s)$ denotes a set of query terms of Q_s, $O(Q_s)$ denotes a set of ordered bigrams extracted from Q_s and $U(Q_s)$ denotes a set of unordered bigrams extracted from Q_s. λ_T, λ_O and λ_U are parameters controling the weights of different parts and are usually set as 0.85, 0.1 and 0.05.

The above retrieval models only use the sub-query Q_s. The following retrieval models attempt to combine the sub-query Q_s with the original query Q.

QL+SubQL denotes a combination of the original query and the sub-query, where both parts use the query likelihood model. The score of a document using this model can be calculated as follows:

$$score(D, Q, Q_s) = \alpha score_{QL}(D, Q) + (1-\alpha) score_{QL}(D, Q_s) \quad (14)$$

where α is a parameter weighting the original query and the sub-query. α is set as 0.8 in this paper.

DM+SubQL uses the sequential dependency model for the original query and uses the query likelihood model for the sub-query. The score of a document is calculated as follows:

$$score(D, Q, Q_s) = \alpha score_{DM}(D, Q) + (1-\alpha) score_{QL}(D, Q_s) \quad (15)$$

Table 2: Example of Indri queries.

```
Q : jobs outsourced india
Q_s: jobs india
SubQL: #combine(jobs india)
SubDM: #weight(
        0.85 #combine(jobs india)
        0.1 #combine(#1(jobs india))
        0.05 #combine(#uw8(jobs india)))
QL+SubQL: #weight(
        0.8 #combine(jobs outsourced india)
        0.2 #combine(jobs india))
DM+SubQL: #weight(
        0.8 #wegiht(
            0.85 #combine(jobs outsourced india)
            0.1 #combine(#1(jobs outsourced)
                         #1(outsourced india))
            0.05 #combine(#uw8(jobs outsourced)
                          #uw8(outsourced india)))
        0.2 #combine(jobs india))
```

Table 3: TREC collections used in experiments

Name	Docs	Topics
Robust04	528,155	301-450,601-700
WT10g	1,692,096	451-550
Gov2	25,205,179	701-850

where α is a parameter and is set as 0.8 in this paper.

The above four retrieval models can all be implemented using the Indri query language [18]. Table 2 shows an example of Indri queries used for each retrieval model.

During the training phase, the retrieval model M is used to calculate $m(\mathbf{y}, M)$ for each sub-query. After learning CRF-perf, the probability distribution $P(\mathbf{y}'|\mathbf{x}')$ is predicted for an unseen query $\mathbf{x}'$. We then consider how to do retrieval for $\mathbf{x}'$ using M and $P(\mathbf{y}'|\mathbf{x}')$. A straightforward method is to select the sub-query $\mathbf{y}'$ with the highest probability and feed it to M. This method is denoted as **Top1**. Another alternative is to feed the top k sub-queries to M respectively and combine the retrieval scores with their corresponding probabilities. This method is denoted as **TopK**. Take **SubQL** for example. The retrieval scores of using top k sub-queries are calculated as follows:

$$score_{QL}(D, \{Q_s\}) = \sum_{i=1}^{k} P(Q_{s_i}|Q) score_{QL}(D, Q_{s_i}) \quad (16)$$

where $P(Q_{s_i}|Q)$ corresponds to the probability of the i^{th} sub-query Q_{s_i}.

6. EXPERIMENTS

In this section, we first describe the experimental configuration, then provide examples of the sub-query distributions generated by CRF-perf and finally report the experimental results.

6.1 Experimental Configuration

Experiments are conducted on three TREC collections (Gov2, Robust04 and WT10g). The statistics of each collection are summarized in Table 3.

For each collection, Indri 2.10 [18] is used to build the index with the Porter Stemmer[21]. No stopword removal is

Table 4: Example of $P(Q_s|Q)$ learned with SubQL and AP. Top four sub-queries are displayed. In the original query Q, the words actually used are bolded and stopwords and stop structures are italicized.

$P(Q_s\|Q)$	Sub-query (Q_s)	MAP
Q: *what is the* **history** *and* **location** *of* **scottish highland games** *in the* **united states**		48.83
0.564	scottish highland games united states	54.28
0.341	location scottish highland games united states	57.04
0.074	history scottish highland games united states	43.73
0.010	history location scottish highland united states	18.38
Q: *give information on* **steps** *to* **manage control** *or* **protect squirrels**		21.03
0.621	steps protect squirrels	31.13
0.324	steps control squirrels	28.59
0.048	steps control protect squirrels	30.35
0.002	steps manage squirrels	25.46
Q: **how** *have* **humans responded** *and* **how should they respond** *to the* **appearance** *of* **coyotes** *in* **urban** *and* **suburban areas**		10.47
0.452	humans responded appearance coyotes urban suburban	31.54
0.103	humans responded respond appearance coyotes urban	26.90
0.086	humans responded respond appearance coyotes suburban	14.09
0.045	humans responded appearance coyotes urban	50.80
Q: *what is* **known about** *the* **culture** *and* **history** *of the* **chaco people** *from* **features** *of the* **chaco culture national historic park**		30.77
0.214	culture history chaco people features chaco	53.16
0.119	culture chaco people features chaco park	46.98
0.102	known chaco people features chaco park	38.60
0.069	history chaco people features chaco park	42.01
Q: *the* **remedies** *and* **treatments given** *to* **lessen** *or* **stop effects** *of* **ovarian cancer**		13.53
0.835	remedies treatments given effects ovarian cancer	23.92
0.073	remedies treatments given lessen ovarian cancer	17.31
0.066	remedies treatments given ovarian cancer	20.37
0.010	remedies treatments lessen effects ovarian cancer	16.79

performed during indexing. For each topic, the *description* part is used as the query. Following Bendersky et al [3], a short list of 35 stopwords and some frequent stop patterns (e.g., "find information") are removed from the description query in order to improve the retrieval performance of the baseline methods. If the length of a query is more than 10 words, all query words are first ranked by their *idf* scores and then the top 10 words are kept for sub-query generating. Following Kumaran and Carvalho [11], only sub-queries with length between three to six words are considered.

The query set is split into a training set and a test set. On the training set, four types of retrieval methods mentioned in Section 5 (SubQL, SubDM, QL+SubQL, DM+SubQL) are used to help learn CRF-perf, respectively. On the test set, the performance of using each retrieval model and sub-queries selected by its corresponding CRF model is reported. Ten-fold cross validation is conducted in this paper. The parameter δ^2 of CRF is set as 100. According to Sutton and McCallum [27], the performance of the CRF model doesn't appear to be sensitive to changes of δ^2.

Several baseline methods are compared. QL denotes the query likelihood language model [20, 29]. DM denotes the sequential dependence model [19]. SRank denotes Kumaran and Carvalho's method [11], which considers sub-query selection as a ranking problem. They used Rank SVM [8] as the ranking model. According to their suggestions, the parameters of Rank SVM are set as follows: RBF kernel is used with γ set as 0.001 and C is set as 0.01. KeyConcept denotes Bendersky and Croft's method [1] that augments the original query by discovering key concepts.

The standard performance measures, mean average precision (MAP) and precision at 10 (P10), are used to measure retrieval performance. The two-tailed t-test is conducted for significance.

6.2 Examples of Selected Sub-queries

First, we present some examples of $P(Q_s|Q)$(or $P(\mathbf{y}|\mathbf{x})$) learned by CRF-perf(SubQL,AP) in Table 4. The retrieval performance (MAP) of the original query and sub-queries is reported on the Gov2 collection. As mentioned previously, some stopwords and stop patterns are removed from the original query. Those words are italicized to improve readability. Note that they are not used for retrieval and sub-query generation.

Table 4 shows that the CRF-pref model can learn a reasonable distribution of sub-queries, which successfully assigns high probabilities to sub-queries that perform better than the original query. For example, given the original query "steps manage control protect squirrels", the top three sub-queries "steps protect squirrels", "steps control squirrels" and "steps control protect squirrels" receive most of the probabilities and all of them perform much better than the original query.

As mentioned above, CRF-perf is a general learning framework that can be adapted to different retrieval models. It is interesting to compare the sub-queries learned based on different retrieval models. Table 5 compares the top one sub-query returned by SubQL and QL+SubQL and some examples are also provided to compare SubDM and DM+SubQL. All models are learned based on AP.

Table 5: Comparisons of the top sub-query learned by CRF-perf with different retrieval models. In the original query Q, the words actually used are bolded and stopwords and stop structures are italicized.

Q: *what* **evidence** *is* **there** *that* **aspirin may help prevent cancer**	
SubQL: evidence aspirin may help cancer	QL+SubQL: aspirin prevent cancer
Q: *what* **states** *or* **localities offer programs** *for* **gifted talented students**	
SubQL: localities offer programs gifted talented students	QL+SubQL: gifted talented students
Q: **illicit activity involving diamonds** *to* **include diamond smuggling**	
SubQL: illicit activity diamonds diamond smuggling	QL+SubQL: diamonds diamond smuggling
Q: *what* **allegations** *have been* **made about enrons culpability** *in the* **california energy crisis**	
SubDM: allegations enrons culpability california energy crisis	DM+SubQL: enrons culpability california energy crisis
Q: *what is the* **history** *and* **location** *of* **scottish highland games** *in the* **united states**	
SubDM: location scottish highland games united states	DM+SubQL: scottish highland games
Q: *where do* **yew trees grow anywhere** *on the* **globe**	
SubDM: yew trees grow globe	DM+SubQL: yew trees globe

Table 6: Performance of retrieval models using sub-queries. ^{q} denotes significantly different with QL and $_d$ denotes significantly different with DM. "1" denotes the Top1 method and "K" denotes the TopK method.

	Gov2		Robust04		Wt10g	
	MAP	P10	MAP	P10	MAP	P10
QL	25.43	52.21	25.49	43.13	19.61	32.68
DM	27.85	54.03	26.83	44.94	20.87	**35.77**
SRank	24.99	50.74	24.78	41.57	19.98	32.06
KeyConcept	27.52	53.83	25.97	41.65	21.01	34.02
SubQL(1)	25.90	51.88	25.43	40.84	18.97	31.55
SubQL(K)	26.66^q	53.36	25.96	41.93	19.27	31.75
QL+SubQL(1)	26.49^q	53.09	26.10	43.53	20.12	32.78
QL+SubQL(K)	26.76^q	53.15	26.20^q	43.21	19.94	33.20
SubDM(1)	28.17^q	53.49	26.56^q	42.69	20.26	33.92
SubDM(K)	28.60^q	53.76	27.07^q	43.69	20.70	34.74
DM+SubQL(1)	28.56^q	$\mathbf{55.91}^q_d$	27.36^q	$\mathbf{45.42}^q$	21.94^q	35.26^q
DM+SubQL(K)	$\mathbf{28.70}^q_d$	55.37^q	$\mathbf{27.37}^q_d$	45.14^q	$\mathbf{22.17}^q$	35.15^q

Table 5 shows that when the sub-queries are used alone (SubQL and SubDM), CRF-perf tends to select longer sub-queries, since it is safer to cover most of the concepts in the original query. When the sub-queries are combined with the original query (QL+SubQL, DM+SubQL), the CRF-perf model tends to favor shorter queries, since it is reasonable to focus on important concepts when the original query has covered all concepts. For example, given the original query "history location scottish highland games united states", SubDM selects a sub-query "location scottish highland games united states" which covers almost all concepts of the original query, while DM+SubQL simply picks up the most important concept "scottish highland games". Similarly, SubQL selects a subset query "localities offer programs gifted talented students" which only removes "states" from the original query, while QL+SubQL selects the key concept "gifted talented student".

6.3 Retrieval Performance

The first experiment is conducted to compare the proposed four types of retrieval models with baseline methods. Those sub-query based retrieval models include SubQL, QL+SubQL, SubDM and DM+SubDM. Average Precision (AP) is optimized during training phase. Note that, unless otherwise mentioned, AP is optimized in the following experiments. For each model, the performance of using Top1 and TopK (K=10) sub-queries is reported, respectively. The baseline methods are QL (query likelihood language model), DM (sequential dependence model), SRank (Kumaran and Carvalho's method [11]) and KeyConcept (Bendersky and Croft's method [1]). The results are shown in Table 6. The best performance of each column is bolded.

Table 6 shows that SubQL(Top1) is comparable with QL, which indicates that using the top one sub-query does not outperform QL. This observation has been supported by the performance of SRank, a Ranking SVM based method, which also uses the top one sub-query for retrieval. SubQL (Top1) performs slightly better than SRank, especially on Gov2. When top K sub-queries are used, the performance improves. SubQL(TopK) performs better than QL on Gov2. QL+SubQL performs better than QL on all three collections, which indicates that combining the sub-query with the original query is promising. SubDM performs better than DM on three collections, although the improvement is not significant. The best performance is achieved by DM+SubQL, which performs significantly better than DM, a very strong baseline, on Gov2 and Robust04. It indicates that the most effective way is to use the sequential dependence model for the original query, use the query likelihood model for the sub-query and combine them together. DM+SubQL also performs better than KeyConcept, the state-of-the-art technique for improving verbose queries. Generally, the TopK method is better than the Top1 method, which is consistent over different retrieval models.

6.4 Further Analysis of CRF-perf

The second experiment is conducted to help understand where the benefits of CRF-perf come from. We compare CRF-perf with two other methods. The first method considers the sub-query selection as a classification problem, where a classifier is trained to decide the label of each word independently. For a query in the training set, the labels of query words are decided by its best sub-query. A standard SVM classifier [7] is used here[2], since it has been successfully applied to a variety of classification tasks. This method is denoted as SVM. The features used to describe a single word are displayed in Table 7. The comparisons between

[2]Note that the SVM used here is a standard classification method that assigns labels to each single word. It is different with the RankSVM used by Kumaran and Carvalho [11] that ranks sub-queries.

Table 7: Features used to describe a single word

uTF	unigram term frequency
uDF	unigram document frequency
uNGram	unigram count in Google nGram
uWiki	unigram count of matching Wiki titles
uMSNLog	unigram count in MSN query logs
uPosTag	unigram pos-tag="NN", "VB", "JJ"
isPartOfNP	part of noun phrases?
isPartOfPER	part of person names?
isPartOfLOC	part of location names?
isPartOfORG	part of organization names?

Table 8: Comparisons of SVM, CRF and CRF-perf. SubQL(1) is the retrieval model.

	Gov2		Robust04		Wt10g	
	MAP	P10	MAP	P10	MAP	P10
SVM	23.91	46.04	20.68	33.41	19.45	31.96
CRF	24.76	48.99	23.21	36.99	17.48	29.48
CRF-perf	25.90	51.88	25.43	40.84	18.97	31.55

SVM and CRF-perf help indicate whether it is helpful to consider sub-query selection as a sequential labeling problem instead of a classification problem. The second method is the ordinary CRF (denoted as CRF). As discussed in the Introduction, the ordinary CRF uses the sub-query with the best performance as the gold-standard label sequence, while CRF-perf considers all sub-queries with their performance. The comparisons between CRF and CRF-perf help indicate whether it is helpful to consider all sub-queries. Table 8 shows the retrieval performance of SVM, CRF and CRF-perf. Here, SubQL(1) is used as the retrieval model, which means using the top one sub-query for retrieval.

Table 8 shows that CRF and CRF-perf perform better than SVM on Gov2 and Robust04. CRF-perf is comparable with SVM on Wt10g. These results support that modeling the sub-query selection as a sequential labeling problem is indeed helpful. Also, CRF-perf outperforms CRF on all three collections, thus it is important to consider all sub-queries with their performance for sub-query selection.

Some examples of the sub-query returned by SVM, CRF and CRF-perf are provided in Table 9 to help understand their differences. SVM is generally able to select important nouns or noun phrases from the original query, since in most cases the selection of nouns can be decided independently. However, it has trouble in selecting verbs and adjectives, since the selection of these types of words usually depends on the relations to other words, which cannot be captured by SVM. For example, SVM selects the important nouns "maryland" and "chesapeake bay", but misses the key verb "clean". More examples can be found in Table 9. CRF, on the other hand, does consider the relations between query words, thus it can make better decisions. However, CRF is trained based on the best sub-queries. As discussed in the Introduction, the best sub-queries are usually short and may cause overfitting on the collection. Therefore, CRF tends to select short sub-queries and this can lead to mistakes. Sometimes, CRF selects good sub-queries such as "aspirin prevent cancer" (see Table 9), but, in most cases, the sub-queries selected by CRF are too short to cover all important concepts of the original queries such as "gastric bypass surgery" and "people puerto rico" (see Table 9). In contrast, the sub-query selected by CRF-perf may not be the perfect one, but it reliably improves the original query. This property is obtained by optimizing the expected, not the best, retrieval performance on the training set.

6.5 Effects of Different Feature Sets

The third experiment is conducted to explore the relative effect of different types of features (Independency Features, Local Dependency Features and Global Dependency Features). Table 10 reports the mean average precision (MAP) of different retrieval models with different types of features.

Table 10 shows that the best feature set is not consistent over different retrieval models and different collections. On Gov2, different retrieval models prefer different features. On Robust04, combining all three types of features together is clearly the best choice. On Wt10g, only using the Independency Features and the Global Dependency Features (I+G) seems to be the best choice. It is also interesting to notice that the best feature set almost always involves Global Dependency Features, which shows that capturing global dependencies of queries is important for sub-query selection. The observations using Top1 and using TopK are consistent. Generally, combining different types of features together is more effective than only using a single type of features. Using all features together can achieve good performance for most cases.

6.6 Effects of using Sub-query Distribution

Table 6 has shown that using the top ten sub-queries is better than only using the top one sub-query. In this subsection, we explore the effect of k, i.e. the number of sub-queries used. The results are displayed in Fig. 2. The retrieval model used is SubQL.

Fig. 2 shows that on three collections the most obvious performance improvement happens when k increases from one to five and there is not much difference when k is bigger than five. The reason is that the top five sub-queries usually receive most of the probabilities of the distribution. The above observations support using a distribution of sub-queries for improving retrieval performance. Thus, instead of returning a single sub-query, the model that can generate the sub-query distribution is preferred.

6.7 Train on Different Performance Measures

Since all above results are reported using CRF-perf trained on AP, it is interesting to compare them with the results using CRF-perf trained on P10. Table 11 reports the performance of CRF-perf trained on different performance measures (i.e. AP and P10).

Table 11 shows that the results using CRF-perf trained on different performance measures are similar. Generally, the CRF-perf trained on AP can produce slightly better performance than the model trained on P10. Although we expect that the model trained on P10 can generate better precision scores than the model trained on AP, actually the model trained on AP still performs better on P10 except for Wt10g. A possible explanation is that the AP metric allows the CRF model to distinguish between query subsets that would appear identical using the P10 metric.

7. CONCLUSION

Selecting sub-queries provides an effective way to improve the retrieval performance of verbose queries. In this paper, the sub-query selection problem is modeled as a sequential

Table 9: Comparisons of SVM, CRF and CRF-perf. In the original query Q, stopwords and stop structures are italicized and the words actually used are bolded.

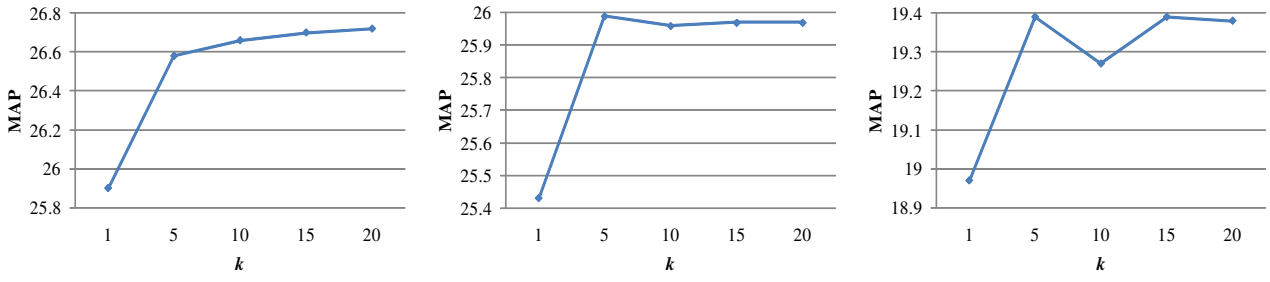
Q: *what is the* **state** *of* **maryland** *doing to* **clean up** *the* **chesapeake bay**	
SVM: state maryland chesapeake bay (MAP=11.65)	CRF-perf: maryland clean chesapeake bay (MAP=30.69)
Q: *what* **would cause** *a* **lowered white blood cell count**	
SVM: white blood cell count (MAP=13.58)	CRF-perf: cause lowered white blood cell count (MAP=20.97)
Q: **where** *have* **dams** *been* **removed** *and what has been the* **environmental impact**	
SVM: dams impact (MAP=7.18)	CRF-perf: dams removed environmental impact (MAP=21.97)
Q: *what* **evidence** *is* **there** *that* **aspirin may help prevent cancer**	
CRF: aspirin prevent cancer (MAP=40.99)	CRF-perf: evidence aspirin may help cancer (MAP=29.67)
Q: *what are* **some** *of the* **possible complications** *and* **potential dangers** *of* **gastric bypass surgery**	
CRF: gastric bypass surgery (MAP=35.81)	CRF-perf: complications dangers gastric bypass surgery (MAP=51.27)
Q: *do* **people** *in* **puerto rico want** *for it to* **become** *a* **us state**	
CRF: people puerto rico (MAP=32.57)	CRF-perf: people puerto rico want state (MAP=45.92)

Table 10: The mean average precision (MAP) of using different types of features. "I" denotes the Independency Features, "L" denotes the Local Dependency Features and "G" denotes the Global Dependency Features.

	Gov2				Robust04				Wt10g			
Top1	SubQL	QL+SubQL	SubDM	DM+SubQL	SubQL	QL+SubQL	SubDM	DM+SubQL	SubQL	QL+SubQL	SubDM	DM+SubQL
I	25.80	26.31	27.83	28.50	24.32	25.35	25.64	26.84	20.66	20.28	21.28	**22.63**
L	25.69	26.30	**28.19**	28.39	24.82	25.88	26.24	27.12	18.68	20.10	20.26	21.64
G	24.86	26.01	27.13	28.03	24.91	25.69	26.10	26.89	20.40	19.57	21.28	21.00
I+L	25.00	26.59	27.65	28.36	24.63	25.88	25.57	27.30	19.57	20.18	20.55	21.92
I+G	25.76	26.45	27.95	**28.73**	24.57	25.67	26.12	26.99	**21.62**	**21.10**	**22.37**	22.32
L+G	25.40	**26.83**	27.72	28.43	25.03	25.77	25.83	26.90	19.21	20.17	19.99	21.79
I+L+G	**25.90**	26.49	28.17	28.56	**25.43**	**26.10**	**26.56**	**27.36**	18.97	20.12	20.26	21.94
TopK	SubQL	QL+SubQL	SubDM	DM+SubQL	SubQL	QL+SubQL	SubDM	DM+SubQL	SubQL	QL+SubQL	SubDM	DM+SubQL
I	26.20	26.55	**29.04**	28.71	25.53	25.63	26.77	26.86	21.00	20.34	22.19	22.73
L	26.27	26.32	28.57	28.54	25.04	25.88	26.44	27.25	19.33	20.10	20.51	21.55
G	25.78	26.19	28.30	28.49	25.69	25.81	26.87	26.99	21.15	20.11	22.14	21.98
I+L	26.02	26.78	28.23	28.62	25.46	26.08	26.70	27.32	19.54	19.92	20.55	21.96
I+G	26.49	26.62	28.84	**28.96**	25.77	26.12	26.75	27.25	**21.84**	**21.11**	**23.21**	**22.76**
L+G	26.57	**26.90**	28.79	28.60	25.59	26.19	26.44	27.06	19.38	19.99	20.59	22.47
I+L+G	**26.66**	26.76	28.60	28.70	**25.96**	**26.20**	**27.07**	**27.37**	19.27	19.94	20.70	22.17

Table 11: Comparisons of CRF-perf trained on AP and P10.

	Gov2				Robust04				Wt10g			
	train on AP		train on P10		train on AP		train on P10		train on AP		train on P10	
	MAP	P10	MAP	P10	MAP	P10	MAP	P10	MAP	P10	MAP	P10
SubQL(1)	25.90	51.88	24.57	49.26	25.43	40.84	24.37	38.96	18.97	31.55	18.86	30.93
SubQL(K)	26.66	53.36	25.65	52.15	25.96	41.93	24.93	39.80	19.27	31.75	18.54	30.41
QL+SubQL(1)	26.49	53.09	26.19	52.15	26.10	43.53	25.33	42.33	20.12	32.78	19.92	33.40
QL+SubQL(K)	26.76	53.15	26.19	52.35	26.20	43.21	25.70	42.93	19.94	33.20	19.82	33.30
SubDM(1)	28.17	53.49	27.32	54.30	26.56	42.69	24.93	41.29	20.26	33.92	21.52	**35.98**
SubDM(K)	28.60	53.76	**28.69**	55.17	27.07	43.69	26.17	43.13	20.70	34.74	21.36	**35.98**
DM+SubQL(1)	28.56	**55.91**	28.12	54.97	27.36	**45.42**	26.19	43.94	21.94	**35.26**	20.86	34.33
DM+SubQL(K)	**28.70**	55.37	28.47	**55.30**	**27.37**	45.14	**26.56**	**44.14**	**22.17**	35.15	**21.70**	34.23



(a) Gov2 *(b) Robust04* *(c) Wt10g*

Figure 2: The influence of the number of sub-queries (k)

labeling problem. A novel Conditional Random Field model is proposed to capture the local and global dependencies underlying the verbose query and to directly optimize the retrieval performance on the training set. Four types of retrieval models are proposed to incorporate sub-queries. Experiments show that the proposed CRF model successfully selects high-quality sub-queries for different retrieval models. The best performance is observed when the selected sub-queries are combined with the original query. Besides generating sub-query distribution, the CRF-perf model is also promising to generate reformulated-query distribution. Studying how to combine these two types of distributions within the same CRF-based framework will be an interesting future issue.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant #IIS-0711348, and in part by NSF grant #IIS-0534383. 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. We appreciate Kumaran and Carvalho [11] and Bendersky et al [3] to provide the codes and features of their work.

8. REFERENCES

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

[2] M. Bendersky and W. B. Croft. Analysis of long queries in a large scale search log. In *Workshop of WSDM09*, pages 8–14, Barcelona, Spain, 2009.

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

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

[5] B. He and I. Ounis. Inferring query performance using pre-retrieval predictors. In *SPIRE04*, pages 43–54, 2004.

[6] G. E. Hinton. Training products of experts by minimizing contrastive divergence. Technical report, Gatsby Computational Neuroscience Unit, 2000.

[7] T. Joachims. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT-Press, 1999.

[8] T. Joachims. Optimizing search engines using clickthrough data. In *SIGKDD02*, pages 133–142, Alberta, Canada, 2002.

[9] D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *ACL03*, pages 423–430, Sapporo, Japan, 2003.

[10] G. Kumaran and J. Allan. A case for shorter queries, and helping users creat them. In *ACL07*, pages 220–227, Rochester,New York, 2007.

[11] G. Kumaran and V. R. Carvalho. Reducing long queries using query quality predictors. In *SIGIR09*, pages 564–571, Boston, MA, 2009.

[12] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *ICML01*, pages 282–289, Williamstown, MA, 2001.

[13] M. Lease. An improved Markov random field model for supporting verbose queries. In *SIGIR09*, pages 476–483, Boston,MA, 2009.

[14] M. Lease, J. Allan, and W. B. Croft. Regression rank: learning to meet the oppotunity of descriptive queries. In *SIGIR05*, pages 472–479, Salvador,Brazil, 2005.

[15] X. Li, Y.-Y. Wang, and A. Acero. Extracting structured information from user queries with semi-supervised conditional random fields. In *SIGIR09*, pages 572–579, Boston, MA, 2009.

[16] M.-C. Marneffe and C. D. Manning. *Stanford typed dependencies manual*. Stanford.

[17] A. McCallum and F. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *HLT-NAACL03*, pages 188–191, Edmonton, Canada, 2003.

[18] 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.

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

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

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

[22] T. Qin, T.-Y. Liu, X.-D. Zhang, D.-S. Wang, and H. Li. Global ranking using continuous conditional random fields. In *NIPS08*, 2008.

[23] S. Robertson. On gmap and other transformations. In *CIKM05*, pages 78–83, Arlington, VA, 2005.

[24] Y. Z. S. Cronen-Townsend and W. B. Croft. Predicting query performance. In *SIGIR02*, pages 299–306, New York, NY, 2002.

[25] B. Settles. Abner: an open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics*, 21(2):237–242, 2005.

[26] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *HLT-NAACL03*, pages 134–141, Edmonton, Canada, 2003.

[27] C. Sutton and A. McCallum. *An introduction to conditional random fields for relational learning*. MIT Press, 2006.

[28] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *UAI02*, 2002.

[29] 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.