A Unified and Discriminative Model for Query Refinement

Jiafeng Guo[†*], Gu Xu[‡], Hang Li[‡], Xueqi Cheng[†]

[†]Institute of Computing Technology, CAS
Beijing, P.R.China
guojiafeng@software.ict.ac.cn,cxq@ict.ac.cn

[‡]Microsoft Research Asia
Beijing, P.R.China
{guxu, hangli}@microsoft.com

ABSTRACT

This paper addresses the issue of query refinement, which involves reformulating *ill-formed* search queries in order to enhance relevance of search results. Query refinement typically includes a number of tasks such as spelling error correction, word splitting, word merging, phrase segmentation, word stemming, and acronym expansion. In previous research, such tasks were addressed separately or through employing generative models. This paper proposes employing a unified and discriminative model for query refinement. Specifically, it proposes a Conditional Random Field (CRF) model suitable for the problem, referred to as Conditional Random Field for Query Refinement (CRF-QR). Given a sequence of query words, CRF-QR predicts a sequence of refined query words as well as corresponding refinement operations. In that sense, CRF-QR differs greatly from *conventional* CRF models. Two types of CRF-QR models, namely a basic model and an extended model are introduced. One merit of employing CRF-QR is that different refinement tasks can be performed simultaneously and thus the accuracy of refinement can be enhanced. Furthermore, the advantages of discriminative models over generative models can be fully leveraged. Experimental results demonstrate that CRF-QR can significantly outperform baseline methods. Furthermore, when CRF-QR is used in web search, a significant improvement of relevance can be obtained.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Performance, Theory

Keywords

Query Refinement, Conditional Random Fields, Web Search

*This work was conducted at Microsoft Research Asia.

SIGIR'08, July 20–24, 2008, Singapore.

1. INTRODUCTION

In modern Information Retrieval (IR), search is formalized as a problem of document ranking based on degree of matching between query terms and document terms. Therefore, how to resolve a mismatch between query terms and document terms becomes one of the biggest challenges for IR. For example, if a document contains "New York Times" while the user types "ny times", typically the document would not be retrieved at a search system. Spink et al. [23] observe that users have to reformulate their search queries 40% to 52% of the time in order to find what they want. In fact, many ill-formed queries can be found from the query logs of web search engines. Queries may contain misspelled words, mistakenly split words, or mistakenly merged words. Moreover, queries may include phrases that should be quoted (using the quotation operator), words that should be properly stemmed, or acronyms that should be expanded.

The question then becomes whether we can offer a solution during search which automatically reformulates queries, in order to better represent users' search needs and help users more easily find the relevant information. This is what we mean by query refinement as address in this paper. Note that for simplicity we only consider replacing the original query with the refined query in search (e.g., changing "ny times" to "new york times" and searching with it), but not combining the two.

There is previous work on query refinement. However, the issue was tackled in separate tasks or by employing generative models. For example, Li et al. conducted spelling error correction for web search by using a Maximum Entropy model as well as the Source Channel model [15]. Peng et al. performed automatic word stemming for web search by means of a Statistical Language model [17].

We propose exploiting a unified and discriminative model in query refinement, specifically (1) conducting various query refinement tasks in a unified framework, and (2) employing a special CRF model called CRF-QR to accomplish the tasks.

One advantage of employing CRF-QR is that the accuracy of query refinement can be enhanced. This is because the tasks of query refinement are often mutually dependent, and need to be addressed at the same time, e.g., refining the query "paper on machin learn" to "paper on 'machine learning"'. Alternatively, we could employ a cascaded approach in which we perform the refinement tasks one by one. However, the accuracy of this approach might not be high, because the dependencies between the tasks cannot be handled properly and errors can be accumulated through the refinement processes.

Query refinement is by nature a structured prediction problem which seeks to predict the latent structure of an observation sequence. Therefore, when we employ a discriminative model in query refinement, we enjoy all its advantages as compared with employing a generative model. A straightforward application of existing models, for example conventional CRF, would not work, because the output space would necessarily become extremely large.

We instead propose a new CRF model for query refinement, referred to as CRF-QR. The CRF-QR model is largely different from conventional CRF. Given a sequence of query words, CRF-QR predicts a sequence of refined query words as well as corresponding refinement operations. The refinement operations are defined for the query refinement tasks. We consider two types of CRF-QR; one is called basic model and the other extended model. The former is used when only one refinement task can be applied to a word, while the latter is used when multiple refinement tasks can be applied to a word (e.g., "learm" to "learn" and then "learning"). In learning, we employ Maximum Likelihood Estimation to estimate the parameters of the model. In prediction, we employ the Viterbi Algorithm to find the best sequence of refined query words.

Without loss of generality, we consider spelling error correction, word merging, word splitting, and phrase segmentation as example refinement tasks in our experiments. Experimental results show that our approach (CRF-QR) significantly outperforms the cascaded approach and generative approach to query refinement. Furthermore, when used in search, our approach can help significantly improve relevance.

2. RELATED WORK

Query refinement is a problem already identified and studied in IR (cf., [27]). Much of the previous work, however, only focused on one task of refinement or resorted to generative models. Li et al. [15] proposed a method for spelling error correction by using a Maximum Entropy (ME) model as well as the Source Channel model. They utilized distributional similarities between the query word and its correction candidate as features in the ME model. Cucerzan and Brill [6] addressed a more generalized spelling correction task using a Source Channel model and query log data. They made use of bigrams as the source model and Weighted Edit Distance as the channel model. However, it is less possible to extend their approach to handle other alteration types, e.g. phrase segmentation. Peng et al. [17] proposed a method for conducting stemming on head words of queries. They employed a Statistical Language model in stemming candidate selection. Risvik et al. [20] proposed a method for phrase segmentation using the so-called "connexity measures". A "segmentation score" is computed from connexity values and used as a criterion for segmentation. (See also [1][5][3]). Table 1 gives a summary on the previous work.

There are several other problems relevant to query refinement, although they are slightly different. In query expansion one adds new terms to the query to overcome the term mismatch problem [4][18]. There is no assumption that errors exist in the queries submitted by users. The so-called global analysis [8][10] and local analysis [21][22][28] are usually used in query expansion. In query suggestion we present related queries to the user to enhance his/her search experience [2][7][13]. The suggested queries can be semantically different from the input query. Search log data is usually used in query suggestion. Query substitution by Jones et al. [12] is similar to query refinement. The key idea of the work was to replace the current query with a new query that can improve search relevance, mined from search log data. Two models (linear classification and linear regression) were trained by using labeled data and employed in the substitution decision. The major difference between query substitution and query refinement is that the former is used to consider *inter* query relations, while the latter considers *intra* query relations.(See also [11].)

Table 1: Previous Work on Query Refinement

Work	Task	Approach
[6][1][15]	spelling correction	generative
[15][5]	spelling correction	discriminative
[17]	word stemming	generative
[20]	phrase segmentation	generative
[3]	phrase segmentation	discriminative

Structured prediction is a problem in machine learning, in which given an observation sequence we are to predict the latent structure of the sequence. Many application tasks in information extraction, natural language processing, bioinformatics, and speech recognition can be formalized as structured prediction. One approach to the problem is to employ a discriminative model. For example, Lafferty et al. proposed using CRF [14] for structured prediction. CRF is a conditional probability distribution of structure given observation sequence. The most widely used CRF model is the one that predicts a label sequence (structure) given an observation sequence. Maximum Likelihood Estimation can be used in learning and the Viterbi Algorithm in prediction. Taskar et al. [25] and Tsochantaridis et al. [26] addressed the structured prediction issue by means of Support Vector Machine (SVM). Another approach to structured prediction is to employ a generative model. A typical model in this approach is Hidden Markov Model (HMM)[19], which represents a joint probability distribution of observation sequence and structure. Discriminative learning offers several advantages when compared with generative learning. Learning is equivalent to modeling the mapping relations from observation sequence to structure but not to estimating the joint probability distribution between observation sequence and structure. Thus, it can be conducted more effectively toward the goal of enhancing prediction performance. Moreover, it is easier to incorporate various features into the model. As a result, a discriminative model can usually achieve better accuracies than a generative model. However, exploiting a generative model also has its merits. Usually there is no need to use labeled data in training; creating such data is costly anyway.

CRF-QR proposed in this paper is designed for query refinement and differs greatly from existing CRF models. The conventional CRF model is defined on a label sequence (a chain) and is used for sequence data labeling [14]. Dynamic CRF proposed by Sutton et al. is somewhat similar to CRF-QR, in which multiple labels can be assigned to an observation in the observation sequence [24]. However, Dynamic CRF does not have the operations that CRF-QR has.

3. QUERY REFINEMENT

Query refinement is a problem as follows. Given an ill-

Table 2: Examples of Query Refinement

Original Query	Refined Query
sytem requirement	system requirement
you tube	youtube
universityof california	"university of california"
data mine	"data mining"
the office show	"the office" show
on line book store	online bookstore
papers on machin learn	papers on "machine learning"
system of a dowm	"system of a down"
las vegas cart race	"las vegas" "kart racing"
south sea port new york	south seaport "new york"
chicargo news paper	chicago newspaper

formed query from the user, we refine the query and help the user to better retrieve documents. A number of refinement tasks are involved: spelling error correction, word splitting, word merging, phrase segmentation, word stemming, and acronym expansion. Table 2 shows some examples of query refinement. For instance, if the query typed by the user is "papers on machin learn", then it is very likely that no relevant documents will be retrieved. This is because the spelling error "machin" should be corrected, the word "learn" should be stemmed as "learning", and it is better to group the phrase "machine learning" together (use the quotation operator).

As can be seen from Table 2, refinement tasks are often mutually dependent. In the above example, stemming on "learn" needs help from spelling error correction on "machin", and vice versa. Furthermore, phrase segmentation on "machine learning" depends on the stemming and the spelling error correction, and vice versa. Therefore, it is better to employ a unified framework with which we perform all the refinement tasks simultaneously. In this way, we are able to significantly boost the accuracy of query refinement.

Query refinement is by nature a structured prediction problem. Specifically, given a sequence of query words, we predict the most likely sequence of refined query words. Given enough labeled training data, we are able to train a reasonably effective model by using discriminative learning.

4. OUR APPROACH

We propose employing a unified and discriminative model in query refinement, specifically the CRF-QR model. There are two types of models: a basic model and an extended model. For ease of explanation, we explain the former model first. In our experiments we used the latter model.

4.1 Basic Model

Let us first look at the basic CFR-QR model. In the basic model, we assume that given a query word only one refinement task (e.g., spelling error correction or word stemming) can be applied.

Let $\boldsymbol{x}$ denote a sequence of query words, and let $\boldsymbol{y}$ denote a corresponding sequence of refined query words. Without loss of generality, we assume that the sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same length. (When it is not the case, we can normalize them to the longest.) We use $\boldsymbol{x} = x_1 x_2 \dots x_n$ and $\boldsymbol{y} = y_1 y_2 \dots y_n$ to denote the words in the query and in the refined query, respectively. Here n denotes the length of the sequences.

A straightforward approach would be to employ a conditional probability model $\Pr(\boldsymbol{y}|\boldsymbol{x})$ for query refinement. The conditional probability model can be defined as a conventional CRF model on a chain, in which adjacent y's are dependent on each other and individual y's are dependent on all the x's. It is depicted in the graphical model in Figure 1(a) (for ease of presentation, we show the case in which individual y's only depend on the corresponding x's). As a result, the query refinement task becomes that of finding the sequence $\boldsymbol{y}^*$ satisfying $\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} \Pr(\boldsymbol{y}|\boldsymbol{x})$ where the conditional probability is calculated with the CRF model. However, the learning of such a model would be intractable, because the space of y is as large as the space of x (this means that any word can be mapped to any other word) and a large amount of data is needed for training.

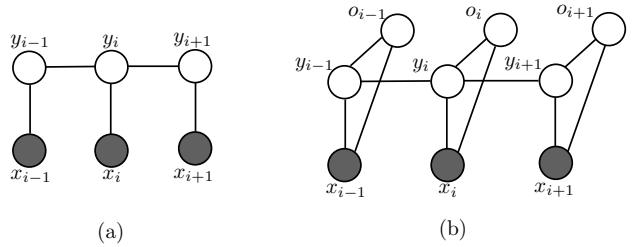

Figure 1: Graphical representation of (a) conventional CRF, and (b) Basic CRF-QR

The key idea in CRF-QR is to introduce operations and incorporate the operations into the conditional probability model. In this way, we can make the construction and utilization of the model feasible, as explained below. For the task of spelling error correction, for example, we can consider the following refinement operations: deletion, insertion, substitution, and transposition. For word stemming, we can introduce operations such as $+s$, $+ed$, and $+ing$. Similarly, we can define operations for other refinement tasks, as described in Table 3. (Note that the operations in a task are mutually exclusive.) In fact, more complicated refinement operations can be easily incorporated in our model.

We let $\boldsymbol{o}$ denote a sequence of refinement operations $\boldsymbol{o} = o_1 o_2 \dots o_n$. We employ a conditional model $\Pr(\boldsymbol{y}, \boldsymbol{o}|\boldsymbol{x})$ in query refinement. We call the model CRF-QR. We assume that adjacent y's are mutually dependent, o's are independent from one another, and individual y's are dependent on corresponding o's and all the x's. Note that it is not necessary to assume that the dependency exists between o's because it is already somehow captured by the dependency between y's; this also makes the model simple. CRF-QR is actually a graphical model, in which vertexes represent refined query words, edges dependencies, and conditional vertexes input query words, as depicted in Figure 1(b) (for ease of presentation, we show the case in which individual y's only depend on the corresponding o's and x's). Therefore, query refinement becomes the task of finding the sequence $\boldsymbol{y}^*$ satisfying $\boldsymbol{y}^*\boldsymbol{o}^* = \arg\max_{\boldsymbol{yo}} \Pr(\boldsymbol{y}, \boldsymbol{o}|\boldsymbol{x})$.

In general, a graphical model can be written as a product of potential functions over the maximal cliques of the graph. In our case, there are two types of maximal cliques in the graph. One is the edge between y_{i-1} and y_i, $i = 1, 2...n$. (Here we assume that there is a start node y_0). The other is the edge between y_i and o_i conditioned on $\boldsymbol{x}$, $i = 1, 2...n$.

Table 3: Query Refinement Tasks and Corresponding Operations

Task	Operation	Description
Spelling Correction	Deletion	Delete a letter in the word
	Insertion	Insert a letter into the word
	Substitution	Replace a letter in the word with another letter
	Transposition	Switch two letters in the word
Word Splitting	Splitting	Split one word into two words
Word Merging	Merging	Merge two words into one word
Phrase Segmentation	Begin	Mark a word as beginning of phrase
	Middle	Mark a word as middle of phrase
	End	Mark a word as end of phrase
	Out	Mark a word as out of phrase
Word Stemming	+s/-s	Add or Remove suffix '-s'
	+ed/-ed	Add or Remove suffix '-ed'
	+ing/-ing	Add or Remove suffix '-ing'
Acronym Expansion	Expansion	Expand acronym

Therefore, we have

$$\Pr(\boldsymbol{y}, \boldsymbol{o}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{i=1}^{n} \phi(y_{i-1}, y_i)\phi(y_i, o_i, \boldsymbol{x}) \quad (1)$$

where $\phi(y_{i-1}, y_i)$ is the potential function associated with edge y_{i-1} and y_i, $\phi(y_i, o_i, \boldsymbol{x})$ is the potential function associated with edge y_i and o_i, and Z is the normalizing factor. The potential functions are strictly positive and real valued functions.

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y},\boldsymbol{o}} \prod_{i=1}^{n} \phi(y_{i-1}, y_i)\phi(y_i, o_i, \boldsymbol{x})$$

Here we assume that the potential functions are exponential functions. Then we have

$$\phi(y_{i-1}, y_i) = \exp(\sum_k \lambda_k f_k(y_{i-1}, y_i)) \quad (2)$$

$$\phi(y_i, o_i, \boldsymbol{x}) = \exp(\sum_k \lambda_k h_k(y_i, o_i, \boldsymbol{x})) \quad (3)$$

where $f_k(y_{i-1}, y_i)$ and $h_k(y_i, o_i, \boldsymbol{x})$ denote features and λ_k denotes parameter. Substituting Equation (2) and (3) into (1), we obtain the basic CRF-QR model

$$\Pr(\boldsymbol{y}, \boldsymbol{o}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp(\sum_{i=1}^{n}(\sum_k \lambda_k f_k(y_{i-1}, y_i) + \sum_k \lambda_k h_k(y_i, o_i, \boldsymbol{x})))$$

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y},\boldsymbol{o}} \exp(\sum_{i=1}^{n}(\sum_k \lambda_k f_k(y_{i-1}, y_i) + \sum_k \lambda_k h_k(y_i, o_i, \boldsymbol{x})))$$

The refinement (refined word) at each position y_i has a dependency on the operation at position o_i and the entire input query $\boldsymbol{x}$. In CRF-QR, we use features $h(y_i, o_i, \boldsymbol{x})$ to represent the dependency relationship.

It is supposed that there exists dependency between adjacent words in $\boldsymbol{y}$. This is reasonable because dependency should only exist between refined 'true' words. Instead, we do not need to assume that dependency exists between operations in $\boldsymbol{o}$, because that has already been reflected in $\boldsymbol{y}$. The independency assumption in $\boldsymbol{o}$ will also make the construction and utilization of the model efficient. Moreover, we do not need to model the dependency between ill-formed words in $\boldsymbol{x}$, because correct inputs are all alike but every incorrect input may be incorrect in its own way. In CRF-QR, we use features $f(y_{i-1}, y_i)$ to represent the dependency relationship.

The CRF-QR model is suitable for query refinement and is easy to learn. With the embedding of the refinement operations in the model, the number of parameters in CRF-QR $P(\boldsymbol{y}, \boldsymbol{o}|\boldsymbol{x})$ becomes significantly smaller than that in the conventional CRF $P(\boldsymbol{y}|\boldsymbol{x})$. First, the mapping from x's to y's will not be completely free. Given x and o, the possible y's will become limited. For example, if x and o are "machin" and insertion operation in spelling error correction respectively, then y can only be "machina", "machine", etc. It turns out, therefore, that o works as a *constraint* in the model to drastically reduce the space of y for given x. Next, since the number of operations is small (each task only has several operations), we can index the feature $h(y, o, \boldsymbol{x})$ by indexing o. The learning of CRF-QR, then, becomes very efficient.

4.2 Learning and Prediction

In learning, we assume that labeled data $(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}, \boldsymbol{o}^{(1)})$, ..., $(\boldsymbol{x}^{(N)}, \boldsymbol{y}^{(N)}, \boldsymbol{o}^{(N)})$ is given, where $\boldsymbol{x}^{(i)}$ denotes a query, $\boldsymbol{y}^{(i)}$ a refined query, and $\boldsymbol{o}^{(i)}$ a sequence of operations $i = 1, \ldots, N$. Given the training data, we maximize the regularized log-likelihood function of the training data with respect to the model, and then obtain the parameter $\hat{\lambda}$.

$$\hat{\lambda} = \arg\max_{\lambda} \left\{ \sum_{i=1}^{N} \log(\Pr_{\lambda}(\boldsymbol{y}^{(i)}, \boldsymbol{o}^{(i)}|\boldsymbol{x}^{(i)})) - C\|\lambda\|_2 \right\}$$

where C denotes coefficient and $\|\cdot\|_2$ denotes L_2 norm. In this paper, we employ Quasi-Newton Method to conduct the optimization, specifically we use the L-BFGS algorithm [16]. Because the log-likelihood function is convex, it is guaranteed that the global optimal solution will be found.

In prediction, given a query $\boldsymbol{x}$ we employ the Viterbi algorithm to find the most likely refined query $\boldsymbol{y}^*$ satisfying

$$\boldsymbol{y}^*\boldsymbol{o}^* = \arg\max_{\boldsymbol{y},\boldsymbol{o}} \Pr(\boldsymbol{y}, \boldsymbol{o}|\boldsymbol{x})$$

4.3 Features

The feature $f(y_{i-1}, y_i)$ represents the relation on adjacent words y_{i-1} and y_i in the refined query. We can define it as

$$f(y_{i-1}, y_i) = \log \Pr(y_i|y_{i-1})$$

where $\Pr(y_i|y_{i-1})$ denotes the conditional probability of observing y_i given y_{i-1} in a corpus. We can use, for example, bigrams in web pages or query logs to estimate the probabilities. These data are much larger than labeled data for training CRF-QR, and thus more accurate estimation can be obtained.

The feature $h(y_i, o_i, \boldsymbol{x})$, indexed by o_i, represents the refinement operation of o_i from x_i to y_i, conditioned on $\boldsymbol{x}$. It is defined as a binary feature in this paper:

$$h(y_i, o_i, \boldsymbol{x}) = \begin{cases} 1, & \text{certain condition satisfied, given } o_i \\ 0, & \text{otherwise} \end{cases}$$

Note that $h(y_i, o_i, \boldsymbol{x})$ can be simplified as $h(y_i, o_i, x_i)$, $h(y_i, o_i, \_)$, or $h(\_, o_i, x_i)$. For example, whether the frequency of query word x_i is higher than the frequency of refined word y_i in the corpus when o_i is insertion, or whether the refined word y_i is in the lexicon when o_i is deletion, or whether the query word x_i is a stop word when o_i is substitution.

In this paper, we make use of the following features:

Lexicon-based feature representing whether a query word or a refined query word is in a lexicon or a stopword list.

Position-based feature representing whether a query word is at the beginning, middle, or end of the query.

Word-based feature representing whether a query word consists of digit, alphabet, or a mix of the two, and whether the length of a query word is in a certain range.

Corpus-based feature representing whether the frequency of a query word or a refined query word in the corpus exceeds a certain threshold.

Query-based feature representing whether the query is a single word query or multi-word query.

4.4 Extended Model

If we assume that only one refinement task can be applied to a query word, then we may directly employ the basic CRF-QR model. In practice, multiple refinement tasks may be needed for a query word (e.g., spelling error correction and word stemming). Thus the use of the basic model is not enough.

We consider the use of an extended CRF-QR model in this case. In the extended model, we use multiple sequences of operations as well as their corresponding sequences of intermediate results. In this way, we keep the number of parameters the same as that in the basic CRF-QR model.

Suppose that for the query word x_i, we have multiple sequences of operations on it, resulting in different refined words y_i's. One sequence of operations is denoted as $\vec{o}_i = o_{i,1}o_{i,2}, \cdots, o_{i,m_i}$ where m_i stands for number of operations in the sequence. Among the operations, one operation is from one task and generates one intermediate result (note that each task can only be applied once in each sequence). For the sequence of operations $\vec{o}_i = o_{i,1}o_{i,2}, \cdots, o_{i,m_i}$ we have a sequence of intermediate results $\vec{z}_i = z_{i,1}z_{i,2}z_{i,m_{i-1}}$. Here for convenience, we let $z_{i,0} = x_i$ and $z_{i,m_i} = y_i$. Note that m_i can be different for different positions.

We retain all the possible sequences of refinement operations on a query word and make a prediction at the final stage. Even though the number of possible sequences of refinement operations is of exponential order, we can still handle them when the number of tasks is small, which is the case in this paper. When the number of tasks becomes large, we can use heuristics to reduce the number of possible sequences. We leave it as future work.

Again we assume that the operations are mutually independent, while the adjacent words in the final output query are mutually dependent. Figure 2 shows the relationships.

Similarly, we define the conditional probability distribution

$$\Pr(\boldsymbol{y}, \vec{\boldsymbol{o}}, \vec{\boldsymbol{z}}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{i=1}^{n} (\phi(y_{i-1}, y_i) \prod_{j_i=1}^{m_i} \phi(z_{i,j_i}, o_{i,j_i}, z_{i,j_i-1}))$$

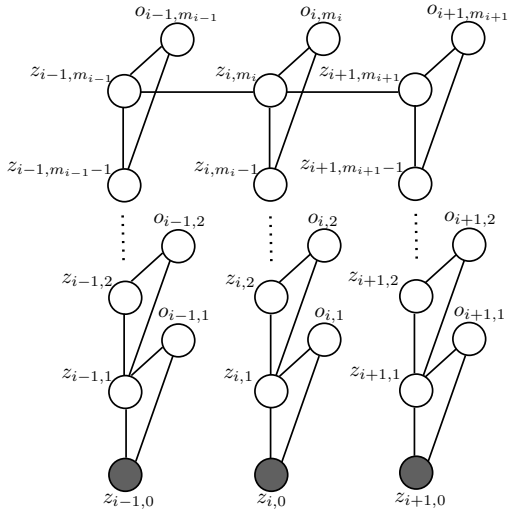

Figure 2: Graphical representation of Extended CRF-QR

Table 4: Example Queries

seamans	free online games
Jessica Sinpson	BALLBEARINGS
gulf shores	babiesrus
University of Georgia	c-172 rg fuel system
what is #1 city	kelly blue book
Merriam webster	room by room
portland museum of art	soap city
listings of human growth homones used	weems
Inn at Little Washington hotel	bridesmaid's dresses
2pac arts centre	vw for sale

as the extended CRF-QR model and employ it in query refinement. Here, Z is the normalizing factor, and we have

$$\phi(y_{i-1}, y_i) = \exp(\sum_k \lambda_k f_k(y_{i-1}, y_i))$$

$$\phi(z_{i,j_i}, o_{i,j_i}, z_{i,j_i-1}) = \exp(\sum_k \lambda_k h_k(z_{i,j_i}, o_{i,j_i}, z_{i,j_i-1}))$$

We can define features in a similar way as we do in the basic model. We can also use the same learning and prediction methods in the basic model for the extended model. Note that in prediction a refined word can be obtained by multiple ways of combining operations (e.g., first splitting then insertion, or first insertion then splitting). In such cases, we just adopt the one in the predicted result.

5. EXPERIMENTAL RESULTS

5.1 Data Set

In our experiments we made use of a real data set consisting of 10,000 queries. The queries were randomly selected from the query log of a commercial web search engine. The average length of queries is 2.8 words. Table 4 shows 20 randomly selected queries.

We asked four human annotators to manually refine the ill-formed queries. We considered four types (tasks) of refinement and the human annotators only made corrections belonging to these types. The four types are spelling error correction, word merging, word splitting, and phrase segmentation (adding quotation operators to phrases). This is

Table 5: Statistics on Annotated Data

Refinement Task	Num. of Refined Queries
Spelling Correction	733
Word Splitting	221
Word Merging	323
Phrase Segmentation	5,876

because for those types, we were able to define a clear guideline. We leave the annotation of other types of refinement such as word stemming as future work. Note that for word stemming it is not easy to make a refinement judgment, because the effectiveness of a refinement also depends on the contents of document collection. Even with four types of refinement, we are still able to verify the effectiveness of the model we propose in this paper. In the annotation, if there was a disagreement among the annotators, we took a majority vote.

Table 5 gives some statistics of the annotated data. Among the 10,000 queries, 6,421 queries are refined. Note that the majority of the refinements are phrase segmentation. Furthermore, there are 649 queries with multiple refinements. The labeled data were further divided into a training set containing 7,000 queries and a test set containing 3,000 queries.

For language model estimation, we made use of a collection of 50 million web pages. Texts were extracted from the web pages and bigrams were then created from the data.

5.2 Baseline Methods

We adopted the cascaded approach and the generative approach as baselines.

The cascaded approach builds one sub-model for each of the four refinement tasks. The sub-models have the same model structure and feature set as the model in our approach; the only difference is that each of them only addresses one type of refinement. In testing, we sequentially connect the sub-models in different orders. In our experiments, we put phrase segmentation at the end (it seems reasonable to do so) and all the other three tasks in different permutations at the beginning. This gives us six options for the cascaded approach, denoted as Cascaded1∼Cascaded6.

The generative approach exploits a Source Channel model and Mutual Information in query refinement. For spelling error correction, word splitting, and word merging, we use the Source Channel model. We assume that the model has equal translation probabilities (channel model) and we only use language probabilities (source model) in refined query selection. For phrase segmentation, we use Mutual Information to identify phrases [12] [17]. Specifically, we compute a Mutual Information score for each pair of adjacent words using the corpus data; if it exceeds a predefined threshold, we skip it, otherwise, we set a phrase boundary at the position. Here the threshold is tuned to be optimal on the training set.

5.3 Experiment on Query Refinement

In this experiment, we compared the performances of different approaches to query refinement. We made the evaluations at query level (note that a query may have several refinements) and used precision (Pre.), recall (Rec.), F1 score (F1), and Accuracy (Acc.) as evaluation measures.

Table 6 shows the results of query refinement by our approach (extended CRF-QR model) and the baseline methods. From the results in Table 6, we can see that CRF-QR outperforms the baseline methods in terms of all measures. When compared with the best of the baseline, the relative improvement by CRF-QR is 2.26% and 1.21% in terms of F1 and accuracy respectively. We conducted a sign test on the results, which shows that all the improvements are statistical significant (p-value < 0.01).

Table 6: Comparisons between CRF-QR and Baselines on Query Refinement at Query Level (%)

	Pre.	Rec.	F1	Acc.
CRF-QR	**54.48**	**40.75**	**46.63**	**56.27**
Cascaded1	53.38	39.71	45.54	55.57
Cascaded2	53.38	39.71	45.54	55.57
Cascaded3	53.38	39.71	45.54	55.57
Cascaded4	53.45	39.76	45.60	55.60
Cascaded5	53.45	39.76	45.60	55.60
Cascaded6	53.45	39.76	45.60	55.60
Generative	30.46	32.95	31.66	39.10

Table 7 gives the results of different refinement tasks of CRF-QR and the baseline methods. From the results, we can observe the same tendencies, that is, CRF-QR performs the best.

CRF-QR works better than all the cascaded methods, no matter which order we use for cascading. The result suggests that it is better to conduct query refinement tasks in a unified framework. The performance of the generative approach is the lowest, indicating that using only a Statistical Language model and Mutual Information in query refinement is not sufficient. With the use of additional features and a discriminative model, CRF-QR can achieve a much better performance than the generative approach.

5.4 Discussion

We conducted analysis on the refinement results to see the differences between CRF-QR and the baseline methods. It seems that the cascaded approach suffers from the neglect of mutual dependencies between query refinement tasks. For example, it could not correctly refine the query "nypark hitel" to "ny 'park hotel"'. Specifically, it could not simultaneously change "hitel" to "hotel" and change "nypark" to "ny park", because one operation needed to leverage the result of the other. Furthermore, in the cascaded approach, an error made at an early stage can be propagated to the later stages. In one option, the cascaded approach conducted spelling error correction first, and then word splitting, word merging, and phrase segmentation. It incorrectly altered the query "bankin las vegas" into "banking 'las vegas"', while the ideal output would be "bank in 'las vegas"'. It failed in that case, because "bankin" was incorrectly changed to "banking" at the beginning and there was no chance to reverse the decision later.

It seems that one can carry out the learning for query refinement more effectively by training a unified model (CRF-QR) at one time than from training several sub-models separately. We used the sub-models in the cascaded approach separately for the corresponding tasks in testing (note that in the cascaded approach the sub-models were trained independently in training and cascaded together in testing). We then forcibly replaced the parameters in the sub-models with the corresponding parameters in the unified model (CRF-QR), and used the new sub-models in the testing again. We

Table 7: Comparisons between CRF-QR and Baselines on Query Refinement Tasks (%)

	Spelling Correction			Word Splitting			Word Merging			Phrase Segmentation		
	Pre.	Rec.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1
CRF-QR	77.16	71.84	**74.40**	76.47	76.47	**76.47**	88.89	82.35	**85.50**	69.21	50.78	**58.58**
Cascaded1	73.91	68.39	71.04	74.19	67.65	70.77	86.67	76.47	81.25	69.00	50.13	58.07
Cascaded2	73.91	68.39	71.04	74.19	67.65	70.77	86.67	76.47	81.25	69.00	50.13	58.07
Cascaded3	74.68	67.43	70.87	70.59	70.59	70.59	86.67	76.47	81.25	69.01	50.16	58.09
Cascaded4	75.16	67.43	71.09	70.59	70.59	70.59	86.89	77.94	82.17	69.01	50.16	58.09
Cascaded5	74.38	68.39	71.26	74.19	67.65	70.77	86.89	77.94	82.17	69.00	50.13	58.07
Cascaded6	75.16	67.43	71.09	70.59	70.59	70.59	86.89	77.94	82.17	69.01	50.16	58.09
Generative	30.86	92.57	46.29	39.06	59.52	47.17	34.44	84.93	49.01	57.36	53.47	55.35

Table 8: Comparisons between Unified and Independent Training on Refinement Tasks (%)

		Pre.	Rec.	F1
Spelling Correction	Unified	75.16	69.54	**72.24**
	Independent	73.91	68.39	71.04
Word Splitting	Unified	72.22	76.47	**74.29**
	Independent	70.59	70.59	70.59
Word Merging	Unified	83.82	83.82	**83.82**
	Independent	86.89	77.94	82.17
Phrase Segmentation	Unified	67.67	51.13	**58.25**
	Independent	67.47	50.71	57.90

Table 9: Results on Relevance Search with Entire Query Set (NDCG@3)

	Before	After
Human (Upper Bound)	0.265	0.304 (+14.7%)
CRF-QR	0.265	0.288 (+8.7%)

Table 10: Results on Relevance Search with Refined Queries (NDCG@3)

	Refined	Before	After
Human (Upper Bound)	2023	0.254	0.312 (+22.8%)
CRF-QR	1546	0.258	0.304 (+17.7%)

made comparisons between the two cases. Table 8 shows the results. We can see that the use of parameters trained in a unified framework (Unified) outperforms the use of the parameters trained independently (Independent). This seems to indicate that even for addressing individual refinement tasks, it is still better to consider the effects of all the tasks together in training.

The generative approach produced more incorrect results because it could only rely on the Statistical Language model and Mutual Information scores in its prediction. For example, it altered the query "pick up stix" to "pick up six", and altered the query "door to door" to "'door to' door". These types of errors occurred mainly because the information used was insufficient for making correct refinement decisions.

We further made error analysis on CRF-QR. There were three types of errors. (1) Errors were mainly made by one of the refinement tasks. This is also reflected in the experiment results in Table 7; for a few tasks, the recall of CRF-QR is lower than the generative approach. For example, CRF-QR could not recognize the spelling error in the query "parnell roberts", where "parnell" should be "pernell". We may reduce such kinds of errors by either adding new features or increasing the data size for language model training. (2) Another error type, although rare, was caused by a competition between refinement tasks. For example, the query "skate board dudes" was changed to "'skate board' dudes" by CRF-QR, while the ideal output would be "skateboard dudes". CRF-QR could not make a correct refinement because the refined query generated by applying phrase segmentation has a higher probability than the one generated by applying word merging. These sorts of errors are also related to features and training data size. (3) Some queries were difficult to refine even for humans, since they were short and included unknown words. This was particularly true for phrase segmentation. For example, the query "ohio buckeye card" was changed to "'ohio buckeye' card", while the ideal refinement would be "ohio 'buckeye card"'.

5.5 Experiment on Relevance Search

We evaluated CRF-QR in search relevance improvements. As a relevance measure, we utilized the Normalized Discounted Cumulative Gain (NDCG@3)[9].

We submitted the original queries and the refined queries in the test set to a commercial search engine. For each query the system returned 10 to 1,000 results. We asked human annotators to make judgments on the relevance of the top 3 documents with respect to the queries. There are five levels of relevance: perfect, excellent, good, fair, and bad.

Table 9 shows the results. "Before" stands for the results before query refinement and "After" stands for the results after it. The numbers in the parentheses are the relative improvement. The results indicate that when applied in web search, CRF-QR can help significantly improve search relevance. NDCG@3 can be enhanced by 8.7%. Table 10 shows the results on relevance search using only the refined queries. When we only look at the results of the refined queries, CRF-QR can help improve NDCG@3 by about 17.7%. We conducted a t-test on the results, which indicate that all the improvements are statistically significant (p-value<0.01). In the tables we include the search results using human refined queries that work as upper bounds of CRF-QR.

We also investigate the influence of each refinement task. Table 11 shows the relevance search results by refinement tasks. We can see that all refinement tasks can help improve the relevance. Spelling error correction seems to be most effective, even though the number of affected queries is not very large. Phrase segmentation affects more queries, even though its improvement on relevance is the smallest. T-test results show that all the improvements are statistically significant (p-value<0.01).

Table 11: Results on Relevance Search by Query Refinement Tasks (NDCG@3)

		Refined	Before	After
Spelling Correction	Human	208	0.093	0.339
	Unified	163	0.078	0.322
Word Splitting	Human	61	0.190	0.333
	Unified	51	0.180	0.294
Word Merging	Human	120	0.198	0.305
	Unified	111	0.207	0.278
Phrase Segmentation	Human	1881	0.281	0.308
	Unified	1351	0.276	0.288

6. CONCLUSION

We have investigated the problem of query refinement, which is changing ill-formed queries for users before submitting them to the search engine. Query refinement includes a number of tasks such as spelling error correction, word splitting, word merging, phrase segmentation, word stemming, and acronym expansion.

Previous work addressed the problem either separately or by using generative models. In this paper we show that employing a unified and discriminative model in query refinement is effective. Since the query refinement tasks usually are mutually dependent, it is better to employ a unified framework to enhance the performance. Furthermore, employing a discriminative model can achieve better performance than employing a generative model if there is enough labeled training data.

We describe our new CRF model for performing query refinement, called CRF-QR. The model is unique in that it predicts a sequence of refined query words as well as corresponding operations given a sequence of query words. There are two variants of the model: a basic model and an extended model.

Experimental results on a large real data set show that our approach significantly outperforms the cascaded approach and generative approach. The results indicate that for query refinement it is better to adopt the CRF-QR model proposed in this paper. Experimental results also show that when using the refined queries with CRF-QR, a significant improvement can be obtained on search relevance.

There are other types of query refinement that have the potential to improve search relevance, but are not investigated in this paper, for example, word stemming and acronym expansion. We plan to investigate these problems in the future. The proposed unified and discriminative model would not be limited to query refinement. It is potentially useful for other query transformation problems, for example, transformation of natural language queries to phrasal queries. We plan to investigate these issues as well.

7. REFERENCES

[1] F. Ahmad and G. Kondrak. Learning a spelling error model from search query logs. In *Proceedings of EMNLP 2005*, pages 955–962, 2005.

[2] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Knowledge Discovery and Data Mining*, pages 407–416, 2000.

[3] S. Bergsma and Q. I. Wang. Learning noun phrase query segmentation. In *Proceedings of EMNLP-CoNLL 2007*, pages 819–826, 2007.

[4] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using smart: Trec 3. In *Text Retrieval Conference*, pages 69–80, 1994.

[5] Q. Chen, M. Li, and M. Zhou. Improving query spelling correction using web search results. In *Proceedings of EMNLP-CoNLL 2007*, pages 181–189, 2007.

[6] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP 2004*, pages 293–300, 2004.

[7] A. Feuer, S. Savev, and J. A. Aslam. Evaluation of phrasal query suggestions. In *Proc. of CIKM '07*, November, 2007.

[8] W. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures & Algorithms*. Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[9] K. Jarvelin and J. Kekalainen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[10] Y. Jing and W. Croft. An association thesaurus for information retrieval. In *Proceedings of RIAO 94*, pages 146–160, 1994.

[11] R. Jones and D. C. Fain. Query word deletion prediction. In *SIGIR*, pages 435–436, 2003.

[12] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW '06*, pages 387–396, 2006.

[13] R. Kraft and J. Zien. Mining anchor text for query refinement. In *WWW '04, New York, USA*, 2004.

[14] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01*, 2001.

[15] M. Li, M. Zhu, Y. Zhang, and M. Zhou. Exploring distributional similarity based models for query spelling correction. In *Proceedings of COLING-ACL 2006*, pages 1025–1032, 2006.

[16] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45:503–528, 1989.

[17] F. Peng, N. Ahmed, X. Li, and Y. Lu. Context sensitive stemming for web search. In *SIGIR '07*, pages 23–27, July 2007.

[18] Y. Qiu and H.-P. Frei. Concept-based query expansion. In *SIGIR*, pages 160–169, 1993.

[19] L. R. Rabiner and B. H. Juang. An introduction to hidden markov models. *IEEE Acoustics, Speech & Signal Processing Magazine*, 3:4–16, 1986.

[20] K. M. Risvik, T. Mikolajewski, and P. Boros. Query segmentation for web search. In *WWW*, 2003.

[21] J. Rocchio. Relevance feedback in information retrieval. In *The Smart Retrieval system—Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, 1971.

[22] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. In *JASIS*, pages 288–297, 1999.

[23] A. Spink, B. J. Jansen, D. Wolfram, and T. Saracevic. From e-sex to e-commerce: Web search changes. *IEEE Computer*, 35(3):107–109, 2002.

[24] C. Sutton, K. Rohanimanesh, and A. McCallum. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *ICML*, 2004.

[25] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *ICML '05, Bonn, Germany*, August 2005.

[26] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[27] B. Vélez, R. Weiss, M. A. Sheldon, and D. K. Gifford. Fast and effective query refinement. In *SIGIR*, pages 6–15, 1997.

[28] J. Xu and B. Croft. Query expansion using local and global document analysis. In *SIGIR*, 1996.