A Discriminative Model for Query Spelling Correction with Latent Structural SVM

Huizhong Duan, Yanen Li, ChengXiang Zhai and Dan Roth

University of Illinois at Urbana-Champaign 201 N Goodwin Ave Urbana, IL 61801 {duan9, yanenli2, czhai, danr}@illinois.edu

Abstract

Discriminative training in query spelling correction is difficult due to the complex internal structures of the data. Recent work on query spelling correction suggests a two stage approach a noisy channel model that is used to retrieve a number of candidate corrections, followed by discriminatively trained ranker applied to these candidates. The ranker, however, suffers from the fact the low recall of the first, suboptimal, search stage.

This paper proposes to directly optimize the search stage with a discriminative model based on latent structural SVM. In this model, we treat query spelling correction as a multiclass classification problem with structured input and output. The latent structural information is used to model the alignment of words in the spelling correction process. Experiment results show that as a standalone speller, our model outperforms all the baseline systems. It also attains a higher recall compared with the noisy channel model, and can therefore serve as a better filtering stage when combined with a ranker.

1 Introduction

Query spelling correction has become a crucial component in modern information systems. Particularly, search engine users rely heavily on the query correction mechanism to formulate effective queries. Given a user query q, which is potentially misspelled, the goal of query spelling correction is to find a correction of the query c that could lead to a better search experience. A typical query spelling correction system employs a noisy channel model (Kernighan et al., 1990). The model assumes that the correct query c is formed in the user's mind before entering the noisy channels, e.g., typing, and get misspelled. Formally, the model maximizes the posterior probability p(c|q):

$$\hat{c} = \arg\max_{c} p(c|q). \tag{1}$$

Applying Bayes rule, the formulation can be rewritten as:

$$\hat{c} = \arg \max_{c} p(q|c)p(c)$$

=
$$\arg \max_{c} [\log p(q|c) + \log p(c)].$$
(2)

The model uses two probabilities. The prior probability p(c) represents how likely it is that c is the original correct query in the user's mind. The probability is usually modeled by a language model estimated from a sizable corpus. The transformation probability p(q|c) measures how likely it is that q is the output given that c has been formed by the user. This probability can be either heuristic-based (edit distance) or learned from samples of well aligned corrections. One problem with the noisy channel model is that there is no weighting for the two kinds of probabilities, and since they are estimated from different sources, there are usually issues regarding their scale and comparability, resulting in suboptimal performance (Gao et al., 2010). Another limitation of this generative model is that it is not able to take advantage of additional useful features.

A discriminative model may solve these problems by adding the flexibility of using features and applying weights. But training such a model is not easy. The difficulty is that the output space of query correction is enormous, as the candidate corrections for each a query term could be the entire vocabulary. This is even worse when word boundary errors (i.e. merging and splitting of words) exist. The problem is intractable with standard discriminative models as we cannot enumerate every candidate correction.

To solve the problem, (Gao et al., 2010) proposed a two stage approach. In this approach, a ranker is trained to score each candidate correction of a query. When a query is issued, the system first uses the noisy channel model with a standard search algorithm to find the 20 best candidates. Then the ranker is used to re-rank these candidates and find the best correction for the query. This ranker based system has one critical limitation, though. Since the ranking stage is decoupled from the search, it relies on the outsourced search algorithm to find the candidates. Because query spelling correction is an online operation, only a small number of candidates can enter the ranker due to efficiency concerns, thus limiting the ability of the ranker to the ceiling of recall set by the suboptimal search phase.

The research question we address here is whether we can directly optimize the search phase of query spelling correction using a discriminative model without loss of efficiency. More specifically, we want 1) a learning process that is aware of the search phase and interacts with its result; 2) an efficient search algorithm that is able to incorporate the learned model and guide the search to the target spelling correction.

In this paper, we propose a new discriminative model for query correction that maintains the advantage of a discriminative model in accommodating flexible combination of features and naturally incorporates an efficient search algorithm in learning and inference. Similarly to (Chang et al., 2010) we collapse a two stage process into a single discriminatively trained process, by considering the output of the first stage as an intermediate latent representation for the joint learning process. Specifically, we make use of the latent structural SVM (LS-SVM) (Yu and Joachims, 2009) formulation. We formulate the problem query spelling correction as a multiclass classification problem on structured inputs and outputs. The advantage of the structural SVM model is that it allows task specific, customizable solutions for the inference problem. This allows us to adapt the model to make it work directly with the search algorithm we use for finding the best correction of the query. To account for word boundary errors, we model the word alignment between the query and the correction as a latent structural variable. The LS-SVM model allows us to jointly search over the output space and the latent structure space.

As the inference algorithm in the proposed discriminative model we use an algorithm that resembles a traditional noisy channel model. To adapt the LS-SVM model to enable the efficient search of query spelling correction, we study how features can be designed. We analyze the properties of features that can be used in the search algorithm and propose a criteria for selecting and designing new features. We demonstrate the use of the criteria by designing separate features for different types of spelling errors (e.g. splitting, merging). With the proposed discriminative model, we can directly optimize the search phase of query spelling correction without loss of efficiency. Our model can be used not only as a standalone speller with high accuracy, but also as a high recall candidate generation stage for a ranker based system.

Experiments verify the effectiveness of the discriminative model, as the accuracy of correction can be improved significantly over baseline systems including an award winning query spelling system. Even though the optimization is primarily based on the top correction, the weights trained by LS-SVM can be used to search for more candidate corrections. The improvement in recall at different levels over the noisy channel model demonstrates that our model is superior even when used in the two-stage approach..

2 Related Work

Spelling correction has a long history (Levenshtein, 1966). Traditional techniques were on small scale and depended on having a small trusted lexicons (Kukich, 1992). Later, statistical generative models were shown to be effective in spelling correction, where a source language model and an error model were identified as two major components

(Brill and Moore, 2000). Note that we are not dealing here with the standard models in context sensitive spelling (Golding and Roth, 1999) where the set of candidate correction is a known "confusion set". Query spelling correction, a special form of the problem, has received much attention in recent years. Compared with traditional spelling correction task, query spelling deals with more complex types of misspellings and a much larger scale of language. Research in this direction includes utilizing large web corpora and query log (Chen et al., 2007; Cucerzan and Brill, 2004; Ahmad and Kondrak, 2005), employing large-scale n-gram models, training phrase-based error model from clickthrough data (Sun et al., 2010) and developing additional features (Gao et al., 2010).

Query alteration/refinement is a very relevant topic to query spelling correction. The goal of query alteration/refinement is to modify the ineffective query so that it could . Researches on this track include query expansion (Xu and Croft, 1996; Qiu and Frei, 1993; Mitra et al., 1998), query contraction(Kumaran and Allan, 2008; Bendersky and Croft, 2008; Kumaran and Carvalho, 2009) and other types of query reformulations for bridging the vocabulary gap (Wang and Zhai, 2008). (Guo et al., 2008) proposed a unified model to perform a broad set of query refinements including correction, segmentation and stemming. However, it has very limited ability in query correction. In this paper, we study the discriminative training of query spelling correction, which is potentially beneficial to many existing studies.

Noisy channel model (or source channel model) has been widely used in NLP. Many approaches have been proposed to perform discriminative training of the model (McCallum et al., 2000; Lafferty, 2001). However, these approaches mostly deal with a relatively small search space where the number of candidates at each step is limited (e.g. POS tagging). A typically used search algorithm is dynamic programming. In spelling correction, however, the search space is much bigger and the existing approaches featuring dynamic programming are difficult to be applied.

Structural learning and latent structural learning has been studied a lot in NLP in recent years(Chang et al., 2010; Dyer et al., 2011), and has been

shown to be useful in a range of NLP applications from Textual Entailment, Paraphrasing and Transliteration (Chang et al., 2010) to sentiment analysis (Yessenalina et al., 2010).

Work has also been done on integrating discriminative learning in search. Freitag and Khadivi used a perceptron algorithm to train for sequence alignment problem. A beam search algorithm was utilized in the search (Freitag and Khadivi, 2007). Daume et al. proposed the Searn framework for search based structural prediction (Daume et al., 2009). Our model differs from the Searn framework in that it learns to make global decisions rather than accumulating local decisions. The global decision was made possible by an efficient search algorithm.

Query spelling correction also shares many similarities with statistical machine translation (SMT). Sun et al. (2010) has formulated the problem within an SMT framework. However, SMT usually involves more complex alignments, while in query spelling correction search is the more challenging part. Our main contribution in this paper is a novel unified way to directly optimize the search phase of query spelling correction with the use of LS-SVM.

3 Discriminative Model for Query Spelling Correction Based on LS-SVM

In this section, we first present the discriminative formulation of the problem of query spelling correction. Then we introduce in detail the model we use for solving the problem.

3.1 The Discriminative Form of Query Spelling Correction

In query spelling correction, given a user entered query q, which is potentially misspelled, the goal is to find a correction c, such that it could be a more effective query which improves the quality of search results. A general discriminative formulation of the problem is of the following form:

$$f(q) = \underset{c \in \mathcal{V}*}{\arg\max[w \cdot \Psi(q, c)]},$$
 (3)

where $\Psi(q, c)$ is a vector of features and w is the model parameter. This discriminative formulation is more general compared to the noisy channel model. It has the flexibility of using features and applying

weights. The noisy channel model is a special case of the discriminative form where only two features, the source probability and the transformation probability, are used and uniform weightings are applied. However, this problem formulation does not give us much insight on how to proceed to design the model. Especially, it is unclear how $\Psi(q, c)$ can be computed.

To enhance the formulation, we explore the fact that spelling correction follows a word-by-word procedure. Let us first consider a scenario where word boundary errors does not exist. In this scenario, each query term matches and only matches to a single term in the correction. Formally, let us denote $q = q_1, ..., q_n$ and $c = c_1, ..., c_m$ as structured objects from the space of \mathcal{V}^* , where \mathcal{V} is our vocabulary of words and \mathcal{V}^* is all possible phrases formed by words in \mathcal{V} . Both q and c have an intrinsic sequential structure. When no word boundary error exists, |c| = |q| holds for any candidate correction $c. q_i$ and c_i establish a one-to-one mapping. In this case, we have a more detailed discriminative form:

$$f(q) = \arg\max_{c \in \mathcal{V}^{|q|}} [w \cdot (\Psi_0 + \sum_{i=1}^{|q|} \Psi_1(q_i, c_i))], \quad (4)$$

where Ψ_0 is a vector of normalizing factors, $\Psi_1(q_i, c_i)$ is the decomposed computation of $\Psi(q, c)$ for each query term q_i and c_i , for i = 1 to |q|.

Equation 4 is a clearer formulation. The major challenge of solving this discriminative problem is the complexity. Theoretically, each term has $|\mathcal{V}|$ candidates and it is impossible to enumerate over all possible combinations. To make it even worse, merging and splitting errors are quite common in misspelling. As a result, the assumption of one-to-one mapping does not hold in practice.

To account for these word boundary errors and enhance the discriminative formulation, we introduce a latent variable a to model the unobserved structural information. More specifically, $a = a_1, a_2, ..., a_{|a|}$ is the alignment between q and c. Each alignment node a_t is a represented by a quadruple $(q_{start}, q_{end}, c_{start}, c_{end})$. Figure 1 shows a common merge error and its best alignment. The phrase "credit card", in this case, is incorrectly merged into one word "creditcard" by the user. Figure 2 shows



Figure 1: Example of Merge Error and Alignment



Figure 2: Example of Split Error and Alignment

the best alignment for a common split error, where the word "gamespot" is incorrectly split into a two word phrase "game spot".

Taking into consideration the latent variable, we arrive at our final discriminative form of query spelling correction:

$$f(q) = \arg \max_{(c,a) \in \mathcal{V}^n \times \mathcal{A}} [w \cdot \Psi(q, c, a)]$$

=
$$\arg \max_{(c,a) \in \mathcal{V}^* \times \mathcal{A}} [w \cdot (\Psi_0 + \sum_{t=0}^{|a|} \Psi_1(q_{a_t}, c_{a_t}, a_t))],$$

(5)

The challenges of successfully applying a discriminative model to this problem formulation are 1) how can we design a learning algorithm to learn the model parameter w to directly optimize the maximization problem; 2) how can we solve the maximization efficiently without having to enumerate all candidates; 3) how can we design features to guarantee the correctness of the search algorithm. In the following subsections we introduce our solutions to the three challenges in detail.

3.2 Latent Structural SVM

We employ the latent structural SVM (LS-SVM) model for learning the discriminative model of query spelling correction. LS-SVM is a large margin method that deals with structured prediction problems with latent structural information (Yu and Joachims, 2009). LS-SVM has the merit of allowing

task specific, customizable solutions for the inference problem. This makes it easy to adapt to learning the model parameters for different problems. The following is a brief introduction of LS-SVM that largely mirrors the work by (Yu and Joachims, 2009).

Without loss of generality, let us aim at learning a prediction function $f : \mathcal{X} \to \mathcal{Y}$ that maps input $x \in \mathcal{X}$ to an output $y \in \mathcal{Y}$ with latent structural information $h \in \mathcal{H}$. The decision function is of the following form:

$$f(x) = \underset{(y,h)\in\mathcal{Y}\times\mathcal{H}}{\arg\max} [w \cdot \Psi(x,y,h)], \qquad (6)$$

where $\Psi(x, y, h)$ is the set of feature functions defined jointly over the input x, the output y and the latent variable h. w is the parameter of the model. Given a set of training examples that consist of input and output pairs $\{(x_1, y_1), ...(x_n, y_n)\} \in (\mathcal{X} \times \mathcal{Y})^n$, the LS-SVM method solves the following optimization problem:

$$\min_{w} \frac{1}{2} \|w\|^{2} + C \sum_{\substack{i=1\\n}}^{n} \max_{(\hat{y}, \hat{h}) \in \mathcal{Y} \times \mathcal{H}} [w \cdot \Psi(x_{i}, \hat{y}, \hat{h}) + \Delta(y_{i}, \hat{y})]$$
$$-C \sum_{\substack{i=1\\n}}^{n} \max_{h \in \mathcal{H}} [w \cdot \Psi(x_{i}, y_{i}, h)],$$
(7)

where $\Delta(y_i, \hat{y})$ is the loss function for the *i*th example. The details of the derivation is omitted in this paper. Readers who are interested can read more from (Yu and Joachims, 2009).

There are two maximization problems that are essential in Equation 7. The first one is the loss augmented decision function:

$$\max_{(\hat{y},\hat{h})\in\mathcal{Y}\times\mathcal{H}} [w\cdot\Psi(x_i,\hat{y},\hat{h}) + \Delta(y_i,\hat{y})], \quad (8)$$

and the second is the inference of latent variable given the label of the training data:

$$\max_{h \in \mathcal{H}} [w \cdot \Psi(x_i, y_i, h)].$$
(9)

The Latent Structural SVM framework does not specify how the maximization problems in Equation

8 and Equation 9 are solved, as well as the inference problem in 6. These maximization problems are task dependent. Being able to efficiently solve them is the key to successfully applying the Latent Structural SVM method. We will show in detail how we solve these maximization problems to make LS-SVM work for query spelling correction in the following subsection.

For training the LS-SVM model, a Concave-Convex Procedure (CCCP) was proposed to solve this optimization problem (Yu and Joachims, 2009). The method resembles the Expect-Maximization (EM) training method as it updates the model by iteratively recomputing the latent variable. However, rather than performing "sum-product" training as in EM where a distribution over the hidden variable is maintained, the CCCP method used for LS-SVM is more similar to the "max-product" paradigm where we "guess" the best hidden variable in each iteration, except here we "guess" by minimizing a regularized loss function instead of maximizing the likelihood.

3.3 Solving the Inference Problems

The essential inference problem is to find the correction that maximizes the scoring function according to the model (i.e., the decision function in Equation 6). For this purpose we design a best first search algorithm similar to the standard search algorithm in the noisy channel model. The essence of the search algorithm is to bound the score of each candidate so that we could evaluate the most promising candidates first. The algorithm is given in Algorithm 1.

Essentially, the algorithm maintains a priority queue of all search paths. Each time the best path is de-queued, it is expanded with up to m - 1 words in q by searching over a vocabulary trie of up to m-gram. Each path is represented as a quadruple (pos, str, sc, a), representing the current term position in query, the string of the path, the path's score and the alignment so far. The priority queue is sorted according to the score of each path in descending order. The GetSuggestions() function retrieves the top n similar words to the given word with a vocabulary trie according to an error model.

Splitting errors are dealt with in Algorithm 1 by "looking forward" m words in the query when generating candidate words. Merging errors are accounted for by including up to m-gram in the vocab-

ulary trie. It is worth mentioning that performance of Algorithm 1 could be further improved by computing heuristic scores for each path.

Algorithm 1: Best First Search Algorithm				
Input : Vocabulary Trie V , query q , output size k ,				
max order m , candidate pool size n				
Output : List l of top k corrections for q				
1 Initialize List l ;				
2 Initialize PriorityQueue <i>pq</i> ;				
3 Enqueue to pq a start path with position set to 0,				
string set to empty string, score set to $w \cdot \Psi_0$, and				
path alignment set to empty set;				
4 while pq is not Empty do				
5 Path $\pi \leftarrow pq$.Dequeue();				
6 if π .pos < q.terms.length then				
7 for $i \leftarrow 0$ to m do				
8 $ph \leftarrow q.terms[\pi.pos + 1\pi.pos + i];$				
9 $sug \leftarrow \text{GetSuggestions}(ph, V, n);$				
10 foreach s in sug do				
11 $pos' \leftarrow \pi.pos + i;$				
12 $str' \leftarrow concat(\pi.str, s.str);$				
13 $a' \leftarrow \pi.a \cup s.a;$				
14 $sc' \leftarrow \pi.sc + w \cdot \Psi_1(q_{s.a}, c_{s.a}, s.a);$				
15 Enqueue pq with the new path				
(pos', str', sc', a');				
16 else				
17 Add suggestion string π .str to l;				
18 if $l.Count > k$ then return l ;				
19 return l;				

As Algorithm 1 originates from the noisy channel model, the two known features that work with the algorithm are $\log p(c)$ and $\log p(q|c)$ from the noisy channel model. However, it is unknown whether other features can work with the search algorithm and how we can develop new features to ensure it. After analyzing the properties of the features and the search algorithm, we find that a feature ψ has to satisfy the following monotonicity constraint in order to be used in Algorithm 1.

Monotonicity Property. Given query q, for any alignment $A_t = A_{t-1} \cup \{a_t\}$ at time t, $\psi(q_{A_t}, c_{A_t}, A_t) \leq \psi(q_{A_{t-1}}, c_{A_{t-1}}, A_{t-1})$, where q_{A_t} is the concatenation of q_{a_0} to q_{a_t} and c_{A_t} is the concatenation of c_{a_0} to c_{a_t} .

That is, the value of the feature (which is computed in an accumulative manner) cannot increase as the candidate is extended with a new term at any search step. This ensures that the score of the best candidate at any search step is guaranteed to be higher than the score of any future candidates. It also implies $\psi_t(q_{a_t}, c_{a_t}, a_t) \leq 0$ for any $t \in T$. The monotonicity feature ensures the correctness of Algorithm 1. We show how we design features with the guidance of the monotonicity constraint in Section 4.

The solution to to the loss augmented inference depends on the loss function we use. In spelling correction, usually only one correction is valid for an input query. Therefore, we apply the 0-1 loss to our model:

$$\Delta(c,\hat{c}) = \begin{cases} 0 & c = \hat{c} \\ 1 & c \neq \hat{c} \end{cases}$$
(10)

Given this loss function, the loss augmented inference problem can be solved easily with an algorithm similar to Algorithm 1. This is done by initializing the loss to be 1 at the beginning of each search path. During the search procedure, we check if the loss decreases to 0 given the correction string so far. If this is the case, we decreases the score by 1 and add the path back to the priority queue. More advanced functions may also be used (Dreyer et al., 2006), which may lead to better training performance. We plan to further study different loss functions in our future work.

The inference of the latent alignment variable can be solved with dynamic programming, as the number of possible alignments is limited given the query and the correction.

4 Features

In the following discussions, we will describe how the features in our discriminative model are developed under the guidance of the monotonicity constraint.

4.1 Source Probability and Transformation Probability

We know from empirical experience that the source probability and the transformation probability are the two most important features in query spelling correction. We include them in our model in a normalized form. Taking the source probability for example, we define the following feature:

$$\psi(q, c, a) = \frac{\mu + \sum_{1}^{|a|} \log p(c)}{\mu} = 1 + \sum_{1}^{|a|} \frac{\log p(c)}{\mu},$$
(11)

where μ is a normalizing factor computed as:

$$\mu = -|q|\log p_{min},\tag{12}$$

where p_{min} is the smallest probability we use in practice.

The formula fits the general form we define in 5 in that $\psi_0 = 1$ and $\psi_1(q_{a_t}, c_{a_t}, a_t) = \frac{\log p(c)}{\mu}$ for any t = 1 to |a|.

Similarly, we have the follow feature for the transformation probability:

$$\psi'(q,c,a) = \frac{\mu + \sum_{1}^{|a|} \log p(q|c)}{\mu} \\ = 1 + \sum_{1}^{|a|} \frac{\log p(q|c)}{\mu}.$$
(13)

We use the web Microsoft n-gram model¹ to compute source model p(c). We train the unigram transformation model for the transformation probability p(q|c) according to (Duan and Hsu, 2011).

In generative models, we treat transformation probabilities from merging and splitting errors in the same way as single word errors. In our discriminative model we can assign separate weight to the transformation probabilities resulted from different types of errors. This allows fine tuning of the query spelling correction system, making it more adaptive to environments where the ratio of different types of errors may vary. Moreover, the model also allows us to include language models trained over different resources, such as query log, title of webpages or anchor texts.

4.2 Local Heuristic Features

Despite the goal of query spelling correction is to deal with misspellings, in real world most queries are correctly spelled. A good query spelling correction system shall prevent as much as possible from misjudging an correctly spelled query as misspelled. With this idea in mind, we invent some heuristic functions to avoid misjudging.

¹http://research.microsoft.com/enus/collaboration/focus/cs/web-ngram.aspx **Local Heuristic 1.** When a query term is matched against trustable vocabulary, it increases the chance that the term is already in its correct form. For example, we extract a reliable vocabulary from the title field of Wikipedia². We therefore design the following feature:

$$\phi(q,c,a) = 1 + \sum_{t=1}^{|a|} \phi_1(q_{a_t}, c_{a_t}, a_t), \qquad (14)$$

where $\phi_1(q_{a_t}, c_{a_t}, a_t)$ is defined as:

$$\phi_{1}(q_{a_{t}}, c_{a_{t}}, a_{t}) = \begin{cases} 0 & q_{a_{t}} \notin \mathcal{W} \\ 0 & q_{a_{t}} \in \mathcal{W}, q_{a_{t}} = c_{t} \\ -\frac{1}{|q|} & q_{a_{t}} \in \mathcal{W}, q_{a_{t}} \neq c_{a_{t}} \end{cases}$$
(15)

where W is the vocabulary of Wikipedia titles. Since |q| > |a| always holds, the feature is normalized between 0 and 1.

Local Heuristic 2. Another heuristic is that words with numbers in it, despite usually not included in any vocabulary, should be treated carefully as they tend to be correct words. Such words could be a model, a serial number or a special entity name. Since the number keys on keyboard are away from the letter keys, they are more likely to be intentionally typed in if found in user queries. Similar to Heuristic 1, we design the following feature to capture this heuristic:

$$\phi'(q,c,a) = 1 + \sum_{t=1}^{|a|} \phi'_1(q_{a_t},c_{a_t},a_t), \qquad (16)$$

where $\phi'_1(q_{a_t}, c_{a_t}, a_t)$ is defined as:

$$\phi_1'(q_{a_t}, c_{a_t}, a_{a_t}) = \begin{cases} 0 & [0...9] \notin q_{a_t} \\ 0 & [0...9] \in q_{a_t}, q_{a_t} = c_{a_t} \\ -\frac{1}{|q|} & [0...9] \in q_{a_t}, q_{a_t} \neq c_{a_t} \end{cases}$$
(17)

4.3 Global Heuristic Features

Some global heuristics are also important in query spelling correction. For instance, the total number

²http://www.wikipedia.org

of words being corrected in the query may be an indicator of whether the system has leaned towards overcorrecting. To account for this global heuristic, we design the following feature:

$$\varphi(q,c,a) = \begin{cases} 1 & wc(q,c,a) < wc_{max} \\ 0 & otherwise \end{cases}$$
(18)

where wc(q, c, a) is the number of word changes at step t, wc_{max} is the maximum number of word changes we allow in our system (in a soft way). Similarly, other thresholded features can be designed such as the number of total edit operations. The use of global features is similar to the use of loss function in the search algorithm.

5 Experiments

In order to test the effectiveness and efficiency of our proposed discriminative training method, in this section we conduct extensive experiments on two web query spelling datasets. Below we first present the dataset and evaluation metrics, followed by the experiment results on query spelling correction.

5.1 Dataset Preparation

The experiments are conducted on two query spelling correction datasets. One is the TREC dataset based on the publicly available TREC queries (2008 Million Query Track). This dataset contains 5892 queries and the corresponding corrections annotated by the MSR Speller Challenge ³ organizers. There could be more than one plausible corrections for a query. In this dataset only 5.3% of queries are judged as misspelled.

We have also annotated another dataset that contains 4926 MSN queries, where for each query there is at most one correction. Three experts are involved in the annotation process. For each query, we consult the speller from two major search engines (i.e. Google and Bing). If they agree on the returned results (including the case if the query is just unchanged), we take it as the corrected form of the input query. If the results are not the same from the two, as least one human expert will manually annotate the most likely corrected form of the query. Finally, about 13% of queries are judged as misspelled in this dataset, which is close to the error rate of real web queries. We've made this dataset publicly available to all researchers⁴.

Both the two datasets are split randomly into two equal subsets for training and testing.

5.2 Evaluation Metrics

We evaluate our system based on the evaluation metrics proposed in Microsoft Speller Challenge, including expected precision, expected recall and expected F1 measure.

Let q be a user query and $C(q) = (c^1, c^2, c^k)$ be the set of system output with posterior probabilities $P(c^i|q)$. Let S(q) denote the set of plausible spelling variations annotated by the human experts for q. Expected Precision is computed as:

$$Precision = \frac{1}{|Q|} \sum_{q \in Q} \sum_{c \in C(q)} I_p(c,q) P(c|q), \quad (19)$$

where $I_p(c,q) = 1$ if $c \in S(q)$, and 0 otherwise. And expected recall is defined as:

$$Recall = \frac{1}{|Q|} \sum_{q \in Q} \sum_{a \in S(q)} I_r(C(q), a) / |S(q)|,$$
(20)

where $I_r(C(q), a) = 1$ if $a \in C(q)$ for $a \in S(q)$, and 0 otherwise. We use R@N to denote recall for systems limited to output top N corrections.

Expected F1 measure can be computed as:

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$
(21)

5.3 Experiment Results

Table 1 compares the performance of our LS-SVM based model with two strong baseline systems. The first baseline system is an Echo system which simply echos the input. The echo system is usually considered as a strong baseline in query spelling correction as the majority of the queries are correctly spelled queries. The second baseline Lueck-2011 we use is a award winning speller system⁵ (Luec, 2011), which was ranked at the first place in Microsoft Spelling Challenge 2011.

³http://web-ngram.research.microsoft.com/spellerchallenge/

⁴http://times.cs.uiuc.edu/duan9/msn_speller.tar.gz

⁵http://www.phraselink.com

Table 1. LSS VM VS Baselines Serving as Standarone Spener							
		All Queries			Misspelled Queries		
Dataset	Method	Precision	R@10	F1	Precision	R@10	F1
TREC	Echo	0.949	0.876	0.911	0	0	0
	Lueck-2011	0.963	0.932	0.947	0.391	0.479	0.430
	LS-SVM	0.955	0.944	0.949	0.331	0.678^{\dagger}	0.445 [†]
MSN	Echo	0.869	0.869	0.869	0	0	0
	Lueck-2011	0.896	0.921	0.908	0.334	0.397	0.363
	LS-SVM	0.903	0.953	0.928	0.353 [†]	0.662^{\dagger}	0.461 [†]

Table 1: LSSVM vs Baselines Serving as Standalone Speller

We show performances for the entire query sets as well as the query sets consisting only the misspelled queries. As we can see, our system outperforms both baseline systems on almost all metrics, except the precision of Lueck-2011 is better than ours on TREC dataset. We perform statistical test and measures where our system shows statistical significant improvement over both baseline systems are noted by \dagger . It is theoretically impossible to achieve statistical significance in the entire query set as majority queries have almost identical performance in different systems due to the large amount of correct queries. But our method shows significant improvement in the dealing with the misspelled queries. This experiment verified the effectiveness of our proposed discriminative model. As a standalone speller, our system achieves very high accuracy.

Despite we are primarily focused on optimizing the top correction in our discriminative model, we can also use the trained system to output multiple candidate corrections. Table 2 compare our system with the noisy channel model (N-C) in terms of recall at different levels of cutoff. For all levels, we see that our system achieves higher recall than the noisy channel model. This indicates that when used together with a secondary ranker, our system serves as a better filtering method than the unoptimized noisy channel model. Since the ranker makes use of arbitrary features, it has the potential of further improving the accuracy of query spelling correction. We plan to further explore this idea as a future work.

In Table 3 we study the effect of treating the transformation probability of merging and splitting errors as separate features and including the local and global heuristic features (rich features). We see that

Table 2: LS-SVM vs Noisy Channel Model Serving as Filtering Method

Dataset	Method	R@5	R@10	R@20
TREC	N-C	0.896	0.899	0.901
	LS-SVM	0.923	0.944	0.955
MSN	N-C	0.870	0.873	0.876
	LS-SVM	0.950	0.953	0.960

the precision of query spelling correction can benefits from the use of rich features. However, it does not result in much improvement in recall. This is reasonable as the additional features are primarily designed to improve the accuracy of the top correction generated by the system. In doing so, it actually regularizes the ability of the system in retrieving diversified results. For instance, the global heuristic feature on the number of word change tries to prevent the system from returning candidates having more than a certain number of changed words. For the TREC collection where more than one corrections can be labeled for a query, this phenomena is aggravated.

Table 3: LSSVM w/ and w/o Rich Features

Tuble 5: EBS (111 m und mo fuen f eutares						
Dataset	Method	Precision	R@10	F1		
TREC	w/o	0.942	0.946	0.944		
	w/	0.955	0.944	0.949		
MSN	w/o	0.898	0.952	0.924		
	w/	0.903	0.953	0.928		

6 Conclusions

In this paper, we present a novel discriminative model for query spelling correction. The paper made the following contributions:

First, to the best of our knowledge, this is a novel exploration of directly optimizing the search phase in query spelling correction with a discriminative model. By modeling word alignment as the latent structural information, our formulation also deals with word boundary errors. We propose to use LS-SVM for learning the discriminative model which naturally incorporates search in the learning process. Second, we develop an efficient search algorithm that solves the inference problems in the LS-SVM based model. We analyze the criteria for selecting and designing features to ensure the correctness and efficiency of the search algorithm. Third, we explore effective features to improve the accuracy of the model. Finally, experiments are conducted to verify the effectiveness of the proposed model. It is shown that as a standalone speller our system achieves high accuracy. When used in a two stage approach, it attains higher recall than the noisy channel model and can thus serve as a superior method for candidate generation. We also verify that through the use of rich features, we can further improve the accuracy of our query spelling correction system.

7 Acknowledgments

This paper is based upon work supported in part by MIAS, the Multimodal Information Access and Synthesis center at UIUC, part of CCICADA, a DHS Center of Excellence, and by the National Science Foundation under grant CNS-1027965, and by a Microsoft grant.

References

- F. Ahmad and G. Kondrak. 2005. Learning a spelling error model from search query logs. In *HLT/EMNLP*. The Association for Computational Linguistics.
- M. Bendersky and W. B. Croft. 2008. Discovering key concepts in verbose queries. In *Proceedings of the 31st* annual international ACM SIGIR conference on Research and development in information retrieval, SI-GIR '08. ACM, New York, NY, USA, 491-498.
- E. Brill and R. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong.
- M. Chang, D. Goldwasser, D. Roth and V. Srikumar. 2010. Discriminative Learning over Constrained Latent Representations. In *Proceedings of NAACL*.

- Q. Chen, M. Li, and M. Zhou. 2007. Improving query spelling correction using web search results. In *EMNLP-CoNLL*, pages 181–189.
- S. Cucerzan and E. Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- H. Daume, J. Langford and D. Marcu. 2009. Searchbased Structured Prediction. Machine Learning Journal (MLJ).
- M. Dreyer, D. Smith and N. Smith. 2006. Vine parsing and minimum risk reranking for speed and precision. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*. 201-205.
- H. Duan and B.-J. P. Hsu. 2011. Online spelling correction for query completion. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 117–126, New York, NY, USA.
- C. Dyer, J. H. Clark, A. Lavie, and N. A. Smith. 2011. Unsupervised Word Alignment with Arbitrary Features. In *Proceedings of ACL*.
- D. Freitag, S. Khadivi. 2007. A Sequence Alignment Model Based on the Averaged Perceptron. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. 238-247.
- J. Gao, X. Li, D. Micol, C. Quirk, and X. Sun. 2010. A large scale ranker-based system for search query spelling correction. In *COLING*, pages 358–366.
- A. R. Golding and D. Roth 1999. A Winnow based approach to Context-Sensitive Spelling Correction. In *Machine Learning*, vol 34, pages 107–130.
- J. Guo, G. Xu, H. Li, and X. Cheng. 2008. A unified and discriminative model for query refinement. In *Proceedings of the 31st annual international ACM SIGIR*, SIGIR '08, pages 379–386, New York, NY, USA.
- C. John Yu and T. Joachims. 2009. Learning structural SVMs with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning* (ICML '09). ACM, New York, NY, USA, 1169-1176.
- M. D. Kernighan, K. W. Church, W. A. Gale. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th conference on Computational linguistics*. 205-210. August 20-25, 1990, Helsinki, Finland.
- K. Kukich. 1992. Techniques for automatically correcting words in text. *ACM computing surveys*, 24(4).
- G. Kumaran and J. Allan. 2008. Effective and efficient user interaction for long queries. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08. ACM, New York, NY, USA.

- G. Kumaran and V. R. Carvalho. 2009. Reducing long queries using query quality predictors. In Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '09. ACM, New York, NY, USA, 564-571.
- J. Lafferty. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01). 282–289.
- V. I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, 10(8), 707-710.
- G. Luec. 2011. A data-driven approach for correcting search quaries. In *Spelling Alteration for Web Search Workshop*.
- A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum Entropy Markov Models for Information Extraction and Segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning* (ICML '00). 591-598.
- M. Mitra, A. Singhal, and C. Buckley. 1998. Improving automatic query expansion. In *Proceedings of the 21st* annual international ACM SIGIR conference on Research and development in information retrieval, SI-GIR '98.
- Y. Qiu and H. Frei. 1993. Concept based query expansion. In Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '93. ACM, New York, NY, USA, 160-169.
- X. Sun, J. Gao, D. Micol, and C. Quirk. 2010. Learning phrase-based spelling error models from clickthrough data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 266–274, Stroudsburg, PA, USA.
- X. Wang, C. Zhai. 2008. Mining Term Association Patterns from Search Logs for Effective Query Reformulation. In Proceedings of the 17th ACM International Conference on Information and Knowledge Management 2008, CIKM'08. 479-488.
- J. Xu and W. B. Croft. 1996. Query expansion using local and global document analysis. In *Proceedings of* the 19th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '96. ACM, New York, NY.
- A. Yessenalina, Y. Yue, C. Cardie. 2010. Multilevel Structured Models for Document-level Sentiment Classification. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP '10). 10461056.