Learning Phrase-Based Spelling Error Models from Clickthrough Data

Xu Sun[*]
Dept. of Mathematical Informatics
University of Tokyo, Tokyo, Japan
xusun@mist.i.u-tokyo.ac.jp

Jianfeng Gao
Microsoft Research
Redmond, WA, USA
jfgao@microsoft.com

Daniel Micol
Microsoft Corporation
Munich, Germany
danielmi@microsoft.com

Chris Quirk
Microsoft Research
Redmond, WA, USA
chrisq@microsoft.com

Abstract

This paper explores the use of clickthrough data for query spelling correction. First, large amounts of query-correction pairs are derived by analyzing users' query reformulation behavior encoded in the clickthrough data. Then, a phrase-based error model that accounts for the transformation probability between multi-term phrases is trained and integrated into a query speller system. Experiments are carried out on a human-labeled data set. Results show that the system using the phrase-based error model outperforms significantly its baseline systems.

1 Introduction

Search queries present a particular challenge for traditional spelling correction methods for three main reasons (Ahmad and Kondrak, 2004). First, spelling errors are more common in search queries than in regular written text: roughly 10-15% of queries contain misspelled terms (Cucerzan and Brill, 2004). Second, most search queries consist of a few key words rather than grammatical sentences, making a grammar-based approach inappropriate. Most importantly, many queries contain search terms, such as proper nouns and names, which are not well established in the language. For example, Chen et al. (2007) reported that 16.5% of valid search terms do not occur in their 200K-entry spelling lexicon.

Therefore, recent research has focused on the use of Web corpora and query logs, rather than human-compiled lexicons, to infer knowledge about misspellings and word usage in search queries (e.g., Whitelaw et al., 2009). Another important data source that would be useful for this purpose is clickthrough data. Although it is well-known that clickthrough data contain rich information about users' search behavior, e.g., how a user (re-) formulates a query in order to find the relevant document, there has been little research on exploiting the data for the development of a query speller system.

In this paper we present a novel method of extracting large amounts of query-correction pairs from the clickthrough data. These pairs, implicitly judged by millions of users, are used to train a set of spelling error models. Among these models, the most effective one is a phrase-based error model that captures the probability of transforming one multi-term phrase into another multi-term phrase. Comparing to traditional error models that account for transformation probabilities between single characters (Kernighan et al., 1990) or sub-word strings (Brill and Moore, 2000), the phrase-based model is more powerful in that it captures some contextual information by retaining inter-term dependencies. We show that this information is crucial to detect the correction of a query term, because unlike in regular written text, any query word can be a valid search term and in many cases the only way for a speller system to make the judgment is to explore its usage according to the contextual information.

We conduct a set of experiments on a large data set, consisting of human-labeled

[*] The work was done when Xu Sun was visiting Microsoft Research Redmond.

query-correction pairs. Results show that the error models learned from clickthrough data lead to significant improvements on the task of query spelling correction. In particular, the speller system incorporating a phrase-based error model significantly outperforms its baseline systems.

To the best of our knowledge, this is the first extensive study of learning phase-based error models from clickthrough data for query spelling correction. The rest of the paper is structured as follows. Section 2 reviews related work. Section 3 presents the way query-correction pairs are extracted from the clickthrough data. Section 4 presents the baseline speller system used in this study. Section 5 describes in detail the phrase-based error model. Section 6 presents the experiments. Section 7 concludes the paper.

2 Related Work

Spelling correction for regular written text is a long standing research topic. Previous researches can be roughly grouped into two categories: correcting non-word errors and real-word errors.

In non-word error spelling correction, any word that is not found in a pre-compiled lexicon is considered to be misspelled. Then, a list of lexical words that are *similar* to the misspelled word are proposed as candidate spelling corrections. Most traditional systems use a manually tuned similarity function (e.g., edit distance function) to rank the candidates, as reviewed by Kukich (1992). During the last two decades, statistical error models learned on training data (i.e., query-correction pairs) have become increasingly popular, and have proven more effective (Kernighan et al., 1990; Brill and Moore, 2000; Toutanova and Moore, 2002; Okazaki et al., 2008).

Real-word spelling correction is also referred to as context sensitive spelling correction (CSSC). It tries to detect incorrect usages of a valid word based on its context, such as "peace" and "piece" in the context "a _ of cake". A common strategy in CSSC is as follows. First, a pre-defined confusion set is used to generate candidate corrections, then a scoring model, such as a trigram language model or naïve Bayes classifier, is used to rank the candidates according to their context (e.g., Golding and Roth, 1996; Mangu and Brill, 1997; Church et al., 2007).

When designed to handle regular written text, both CSSC and non-word error speller systems rely on a pre-defined vocabulary (i.e., either a lexicon or a confusion set). However, in query spelling correction, it is impossible to compile such a vocabulary, and the boundary between the non-word and real-word errors is quite vague. Therefore, recent research on query spelling correction has focused on exploiting noisy Web data and query logs to infer knowledge about misspellings and word usage in search queries. Cucerzan and Brill (2004) discuss in detail the challenges of query spelling correction, and suggest the use of query logs. Ahmad and Kondrak (2005) propose a method of estimating an error model from query logs using the EM algorithm. Li et al. (2006) extend the error model by capturing word-level similarities learned from query logs. Chen et al. (2007) suggest using web search results to improve spelling correction. Whitelaw et al. (2009) present a query speller system in which both the error model and the language model are trained using Web data.

Compared to Web corpora and query logs, clickthrough data contain much richer information about users' search behavior. Although there has been a lot of research on using clickthrough data to improve Web document retrieval (e.g., Joachims, 2002; Agichtein et al., 2006; Gao et al., 2009), the data have not been fully explored for query spelling correction. This study tries to learn error models from clickthrough data. To our knowledge, this is the first such attempt using clickthrough data.

Most of the speller systems reviewed above are based on the framework of the source channel model. Typically, a language model (source model) is used to capture contextual information, while an error model (channel model) is considered to be *context free* in that it does not take into account any contextual information in modeling word transformation probabilities. In this study we argue that it is beneficial to capture contextual information in the error model. To this end, inspired by the phrase-based statistical machine translation (SMT) systems (Koehn et al., 2003; Och and Ney, 2004), we propose a phrase-based error model where we assume that query spelling correction is performed at the phrase level.

In what follows, before presenting the phrase-based error model, we will first describe the clickthrough data and the query speller system we used in this study.

3 Clickthrough Data and Spelling Correction

This section describes the way the query-correction pairs are extracted from clickthrough data. Two types of clickthrough data are explored in our experiment.

The clickthrough data of the first type has been widely used in previous research and proved to be useful for Web search (Joachims, 2002; Agichtein et al., 2006; Gao et al., 2009) and query reformulation (Wang and Zhai, 2008; Suzuki et al., 2009). We start with this same data with the hope of achieving similar improvements in our task. The data consist of a set of query sessions that were extracted from one year of log files from a commercial Web search engine. A query session contains a query issued by a user and a ranked list of links (i.e., URLs) returned to that same user along with records of which URLs were clicked. Following Suzuki et al. (2009), we extract query-correction pairs as follows. First, we extract pairs of queries Q_1 and Q_2 such that (1) they are issued by the same user; (2) Q_2 was issued within 3 minutes of Q_1; and (3) Q_2 contained at least one clicked URL in the result page while Q_1 did not result in any clicks. We then scored each query pair (Q_1, Q_2) using the edit distance between Q_1 and Q_2, and retained those with an edit distance score lower than a pre-set threshold as query correction pairs.

Unfortunately, we found in our experiments that the pairs extracted using the method are too noisy for reliable error model training, even with a very tight threshold, and we did not see any significant improvement. Therefore, in Section 6 we will not report results using this dataset.

The clickthrough data of the second type consists of a set of *query reformulation sessions* extracted from 3 months of log files from a commercial Web browser. A query reformulation session contains a list of URLs that record user behaviors that relate to the query reformulation functions, provided by a Web search engine. For example, almost all commercial search engines offer the "did you mean" function, suggesting a possible alternate interpretation or spelling of a user-issued query. Figure 1 shows a sample of the query reformulation sessions that record the "did you mean" sessions from three of the most popular search engines. These sessions encode the same user behavior: A user first queries for "harrypotter sheme park", and then clicks on the resulting spelling suggestion "harry potter theme park". In our experiments, we "reverse-engineer" the parameters from the URLs of these sessions, and deduce how each search engine encodes both a query and the fact that a user arrived at a URL by clicking on the spelling suggestion of the query – an important indication that the spelling suggestion is desired. From these three months of query reformulation sessions, we extracted about 3 million query-correction pairs. Compared to the pairs extracted from the clickthrough data of the first type (query sessions), this data set is much cleaner because all these spelling corrections are actually clicked, and thus judged implicitly, by many users.

Google:

```
http://www.google.com/search?
hl=en&source=hp&
q=harrypotter+sheme+park&aq=f&oq=&aqi=
```

```
http://www.google.com/search?
hl=en&ei=rnNAS8-oKsWe_AaB2eHlCA&
sa=X&oi=spell&resnum=0&ct=
result&cd=1&ved=0CA4QBSgA&
q=harry+potter+theme+park&spell=1
```

Yahoo:

```
http://search.yahoo.com/search;
_ylt=A0geu6ywckBL_XIBSDtXNyoA?
p=harrypotter+sheme+park&
fr2=sb-top&fr=yfp-t-701&sao=1
```

```
http://search.yahoo.com/search?
ei=UTF-8&fr=yfp-t-701&
p=harry+potter+theme+park
&SpellState=n-2672070758_q-tsI55N6srhZa.
qORA0MuawAAAA%40%40&fr2=sp-top
```

Bing:

```
http://www.bing.com/search?
q=harrypotter+sheme+park&form=QBRE&qs=n
```

```
http://www.bing.com/search?
q=harry+potter+theme+park&FORM=SSRE
```

Figure 1. A sample of query reformulation sessions from three popular search engines. These sessions show that a user first issues the query "harrypotter sheme park", and then clicks on the resulting spell suggestion "harry potter theme park".

In addition to the "did you mean" function, recently some search engines have introduced two new spelling suggestion functions. One is the "auto-correction" function, where the search engine is confident enough to automatically apply the spelling correction to the query and execute it to produce search results for the user. The other is the "split pane" result page, where one half portion of the search results are produced using the original query, while the other half, usually visually separate portion of results are produced using the auto-corrected query.

In neither of these functions does the user ever receive an opportunity to approve or disapprove of the correction. Since our extraction approach focuses on user-approved spelling suggestions, we ignore the query reformulation sessions recording either of the two functions. Although by doing so we could miss some basic, obvious spelling corrections, our experiments show that the negative impact on error model training is negligible. One possible reason is that our baseline system, which does not use any error model learned from the clickthrough data, is already able to correct these basic, obvious spelling mistakes. Thus, including these data for training is unlikely to bring any further improvement.

We found that the error models trained using the data directly extracted from the query reformulation sessions suffer from the problem of underestimating the self-transformation probability of a query $P(Q_2=Q_1|Q_1)$, because we only included in the training data the pairs where the query is different from the correction. To deal with this problem, we augmented the training data by including correctly spelled queries, i.e., the pairs (Q_1, Q_2) where $Q_1 = Q_2$. First, we extracted a set of queries from the sessions where no spell suggestion is presented or clicked on. Second, we removed from the set those queries that were recognized as being auto-corrected by a search engine. We do so by running a sanity check of the queries against our baseline spelling correction system, which will be described in Section 6. If the system thinks an input query is misspelled, we assumed it was an obvious misspelling, and removed it. The remaining queries were assumed to be correctly spelled and were added to the training data.

4 The Baseline Speller System

The spelling correction problem is typically formulated under the framework of the source channel model. Given an input query $Q = q_1 ... q_I$, we want to find the best spelling correction $C = c_1 ... c_J$ among all candidate spelling corrections:

$$C^* = \underset{C}{\operatorname{argmax}} P(C|Q) \tag{1}$$

Applying Bayes' Rule and dropping the constant denominator, we have

$$C^* = \underset{C}{\operatorname{argmax}} P(Q|C)P(C) \tag{2}$$

where the error model $P(Q|C)$ models the transformation probability from C to Q, and the language model $P(C)$ models how likely C is a correctly spelled query.

The speller system used in our experiments is based on a ranking model (or ranker), which can be viewed as a generalization of the source channel model. The system consists of two components: (1) a candidate generator, and (2) a ranker.

In candidate generation, an input query is first tokenized into a sequence of terms. Then we scan the query from left to right, and each query term q is looked up in lexicon to generate a list of spelling suggestions c whose edit distance from q is lower than a preset threshold. The lexicon we used contains around 430,000 entries; these are high frequency query terms collected from one year of search query logs. The lexicon is stored using a trie-based data structure that allows efficient search for all terms within a maximum edit distance.

The set of all the generated spelling suggestions is stored using a lattice data structure, which is a compact representation of exponentially many possible candidate spelling corrections. We then use a decoder to identify the top twenty candidates from the lattice according to the source channel model of Equation (2). The language model (the second factor) is a backoff bigram model trained on the tokenized form of one year of query logs, using maximum likelihood estimation with absolute discounting smoothing. The error model (the first factor) is approximated by the edit distance function as

$$-\log P(Q|C) \propto \text{EditDist}(Q, C) \tag{3}$$

The decoder uses a standard two-pass algorithm to generate 20-best candidates. The first pass uses the Viterbi algorithm to find the best C according to the model of Equations (2) and (3). In the second pass, the A-Star algorithm is used to find the 20-best corrections, using the Viterbi scores computed at each state in the first pass as heuristics. Notice that we always include the input query Q in the 20-best candidate list.

The core of the second component of the speller system is a ranker, which re-ranks the 20-best candidate spelling corrections. If the top C after re-ranking is different than the original query Q, the system returns C as the correction.

Let $\mathbf{f}$ be a feature vector extracted from a query and candidate spelling correction pair (Q, C). The ranker maps $\mathbf{f}$ to a real value y that indicates how likely C is a desired correction of Q. For example, a linear ranker simply maps $\mathbf{f}$ to y with a learned weight vector $\mathbf{w}$ such as $y = \mathbf{w} \cdot \mathbf{f}$, where $\mathbf{w}$ is optimized w.r.t. accuracy on a set of hu-

C:	"disney theme park"	*correct query*
S:	["disney", "theme park"]	*segmentation*
T:	["disnee", "theme part"]	*translation*
M:	(1 → 2, 2→ 1)	*permutation*
Q:	"theme part disnee"	*misspelled query*

Figure 2: Example demonstrating the generative procedure behind the phrase-based error model.

man-labeled (Q, C) pairs. The features in **f** are arbitrary functions that map (Q, C) to a real value. Since we define the logarithm of the probabilities of the language model and the error model (i.e., the edit distance function) as features, the ranker can be viewed as a more general framework, subsuming the source channel model as a special case. In our experiments we used 96 features and a non-linear model, implemented as a two-layer neural net, though the details of the ranker and the features are beyond the scope of this paper.

5 A Phrase-Based Error Model

The goal of the phrase-based error model is to transform a correctly spelled query C into a misspelled query Q. Rather than replacing single words in isolation, this model replaces sequences of words with sequences of words, thus incorporating contextual information. For instance, we might learn that "*theme part*" can be replaced by "*theme park*" with relatively high probability, even though "*part*" is not a misspelled word. We assume the following generative story: first the correctly spelled query C is broken into K non-empty word sequences $\mathbf{c}_1, \ldots, \mathbf{c}_k$, then each is replaced with a new non-empty word sequence $\mathbf{q}_1, \ldots, \mathbf{q}_k$, and finally these phrases are permuted and concatenated to form the misspelled Q. Here, **c** and **q** denote consecutive sequences of words.

To formalize this generative process, let S denote the segmentation of C into K phrases $\mathbf{c}_{1\ldots}\mathbf{c}_K$, and let T denote the K replacement phrases $\mathbf{q}_1\ldots\mathbf{q}_K$ – we refer to these ($\mathbf{c}_i$, $\mathbf{q}_i$) pairs as *bi-phrases*. Finally, let M denote a permutation of K elements representing the final reordering step. Figure 2 demonstrates the generative procedure.

Next let us place a probability distribution over rewrite pairs. Let $B(C, Q)$ denote the set of S, T, M triples that transform C into Q. If we assume a uniform probability over segmentations, then the phrase-based probability can be defined as:

$$P(Q|C) \propto \sum_{\substack{(S,T,M)\in \\ B(C,Q)}} P(T|C,S) \cdot P(M|C,S,T) \quad (4)$$

As is common practice in SMT, we use the maximum approximation to the sum:

$$P(Q|C) \approx \max_{\substack{(S,T,M)\in \\ B(C,Q)}} P(T|C,S) \cdot P(M|C,S,T) \quad (5)$$

5.1 Forced Alignments

Although we have defined a generative model for transforming queries, our goal is not to propose new queries, but rather to provide scores over existing Q and C pairs which act as features for the ranker. Furthermore, the word-level alignments between Q and C can most often be identified with little ambiguity. Thus we restrict our attention to those phrase transformations consistent with a good word-level alignment.

Let J be the length of Q, L be the length of C, and $A = a_1, \ldots, a_J$ be a hidden variable representing the word alignment. Each a_i takes on a value ranging from 1 to L indicating its corresponding word position in C, or 0 if the ith word in Q is unaligned. The cost of assigning k to a_i is equal to the Levenshtein edit distance (Levenshtein, 1966) between the i^{th} word in Q and the k^{th} word in C, and the cost of assigning 0 to a_i is equal to the length of the i^{th} word in Q. We can determine the least cost alignment A^* between Q and C efficiently using the A-star algorithm.

When scoring a given candidate pair, we further restrict our attention to those S, T, M triples that are consistent with the word alignment, which we denote as $B(\text{C}, Q, A^*)$. Here, consistency requires that if two words are aligned in A^*, then they must appear in the same bi-phrase ($\mathbf{c}_i$, $\mathbf{q}_i$). Once the word alignment is fixed, the final permutation is uniquely determined, so we can safely discard that factor. Thus we have:

$$P(Q|C) \approx \max_{\substack{(S,T,M)\in \\ \mathcal{B}(C,Q,A^*)}} P(T|C,S) \quad (6)$$

For the sole remaining factor $P(T|C, S)$, we make the assumption that a segmented query $T = \mathbf{q}_1\ldots\ \mathbf{q}_K$ is generated from left to right by transforming each phrase $\mathbf{c}_1\ldots\mathbf{c}_K$ independently:

```
Input: biPhraseLattice "PL" with length = K & height = L;
Initialization: biPhrase.maxProb = 0;
for (x = 0; x <= K – 1; x++)
    for (y = 1; y <= L; y++)
        for (yPre = 1; yPre <= L; yPre++)
        {
            xPre = x – y;
            biPhrasePre = PL.get(xPre, yPre);
            biPhrase = PL.get(x, y);
            if (!biPhrasePre || !biPhrase)
                continue;
            probIncrs = PL.getProbIncrease(biPhrasePre,
                                           biPhrase);
            maxProbPre = biPhrasePre.maxProb;
            totalProb = probIncrs + maxProbPre;
            if  (totalProb > biPhrase.maxProb)
            {
                biPhrase.maxProb = totalProb;
                biPhrase.yPre = yPre;
            }
        }
Result: record at each bi-phrase boundary its maximum probability (biPhrase.maxProb) and optimal back-tracking biPhrases (biPhrase.yPre).
```

Figure 3: The dynamic programming algorithm for Viterbi bi-phrase segmentation.

$$P(T|C,S) = \prod_{k=1}^{K} P(\mathbf{q}_k|\mathbf{c}_k), \qquad (7)$$

where $P(\mathbf{q}_k|\mathbf{c}_k)$ is a phrase transformation probability, the estimation of which will be described in Section 5.2.

To find the maximum probability assignment efficiently, we can use a dynamic programming approach, somewhat similar to the monotone decoding algorithm described in Och (2002). Here, though, both the input and the output word sequences are specified as the input to the algorithm, as is the word alignment. We define the quantity α_j to be the probability of the most likely sequence of bi-phrases that produce the first j terms of Q and are consistent with the word alignment and C. It can be calculated using the following algorithm:

1. Initialization:

$$\alpha_0 = 1 \qquad (8)$$

2. Induction:

$$\alpha_j = \max_{j' < j, \mathbf{q}=q_{j'+1} \dots q_j} \left\{ \alpha_{j'} P(\mathbf{q}|\mathbf{c}_\mathbf{q}) \right\} \qquad (9)$$

3. Total:

$$P(Q|C) = \alpha_J \qquad (10)$$

	A	B	C	D	E	F
a	#					
d				#		
c			#			
f						#

a	A
adc	ABCD
d	D
dc	CD
dcf	CDEF
c	C
f	F

Figure 4: Toy example of (left) a word alignment between two strings "adcf" and "ABCDEF"; and (right) the bi-phrases containing up to four words that are consistent with the word alignment.

The pseudo-code of the above algorithm is shown in Figure 3. After generating Q from left to right according to Equations (8) to (10), we record at each possible bi-phrase boundary its maximum probability, and we obtain the total probability at the end-position of Q. Then, by back-tracking the most probable bi-phrase boundaries, we obtain B^*. The algorithm takes a complexity of $O(KL^2)$, where K is the total number of word alignments in A^* which does not contain empty words, and L is the maximum length of a bi-phrase, which is a hyper-parameter of the algorithm. Notice that when we set L=1, the phrase-based error model is reduced to a word-based error model which assumes that words are transformed independently from C to Q, without taking into account any contextual information.

5.2 Model Estimation

We follow a method commonly used in SMT (Koehn et al., 2003) to extract bi-phrases and estimate their replacement probabilities. From each query-correction pair with its word alignment (Q, C, A^*), all bi-phrases consistent with the word alignment are identified. Consistency here implies two things. First, there must be at least one aligned word pair in the bi-phrase. Second, there must not be any word alignments from words inside the bi-phrase to words outside the bi-phrase. That is, we do not extract a phrase pair if there is an alignment from within the phrase pair to outside the phrase pair. The toy example shown in Figure 4 illustrates the bilingual phrases we can generate by this process.

After gathering all such bi-phrases from the full training data, we can estimate conditional relative frequency estimates without smoothing. For example, the phrase transformation probability $P(\mathbf{q}|\mathbf{c})$ in Equation (7) can be estimated approximately as

$$P(\mathbf{q}|\mathbf{c}) = \frac{N(\mathbf{c},\mathbf{q})}{\sum_{\mathbf{q}'} N(\mathbf{c},\mathbf{q}')} \qquad (11)$$

where $N(\mathbf{c},\mathbf{q})$ is the number of times that **c** is aligned to **q** in training data. These estimates are useful for contextual lexical selection with sufficient training data, but can be subject to data sparsity issues.

An alternate translation probability estimate not subject to data sparsity issues is the so-called *lexical weight* estimate (Koehn et al., 2003). Assume we have a word translation distribution $t(q|c)$ (defined over individual words, not phrases), and a word alignment A between **q** and **c**; here, the word alignment contains (*i*, *j*) pairs, where $i \in 1..|\boldsymbol{q}|$ and $j \in 0..|\mathbf{c}|$, with 0 indicating an inserted word. Then we can use the following estimate:

$$P_w(\mathbf{q}|\mathbf{c},A) = \prod_{i=1}^{|\mathbf{q}|} \frac{1}{|\{j|(j,i) \in A\}|} \sum_{\forall (i,j) \in A} t(q_i|c_j) \qquad (12)$$

We assume that for every position in **q**, there is either a single alignment to 0, or multiple alignments to non-zero positions in **c**. In effect, this computes a product of per-word translation scores; the per-word scores are averages of all the translations for the alignment links of that word. We estimate the word translation probabilities using counts from the word aligned corpus: $t(q|c) = \frac{N(c,q)}{\sum_{q'} N(c,q')}$. Here $N(c,q)$ is the number of times that the words (not phrases as in Equation 11) c and q are aligned in the training data. These word based scores of bi-phrases, though not as effective in contextual selection, are more robust to noise and sparsity.

Throughout this section, we have approached this model in a noisy channel approach, finding probabilities of the misspelled query given the corrected query. However, the method can be run in both directions, and in practice SMT systems benefit from also including the direct probability of the corrected query given this misspelled query (Och, 2002).

5.3 Phrase-Based Error Model Features

To use the phrase-based error model for spelling correction, we derive five features and integrate them into the ranker-based query speller system, described in Section 4. These features are as follows.

- **Two phrase transformation features:** These are the phrase transformation scores based on relative frequency estimates in two directions. In the correction-to-query direction, we define the feature as $f_{PT}(Q,C,A) = \log P(Q|C)$, where $P(Q|C)$ is computed by Equations (8) to (10), and $P(\mathbf{q}|\mathbf{c_q})$ is the relative frequency estimate of Equation (11).
- **Two lexical weight features:** These are the phrase transformation scores based on the lexical weighting models in two directions. For example, in the correction-to-query direction, we define the feature as$f_{LW}(Q,C,A) = \log P(Q|C)$, where $P(Q|C)$ is computed by Equations (8) to (10), and the phrase transformation probability is the computed as lexical weight according to Equation (12).
- **Unaligned word penalty feature:** the feature is defined as the ratio between the number of unaligned query words and the total number of query words.

6 Experiments

We evaluate the spelling error models on a large scale real world data set containing 24,172 queries sampled from one year's worth of query logs from a commercial search engine. The spelling of each query is judged and corrected by four annotators.

We divided the data set into training and test data sets. The two data sets do not overlap. The training data contains 8,515 query-correction pairs, among which 1,743 queries are misspelled (i.e., in these pairs, the corrections are different from the queries). The test data contains 15,657 query-correction pairs, among which 2,960 queries are misspelled. The average length of queries in the training and test data is 2.7 words.

The speller systems we developed in this study are evaluated using the following three metrics.

- **Accuracy:** The number of correct outputs generated by the system divided by the total number of queries in the test set.
- **Precision:** The number of correct spelling corrections for misspelled queries generated by the system divided by the total number of corrections generated by the system.
- **Recall:** The number of correct spelling corrections for misspelled queries generated by the system divided by the total number of misspelled queries in the test set.

We also perform a significance test, i.e., a t-test with a significance level of 0.05. A significant difference should be read as significant at the 95% level.

#	System	Accuracy	Precision	Recall
1	Source-channel	0.8526	0.7213	0.3586
2	Ranker-based	0.8904	0.7414	0.4964
3	Word model	0.8994	0.7709	0.5413
4	Phrase model (*L*=3)	0.9043	0.7814	0.5732

Table 1. Summary of spelling correction results.

#	System	Accuracy	Precision	Recall
5	Phrase model (*L*=1)	0.8994	0.7709	0.5413
6	Phrase model (*L*=2)	0.9014	0.7795	0.5605
7	Phrase model (*L*=3)	0.9043	0.7814	0.5732
8	Phrase model (*L*=5)	0.9035	0.7834	0.5698
9	Phrase model (*L*=8)	0.9033	0.7821	0.5713

Table 2. Variations of spelling performance as a function of phrase length.

#	System	Accuracy	Precision	Recall
10	*L*=3; 0 month data	0.8904	0.7414	0.4964
11	*L*=3; 0.5 month data	0.8959	0.7701	0.5234
12	*L*=3; 1.5 month data	0.9023	0.7787	0.5667
13	*L*=3; 3 month data	0.9043	0.7814	0.5732

Table 3. Variations of spelling performance as a function of the size of clickthrough data used for training.

In our experiments, all the speller systems are ranker-based. In most cases, other than the baseline system (a linear neural net), the ranker is a two-layer neural net with 5 hidden nodes. The free parameters of the neural net are trained to optimize accuracy on the training data using the back propagation algorithm, running for 500 iterations with a very small learning rate (0.1) to avoid overfitting. We did not adjust the neural net structure (e.g., the number of hidden nodes) or any training parameters for different speller systems. Neither did we try to seek the best tradeoff between precision and recall. Since all the systems are optimized for accuracy, we use accuracy as the primary metric for comparison.

Table 1 summarizes the main spelling correction results. Row 1 is the baseline speller system where the source-channel model of Equations (2) and (3) is used. In our implementation, we use a linear ranker with only two features, derived respectively from the language model and the error model models. The error model is based on the edit distance function. Row 2 is the ranker-based spelling system that uses all 96 ranking features, as described in Section 4. Note that the system uses the features derived from two error models. One is the edit distance model used for candidate generation. The other is a phonetic model that measures the edit distance between the metaphones (Philips, 1990) of a query word and its aligned correction word. Row 3 is the same system as Row 2, with an additional set of features derived from a word-based error model. This model is a special case of the phrase-based error model described in Section 5 with the maximum phrase length set to one. Row 4 is the system that uses the additional 5 features derived from the phrase-based error models with a maximum bi-phrase length of 3.

In phrase based error model, *L* is the maximum length of a bi-phrase (Figure 3). This value is important for the spelling performance. We perform experiments to study the impact of *L*; the results are displayed in Table 2. Moreover, since we proposed to use clickthrough data for spelling correction, it is interesting to study the impact on spelling performance from the size of clickthrough data used for training. We varied the size of clickthrough data and the experimental results are presented in Table 3.

The results show first and foremost that the ranker-based system significantly outperforms the spelling system based solely on the source-channel model, largely due to the richer set of features used (Row 1 vs. Row 2). Second, the error model learned from clickthrough data leads to significant improvements (Rows 3 and 4 vs. Row 2). The phrase-based error model, due to its capability of capturing contextual information, outperforms the word-based model with a small but statistically significant margin (Row 4 vs. Row 3), though using phrases longer ($L > 3$) does not lead to further significant improvement (Rows 6 and 7 vs. Rows 8 and 9). Finally, using more clickthrough data leads to significant improvement (Row 13 vs. Rows 10 to 12). The benefit does not appear to have peaked – further improvements are likely given a larger data set.

7 Conclusions

Unlike conventional textual documents, most search queries consist of a sequence of key words, many of which are valid search terms but are not stored in any compiled lexicon. This presents a challenge to any speller system that is based on a dictionary.

This paper extends the recent research on using Web data and query logs for query spelling correction in two aspects. First, we show that a large amount of training data (i.e. query-correction pairs) can be extracted from clickthrough data, focusing on query reformulation sessions. The resulting data are very clean and effective for error model training. Second, we argue that it is critical to capture contextual information for query spelling correction. To this end, we propose a new phrase-based error model, which leads to significant improvement in our spelling correction experiments.

There is additional potentially useful information that can be exploited in this type of model. For example, in future work we plan to investigate the combination of the clickthrough data collected from a Web browser with the noisy but large query sessions collected from a commercial search engine.

Acknowledgments

The authors would like to thank Andreas Bode, Mei Li, Chenyu Yan and Galen Andrew for the very helpful discussions and collaboration.

References

Agichtein, E., Brill, E. and Dumais, S. 2006. Improving web search ranking by incorporating user behavior information. In *SIGIR*, pp. 19-26.

Ahmad, F., and Kondrak, G. 2005. Learning a spelling error model from search query logs. In *HLT-EMNLP*, pp 955-962.

Brill, E., and Moore, R. C. 2000. An improved error model for noisy channel spelling correction. In *ACL*, pp. 286-293.

Chen, Q., Li, M., and Zhou, M. 2007. Improving query spelling correction using web search results. In *EMNLP-CoNLL*, pp. 181-189.

Church, K., Hard, T., and Gao, J. 2007. Compressing trigram language models with Golomb coding. In *EMNLP-CoNLL*, pp. 199-207.

Cucerzan, S., and Brill, E. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *EMNLP*, pp. 293-300.

Gao, J., Yuan, W., Li, X., Deng, K., and Nie, J-Y. 2009. Smoothing clickthrough data for web search ranking. In *SIGIR*.

Golding, A. R., and Roth, D. 1996. Applying winnow to context-sensitive spelling correction. In *ICML*, pp. 182-190.

Joachims, T. 2002. Optimizing search engines using clickthrough data. In *SIGKDD*, pp. 133-142.

Kernighan, M. D., Church, K. W., and Gale, W. A. 1990. A spelling correction program based on a noisy channel model. In *COLING*, pp. 205-210.

Koehn, P., Och, F., and Marcu, D. 2003. Statistical phrase-based translation. In *HLT/NAACL*, pp. 127-133.

Kukich, K. 1992. Techniques for automatically correcting words in text. *ACM Computing Surveys*. 24(4): 377-439.

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707-710.

Li, M., Zhu, M., Zhang, Y., and Zhou, M. 2006. Exploring distributional similarity based models for query spelling correction. In *ACL*, pp. 1025-1032.

Mangu, L., and Brill, E. 1997. Automatic rule acquisition for spelling correction. In *ICML*, pp. 187-194.

Och, F. 2002. *Statistical machine translation: from single-word models to alignment templates.* PhD thesis, RWTH Aachen.

Och, F., and Ney, H. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4): 417-449.

Okazaki, N., Tsuruoka, Y., Ananiadou, S., and Tsujii, J. 2008. A discriminative candidate generator for string transformations. In *EMNLP*, pp. 447-456.

Philips, L. 1990. Hanging on the metaphone. *Computer Language Magazine*, 7(12):38-44.

Suzuki, H., Li, X., and Gao, J. 2009. Discovery of term variation in Japanese web search queries. In *EMNLP*.

Toutanova, K., and Moore, R. 2002. Pronunciation modeling for improved spelling correction. In *ACL*, pp. 144-151.

Wang, X., and Zhai, C. 2008. Mining term association patterns from search logs for effective query reformulation. In *CIKM*, pp. 479-488.

Whitelaw, C., Hutchinson, B., Chung, G. Y., and Ellis, G. 2009. Using the web for language independent spellchecking and autocorrection. In *EMNLP*, pp. 890-899.