Spelling correction as an iterative process that exploits the collective knowledge of web users

Silviu Cucerzan and Eric Brill
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{silviu,brill}@microsoft.com

Abstract

Logs of user queries to an internet search engine provide a large amount of implicit and explicit information about language. In this paper, we investigate their use in spelling correction of search queries, a task which poses many additional challenges beyond the traditional spelling correction problem. We present an approach that uses an iterative transformation of the input query strings into other strings that correspond to more and more likely queries according to statistics extracted from internet search query logs.

1 Introduction

The task of general purpose spelling correction has a long history (e.g. Damerau, 1964; Rieseman and Hanson, 1974; McIlroy, 1982), traditionally focusing on resolving typographical errors such as insertions, deletions, substitutions, and transpositions of letters that result in *unknown words* (i.e. words not found in a trusted lexicon of the language). Typical word processing spell checkers compute for each unknown word a small set of in-lexicon alternatives to be proposed as possible corrections, relying on information about in-lexicon-word frequencies and about the most common keyboard mistakes (such as typing *m* instead of *n*) and phonetic/cognitive mistakes, both at word level (e.g. the use of *acceptible* instead of *acceptable*) and at character level (e.g. the misuse of *f* instead of *ph*). Very few spell checkers attempt to detect and correct *word substitution errors*, which refer to the use of in-lexicon words in inappropriate contexts and can also be the result of both typographical mistakes (such as typing *coed* instead of *cord*) and cognitive mistakes (e.g. *principal* and *principle*). Some research efforts to tackle this problem have been made; for example Heidorn et al. (1982) and Garside et al. (1987) developed systems that rely on syntactic patterns to detect substitution errors, while Mays et al. (1991) employed word co-occurrence evidence from a large corpus to detect and correct such errors. The former approaches were based on the impractical assumption that all possible syntactic uses of all words (i.e. part-of-speech) are known, and presented both recall and precision problems because many of the substitution errors are not syntactically anomalous and many unusual syntactic constructions do not contain errors. The latter approach had very limited success under the assumptions that each sentence contains at most one misspelled word, each misspelling is the result of a single *point change* (insertion, deletion, substitution, or transposition), and the *defect rate* (the relative number of errors in the text) is known. A different body of work (e.g. Golding, 1995; Golding and Roth, 1996; Mangu and Brill, 1997) focused on resolving a limited number of cognitive substitution errors, in the framework of context sensitive spelling correction (CSSC). Although promising results were obtained (92-95% accuracy), the scope of this work was very limited as it only addressed known sets of commonly confused words, such as {*peace*, *piece*}).

1.1 Spell Checking of Search Engine Queries

The task of web-query spelling correction addressed in this work has many similarities to traditional spelling correction but also poses additional challenges. Both the frequency and severity of spelling errors for search queries are significantly greater than in word processing. Roughly 10-15% of the queries sent to search engines contain errors. Typically, the validity of a query cannot be decided by lexicon look-up or by checking its grammaticality. Because web queries are very short (on average, less than 3 words), techniques that use a multitude of features based on relatively wide context windows, such as those investigated in CSSC, are difficult to apply. Rather than being well-formed sentences, most queries consist of one concept or an enumeration of concepts, many times containing legitimate words that are not found in any traditional lexicon.

Just defining what a valid web query is represents a difficult enterprise. We clearly cannot use only a static trusted lexicon, as many new names and concepts (such as *aznar*, *blog*, *naboo*, *nimh*, *nsync*, and *shrek*) become popular every day and it would

be extremely difficult if not impossible to maintain a high-coverage lexicon. In addition, employing very large lexicons can result in more errors surfacing as word substitutions, which are very difficult to detect, rather than as unknown words.

One alternative investigated in this work is to exploit the continuously evolving expertise of millions of people that use web search engines, as collected in *search query logs* (seen as histograms over the queries received by a search engine). In some sense, we could say that the validity of a word can be inferred from its frequency in what people are querying for, similarly to Wittgenstein's (1968) observation that "the meaning of a word is its use in the language". Such an approach has its own caveats. For example, it would be erroneous to simply extract from web-query logs all the queries whose frequencies are above a certain value and consider them valid. Misspelled queries such as *britny spears* are much more popular than correctly spelled queries such as *bayesian nets* and *amd processors*. Our challenge is to try to utilize query logs to learn what queries are valid, and to build a model for valid query probabilities, despite the fact that a large percentage of the logged queries are misspelled and there is no trivial way to determine the valid from invalid queries.

2 Problem Formulation. Prior Work

Comprehensive reviews of the spelling correction literature were provided by Peterson (1980), Kukich (1992), and Jurafsky and Martin (2000). In this section, we survey a few lexicon-based spelling correction approaches by using a series of formal definitions of the task and presenting concrete examples showing the strengths and the limits corresponding to each situation. We iteratively redefine the problem, starting from an approach purely based on a trusted lexicon and ending up with an approach in which the role of the trusted lexicon is greatly diminished. While doing so, we also make concrete forward steps in our attempt to provide a definition of *valid web queries*.

Let Σ be the alphabet of a language and $L \subset \Sigma^*$ a broad-coverage lexicon of the language. The simplest and historically the first definition of lexicon-based spelling correction (Damerau, 1964) is: Given an unknown word $w \in \Sigma^* \setminus L$, find $w' \in L$ such that $dist(w,w') = \min_{v \in L} dist(w,v)$.

i.e. for any out-of-lexicon word in a text, find the closest word form(s) in the available lexicon and hypothesize it as the correct spelling alternative. *dist* can be any string-based function; for example, it can be the ratio between the number of letters two words do not have in common and the number of letters they share.[1] The two most used classes of distances in spelling correction are edit distances, as proposed by Damerau (1964) and Levenshtein (1965), and correlation matrix distances (Cherkassky et al., 1974). In our study, we use a modified version of the Damerau-Levenshtein edit distance, as presented in Section 3.

One flaw of the preceding formulation is that it does not take into account the frequency of words in a language. A simple solution to this problem is to compute the probability of words in the target language as maximum likelihood estimates (MLE) over a large corpus and reformulate the general spelling-correction problem as follows:

Given $w \in \Sigma^* \setminus L$, find $w' \in L$ such that $dist(w,w') \le \delta$ and $P(w') = \max_{v \in L: dist(w,v) \le \delta} P(v)$.

In this formulation, all in-lexicon words that are within some "reasonable" distance δ of the unknown word are considered as good candidates, the correction being chosen based on its prior probability in the language. While there is an implicit conditioning on the original spelling because of the domain on which the best correction is searched, this objective function only uses the prior probability of words in the language and not the actual distances between each candidate and the input word

One solution that allows using a probabilistic edit distance is to condition the probability of a correction on the original spelling $P(v \mid w)$:

Given $w \in \Sigma^* \setminus L$, find $w' \in L$ such that $dist(w,w') \le \delta$ and $P(w' \mid w) = \max_{v \in L: dist(w,v) \le \delta} P(v \mid w)$.

In a noisy channel model framework, as employed for spelling correction by Kernigham et al. (1990), the objective function can be written by using Bayesian inversion as the product between the prior probability of words in a language $P(v)$ (*the language model*), and the likelihood of misspelling a word *v* as *w*, $P(w \mid v)$ (which models the noisy channel and will be called *the error model*).

In the above formulations, unknown words are corrected in isolation. This is a rather major flaw because context is extremely important for spelling correction, as illustrated in the following example:

power crd → *power cord*
video crd → *video card*

[1] Note that the function does not have to be symmetric; thus, the notation *dist*(*w*,*w*′) is used with a loose sense.

The misspelled word *crd* should be corrected to two different words depending on its contexts.[2]

A formulation of the spelling correction problem that takes into account context is the following:

Given a string $s \in \Sigma^*$, $s = c_l w c_r$, with $w \in \Sigma^* \setminus L$ and $c_l, c_r \in L^*$, find $w' \in L$ such that $dist(w, w') \leq \delta$ and $P(w' | c_l w c_r) = \max_{v \in L: dist(w,v) \leq \delta} P(v | c_l w c_r)$.

Spaces and other word delimiters are ignored in this formulation and the subsequent formulations for simplicity, although text tokenization represents an important part of the spelling-correction process, as discussed in Sections 5 and 6.

The task definitions enumerated up to this point (on which most traditional spelling correction systems are based) ignore word substitution errors. In the case of web searches, it is extremely important to provide correction suggestions for valid words when they are more meaningful as a search query than the original query, for example:

golf war → *gulf war*
sap opera → *soap opera*

This problem is partially addressed by the task of CSSC, which can be formalized as follows:

Given a set of confusable valid word forms in a language $W = \{w_1, w_2, ..., w_n\}$ and a string $s = c_l w_i c_r$, choose $w_j \in W$ such that $P(w_j | c_l w_i c_r) = \max_{k=1..n} P(w_k | c_l w_i c_r)$.

In the CSSC literature, the sets of confusables are presumed known, but they could also be built for each in-lexicon word *w* as all words *w'* with $dist(w, w') \leq \delta$, similarly to the approach investigated by Mays et al. (1991), in which they chose a $\delta = 1$ and employed an edit distance with all point changes having the same cost 1.

The generalized problem of phrasal spelling correction can then be formulated as follows:

Given $s \in \Sigma^*$, find $s' \in L^*$ such that $dist(s, s') \leq \delta$ and $P(s' | s) = \max_{t \in L^*: dist(s,t) \leq \delta} P(t | s)$.

Typically, a correction is desirable when $s \notin L^*$ (i.e. at least one of the component words is unknown) but, as shown above, there are frequent cases (e.g. *golf war*) when sequences of valid words should be changed to other word sequences. Note that word boundaries are hidden in this latter formulation, making it more general and allowing it to cover two other important spelling error classes, concatenation and splitting, e.g.:

power point slides → *powerpoint slides*
chat inspanich → *chat in spanish*

Yet, it still does not account for another important class of cases in web query correction which is represented by out-of-lexicon words that are valid in certain contexts (therefore, $s' \notin L^*$), for example:

amd processors → *amd processors* (no change)

The above phrase represents a legitimate query, despite the fact that it may contain unknown words when employing a traditional English lexicon.

Some even more interesting cases not handled by traditional spellers and also not covered by the latter formulation are those in which in-lexicon words should be changed to out-of-lexicon words, as in the following examples, where two valid words must be concatenated into an out of lexicon word:

gun dam planet → *gundam planet*
limp biz kit → *limp bizkit*

These observations lead to an even more general formulation of the spelling-correction problem:

Given $s \in \Sigma^*$, find $s' \in \Sigma^*$ such that $dist(s, s') \leq \delta$ and $P(s' | s) = \max_{t \in \Sigma^*: dist(s,t) \leq \delta} P(t | s)$.

For the first time, the formulation no longer makes explicit use of a lexicon of the language.[3] In some sense, the actual language in which the web queries are expressed becomes less important than the query-log data from which the string probabilities are estimated. This probability model can be seen as a substitute for a measure of the meaningfulness of strings as web-queries. For example, an implausible random noun phrase in any of the traditional corpora such as *sad tomatoes* is meaningful in the context of web search (being the name of a somewhat popular music band).

3 The Error Model. String Edit Functions

All formulations of the spelling correction task given in the previous section used a string distance function and a threshold to restrict the space in which alternative spellings are searched. Various previous work has addressed the problem of choosing appropriate functions (e.g. Kernigham et al. 1990, Brill and Moore, 2002; Toutanova and Moore, 2003).

[2] To simplify the exposition, we only consider two highly probable corrections, but other valid alternatives exist, e.g. *video cd*.

[3] A trusted lexicon may still be used in the estimation of the language model probability for the computation of $P(t | s)$.

The choice of distance function d and threshold δ could be extremely important for the accuracy of a speller. At one extreme, the use of a too restrictive function/threshold combination can result in not finding the best correction for a given query. For example, using the vanilla Damerau-Levenshtein edit distance (defined as the minimum number of point changes required to transform a string into another, where a point change is one of the following operations: insertion of a letter, deletion of a letter, and substitution of one letter with another letter) and a threshold $\delta = 1$, the correction *donadl duck* → *donald duck* would not be possible. At the other extreme, the use of a less limiting function might have as consequence suggesting very unlikely corrections. For example, using the same classical Levenshtein distance and $\delta = 2$ would allow the correction of the string *donadl duck*, but will also lead to bad corrections such as *log wood* → *dog food* (based on the frequency of the queries, as incorporated in $P(s)$). Nonetheless, large distance corrections are still desirable in a diversity of situations, for example:

platnuin rings → *platinum rings*
ditroitigers → *detroit tigers*

The system described in this paper makes use of a modified context-dependent weighted Damerau-Levenshtein edit function which allows insertion, deletion, substitution, immediate transposition, and long-distance movement of letters as point changes, for which the weights were interactively refined using statistics from query logs.

4 The Language Model. Exploiting Large Web Query Logs

A misspelling such as *ditroitigers* is far from the correct alternative and thus, it might be extremely difficult to find its correct spelling based solely on edit distance. Nonetheless, the correct alternative could be reached by allowing intermediate *valid correction steps*, such as *ditroitigers* → *detroittigers* → *detroit tigers*. But what makes *detroittigers* a valid correction step? Recall that the last formulation of spelling correction in Section 3 did not explicitly use a lexicon of the language. Rather, any string that appears in the query log used for training can be considered a valid correction and can be suggested as an alternative to the current web query based on the relative frequency of the query and the alternative spelling. Thus, a spell checker built according to this formulation could suggest the correction *detroittigers* because this alternative occurs frequently enough in the employed query log. However, *detroittigers* itself could be corrected to *detroit tigers* if presented as a stand-alone query to this spell checker, based on similar query-log frequency facts, which naturally leads to the idea of an iterative correction approach.

albert einstein	4834
albert einstien	525
albert einstine	149
albert einsten	27
albert einsteins	25
albert einstain	11
albert einstin	10
albert eintein	9
albeart einstein	6
aolbert einstein	6
alber einstein	4
albert einseint	3
albert einsteirn	3
albert einsterin	3
albert eintien	3
alberto einstein	3
albrecht einstein	3
alvert einstein	3

Table 1. Counts of different (mis)spellings of Albert Einstein's name in a web query log.

Essential to such an approach are three typical properties of the query logs (e.g. see Table 1):

- words in the query logs are misspelled in various ways, from relatively easy-to-correct misspellings to very-difficult-to-correct ones, that make the user's intent almost impossible to recognize;
- the less *malign* (difficult to correct) a misspelling is the more frequent it is;
- the correct spellings tend to be more frequent than misspellings.

In this context, the spelling correction problem can be given the following iterative formulation:

Given a string $s_0 \in \Sigma^*$, find a sequence $s_1, s_2, \ldots s_n \in \Sigma^*$ such that $dist(s_i, s_{i+1}) \le \delta$, $P(s_{i+1} \mid s_i) = \max_{t \in \Sigma^*: dist(s_i, t) \le \delta} P(t \mid s_i)$, $\forall i \in 0..n-1$, and $P(s_n \mid s_n) = \max_{t \in \Sigma^*: dist(s_n, t) \le \delta} P(t \mid s_n)$.

An example of correction that can be made by iteratively applying the base spell checker is:

anol scwartegger → *arnold schwarzenegger*

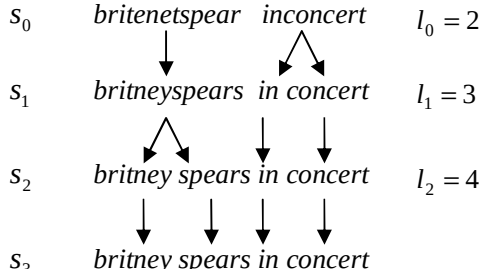
Misspelled query:	*anol scwartegger*
First iteration:	*arnold schwartnegger*
Second iteration:	*arnold schwarznegger*
Third iteration:	*arnold schwarzenegger*
Fourth iteration:	no further correction

Up to this point, we underspecified the notion of string in the task formulations given. One possibility is to consider whole queries as the strings to be corrected and iteratively search for better logged queries according to the agreement between their relative frequencies and the character error model. This is equivalent to identifying all queries in the query log that are misspellings of other queries and for any new query, find a correction sequence of logged queries. While such an approach exploits the vast information available in web-query logs, it only covers exact matches of the queries that appear in these logs and provides a low coverage of infrequent queries. For example, a query such as *britnet spear inconcert* could not be corrected if the correction *britney spears in concert* does not appear in the employed query log, although the substring *britnet spear* could be corrected to *britney spears*.

To address the shortcomings of such an approach, we propose a system based on the following formulation, which uses query substrings:

Given $s_0 \in \Sigma^*$, find a sequence $s_1, s_2, ...s_n \in \Sigma^*$, such that for each $i \in 0..n-1$ there exist the decompositions $s_i = w_{i,0}^1 ... w_{i,0}^{l_i}, \mathrm{s}_{i+1} = w_{i+1,1}^1 ... w_{i+1,1}^{l_i}$, where $w_{j,h}^k$ are words or groups of words such that $dist(w_{i,0}^k, w_{i+1,1}^k) \le \delta$, $\forall i \in 0..n-1$, $\forall k \in 1..l_i$ and $P(s_{i+1} \mid s_i) = \max_{t \in \Sigma^*: dist(s_i, t) \le \delta^*} P(t \mid s_i)$, $\forall i \in 0..n-1$, and $P(s_n \mid s_n) = \max_{t \in \Sigma^*: dist(s_n, t) \le \delta^*} P(t \mid s_n)$.

Note that the length of the string decomposition may vary from one iteration to the next one, for example:



In the implementation evaluated in this paper, we allowed decompositions of query strings into words and word bigrams. The tokenization process uses space and punctuation delimiters in addition to the information provided about multi-word compounds (e.g. *add-on* and *back-up*) by a trusted English lexicon with approximately 200k entries. By using the tokenization process described above, we extracted word unigram and bigram statistics from query logs to be used as the system's language model.

5 Query Correction

An input query is tokenized using the same space and word-delimiter information in addition to the available lexical information as used for processing the query log. For each token, a set of alternatives is computed using the weighted Levenshtein distance function described in Section 3 and two different thresholds for in-lexicon and out-of-lexicon tokens

Matches are searched in the space of word unigrams and bigrams extracted from query logs in addition to the trusted lexicon. Unigrams and bigrams are stored in the same data structure on which the search for correction alternatives is done. Because of this, the proposed system handles concatenation and splitting of words in exactly the same manner as it handles transformations of words to other words.

Once the sets of all possible alternatives are computed for each word form in the query, a modified Viterbi search (in which the transition probabilities are computed using bigram and unigram query-log statistics and output probabilities are replaced with inverse distances between words) is employed to find the best possible alternative string to the input query under the following constraint: no two adjacent in-vocabulary words are allowed to change simultaneously. This constraint prevents changes such as *log wood* → *dog food*. An algorithmic consequence of this constraint is that there is no need to search all the possible paths in the trellis, which makes the modified search procedure much faster, as described further. We assume that the list of alternatives for each word is randomly ordered but the input word is on the first position of the list when the word is in the trusted lexicon. In this case, the searched paths form what we call *fringes*. Figure 1 presents an example of a trellis in which w^1, w^2 and w^3 are in-lexicon word forms. Observe that instead of computing the cost of $k_1 \cdot k_2$ possible paths between the alternatives corresponding to w^1 and w^2, we only need to compute the cost of $k_1 + k_2$ paths.

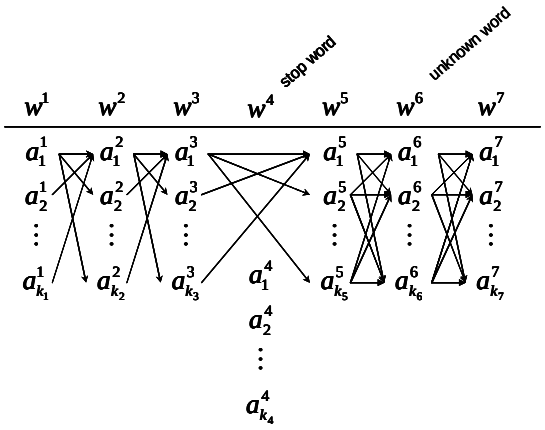


Figure 1. Example of trellis of the modified Viterbi search

Because we use word-bigram statistics, stop words such as prepositions and conjunctions may interfere negatively with the best path search. For example, in correcting a query such as *platunum and rigs*, the language model based on word bigrams would not provide a good context for the word form *rigs*.

To avoid this type of problems, stop words and their most likely misspelling are given a special treatment. The search is done by first ignoring them, as in Figure 1, where w^4 is presumed to be such a word. Once a best path is found by ignoring stop words, the best alternatives for the skipped stop words (or their misspellings) are computed in a second Viterbi search with fringes in which the extremities are fixed, as presented in Figure 2.

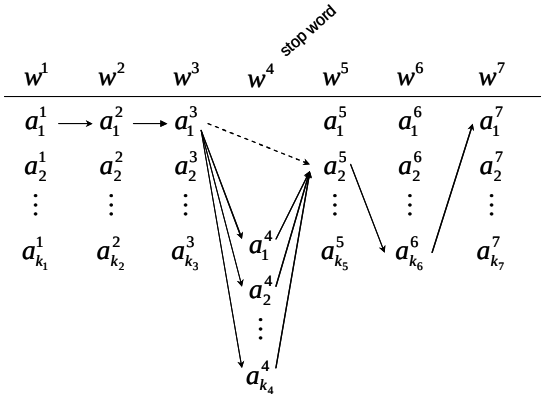


Figure 2. Modified Viterbi search – stop-word treatment

The approach of search with fringes coupled with an iterative correction process is both very efficient and very effective. In each iteration, the search space is much reduced. Changes such as *log wood* → *dog food* are avoided because they can not be made in one iteration and there are no intermediate corrections conditionally more probable than the left-hand-side query (*log wood*) and less probable than the right-hand-side query (*dog food*).

An iterative process is prone to other types of problems. Short queries can be iteratively transformed into other un-related queries; therefore, changing such queries is restricted additionally in our system. Another restriction we imposed is to not allow changes of in-lexicon words in the first iteration, so that easy-to-fix unknown-word errors are handled before any word substitution error.

6 Evaluation

For this work, we are concerned primarily with recall because providing good suggestions for misspelled queries can be viewed as more important than abstaining to provide alternative query suggestions for valid queries as long as these suggestions are reasonable (for example, suggesting *cowboy ropes* for *cowboy robes* may not have major cost to a user). A real system would have a component that decides whether to surface a spelling suggestion based on where we want to be on the ROC curve, thus negotiating between precision and recall.

One problem with evaluating a spell checker designed to correct search queries is that evaluation data is hard to get. Even if the system were used by a search engine and click-through information were available, such information would provide only a crude measure of precision and would not allow us to measure recall, by capturing only cases in which the corrections proposed by that particular speller are clicked on by the users.

We performed two different evaluations of the proposed system.[4] The first evaluation was done on a test set comprising 1044 unique randomly sampled queries from a daily query log, which were annotated by two annotators. Their inter-agreement rate was 91.3%. 864 of these queries were considered valid by both annotators; for the other 180, the annotators provided spelling corrections. The overall agreement of our system with the annotators was 81.8%. The system suggested 131 alternative queries for the valid set, counted as false positives, and 156 alternative queries for the misspelled set. Table 2 shows the accuracy obtained by the proposed system and results from an ablation study where we disabled various components of the system, to measure their influence on performance.

[4] The test data sets can be downloaded from http://research.microsoft.com/~silviu/Work

	All queries	Valid	Misspelled
Nr. queries	1044	864	180
Full system	**81.8**	**84.8**	**67.2**
No lexicon	70.3	72.2	61.1
No query log	77.0	82.1	52.8
All edits equal	80.4	83.3	66.1
Unigrams only	54.7	57.4	41.7
1 iteration only	80.9	88.0	47.2
2 iterations only	81.3	84.4	66.7
No fringes	80.6	83.3	67.2

Table 2. Accuracy of various instantiations of the system

By completely removing the trusted lexicon, the accuracy of the system on misspelled queries (61.1%) was higher than in the case of only using a trusted lexicon and no query log data (52.8%). It can also be observed that the language model built using query logs is by far more important than the channel model employed: using a poorer character error model by setting all edit weights equal did not have a major impact on performance (66.1% recall), while using a poorer language model that only employs unigram statistics from the query logs crippled the system (41.7% recall). Another interesting aspect is related to the number of iterations. Because the first iteration is more conservative than the following iterations, using only one iteration led to fewer false positives but also to a much lower recall (47.2%). Two iterations were sufficient to correct most of the misspelled queries that the full system could correct. While fringes did not have a major impact on recall, they helped avoid false positives (and had a major impact on speed).

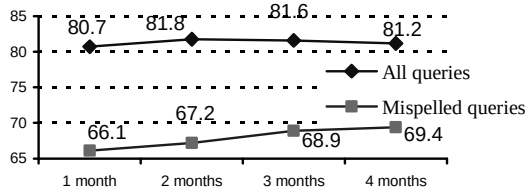


Figure 3. Accuracy and recall as functions of the number of monthly query logs used to train the language model

Figure 3 shows the performance of the full system as a function of the number of monthly query logs employed. While both the total accuracy and the recall increased when using 2 months of data instead of 1 month, by using more query log data (3 and 4 month), the recall (or accuracy on misspelled queries) still improves but at the expense of having more false positives for valid queries, which leads to an overall slightly smaller accuracy.

A post-analysis of the results showed that the system suggested in many cases reasonable corrections but different from the gold standard ones. Many false positives could be considered reasonable suggestions, although it is not clear whether they would have been helpful to the users (e.g. *2002 kawasaki ninja zx6e* → *2002 kawasaki ninja zx6r* was counted as an error, although the suggestion represents a more popular motorcycle model). In the case of misspelled queries in which the user's intent was not clear, the suggestion made by the system could be considered valid despite the fact that it disagreed with the annotators' choice (e.g. *gogle* → *google* instead of the gold standard correction *goggle*).

To address the problems generated by the fact that the annotators could only guess the user intent, we performed a second evaluation, on a set of queries randomly extracted from query log data, by sampling pairs of successive queries (q_1, q_2) sent by the same users in which the queries differ from one another by an un-weighted edit distance of at most 1+(len(q_1)+len(q_2))/10 (i.e. allow a point change for every 5 letters). We then presented the list to human annotators who had the option to reject a pair, choose one of the queries as a valid correction of the other, or propose a correction for both when none of them were valid but the intended valid query was easy to guess from the sequence, as in example 3 below:

(*audio flie, audio file*) ⇒ *audio file*
(*bueavista, buena vista*) ⇒ *buena vista*
(*carrabean nooms, carrabean rooms*) ⇒ *caribbean rooms*

Table 3 shows the performance obtained by different instantiations of the system on this set.

Full system	**73.1**
No lexicon	59.2
No query log	44.9
All edits equal	69.9
Unigrams only	43.0
1 iteration only	45.5
2 iterations only	68.2
No fringes	71.0

Table 3. Accuracy of the proposed system on a set which contains misspelled queries that the users had reformulated

The main system disagreed 99 times with the gold standard, in 80 of these cases suggesting a different correction. 40 of the corrections were not appropriate (e.g. *porat* was corrected by our system to *pirate* instead of *port* in *chinese porat also called xiamen*), 15 were functionally equivalent

corrections given our target search engine (e.g. *audio flie → audio files* instead of *audio file*), 17 were different valid suggestions (e.g. *bellsouth lphone isting → bellsouth phone listings* instead of *bellsouth telephone listing*), while 8 represented gold standard errors (e.g. the speller correctly suggested *brandy sniffters → brandy snifters* instead of *brandy sniffers*). Out of 19 cases in which the system did not make a suggestion, 13 were genuine errors (e.g. *paul waskiewiscz* with the correct spelling *paul waskiewicz*), 4 were cases in which the original input was correct, although different from the user's intent (e.g. *cooed* instead of *coed*) and 2 were gold standard errors (e.g. *commandos 3 walkthrough* had the wrong correction *commando 3 walkthrough*, as this query refers to a popular videogame called "commandos 3").

Differences	Gold std errors	Format	Diff. valid	Real Errors
80+19	8+2	15+0	17+4	40+13

The above table shows a synthesis of this error analysis on the second evaluation set. The first number in each column refers to a precision error (i.e. the speller suggested something different than the gold standard), while the second refers to a recall error (i.e. no suggestion).

As a result of this error analysis, we could arguably consider that while the agreement with the gold standard experiments are useful for measuring the relative importance of components, they do not give us an absolute measure of system usefulness/accuracy.

Agreement	Correctness	Precision	Recall
73.1	**85.5**	88.4	85.4

In the above table, we consider correctness as the relative number of times the suggestion made by the speller was correct or reasonable; precision measures the number of correct suggestions in the total number of spelling suggestions made by the system; recall is computed as the relative number of correct/reasonable suggestions made when such suggestions were needed.

As an additional verification and to confirm the difficulty of the test queries, we sent a set of them to Google and observed that Google speller's agreement with the gold standard was slightly lower than our system's agreement.

7 Conclusion

To our knowledge, this paper is the first to show a successful attempt of using the collective knowledge stored in search query logs for the spelling correction task. We presented a technique to mine this extremely informative but very noisy resource that actually exploits the errors made by people as a way to do effective query spelling correction. A direction that we plan to investigate is the adaptation of such a technique to the general purpose spelling correction, by using statistics from both query-logs and large office document collections.

Acknowledgements

We wish to thank Robert Ragno and Robert Rounthwaite for helpful comments and discussions.

References

Brill, E. and R. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the ACL 2000*, pages 286-293.

Cherkassky, V., N. Vassilas, G.L. Brodt, R.A. Wagner, and M.J. Fisher. 1974. The string to string correction problem. In *Journal of ACM*, 21(1):168-178.

Damerau, F.J. 1964. A technique for computer detection and correction of spelling errors. In *Communications of ACM*, 7(3):171-176.

Garside, R., G. Leech and G. Sampson. 1987. *Computational analysis of English: a corpus-based approach*, Longman.

Golding, A.R. 1995. A Bayesian hybrid method for context-sensitive spelling correction. In *Proceedings of the Workshop on Very Large Corpora*, pages 39-53.

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

Heidorn, G.E., K. Jensen, L.A. Miller, R.J. Byrd and M.S. Chodorow. 1982. The EPISTLE text-critiquing system. In *IBM Systems Journal*, 21(3):305-326.

Jurafsky, D. and J.H. Martin. 2000. *Speech and language processing*. Prentice-Hall.

Kernighan, M., K. Church, and W. Gale. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of COLING 1990*.

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

Mays, E., F.J. Damerau and R.L. Mercer. 1991. Context-based spelling correction. In *Information Processing and Management*, 27(5):517-522.

Mangu, L. and E. Brill. 1997. Automatic rule acquisition for spelling correction. In *Proceedings of the ICML 1997*, pages 734-741.

McIlroy, M.D. 1982. Development of a spelling list. In *J-IEEE-TRANS-COMM*, 30(1);91-99.

Peterson, J.L. 1980. *Computer programs for spelling correction: an experiment in program design*. Springer-Verlag.

Rieseman, E.M. and A.R. Hanson. 1974. A contextual post-processing system for error correction using binary n-grams. In IEEE *Transactions on Computers*, 23(5):480-493.

Toutanova, K. and R. C. Moore. 2002. Pronunciation Modeling for Improved Spelling Correction. In *Proceedings of the ACL 2002*.pages 141-151.

Wittgenstein, L. 1968. *Philosophical Investigations*. Basil Blackwell, Oxford.