Learning a Spelling Error Model from Search Query Logs

Farooq Ahmad
Department of Electrical and
Computer Engineering
University of Alberta
Edmonton, Canada
farooq@ualberta.ca

Grzegorz Kondrak
Department of Computing Science
University of Alberta
Edmonton, Canada
kondrak@cs.ualberta.ca

Abstract

Applying the noisy channel model to search query spelling correction requires an error model and a language model. Typically, the error model relies on a weighted string edit distance measure. The weights can be learned from pairs of misspelled words and their corrections. This paper investigates using the Expectation Maximization algorithm to learn edit distance weights directly from search query logs, without relying on a corpus of paired words.

1 Introduction

There are several sources of error in written language. Typing errors can be divided into two groups (Kucich, 1992): typographic errors and cognitive errors. Typographic errors are the result of mistyped keys and can be described in terms of keyboard key proximity. Cognitive errors on the other hand, are caused by a misunderstanding of the correct spelling of a word. They include phonetic errors, in which similar sounding letter sequences are substituted for the correct sequence; and homonym errors, in which a word is substituted for another word with the same pronunciation but a different meaning. Spelling errors can also be grouped into errors that result in another valid word, such as homonym errors, versus those errors that result in a non-word. Generally non-word errors are easier to detect and correct. In addition to its traditional use in word processing, spelling correction also has applications in optical character recognition and handwriting recognition. Spelling errors in this context are caused by inaccurate character recognition.

Spelling correction is a well developed research problem in the field of computational linguistics. The first dictionary based approach to spelling correction (Damerau, 1964) considers all words that can not be found in a dictionary as misspellings. The correct word is found by making a single edit operation (insertion, deletion, or substitution) on the misspelled word and re-checking the dictionary for the inclusion of the altered version. This method works well for correcting most typos, but often misspelled words are off by more than one character. A method of quantifying string-to-string distance is introduced in (Wagner and Fischer, 1974), allowing the consideration of multiple edit operations when determining candidate corrections. Each edit operation is assigned a fixed cost. Edit operations, though, can be more accurately modelled by considering every possible insertion, deletion, and substitution operation individually instead of having a fixed cost for each operation. For example, the application of probabilistic models to spelling correction is explored in (Kernighan, Church, and Gale, 1990), in which a confusion matrix describes the probability of each letter being substituted for another. The Bayesian noisy channel model is used to determine the the error probabilities, with the simplifying assumption that each word has at most one spelling error. In (Ristad and Yianilos, 1997), a probabilistic model of edit distance is learned from pairs of misspelled words and their corrections. This extends Kernighan's approach by allowing multiple edit operations rather than assuming a single edit. The probability of edit operations is learned from a corpus of pairs of misspelled words and corrections.

Search query correction is an interesting branch of spelling correction. Due to the wide variety of search queries, dictionary based spelling correction is not adequate for correcting search terms. The concept of using query logs to aid in spelling correction is explored in (Brill and Cucerzan, 2004). It is noted that using traditional Levenshtein distance as an error model can lead to inappropriate corrections, so a weighted distance measure is used instead.

This paper focuses on deriving a language model and probabilistic error model directly from search query logs without requiring a corpus of misspelled words paired with their corrections. The task of search query spelling correction is analyzed, and an implementation of the Expectation Maximization (EM) algorithm to learn an error model is described, with reference to similar approaches. In Section 2, the make-up of search queries is analyzed in the context of spelling correction. Section 3 details the noisy channel model spelling correction framework and describes how the EM algorithm is applied to learn an error model. The learned error model is explored in Section 4. The derived model is tested in Section 5 by comparing its performance in the single word spelling correction task to popular spell checking applications. Finally, conclusions and directions for future work are presented in Section 6.

2 Analysis of Search Queries

Search queries present a difficult challenge for traditional spelling correction algorithms. As mentioned above, dictionary-based approaches cannot be used since many search terms include words and names that are not well established in the language. Furthermore, search queries typically consist of a few key words rather than grammatically correct sentences, making grammar-based approaches inappropriate. In addition, spelling errors are more common in search queries than in regular written text, as approximately 10-15 % of search queries contain a misspelling (Brill and Cucerzan, 2004). The suitability of query logs as a corpus for spelling correction is investigated in this section.

The metaspy website[1] displays search queries submitted to the popular metacrawler search engine in real time. Over a period of five days in the last week of March 2005, 580,000 queries were extracted from the site. Several interesting observations can be made from the analysis of the search queries.

[1] www.metaspy.com

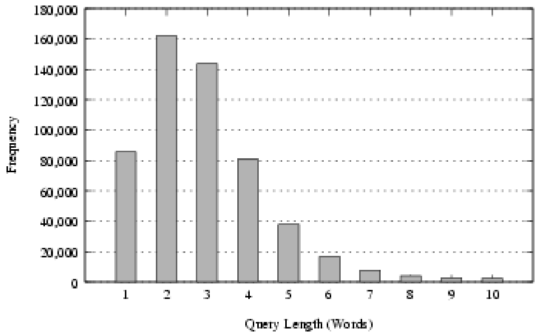

Figure 1: Query Length Frequency Histogram

2.1 Query Length

On average, each query consisted of approximately 3 words. Figure 1 shows the distribution of query lengths.

As illustrated in Figure 1, over 80% of queries include more than one search term. Thus word n-gram probabilities provide useful statistical knowledge that can be exploited to improve spelling correction. Although word cooccurrences are not used for spelling correction in this paper, the possibilities for n-gram analysis are explored in Section 3.2. The longer queries ($>$5 terms) often contain quotations, song lyric excerpts or very specific product names.

The frequency of words in written text has been shown to follow Zipf's law. That is, if the words are ordered in terms of frequency, the relationship between frequency and rank can be approximated with the following equation.

$$F \approx \frac{C}{r^m} \tag{1}$$

where F is the frequency, r is rank, C is a constant, and m is an exponent close to 1. In logarithmic form,

$$\log(F) = \log(C) - m * \log(r) \tag{2}$$

The frequency and rank of search query tokens approximately follow the same distribution, with some deviation at the high and low ends. Figure 2 shows the frequency distribution for dictionary and

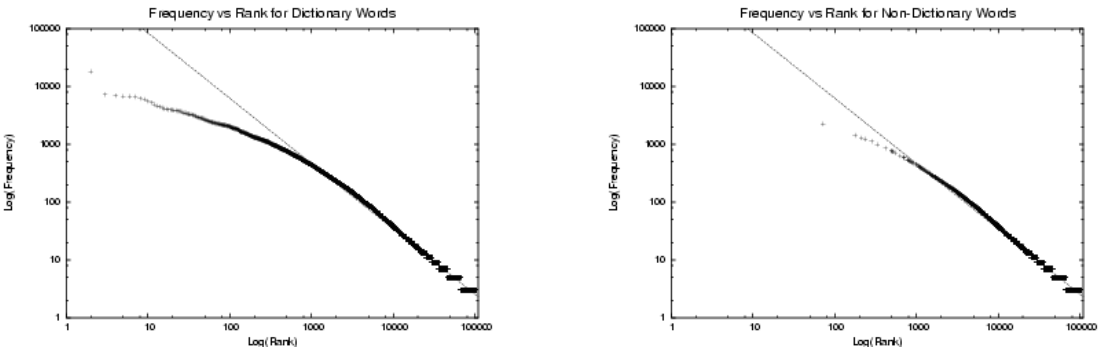

Figure 2: Token Frequency vs. Rank for Dictionary and Non-Dictionary Words

non-dictionary search query tokens. The word list available on most Unix systems */usr/dict/words* is a comprehensive list that contains 96,274 words, including names, plurals, verbs in several tenses, and colloquialisms. Following tokenization of the query logs, the tokens were divided into dictionary and non-dictionary words. The frequency-rank relationship is similar for both types of words, except that nearly all of the 100 most frequent query tokens are dictionary words. The exponent m, the (negative) slope of the linear best fit lines shown in Figure 2, was determined to be 1.11 for dictionary words, and 1.14 for non-dictionary words. As in (Baeza-Yates, 2005), the exponent is slightly higher than 1, partly due to the less frequent use of function words such as *the* in search queries relative to formal writing.

Although the majority of search tokens can be found in a standard dictionary, a large proportion of the less common queries are not dictionary words. In fact, 73% of unique word types were not found in the dictionary. Taking token frequency into consideration, these non-dictionary tokens account for approximately 20% of query search words, including correctly and incorrectly spelled words. However, the majority of the non-dictionary tokens are correctly spelled words, illustrating the unsuitability of traditional dictionary based spelling correction for search query correction.

What are these non-dictionary words? An analysis of the top two hundred non-dictionary words in the query logs allows categorization into a few main groups. The percentage of non-dictionary words belonging to each category, and some examples from each category are shown in Table 1. The first category, *e-speak*, includes words and abbreviations that are commonly used online, but have not crossed over into common language. This category includes words such as *pics, multiplayer*, and *clipart*. The second category is closely related to the first, and includes company and product names, such as *google, xbox*, and *hotmail*. Many of these terms refer to online entities or computer games. Incorrectly spelled words are another main class of non-dictionary tokens. Among the top 20 non-dictionary tokens are words with missing punctuation, such as *womens* and *childrens*, or with missing spaces, such as *realestate*. Names of people and locations are also common search queries, as well as medical terminology. Finally, foreign words make up another class of words that are not found in an (English) dictionary. The 20 highest frequency non-dictionary tokens from the extracted query logs are *pics, html, multiplayer, googletestad, google, xbox, childrens, ebay, angeles, hotmail, womens, ERBS, clipart, playstation, ratzinger, Ilios, lettre, realestate, tech* and *mallorca*.

	Word Class	**Percent**	**Examples**
1	E-speak & new words	45%	pics, html, multiplayer, clipart, mpeg, midi
2	Companies & Products	18%	google, xbox, ebay, hotmail , playstation
3	Proper Names	16%	(los) angeles, ratzinger, ilios, mallorca
4	Medical terms	5%	ERBS, mesothelioma, neuropsychological, alzheimers
5	Misspellings	9%	womens, realestate
6	Foreign Words	6%	lettre, para

Table 1: Classes of Non-Dictionary Words

3 Spelling Correction for Search Queries

The spelling correction problem can be considered in terms of the noisy channel model, which considers the misspelled word v to be a corrupted version of the correctly spelled word w.

$$P(w|v) = \frac{P(v|w)P(w)}{P(v)} \quad (3)$$

Finding the best candidate correction W involves maximizing the above probability.

$$W = argmax_w P(v|w)P(w) \quad (4)$$

The denominator $P(v)$ in Equation 3 is the same for all w and can be eliminated from the calculation. $P(v|w)$ models the errors that corrupt string w into string v, and $P(w)$ is the language model, or prior probability, of word w.

3.1 Error Model

Given two strings v and w, $P(v|w)$ is the probability that v is transmitted given that the desired word is w. One method of describing the noise model is to consider $P(v|w)$ to be proportional to the number of edit operations required to transform w into v. This gives

$$P(v|w) \propto ED(v, w) \quad (5)$$

where $ED(v, w)$ is the edit distance between v and w.

The traditional edit distance calculation assigns a fixed cost for each insertion, deletion, and substitution operation. For example, each insertion and deletion may be assigned a cost of 1, while substitutions are assigned a cost of 1.5. The edit distance calculation can be accomplished by dynamic programming.

The error model can be improved if each edit operation is considered separately, rather than assigning a fixed cost to each operation. For example, the substitution of the letter i for the letter e may be much more likely than k for e. Thus if a string S_1 differs from string S_2 by one $e \rightarrow i$ substitution, it should be considered more similar to S_2 than a string S_3 that differs from S_1 by an $e \rightarrow k$ substitution.

Generating an accurate error model that considers each edit operation individually requires learning edit distance weights. As described in (Ristad and Yianilos, 1997), character-to-character edit distance costs $ED(e)$ can be related to edit probability $P(e)$ by means of the equation:

$$ED(e) = -\log[P(e)] \quad (6)$$

where e is an edit operation consisting of a substitution of one alphanumeric character for another ($c_1 \rightarrow c_2$), an insertion ($\_ \rightarrow c_1$), or a deletion ($c_1 \rightarrow \_$).

Thus higher probability edits will have lower edit distances, and the string to string edit distance calculation proceeds in the same way as the traditional calculation. This convenient representation allows whole string-to-string edit probability to be expressed in terms of the edit distance of the edit sequence $[e_1 ... e_n]$:

$$\begin{aligned} P(w|v) &= \Pi P(e_i) \\ &= P(e_1) * P(e_2) * ... * P(e_n) \end{aligned} \quad (7)$$

Taking the log of both sides gives

$$\begin{aligned} \log[P(w|v)] = &\log[P(e_1)] + \log[P(e_2)] \\ &+ ... + \log[P(e_n)] \end{aligned} \quad (8)$$

Finally, by combining 6 and 8 we can relate the probability of misspelling a string w as v to string-to-string edit distance.

$$\log[P(w|v)] = -ED(w, v) \quad (9)$$

The edit probabilities can be estimated using the expectation maximization (EM) algorithm as described in Section 3.3.

3.2 Language Model

Along with the error model, a language model is used to determine the most likely correction for every input query. Often, spelling correction programs use N-gram language models that use nearby words to help determine the most probable correction. For example, it is noted in (Brill and Cucerzan, 2004) that employing a trigram language model can substantially improve performance relative to a unigram model. However, if search query logs are not very large, bigram or trigram data may be too sparse to be helpful. Nevertheless, a word unigram model can be used for training the error model. The unigram model is determined by tokenizing the query logs and determining the frequency of each token. The language model $P(w)$ is the frequency of the word $C(w)$ divided by the total number of tokens N in the query log:

$$P(w) = \frac{C(w)}{N} \tag{10}$$

Add-One smoothing is used to account for words not present in query logs.

3.3 Determining Edit Probabilities with Expectation Maximization

The EM algorithm is used to determine the parameters of the probability distribution for a given a set of data. It can be considered to be a soft-clustering algorithm: given several data points, the task is to find the cluster parameters which best represent the data. The EM algorithm is applied iteratively to each data point in a two-step process; the expectation step determines the degree to which data agrees with each cluster/hypothesis, and the maximization step updates the parameters to reflect the inclusion of the new data.

Prior to running the EM algorithm, the edit distance table is seeded with initial values. The initialization stage assigns high probability (low edit distance) to characters being typed correctly, and a lower probability for character substitutions. For each character l, substitution distance is equally distributed over all other characters and the deletion operation ($l \rightarrow \_$). Specifically the initial probability for a character match was set to 90%, and the remaining 10% was equally distributed over the other 26 possible substitutions. Essentially, the first edit distance calculated in the EM algorithm will be equivalent to the fixed-weight Levenshtein distance. After this preprocessing stage, the edit probability matrix is iteratively improved with the E-Step and M-Step described below. The operation of the EM algorithm is illustrated in Figure 3.

For each query token, possible corrections are harvested from the query word list. The entire word list is searched, and any word within a threshold edit distance is considered as a candidate. Since the query logs can be quite large, determining the exact weighted edit distance between the input query and each logged query is quite computationally expensive. Instead, the candidate queries are first narrowed down using a fast approximate string matching algorithm (Wu and Manber, 1990) to determine all candidates within k unweighted edit operations. Then, the candidate queries that are within a second tighter threshold T, based on weighted edit distance, are kept.

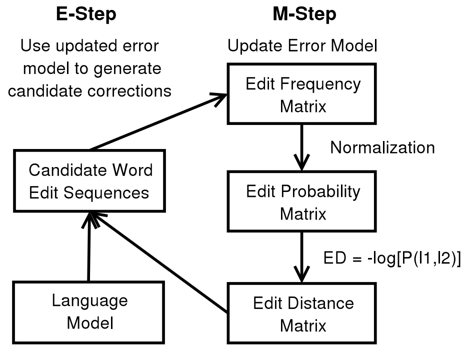

Figure 3: The EM process

$$Candidates(v) = \{w_i | ED(w_i, v) < T\}$$

Generally several words in the query logs will meet the above criteria. The threshold T is chosen to ensure the inclusion of all reasonable corrections, while maintaining a manageable computation time. If T were infinite, every query log token would need to be considered, taking too much time. On the other hand, if T is too small, some corrections may not be considered. In practice, K was set to 3 unweighted edits, and T was set as a constant proportion of word length.

The expectation of each candidate correction is the probability that the word w_i was desired given that the query was v:

$$P(w_i|v) = \frac{P(v|w_i)P(w_i)}{P(v)} \tag{11}$$

where $P(v|w)$ and $P(w)$ are determined using the error and language models described in Equations (9) and (10).

If the value of T is set high enough, it can be assumed that the correct word w is within the set of candidates. So, the sum of probabilities over all candidates is normalized to $P(v)$ in accordance with Bayes Rule of Total Probability.

$$P(v) = \Sigma_j P(v|w_j)P(w_j) \tag{12}$$

Correction	Error Model	Language Model	Total Probability	Normalized
equipment	0.0014	0.00078	1.1e-6	0.77
equpment	0.64	5.0e-7	3.4e-7	0.23
equpiment	0.0005	5.0e-7	1.0e-9	0.0005

Table 2: Candidate Corrections for *equpment*

This gives us the following formula for the expectation value

$$P(w_i|v) = \frac{P(v|w_i)P(w_i)}{\Sigma_j P(v|w_j)P(w_j)} \tag{13}$$

The E-step is used to generate candidate corrections for input query tokens. For example, input query *"equpment"* returns the candidate corrections and their probabilities shown in Table 2.

Note that several incorrectly spelled words, including *"equpment"* itself, are given as candidate corrections. However, the language model derived from the query logs assigns a low probability to the incorrect candidates. In the case of a correctly spelled query, the most likely candidate correction is the word itself. However, occasionally there is a correctly spelled but infrequent word within a small edit distance of another more common word. In this case, the language model will bias the correction probability in favor of an incorrect edit. Nevertheless, overall these cases do not seem to cause a significant impact on the error model except in the case of plural nouns as discussed in Section 4.

The maximization step updates the edit distance probabilities and edit distance table to reflect the query considered in the E-Step. For each candidate correction, the required edits are added to the edit frequency table, weighted by the probability of the correction. Then, the probability of an edit for each character is normalized to 1 and the edit probabilities are stored in a table. Finally, Equation 6 is used to generate the edit probability table. For example, for the input query *"equpment"* in response to the first candidate correction ($equpment \rightarrow equipment$), the following substitution frequencies will each be incremented by 0.77: $e \rightarrow e, q \rightarrow q, u \rightarrow u, i \rightarrow \_, p \rightarrow p, m \rightarrow m, e \rightarrow e, n \rightarrow n, t \rightarrow t$. The ($i \rightarrow \_$) edit represents deletion of the letter i.

Letter	Subs	Letter	Subs
a	e_qo	n	_fkb
b	grnw	o	a_ei
c	_ksm	p	nfrm
d	ds_nk	q	glk_
e	ao_i	r	_sdm
f	btpj	s	_mdn
g	o_ks	t	_yir
h	_rab	u	_rio
i	_aue	v	awcm
j	blhm	w	prgk
k	vots	x	gtms
l	r_is	y	ioaje
m	nkvs	z	skmt

Table 3: Most Common Substitutions

4 The Learned Error Model

Approximately 580,000 queries were extracted from the metaspy site over a period of 5 days. After generating a language model by analyzing token frequencies, the EM algorithm was run on a subset of the queries to find the edit probability matrix.

After 15,000 iterations, several patterns can be observed in the edit distance table. The most common edit operations are shown in Table 3. As expected, vowels are most commonly substituted for other vowels. As can be seen in the table, vowel-to-vowel edits are more probable than vowel-to-consonant transitions. The letter e is most commonly mistyped as a, o, and i; the letter i is most often mistyped as a, u, and e. For the most part, vowel substitutions can be considered to be cognitive errors (except $o \rightarrow i$ may be a cognitive error or typographic error). The effect of keyboard key proximity is also evident; b is often typed as g; d as s; m as n; and so on. Other errors seem to be a result of phonetic similarity; c is misspelled as k and s; q as g and k; and v as w. In general, the edit probabilities roughly match those derived using a corpus of word pairs in (Kernighan, Church, and Gale, 1990).

The insertion probabilities for each letter are shown in Figure 4. Equation 6 is used to convert the edit distances to probabilities. Words in the plural form cause problems for the algorithm, as is illustrated by the high probability of s insertion in Fig-

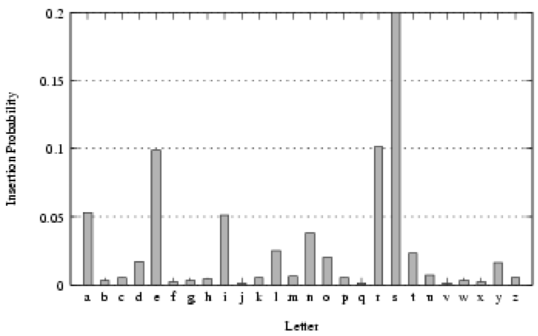

Figure 4: Letter Insertion Probabilities

Spell Checker	ISPELL 3.1.20	ASPELL 0.29	EMBED	Google
Total Tokens	508	508	508	508
Total Found	272 (53.5%)	480 (94.5%)	402 (79.1%)	-
Top 1 (%)	197 (38.8%)	302 (59.5%)	211 (41.5%)	291 (57%)
Top 5 (%)	260 (51.2%)	435 (85.6%)	331 (65.2%)	-
Top 25 (%)	272 (53.5%)	478 (94.1%)	386 (76.0%)	-

Table 4: Spelling Correction Accuracy

ure 4. That is because high frequency query words often appear in both singular and plural form. Every time the singular form is encountered, the plural is considered as a viable candidate correction, and the s insertion probability is increased. Complementarily, every time the plural form is seen, the singular form is considered, increasing the s deletion probability. Indeed, as can be seen in Table 3, deletion is the highest probability operation for the letter s.

5 Testing

To test the accuracy of the error model, the well-known Unix based spelling correction programs *Ispell*[2] and *Aspell*[3] spell checking programs were used for comparison. Ispell generates candidate corrections by considering all words within 1 edit of the misspelled word. *Aspell* uses the *metaphone* algorithm (Philips, 1990), which divides English words into phonetic segments, and generates alternate spellings by substituting similar sounding phones. The test data set[4] consists of 547 misspelled words paired with their best correction as determined by a human expert.Compound words were removed from the test set, leaving 508 misspellings. Several of the misspellings differ from the correction by multiple edit operations. Only the error model learned by the EM algorithm on the search engine queries was used; instead of using the probabilistic language model derived from the query logs and used for training, the word list in */usr/dict/words* was used, with equal probability assigned to each word.

[2]International Ispell Version 3.1.20. http://www.lasr.cs.ucla.edu/geoff/ispell.html

[3]Kevin Atkinson. Aspell Version 0.29. http://aspell.sourceforge.net/

[4]Kevin Atkinson. http://aspell.net/test/

Since the test data is composed of single words of varying prevalence, a language model does not significantly aid correction. In practice, the language model would improve performance.

Table 4 compares the performance of the *Aspell* and *Ispell* spell checkers with the Expectation Maximization Based Edit Distance (EMBED) spelling correction system described in this paper. The percentages refer to the percentage of instances in which the correct correction was within the top N suggestions given by the algorithm. If only the top recommended correction is considered, EMBED fares better than *Ispell*, but worse than *Aspell*. For the top 5 and 25 corrections, the rankings of the algorithms are the same.

As Table 4 shows, in several cases the EMBED algorithm did not find the correction within the top 25 suggestions. Typically, the misspellings that could not be found had large edit distances from their corrections. For example, suggestions for the misspelling ”extions” included ”actions” and ”motions” but not the desired correction ”extensions”. In general, by using a phonetic model to compress English words, *Aspell* can find misspellings that have larger edit distances from their correction. However, it relies on a language specific pronunciation model that is manually derived. EM based spelling correction, on the other hand, can be learned from a unlabeled corpus and can be applied to other languages without modification. Although the test data set was comprised of misspelled dictionary words for the purposes of comparison, the spelling correction system described here can handle a continuously evolving vocabulary. Also, the approach described here can be used to train more general error models.

Comparison to online spelling suggestion systems such as provided by the *Google* search engine is difficult since search results are returned for nearly every query on account of the large lexicon. Consequently, many suggestions provided by *Google* are reasonable, but do not correspond to the golden standard in the test data. For example, ”cimplicity” and ”hallo” are not considered misspellings since several online companies and products contain these terms, and ”verison” is corrected to ”verizon” rather than ”version.” While *Google* returns 291 corrections in agreement with the data set (57%), another 44 were judged to be acceptable corrections, giving an accuracy of 66%. In addition, several of the apparently misspelled test strings are new words, proper names, or commonly accepted alternate spellings that are common on the web, so no suggestions were given. Taking these words into account would further improve the accuracy rating.

6 Conclusions and Future Work

The EM algorithm is able to learn an accurate error model without relying on a corpus of paired strings. The edit probabilities determined using the EM algorithm are similar to error models previously generated using other approaches. In addition, the generated error model can be used to find the correct spelling of misspelled words as described in Section 5. However, there are several improvements that can be made to improve spelling error correction. One step is increasing the size of the corpus. While the corpus included nearly 580,000 queries, several thousand of those queries were correctly spelled words without any misspelled versions in the corpus, or misspelled words without the correctly spelled version available. This results in the misidentification of candidate spelling corrections. Another improvement that can improve candidate correction identification is the use of better language models, as discussed in Section 3.2. Since a large proportion of queries contain more than one word, word n-gram statistics can be used to provide context sensitive spelling correction. Finally, a large proportion of typos involve letter transpositions, and other operations that can not be captured by a single-letter substitution model. In (Brill and Moore, 2000), a more general model allowing generic string to string edits is used, allowing many-to-one and one-to-many character substitution edits. Pronunciation modeling in (Toutanova and Moore, 2002) further improves spelling correction performance.

Acknowledgments

Support for this work was provided by the Natural Sciences and Engineering Research Council of Canada.

References

Baeza-Yates, R. 2005. Web Usage Mining in Search Engines. Chapter 14 in *Web Mining: Applications and Techniques*. Ed. Anthony Scime. New York: Idea Group Publishing, 2005. 307-321.

Brill, E. and Cucerzan, S. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. *Proceedings of EMNLP 04*. 293-300.

Brill, E. and Moore, R. 2000. An improved error model for noisy channel spelling correction. *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics.* 286 - 293.

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

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

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

Philips, L. 1990. Hanging on the metaphone. *Computer Language Magazine.* 7(12):39.

Ristad, E. and Yianilos, P. 1997. Learning string edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 20(5):522-532.

Toutanova, K. and Moore, R. 2002. Pronunciation modeling for improved spelling correction. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics.* 144-151.

Wagner, R. and Fischer, M. January 1974. The string-to-string correction problem. *Journal of the ACM*. 21(1):168-173.

Wu, S. and Manber, U. 1992. Fast text searching allowing errors. *Communications of the ACM*. 35(10):83-91