Spelling Correction Based on User Search Contextual Analysis and Domain Knowledge

Xing Yi, Henry Feild, and James Allan
Center for Intelligent Information Retrieval
University of Massachusetts Amherst
Amherst, MA 01003
{yixing, hfeild, allan}@cs.umass.edu

ABSTRACT

We propose a spelling correction algorithm that combines trusted domain knowledge and query log information for query spelling correction. This algorithm uses query reformulations in the query log and bigram language models built from queries for efficiently and effectively generating correction suggestions and ranking them to find valid corrections. Experimental results show that for both simple unknown word errors and complex word substitution errors, valid corrections mostly appear within the top two ranks.

1. INTRODUCTION

Although developing general purpose spelling correction technology has been well studied for decades, researchers have recently started to investigate using collective knowledge stored in web query logs for the challenging web-query spelling correction task where a high-coverage lexicon is extremely difficult to maintain and many *word substitution errors* like *gulf war* → *golf war* exist [3, 4]. For this challenge, Cucerzan and Brill [3] proposed an effective approach that first iteratively transforms an input query string with spelling errors into plausible corrections, then uses language models built from web query logs to select the most likely correct one. This approach inefficiently generates plausible corrections by directly using string-edit distance. Jones *et al.*[4] proposed an approach of using query reformulations in the same user search session for generating highly relevant query substitutions, which include spelling variants. This approach can efficiently obtain user suggested spelling corrections, but has limited spelling error coverage and does not take advantage of either reliable linguistic source like trusted English lexicons or language models built from query logs.

Here we design a system to combine the advantages of both approaches for web query spelling correction. Our approach is especially useful for specialized web search engines, whose query logs are much smaller than general purpose search engines and are not reliable for building language models for spelling correction. Experimental results show the effectiveness of our approach.

2. OVERVIEW OF THE ALGORITHM

Let Σ be the alphabet of a language and $L_1 \subset \Sigma^\star$ be a broad-coverage lexicon of the language. Let $q_0 = w_0^1 w_0^2 ... w_0^{l_0} \in \Sigma^\star$ be the initial query where l_0 denotes q_0's length and w_0^i denotes the i^{th} word in q_0. Here we ignore the spaces and other word delimiters in the formulations for simplicity. We consider both *unknown word error*, where q_0 contains some unknown word $w_0^i \in \{\Sigma^\star \backslash L_1\}$, and *word substitution error*, where $q_0 \in L_1^\star$ can be reformulated to another much more plausible query, q_1, that has a close string-edit distance to q_0 by substituting one or more words.

We use query reformulations in web query logs for spelling correction. A *query reformulation* $[q_1, q_2]$, also referred to as a *query pair*, is a pair of successive queries in a search session such that q_2 is entered immediately after q_1. We collapse repeated searches for the same terms and only keep query pairs that have no more than one string-edit distance per five letters [3], under the assumption that these pairs are more likely to contain spelling corrections.

Directly matching a new input query to q_i in $[q_i, q_j]$ to find possible spelling errors will have low coverage of errors (pairs extracted from low traffic specialized search engines' query logs are even more sparse). Therefore, we follow Cucerzan and Brill's approach [3] which considers words and word bigrams corrections for each query pair, i.e. we extract appropriate uni/bi-gram correction pairs $[g_i, g_j]$ from each query pair. We aggregate the same $[g_i, g_j]$ in the training data to get the frequency $f_{[g_i,g_j]}$ and further select them for our task according to the following three rules: 1) if g_i is unigram and $g_i \in L_1$, discard $[g_i, g_j]$; 2) if g_j contains some unknown word $w \in \{\Sigma^\star \backslash L_1\}$, discard $[g_i, g_j]$; 3) if both $[g_i, g_j]$ and $[g_j, g_i]$ exist, discard $[g_i, g_j]$ when

$$(f_{[g_i,g_j]} < f_{[g_j,g_i]}) \wedge (f_{[g_i,g_j]} \leq t_1 \vee (f_{[g_i,g_j]}/f_{[g_j,g_i]} < t_2)), \quad (1)$$

where t_1 is a threshold for filtering noisy reformulations ($t_1 = 1$ in experiments) and t_2 is a tuned threshold for filtering relatively infrequent reformulations that may be wrong corrections. We keep the remaining n-gram pairs $[g_i, g_j]$ and $f_{[g_i,g_j]}$ in a reformulation dictionary R_1 for later use.

For efficiently correcting common spelling errors, we can generate a reliable low-coverage static reformulation dictionary R_2. Common errors like *sun burn* → *sunburn* can be put into R_2. We also extract all the bigrams $w^t w^{t+1}$ from the training queries to get a bigram dictionary L_2. If a candidate bigram spelling correction is not in this dictionary (i.e., it does not occur in the training corpus), the bigram is removed from the candidate list. Then we combine L_1, L_2, R_1 and R_2 for detecting spelling errors and generating correction suggestions. The algorithm is described in Figure 1. Note that in steps (2.a) and (4.a), we consider one and two-step reformulations as follows: if $[w_1^i, g_1^i]$ and $[g_1^i, g_2^i]$ are in R_1, both g_1^i and g_2^i are valid reformulations of w_1^i.

For each input query q_0, we generate correction suggestions by using steps 1–4, then use three different ranking

SIGIR'09 July 19–23, Boston, MA, USA
ACM X-XXXXX-XX-X/XX/XX.

Given an input query q_0:
1. Use R_2 to correct obvious spelling errors and output q_1.
2. For each word w_1^i in q_1:
   a) Use R_1 to find one and two-step reformulations of w_1^i
   b) If w_1^i is an unknown word, find other words with close string-edit distances to w_1^i in L_1 and use w_1^{i-1},w_1^{i+1} and L_2 for selecting good reformulations
3. Generate all reformulations $\{q_{2,k}\}$ of q_1 by considering all combinations of each word's reformulation.
4. For each query $q_{2,k}$:
   a) For each bigram $w_2^i w_2^{i+1}$ in $q_{2,k}$:
   – Use R_1 to find one and two-step reformulations of $w_2^i w_2^{i+1}$
   b) Generate all reformulations $\{q_{3,k}\}$ of $q_{2,k}$ by considering all combinations of bigrams' reformulations.
5. Collect and rank all the correction suggestions $\{q_{3,m}\}$.

Figure 1: Algorithm for Generating Spelling Correction Suggestions

methods to rank the correction suggestions in step 5. The first method, called ***ReformF***, uses n-gram reformulation frequency $f_{[g_i,g_j]}$ in R_1 to calculate a ranking score. The basic procedure is to weight reformulations by $f_{[g_i,g_j]}$ in step(2.a) and (4.a) in Figure 1; then weights are multiplied when creating each correction suggestion combination in step(3) and (4.b); a reformulation v^i for an unknown word w^i in step(2.b) is given a weight $e^{-dist(v^i,w^i)}$. This method assumes more frequent reformulations with smaller edit distances are more likely to be valid respellings.

The second method, called ***PriorP***, uses the prior probability $P(q_{3,m})$ of each suggestion $q_{3,m}$ as the ranking score [3]. $P(q_{3,m})$ can be calculated by the bigram language model [2] built from the training queries. Intuitively, this method assumes that suggestions that are more probable queries are also more likely valid respellings.

The third method, called ***PostP***, uses the posterior correction probability $P(q_{3,m}|q_0) \sim P_c(q_0|q_{3,m}) \times P(q_{3,m})$ as the ranking score [3]. We use string-to-string edit probability [1] $P_c(q_1|q_2) = e^{-dist(q_1,q_2)}$ to calculate $P_c(q_0|q_{3,m})$. This method assumes that more probable queries with a smaller edit distance are more likely to be valid respellings.

3. EXPERIMENTS AND ANALYSIS

We utilize a web query log sample from an industrial medical search engine for this study. This query log sample contains more than 50 million queries over a period of ten months starting 11/2007. We use the first half (up through 03/2008) for training and the remaining half for testing. Our primary lexicon, L_1, contains a standard English dictionary from Aspell[1] and a specialized medical dictionary. The second lexicon, L_2, contains about 3M bigrams. A small list of stopwords were removed from the training queries prior to extracting these bigrams in an effort to make better use of context [3]. The first reformulation lookup, R_1, contains 916,530 n-gram reformulation pairs and is extracted from training query reformulations as described in §2. The second reformulation lookup, R_2, contains 4,840 pre-defined common error corrections. We then use the algorithm in Figure 1 for spelling correction.

Our algorithm outputs a ranked list of correction suggestions for each input q_i. We then compute the mean reciprocal rank (**MRR**): $\sum_{q_i} 1/r_{q_i}(q_j)$, where $r_{q_i}(q_j)$ denotes the rank position of the highest ranked ground truth correction (q_j) for the input query q_i.

[1]http://aspell.net/

	ReformF	PriorP	PostP
Aggregate	0.673	0.667	$0.724^{\star\dagger}$
S-Subset	0.709	0.689	$0.750^{\star\dagger}$
U-Subset	0.637	0.645	$0.699^{\star\dagger}$

Table 1: MRRs of three ranking methods by using annotations over misspelled queries. † and ⋆ denote a statistically significant improvement over $PriorP$ and $ReformF$, respectively, with a *p-value* $< .05$ using Fisher's Randomization Test.

To train, we select 500 pairs from the training set with a low edit distance. We use ReformF as the ranking method and tune $t_2 = 1/8$ in equation 1 to have the highest MRR.

For the experiments, we extracted 1000 testing query pairs satisfying the string-edit constraint from the test data and kept distinct pairs. 500 of these are pairs such that q_2 was a respelling of q_1 suggested by the medical search system, and was then selected by the user. We refer to this subset of the query pairs as the ***S-Subset*** and the remaining 500, which are not system-suggested reformulations, as the ***U-Subset***.

We designed the evaluation experiments to investigate our algorithm's performance by asking editors to examine the correction suggestions produced in top 20 ranks by our ranking methods for each query in the two samples. Annotators were asked to indicate if each input query q_i was misspelled, and if so, to indicate which of the suggestions were valid respellings. They were also allowed to manually enter a valid respelling. Eight annotators—seven graduate students and one medical search specialist—judged suggestions from at least 150 input queries each. Annotators were allowed to any Web search engines of their choice to assist their tasks.

Of the 1000 queries, 639 were judged by two annotators. The inter-annotator agreement for whether a query was misspelled is $\kappa = 0.635$. The agreement for the suggested respellings is $\kappa = 0.703$. For 412 (82%) of the query pairs in the *S-Subset*, the query suggested by the search system was marked by an annotator as a valid respelling.

Using the output of our algorithm for the queries in each sample, we calculate the new MRRs of three ranking methods by using the judgments and show the results in Table 1. Recall that if an annotator marked multiple valid respellings for a particular query, the highest ranking respelling is used to compute the reciprocal rank. To combine the annotations for queries judged by two annotators, we randomly chose one set of judgments. The results show that the *PostP* outperforms the two other ranking methods in all cases. *ReformF* performs poorly because the test data contains many spelling mistakes not present in the training data, and thus, there are no direct reformulations of those mistakes in the training set. The performance of *PriorP* is hindered by the exclusion of edit-distance information.

4. ACKNOWLEDGEMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by UpToDate. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor. We wish to thank our annotators, in particular Jerry Greene of UpToDate.

5. REFERENCES

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

[2] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of ACL*, pages 310–318, 1996.

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

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