CloudSpeller: Spelling Correction for Search Queries by Using a Unified Hidden Markov Model with Web-scale Resources

Yanen Li, Huizhong Duan, ChengXiang Zhai

Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801
{yanenli2, duan9, czhai}@illinois.edu

ABSTRACT

Query spelling correction is a crucial component of moden search engines that can help users to express an information need more accurately and thus improve search quality. In participation of the Microsoft Speller Challenge, we proposed and implemented an efficient end-to-end speller correction system, namely CloudSpeller. The CloudSpeller system uses a Hidden Markov model to effectively model major types of spelling errors in a unified framework, in which we integrate a large-scale lexicon constructed using Wikipedia, an error model trained from high confidence correction pairs, and the Microsoft Web N-gram service. Our system achieves excellent performance on two search query spelling correction datasets, reaching 0.970 and 0.940 F1 scores on the TREC dataset and the MSN dataset respectively.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query Alteration*

General Terms

Algorithms, Performance, Experimentation

Keywords

CloudSpeller, Spelling Correction, Hidden Markov Model

1. INTRODUCTION

The Text Information Management group at the University of Illinois at Urbana-Champaign has participated in the Microsoft Speller Challenge. The task of the challenge is to develop a speller that generates most plausible spelling alternatives for a search query. This paper is a report of our proposed methods, experiments and findings about the problem based on the results and analysis on two spelling datasets.

SIGIR'11, July 24–28, 2011, Beijing, China.

Spelling correction has a long history [10]. Traditional spellers focused on dealing with non-word errors caused by misspelling a known word as an invalid word form. They typically rely on manually created lexicons and simple distance measures like Levenshtein distance. Later, statistical models were introduced for spelling correction, in which the error model and n-gram language model are identified as two critical components [3]. Whitelaw *et al.* alleviated the cost of building error model by leveraging the Web to automatically discover the misspelled/corrected word pairs [15].

With the advent of the Web, the research on spelling correction has received much more attention, particularly on the correction of search engine queries. Compared with traditional spelling tasks, it is more difficult as more types of misspelling exist on the Web. Research in this direction includes utilizing large web copora and query log [4, 5, 2], employing large-scale n-gram models [Brants et al. 2007], training phrase-based error model from clickthrough data [13] and developing additional features [7].

To address the challenges of spelling correction for search queries, we propose several novel ideas which are implemented in an efficient end-to-end speller system (called CloudSpeller) of high precision and recall. First, we construct a large and reliable lexicon from Wikipedia. The large size (over 1.2 million words) of the lexicon overcomes the limitation of using a small traditional lexicon, such as identifying human names and neologisms. A clean dictionary is also critical for handling non-word spelling errors, which is the most frequent error type in search queries. Second, we propose a Hidden Markov Model to model all major types of spelling errors into a unified framework. A Top-K paths searching algorithm is designed to efficiently maintain a small number of highly confident correction candidates. Third, the candidate paths are finally re-ranked by a ranker which takes into account two major types of features, one from the error model, the other from the n-gram language model. We train the error model with a set of query correction pairs from the web. Web scale language model is obtained by leveraging the Microsoft Web N-gram service. We demonstrate that these two types of features are sufficient for building a highly accurate speller for web queries. Such a system also has the advantage in efficiency compared to systems employing tens of features [7].

2. PROBLEM SETUP AND CHALLENGES

Given a search query, the task of search query spelling

correction is to find the most effective spelling variant of the query that would retrieve better results than the original query. Formally, let Σ be the alphabet of a language and $L \subset \Sigma^*$ be a large lexicon of the language. We define a general search query correction problem as:

Given query $q \in \Sigma^*$, find top-K $q' \in L$ such that $P(q'|q) \in maxK_{t \in L^*} P(t|q)$,

where $P(q'|q)$ is the conditional probability of q' given q that follows the general noisy channel framework.

The problem of search query spelling correction is significantly harder than the traditional spelling correction. Previous researches show that approximately 10-15% of search queries contain spelling errors [5]. We have identified four major types of errors in search queries. (1) non-word substitution, e.g. insertion, deletion, misspelling of characters. This type of error is most frequent in web queries, and it is not uncommon that up to 3 or 4 letters are misspelled. This type of error can be corrected accurately using the noisy channel model with a trusted lexicon. (2) confusable valid word substitution, e.g. "persian golf" → "persian gulf". This type of error can be effectively addressed by the context sensitive spellers[9]. (3) cancatenation of multiple words, e.g. "unitedstatesofamerica" → "united states of america". This problem can be tackled by n-gram model based word breaker [14] (4) splitting a word into parts, e.g. "power point slides" → "powerpoint slides". For each type of errors in search query spelling correction, there are effective solutions. However, predicting the error type given a query is difficult, and it is quite common that more than one type of errors cooccur in a query. Therefore, a successful speller for web queries requires a solution addressing all types of spelling errors with high accuracy.

3. THE CLOUDSPELLER ARCHITECTURE

The CloudSpeller system accepts a search query as input. Then a unified HMM model generates a small list of most likely candidate corrections (paths). After that, a ranker will score and re-rank those candidate paths, and finally generate the top-K corrections as the output. In this section we describe the unified HMM model and the ranker. We will also describe the critical components our HMM model and ranker rely on, namely the large-scale lexicon, the error model and n-gram model.

3.1 An HMM Model for Query Correction

We adopt a generative model for spelling correction. More specifically, we employ a Hidden Markov Model (HMM) for this purpose. The generative process follows a word-by-word process. At the beginning, the user has a word in its correct form in mind, it is then transformed through a noisy channel and becomes potentially misspelled. In this process, it is not only possible to misspell the given word into another word, but also sometimes possible to split the word into several words, or even combine the two types of misspellings. When the user has a second word in mind, he or she may have similar misspellings as the previous word, but may also incorrectly attach the word (or part of it) to the previous word. Note that this HMM is more general than the existing HMMs used for spelling correction [5] because it can model many different kinds of spelling errors.

Formally, let $\theta = \{A, B, \pi\}$ be the model parameters of the HMM, including the transition probability, emission probability and initial state probability. Given a list of query words (obtained by splitting empty spaces), the states in a state sequence are one-to-one corresponding to the query words except for the merging state. Each state is represented by a phrase in Σ^*. Theoretically the phrase in a state can be chosen arbitrarily, however for the sake of efficiency we reduce the state space by only choosing a phrase in the lexicon L^* such that $dist(s,t) \leq \delta$, where $dist(s,t)$ is the edit distance between the state phrase s and word t in the query. Each state also has the type information; indicating whether the state is a substitution, merging, splitting or NULL state. In order to accommodate a merging state we introduce the NULL state. The NULL state doesn't emit any phrase, and the state transition probability is always equal to 1. For the model parameter A, B, we employ the bigram model probability as the state transition probabilities A and use the error model probabilities as the emission probabilities B. After this treatment, the probability of generating the original query from a state sequence (path) is the product of all phrase-to-phrase error model probabilities and the total language modeling probability of the path.

Figure 1 illustrates our HMM model and a generative example. In this example, there are three potential errors with different error types, e.g. "goverment" → "government" (substitution), "home page" → "homepage" (splitting), "illinoisstate" → "illinois state" (concatenation). The state path showed in Figure 1 is one of the state sequences that can generate the query. Take state s_2 for example, s_2 is represented by phrase *homepage*. Since s_2 is a merging state, it emits a phrase *home page* with probability $P(home\ page|homepage)$ according to the error model. Subsequently s_2 transits to state s_3 with probability $P(s_3|s_2)$ according to the bigram language model.

With this model, we are able to come up with arbitrary corrections instead of limiting ourselves to an incomprehensive set of queries from query log. By simultaneously modeling the misspellings on word boundaries, we are able to correct the query in a more integrated manner. Moreover, to handle the large number of potential corrections, we have designed and implemented a dynamic programming algorithm to compute the Top-K corrections efficiently. If there are n states in a state path, and the maximum number of candidate words for each query term is M, the computational complexity of our algorithm is $O(M^2 \cdot n \cdot K)$. Experiments results show that the recall of Top-40 corrections obtained by this algorithm is about 98.4% in the TREC dataset and 96.9% in the MSN dataset.

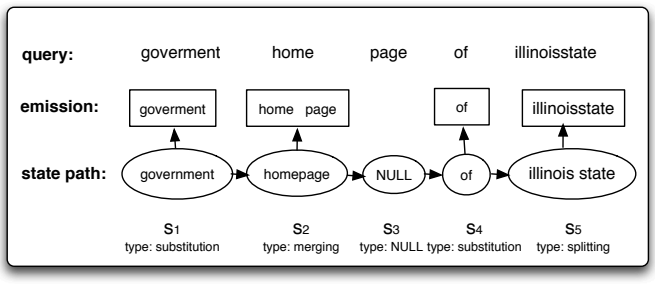

Figure 1: HMM model for query spelling correction

3.2 Candidate Paths Re-ranking

After generating a small list of candidate paths, we propose a ranker to re-rank these paths based on the weighted sum of error model probability and n-gram model probability. We find that probabilities from these two components are most important factors for web query correction. However, a successful speller requires a careful combination of these probabilities. Specifically, for a candidate path q' with n state nodes. We score q' based on the following 7-parameter interpolated model:

$$S_p = \left[\sum_{n=1}^{6} w_n \cdot logP(q_n|q'_n)\right] + w_7 \cdot log(P'(q')) \quad (1)$$

where $P(q_n|q'_n)$ is the error model probability of transforming the phrase q_n from q'_n, $P'(q')$ is the n-gram probability of the phrase in path q'. And $w_n \in \{w_1, ..., w_6\}$ is determined by the type of the state node s_n according to the following rule:

if $q'_n \in L$:
  if q'_n is transformed to q_n by concatenation: $w_n = w_1$
  else if q'_n is transformed to q_n by splitting: $w_n = w_2$
  else: $w_n = w_3$
else if $q'_n \notin L$:
  if q'_n is transformed to q_n by concatenation: $w_n = w_4$
  else if q'_n is transformed to q_n by splitting: $w_n = w_5$
  else: $w_n = w_6$

The model parameters $\{w_1, ..., w_7\}$ are trained by the Powell search algorithm [12] on the development dataset.

3.3 A Large-scale Trusted Lexicon

We find that with a clean vocabulary, it will significantly improve the performance of spelling correction. However, to obtain such a clean vocabulary is usually difficult in practice. To do this, we make use of the Wikipedia data. Particularly, we select the top 2 million words from Wikipedia by their word frequencies, and automatically curate the obtained words by removing those frequent but illegitimate words from the vocabulary. This curate process involves checking if the word appears in the title of a Wikipedia article, comparing the bigram probability of other words etc. Finally we obtained 1.2 million highly reliable words in the vocabulary.

3.4 Error Model Training

The error model intends to model the probability that one word is misspelled into another (either valid or invalid). Previous studies have shown that a weighted edit distance model trained with a sufficient large set of correction pairs could achieve a comparable performance with a sophisticated n-gram model [6]. Meanwhile, a higher order model has more tendency to overfit if the training data is not large enough. Given these considerations, we adopt the weighted edit distance model as the error model in our system. More specifically, we follow the study of Duan and Hsu [6] to model the joint probability of character transformations as the weighted edit distance. In this model, the basic edit operation is defined as a pair of characters from source and destination of the correction, respectively. Null character is included in the vocabulary to model the insertion and deletion operation. The misspelled word and its correction are viewed as generated from a sequence of edit operations. The parameters in this model are trained with an EM algorithm which iteratively maximizes the likelihood of the training set of correction pairs. To obtain a proper set of correction pairs for training, we leverage the existing spelling services of major search engines (Google and Bing). We submit the queries to the spelling services and record the corrections once consensus is reached.

3.5 Use of Web N-gram Model

Another important factor in selecting and ranking the correction candidates is the prior probability of a correction phrase. It represents our prior belief about how likely a query will be chosen by the user without seeing any input from the user. In this work we make use of the Web n-gram service provided by Microsoft [1]. Web n-gram model intends to model the n-gram probability of English phrases with the parameters estimated from the entire Web data. It also differentiates the sources of the data to build different language models from the title, anchor text and body of Web pages, as well as the queries from query log. In our study, we find the **title** model is the most effective for query spelling correction. We hypothesize that this may be because the training data for query model is much noisier. Particularly, misspelled queries may be included in the training data, which makes it less effective for the task of spelling correction. Despite trained with the Web data, Web n-gram model may also suffer from data sparseness in higher order models. To avoid this issue, we make use of the **bigram** model in building our spelling system.

4. EXPERIMENTS AND DISCUSSION

In order to evaluate the performance of CloudSpeller, we have tested it on two query spelling correction datasets. One is the TREC dataset based on the publicly available TREC queries (2008 Million Query Track). This dataset contains 5892 queries and corrections annotated by the Speller Challenge organizers. There could be more than one plausible corrections for a query. In this dataset only 5.3% of queries are judged as misspelled. We also annotated another dataset that contains 4926 from the MSN queries, for each query there is at most only one correction. About 13% of queries are judged as misspelled in this dataset, which is close to the error rate of real web queries. We divide the TREC and MSN datasets into training and test set evenly. CloudSpeller is trained on the training sets and finally evaluated on the TREC test set containing 2947 queries and MSN test set containing 2421 queries.

4.1 Results

We follow the same evaluation metrics as the Speller Challenge, and report results on TREC and MSN datasets in Table 1. Top 40 corrections are used in the default setting of CloudSpeller. The results indicate that CloudSpeller is of very high precision and recall in TREC dataset. In the MSN dataset which is considered harder since it has more misspelled queries, CloudSpeller also achieves high precision of 0.912 and recall of 0.969. This suggests CloudSpeller is very effective for handling spelling errors in search queries overall. We also break down the results by error types so that we can see more clearly the distribution of types of spelling errors and how well our system addressing each type of errors. We present the results of this analysis on Table 2. The breakdown results show that most queries are in the group of "no error", which are much easier to correct than the other

three types. As a result, the overall excellent performance was mostly because the system performed extremely well on the "no error" group. Indeed, the system has substantially lower precision on the queries with the other three types of errors. The splitting errors seem to be the hardest to correct, followed by the concatenation errors, and the substitution errors seem to be relatively easier.

Table 1: Results on TREC and MSN dataset

dataset	#queries	precision	recall	f1
TREC	2947	0.955	0.984	0.970
MSN	2421	0.912	0.969	0.940

Table 2: Results by spelling error type

dataset	error type	% queries	precision	recall	f1
TREC	no error	94.7	0.983	0.986	0.984
	substitution	3.9	0.391	0.970	0.557
	concatenation	0.8	0.352	0.929	0.510
	splitting	0.6	0.301	0.945	0.457
MSN	no error	87.0	0.971	0.973	0.972
	substitution	10.1	0.475	0.904	0.623
	concatenation	1.6	0.328	0.886	0.479
	splitting	1.3	0.304	0.866	0.450

4.2 Number of Spelling Corrections

The Speller Challeng encorages participants to generate all plausible query corrections. However it's unknown that how many spelling corrections are enough. In this section we investigate the effect of number of spelling corrections to the results. We have carried out experiments on five correction size (1,5,10,20,40) on both datasets. Figure 2 summarizes the results. It's clear that a bigger number of corrections leads to higher recall, and the most drastical increase of recall lies from 1 correction to 5 corrections. But the correction size has no effect on precision on both datasets, which suggests that the correction size doesn't affect the top-ranked spelling correction.

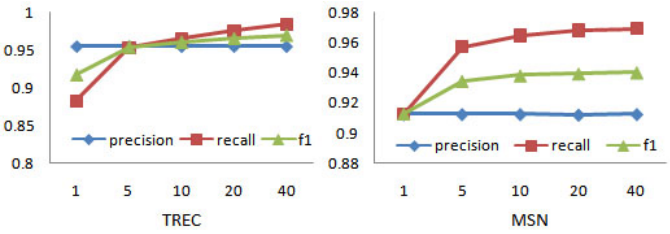

Figure 2: Results by number of corrections

4.3 Effect of the Lexicon Size

The size of the trusted lexicon is an important factor influencing the speller correction result. In order to investigate the effect of lexicon size we conducted a set of experiments on both datasets according to different lexicon sizes (ranging from 100,000 to 900,000). Results in Figure 3 shows the effect of lexicon size is significant on precision: the precision increases as the lexicon size increases. However the recall is not sensitive to the lexicon size.

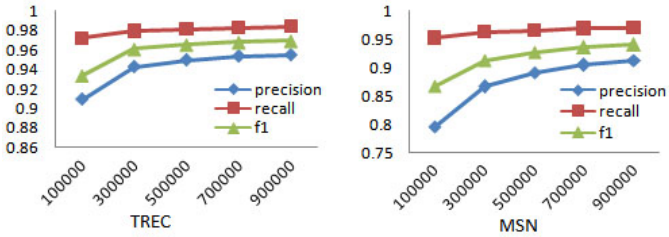

Figure 3: Correction results by lexicion size

4.4 Clean VS Noisy Lexicon

In Table 3 we show the effects of using a clean lexicon for improving the precision and recall of the spelling system. We can see that for both two test dataset, there is noticeable improvement in precision. By removing the noise in the automatically constructed lexicon, our system is able to find matches for candidate queries more precisely. It is interesting that we also observe small improvement in recall for the second test dataset. This is reasonable as we have to limit the number of output candidates in our system due to performance consideration. By reducing the number of matches against the noisy terms, we are able to include more promising results in our ranker, and hence able to improve the overall recall.

Table 3: Clean VS noisy lexicon

dataset	lexicon type	precision	recall	f1
TREC	clean lexicon	0.955	0.984	0.970
	noisy lexicon	0.950	0.984	0.966
MSN	clean lexicon	0.912	0.969	0.940
	noisy lexicon	0.896	0.967	0.930

5. CONCLUSIONS

The key novelty of our system lies in the unified Hidden Markov model that successfully models all major types of spelling errors in search queries, which is under addressed by previous works. The large and clean lexicon, advanced error model and n-gram model are also critical to our system. In the future, we want to directly train the HMM model with examples, removing the need to re-rank the candidate paths. We are also investigating a better way to cache n-gram probabilities, which is crucial to the speed of our system.

6. ACKNOWLEGEMENTS

This work is supported in part by MIAS, the Multimodal Information Access and Synthesis center at UIUC, part of CCICADA, a DHS Center of Excellence, by the National Science Foundation under grants IIS-0713581 and CNS-0834709, and by a gift grant from Microsoft.

7. REFERENCES

[1] http://research.microsoft.com/en-us/collaboration/focus/cs/web-ngram.aspx.

[2] F. Ahmad and G. Kondrak. Learning a spelling error model from search query logs. In *HLT/EMNLP*. The Association for Computational Linguistics, 2005.

[3] E. Brill and R. Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong, 2000.

[4] Q. Chen, M. Li, and M. Zhou. Improving query spelling correction using web search results. In *EMNLP-CoNLL*, pages 181–189. ACL, 2007.

[5] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2004.

[6] H. Duan and B.-J. P. Hsu. Online spelling correction for query completion. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 117–126, New York, NY, USA, 2011. ACM.

[7] J. Gao, X. Li, D. Micol, C. Quirk, and X. Sun. A large scale ranker-based system for search query spelling correction. In C.-R. Huang and D. Jurafsky, editors, *COLING*, pages 358–366. Tsinghua University Press, 2010.

[8] A. R. Golding. A bayesian hybrid method for context-sensitive spelling correction. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 39–53, Boston, MA, 1997. ACL.

[9] A. R. Golding and D. Roth. Applying winnow to context-sensitive spelling correction. *CoRR*, cmp-lg/9607024, 1996. informal publication.

[10] K. Kukich. Techniques for automatically correcting words in text. *ACM computing surveys*, 24(4), 1992.

[11] L. Mangu and E. Brill. Automatic rule acquisition for spelling correction. In *Proceedings of the 14th International Conference on Machine Learning (ICML-97)*, pages 187–194, Nashville, TN, 1997. Morgan Kaufmann.

[12] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964.

[13] X. Sun, J. Gao, D. Micol, and C. Quirk. Learning phrase-based spelling error models from clickthrough data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 266–274, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[14] K. Wang, C. Thrasher, and B.-J. P. Hsu. Web scale nlp: a case study on url word breaking. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 357–366, New York, NY, USA, 2011. ACM.

[15] C. Whitelaw, B. Hutchinson, G. Chung, and G. Ellis. Using the web for language independent spellchecking and autocorrection. In *EMNLP*, pages 890–899. ACL, 2009.