CloudSpeller: Query Spelling Correction 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 an important component of modern search engines that can help users to express an information need more accurately and thus improve search quality. In this work we proposed and implemented an 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 0.960 and 0.937 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

Query Spelling Correction, CloudSpeller

1 Introduction

Search engine users make all kinds of spelling errors frequently, including simple cases such as single character substitution/del/ins, or more complex cases like multiple word concatenation and splitting. Although there are plenty of spelling correction algorithms in the research literature [2, 3, 5], no existing approaches successfully correct all major types of errors mentioned above. While industrial systems such as Google can handle this problem well, it's unclear what they have done, especially what resources they have employed in order to solve the challenges.

In this paper we describe an efficient end-to-end speller system (called CloudSpeller) that can correct all the major types of spelling errors with high precision and recall. CloudSpeller uses a novel Hidden Markov Model to model all major types of spelling errors in a unified framework. An efficient search algorithm for finding top-K paths is designed to search a small number of final corrections directly. The unified HMM 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. And a web-scale language model is obtained by leveraging the Microsoft Web N-gram service. We demonstrate that these two types of features are crucial for building an accurate and efficient speller for web queries. Such a system has an advantage in computing a more complete search space in the first stage than other two-stage query spelling correction methods [4], thus it can also serve as a pre-processing component of a more complex system that employs tens of features for re-ranking. Another component of CloudSpeller that contributes to improvement of performance is a large and reliable lexicon extracted from Wikipedia. The CloudSpeller system is publicly available at http://www.cloudspeller.com.

WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

2 The CloudSpeller Architecture

The CloudSpeller system accepts a search query as input, and outputs a ranked list of possible corrections. Such ranked list of most likely corrections is generated by a unified HMM model. Other critical components include the large-scale trusted lexicon, the error model and web-scale n-gram language model.

2.1 The Unified HMM Model

We adopt a generative model for spelling correction where a possibly mis-spelled query from a user is assumed to be generated from a correctly spelled query using a Hidden Markov Model. The generative process follows a word-by-word process. At the beginning, the user has a word in its correct form in mind. We would then assume that the word is somehow 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 A, emission probability B and initial state probability π. 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. 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 a lexicon such that $dist(s, q_i) \leq \delta$ ($\delta = 3$ in this work), where $dist(s, q_i)$ is the edit distance between the state phrase s and word q_i in the query. Each state also has the type t; indicating whether the state is a substitution, merging, splitting or NULL state (represented by an empty string, and it doesn't emit any phrase). In order to reduce the search space of model parameters A, B, π as

well as to accommodate additional features, we formulate the unified HMM by a set of feature functions. In this formulation, the log-probability of a state sequence s with its type t is represented by the linear combination of these feature functions:

$$G(s,t) = \sum_{i=1}^{n}\sum_{j=1}^{d} \lambda_j \phi_j(s_{i-1}, t_{i-1}, s_i, t_i) + \sum_{i=1}^{n}\sum_{k=1}^{d'} \mu_k f_k(s_i, t_i, q_{[1:n]}) \quad (1)$$

where feature function $\phi_j(s_{i-1}, t_{i-1}, s_i, t_i)$ and $f_k(s_i, t_i, q_{[1:n]})$ can be calculated by the bigram language model and error model probability respectively. And the coefficients λ_j and μ_k are estimated by a discriminative training algorithm on a set of labeled <query, correction> examples. Finally, the best state sequence, which is equivalent to the most likely correction can be found by:

$$s^*t^* = \arg\max_{s,t} G(s,t) \quad (2)$$

Now we describe the specific ways we instantiate the unified HMM model using web-scale resources as follows:

(1) A large-scale trusted lexicon for generating the state space for HMM. We find that a large and clean lexicon will significantly improve the performance of spelling correction while keeping the candidate state space small. Our lexicon is from 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.

(2) Error Model. The feature function $f_k()$ depends on an error model, which measures the probability that one word is misspelled into another. Here we adopt the Weighted Edit Distance (WED) to estimate the error model. More specifically, we first compute the best character-level alignment of two words, and the WED between these two words is the summation of the WED of all aligned character pairs. We model the character-level WED as the character transformation probability. In order to compute such a probability, a large set of query-correction pairs is obtained by leveraging the spelling services from Google and Bing; and then this probability is estimated as the expected number of transformation from one character to another in these aligned pairs. The training queries are from the MSN search query log released by the Microsoft Live Labs in 2006 (6.5 million queries). They are submitted to the spelling services, and the corrections are recorded once consensus is reached.

(3) Web N-gram Model. The feature function $\phi_j()$ is estimated by a n-gram language model, which represents the probability of a state sequence being correct. 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. 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.

3 Evaluation

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 containing 4926 MSN queries. For each query there is at most only one correction. About 13% of queries are judged as misspelled in this dataset. We divide the TREC and MSN datasets into training and test sets 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.

We evaluate our system based on the evaluation metrics proposed in Microsoft Speller Challenge [1], including expected precision, expected recall and expected F1 measure. Results on TREC and MSN datasets are reported in Table 1 at top 10 corrections as output. 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 0.910 and recall of 0.965. This suggests CloudSpeller is very effective for handling spelling errors in search queries overall. We also break down the results in Table 2 by error types to examine how well our system addresses each type of errors. The breakdown results show that most queries are in the group of “no error", which are easier to correct than the other three types. As a result, the overall excellent performance contributes to the high precision and recall on the “no error" group. On the other hand, the system has relatively 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	0.965	0.960
MSN	2421	0.910	0.965	0.937

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 Conclusions

The key novelty of our system lies in the unified Hidden Markov model that successfully models all major types of spelling errors in web queries, which is under addressed by previous works. The large and clean lexicon, error model and n-gram model are also critical to our system. In the future, we want to improve the system by adding more effective features, while ensuring efficient search and evaluation of the whole set of candidates.

5 References

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

[2] E. Brill and R. Moore. An improved error model for noisy channel spelling correction. In *ACL* 2000

[3] Q. Chen, M. Li, and M. Zhou. Improving query spelling correction using web search results. In *EMNLP* 2007.

[4] J. Gao, X. Li, D. Micol, C. Quirk, and X. Sun. A large scale ranker-based system for search query spelling correction. In *COLING* 2010.

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