A Fast and Accurate Method for Approximate String Search

Ziqi Wang[*]
School of EECS
Peking University
Beijing 100871, China
wangziqi@pku.edu.cn

Gu Xu
Microsoft Research Asia
Building 2, No.5 Danling Street,
Beijing 100080, China
guxu@microsoft.com

Hang Li
Microsoft Research Asia
Building 2, No.5 Danling Street,
Beijing 100080, China
hangli@microsoft.com

Ming Zhang
School of EECS
Peking University
Beijing 100871, China
mzhang@net.pku.edu.cn

Abstract

This paper proposes a new method for approximate string search, specifically candidate generation in spelling error correction, which is a task as follows. Given a misspelled word, the system finds words in a dictionary, which are most "similar" to the misspelled word. The paper proposes a probabilistic approach to the task, which is both accurate and efficient. The approach includes the use of a log linear model, a method for training the model, and an algorithm for finding the top k candidates. The log linear model is defined as a conditional probability distribution of a corrected word and a rule set for the correction conditioned on the misspelled word. The learning method employs the criterion in candidate generation as loss function. The retrieval algorithm is efficient and is guaranteed to find the optimal k candidates. Experimental results on large scale data show that the proposed approach improves upon existing methods in terms of accuracy in different settings.

1 Introduction

This paper addresses the following problem, referred to as approximate string search. Given a query string, a dictionary of strings (vocabulary), and a set of operators, the system returns the top k strings in the dictionary that can be transformed from the query string by applying several operators in the operator set. Here each operator is a rule that can replace a substring in the query string with another substring. The top k results are defined in terms of an evaluation measure employed in a specific application. The requirement is that the task must be conducted very efficiently.

Approximate string search is useful in many applications including spelling error correction, similar terminology retrieval, duplicate detection, etc. Although certain progress has been made for addressing the problem, further investigation on the task is still necessary, particularly from the viewpoint of *enhancing both accuracy and efficiency*.

Without loss of generality, in this paper we address candidate generation in spelling error correction. Candidate generation is to find the most possible corrections of a misspelled word. In such a problem, strings are words, and the operators represent insertion, deletion, and substitution of characters with or without surrounding characters, for example, "a"→"e" and "lly"→"ly". Note that candidate generation is concerned with a single word; after candidate generation, the words surrounding it in the text can be further leveraged to make the final *candidate selection*, e.g., Li et al. (2006), Golding and Roth (1999).

In spelling error correction, Brill and Moore (2000) proposed employing a generative model for candidate generation and a hierarchy of trie structures for fast candidate retrieval. Our approach is a discriminative approach and is aimed at improving Brill and Moore's method. Okazaki *et al.* (2008) proposed using a logistic regression model for approximate dictionary matching. Their method is also a discriminative approach, but it is largely different from our approach in the following points. It formalizes the problem as binary classification and

[*] Contribution during internship at Microsoft Research Asia.

assumes that there is only one rule applicable each time in candidate generation. Efficiency is also not a major concern for them, because it is for offline text mining.

There are two fundamental problems in research on approximate string search: (1) how to build a model that can archive both high accuracy and efficiency, and (2) how to develop a data structure and algorithm that can facilitate efficient retrieval of the top k candidates.

In this paper, we propose a probabilistic approach to the task. Our approach is novel and unique in the following aspects. It employs (a) a log-linear (discriminative) model for candidate generation, (b) an effective algorithm for model learning, and (c) an efficient algorithm for candidate retrieval.

The log linear model is defined as a conditional probability distribution of a corrected word and a rule set for the correction given the misspelled word. The learning method employs, in the training process, a criterion that represents the goal of making both accurate and efficient prediction (candidate generation). As a result, the model is optimally trained toward its objective. The retrieval algorithm uses special data structures and efficiently performs the top k candidates finding. It is guaranteed to find the best k candidates without enumerating all the possible ones.

We empirically evaluated the proposed method in spelling error correction of web search queries. The experimental results have verified that the accuracy of the top candidates given by our method is significantly higher than those given by the baseline methods. Our method is more accurate than the baseline methods in different settings such as large rule sets and large vocabulary sizes. The efficiency of our method is also very high in different experimental settings.

2 Related Work

Approximate string search has been studied by many researchers. Previous work mainly focused on efficiency rather than model. Usually, it is assumed that the model (similarity distance) is fixed and the goal is to efficiently find all the strings in the collection whose similarity distances are within a threshold. Most existing methods employ n-gram based algorithms (Behm et al., 2009; Li et al., 2007; Yang et al., 2008) or filtering algorithms (Mihov and Schulz, 2004; Li et al., 2008). Instead of finding all the candidates in a fixed range, methods for finding the top k candidates have also been developed. For example, the method by Vernica and Li (2009) utilized n-gram based inverted lists as index structure and a similarity function based on n-gram overlaps and word frequencies. Yang et al. (2010) presented a general framework for top k retrieval based on n-grams. In contrast, our work in this paper aims to learn a ranking function which can achieve both high accuracy and efficiency.

Spelling error correction normally consists of candidate generation and candidate final selection. The former task is an example of approximate string search. Note that candidate generation is only concerned with a single word. For single-word candidate generation, rule-based approach is commonly used. The use of edit distance is a typical example, which exploits operations of character deletion, insertion and substitution. Some methods generate candidates within a fixed range of edit distance or different ranges for strings with different lengths (Li et al., 2006; Whitelaw et al., 2009). Other methods make use of weighted edit distance to enhance the representation power of edit distance (Ristad and Yianilos, 1998; Oncina and Sebban, 2005; McCallum et al., 2005; Ahmad and Kondrak, 2005).

Conventional edit distance does not take in consideration context information. For example, people tend to misspell “c” to “s” or “k” depending on contexts, and a straightforward application of edit distance cannot deal with the problem. To address the challenge, some researchers proposed using a large number of substitution rules containing context information (at character level). For example, Brill and Moore (2000) developed a generative model including contextual substitution rules; and Toutanova and Moore (2002) further improved the model by adding pronunciation factors into the model. Schaback and Li (2007) proposed a multi-level feature-based framework for spelling error correction including a modification of Brill and Moore's model (2000). Okazaki et al. (2008) utilized substring substitution rules and incorporated the rules into a L_1-regularized logistic regression model. Okazaki et al.'s model is largely different from the model proposed in this paper, although both of them are discriminative models. Their model is a binary classification model and it is assumed that only a single rule is applied in candidate generation.

Since users' behavior of misspelling and correction can be frequently observed in web search log data, it has been proposed to mine spelling-error and correction pairs by using search log data. The mined pairs can be directly used in spelling error correction. Methods of selecting spelling and correction pairs with maximum entropy model (Chen et al., 2007) or similarity functions (Islam and Inkpen, 2009; Jones et al., 2006) have been developed. The mined pairs *can only be used in candidate generation of high frequency typos*, however. In this paper, we work on candidate generation *at the character level*, which can be applied to spelling error correction for both high and low frequency words.

3 Model for Candidate Generation

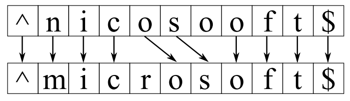
As an example of approximate string search, we consider candidate generation in spelling correction. Suppose that there is a vocabulary $\mathcal{V}$ and a misspelled word, the objective of candidate generation is to select the best corrections from the vocabulary $\mathcal{V}$. We care about both accuracy and efficiency of the process. The problem is very challenging when the size of vocabulary is large, because there are a large number of potential candidates to be verified.

In this paper, we propose a probabilistic approach to candidate generation, which can achieve both high accuracy and efficiency, and is particularly powerful when the scale is large.

In our approach, it is assumed that a large number of misspelled words and their best corrections are given as training data. A probabilistic model is then trained by using the training data, which can assign ranking scores to candidates. The best candidates for correction of a misspelled word are thus defined as those candidates having the highest probabilistic scores with respect to the training data and the operators.

Hereafter, we will describe the probabilistic model for candidate generation, as well as training and exploitation of the model.

Edit-distance based aligment



Derived rules
$\mathrm{n} \rightarrow \mathrm{m}, \phi \rightarrow \mathrm{r}, \mathrm{o} \rightarrow \phi$

Expended rules with context
$\mathrm{n} \rightarrow \mathrm{m}: \;\hat{}\mathrm{n} \rightarrow \hat{}\mathrm{m}, \mathrm{ni} \rightarrow \mathrm{mi}, \hat{}\mathrm{ni} \rightarrow \hat{}\mathrm{mi}$
$\phi \rightarrow \mathrm{r}\;: \mathrm{c} \rightarrow \mathrm{cr}, \mathrm{o} \rightarrow \mathrm{r}, \mathrm{co} \rightarrow \mathrm{cro}$
$\mathrm{o} \rightarrow \phi: \mathrm{oo} \rightarrow \mathrm{o}, \mathrm{of} \rightarrow \mathrm{f}, \mathrm{oof} \rightarrow \mathrm{of}$

Figure 1: Example of rule extraction from word pair

3.1 Model

The operators (rules) represent insertion, deletion, and substitution of characters in a word with or without surrounding context (characters), which are similar to those defined in (Brill and Moore, 2000; Okazaki et al., 2008). An operator is formally represented a rule $\alpha \rightarrow \beta$ that replaces a substring α in a misspelled word with β, where $\alpha, \beta \in \{s | s = t, s = \hat{}t, \text{ or } s = t\$\}$ and $t \in \Sigma^*$ is the set of all possible strings over the alphabet. Obviously, $\mathcal{V} \subset \Sigma^*$. We actually derive all the possible rules from the training data using a similar approach to (Brill and Moore, 2000) as shown in Fig. 1. First we conduct the letter alignment based on the minimum edit-distance, and then derive the rules from the alignment. Furthermore we expand the derived rules with surrounding words. Without loss of generality, we only consider using $+2, +1, 0, -1, -2$ characters as contexts in this paper.

If we can apply a set of rules to transform the misspelled word w_m to a correct word w_c in the vocabulary, then we call the rule set a "transformation" for the word pair w_m and w_c. Note that for a given word pair, it is likely that there are multiple possible transformations for it. For example, both "n"→"m" and "ni"→"mi" can transform "nicrosoft" to "microsoft".

Without loss of generality, we set the maximum number of rules applicable to a word pair to be a fixed number. As a result, the number of possible transformations for a word pair is finite, and usually limited. This is equivalent to the assumption that the number of spelling errors in a word is small.

Given word pair (w_m, w_c), let $R(w_m, w_c)$ denote one transformation (a set of rules) that can rewrite

w_m to w_c. We consider that there is a probabilistic mapping between the misspelled word w_m and correct word w_c plus transformation $R(w_m, w_c)$. We define the conditional probability distribution of w_c and $R(w_m, w_c)$ given w_m as the following log linear model:

$$P(w_c, R(w_m, w_c)|w_m) \tag{1}$$
$$= \frac{\exp\left(\sum_{r \in R(w_m,w_c)} \lambda_r\right)}{\sum_{(w_c', R(w_m, w_c')) \in \mathcal{Z}(w_m)} \exp\left(\sum_{o \in R(w_m, w_c')} \lambda_o\right)}$$

where r or o denotes a rule in rule set R, λ_r or λ_o denotes a weight, and the normalization is carried over $\mathcal{Z}(w_m)$, all pairs of word w_c' in $\mathcal{V}$ and transformation $R(w_m, w_c')$, such that w_m can be transformed to w_c' by $R(w_m, w_c')$. The log linear model actually uses binary features indicating whether or not a rule is applied.

In general, the weights in Equ. (1) can be any real numbers. To improve efficiency in retrieval, we further assume that all the weights are non-positive, i.e., $\forall \lambda_r \leq 0$. It introduces monotonicity in rule application and implies that applying additional rules cannot lead to generation of better candidates. For example, both "office" and "officer" are correct candidates of "ofice". We view "office" a better candidate (with higher probability) than "officer", as it needs one less rule. The assumption is reasonable because the chance of making more errors should be lower than that of making less errors. Our experimental results have shown that the change in accuracy by making the assumption is negligible, but the gain in efficiency is very large.

3.2 Training of Model

Training data is given as a set of pairs $\mathcal{T} = \left\{(w_m^i, w_c^i)\right\}_{i=1}^N$, where w_m^i is a misspelled word and $w_c^i \in \mathcal{V}$ is a correction of w_m^i. The objective of training would be to maximize the conditional probability $P(w_c^i, R(w_m^i, w_c^i)|w_m^i)$ over the training data.

This is not a trivial problem, however, because the "true" transformation $R^*(w_m^i, w_c^i)$ for each word pair w_m^i and w_c^i is not given in the training data. It is often the case that there are multiple transformations applicable, and it is not realistic to assume that such information can be provided by humans or automatically derived. (It is relatively easy to automatically find the pairs w_m^i and w_c^i as explained in Section 5.1).

In this paper, we assume that the transformation that actually generates the correction among all the possible transformations is the one that can give the maximum conditional probability; the exactly same criterion is also used for fast prediction. Therefore we have the following objective function

$$\lambda^* = \arg\max_{\lambda} L(\lambda) \tag{2}$$
$$= \arg\max_{\lambda} \sum_i \max_{R(w_m^i, w_c^i)} \log P(w_c^i, R(w_m^i, w_c^i)|w_m^i)$$

where λ denotes the weight parameters and the $\max$ is taken over the set of transformations that can transform w_m^i to w_c^i.

We employ gradient ascent in the optimization in Equ. (2). At each step, we first find the best transformation for each word pair based on the current parameters $\lambda^{(t)}$

$$R^*(w_m^i, w_c^i) \tag{3}$$
$$= \arg\max_{R(w_m^i, w_c^i)} \log P_{\lambda^{(t)}}(w_c^i, R(w_m^i, w_c^i)|w_m^i)$$

Next, we calculate the gradients,

$$\frac{\partial L}{\partial \lambda_r} = \frac{\sum_i \log P_{\lambda^{(t)}}(w_c^i, R^*(w_m^i, w_c^i)|w_m^i)}{\partial \lambda_r} \tag{4}$$

In this paper, we employ the bounded L-BFGS (Behm et al., 2009) algorithm for the optimization task, which works well even when the number of weights λ is large.

3.3 Candidate Generation

In candidate generation, given a misspelled word w_m, we find the k candidates from the vocabulary, that can be transformed from w_m and have the largest probabilities assigned by the learned model.

We only need to utilize the following ranking function to rank a candidate w_c given a misspelled word w_m, by taking into account Equs. (1) and (2)

$$\text{rank}(w_c|w_m) = \max_{R(w_m, w_c)} \left(\sum_{r \in R(w_m, w_c)} \lambda_r \right) \tag{5}$$

For each possible transformation, we simply take summation of the weights of the rules used in the transformation. We then choose the sum as a ranking score, which is equivalent to ranking candidates based on their largest conditional probabilities.

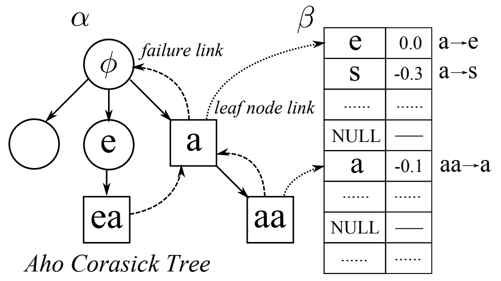


Figure 2: Rule Index based on Aho Corasick Tree.

4 Efficient Retrieval Algorithm

In this section, we introduce how to efficiently perform top k candidate generation. Our retrieval algorithm is guaranteed to find the optimal k candidates with some “pruning” techniques. We first introduce the data structures and then the retrieval algorithm.

4.1 Data Structures

We exploit two data structures for candidate generation. One is a trie for storing and matching words in the vocabulary, referred to as vocabulary trie, and the other based on what we call an Aho-Corasick tree (AC tree) (Aho and Corasick, 1975), which is used for storing and applying correction rules, referred to as rule index. The vocabulary trie is the same as that used in existing work and it will be traversed when searching the top k candidates.

Our rule index is unique because it indexes all the rules based on an AC tree. The AC tree is a trie with “failure links”, on which the Aho-Corasick string matching algorithm can be executed. Aho-Corasick algorithm is a well known dictionary-matching algorithm which can quickly locate all the words in a dictionary within an input string. Time complexity of the algorithm is of linear order in length of input string plus number of matched entries.

We index all the α's in the rules on the AC tree. Each α corresponds to a leaf node, and the β's of the α are stored in an associated list in decreasing order of rule weights λ, as illustrated in Fig. 2. [1]

[1]One may further improve the index structure by using a trie rather than a ranking list to store βs associated with the same α. However the improvement would not be significant because the number of βs associated with each α is usually very small.

4.2 Algorithm

One could employ a naive algorithm that applies all the possible combinations of rules (α's) to the current word w_m, verifies whether the resulting words (candidates) are in the vocabulary, uses the function in Equ. (5) to calculate the ranking scores of the candidates, and find the top k candidates. This algorithm is clearly inefficient.

Our algorithm first employs the Aho-Corasick algorithm to locate all the applicable α's within the input word w_m, from the rule index. The corresponding β's are retrieved as well. Then all the applicable rules are identified and indexed by the applied positions of word w_m.

Our algorithm next traverses the vocabulary trie and searches the top k candidates with some pruning techniques. The algorithm starts from the root node of the vocabulary trie. At each step, it has multiple search branches. It tries to match at the next position of w_m, or apply a rule at the current position of w_m. The following two pruning criteria are employed to significantly accelerate the search process.

1) If the current sum of weights of applied rules is smaller than the smallest weight in the top k list, the search branch is pruned. This criterion is derived from the non-negative constraint on rule weights λ. It is easy to verify that the sum of weights will not become larger if one continues to search the branch because all the weights are non-positive.
2) If two search branches merge at the same node in the vocabulary trie as well as the same position on w_m, the search branches with smaller sum of weights will be pruned. It is based on the dynamic programming technique because we take $\max$ in the ranking function in Equ. 5.

It is not difficult to prove that our algorithm is guaranteed to find the best k candidates in terms of the ranking scores, because we only prune those candidates that cannot give better scores than the ones in the current top k list. Due to the limitation of space, we omit the proof of the theorem that if the weights of rules λ are non-positive and the ranking function is defined as in Equ. 5, then the top k candidates obtained with the pruning criteria are the same as the top k candidates obtained without pruning.

5 Experimental Results

We have experimentally evaluated our approach in spelling error correction of queries in web search. The problem is more challenging than usual due to the following reasons. (1) The vocabulary of queries in web search is extremely large due to the scale, diversity, and dynamics of the Internet. (2) Efficiency is critically important, because the response time of top k candidate retrieval for web search must be kept very low. Our approach for candidate generation is in fact motivated by the application.

5.1 Word Pair Mining

In web search, a search session is comprised of a sequence of queries from the same user within a time period. It is easy to observe from search session data that there are many spelling errors and their corrections occurring in the same sessions. We employed heuristics to automatically mine training pairs from search session data at a commercial search engine.

First, we segmented the query sequence from each user into sessions. If two queries were issued more than 5 minutes apart, then we put a session boundary between them. We used short sessions here because we found that search users usually correct their misspelled queries very quickly after they find the misspellings. Then the following heuristics were employed to identify pairs of misspelled words and their corrections from two consecutive queries within a session:

1) Two queries have the same number of words.
2) There is only one word difference between two queries.
3) For the two distinct words, the word in the first query is considered as misspelled and the second one as its correction.

Finally, we aggregated the identified training pairs across sessions and users and discarded the pairs with low frequencies. Table 1 shows some examples of the mined word pairs.

Misspelled	**Correct**	**Misspelled**	**Correct**
aacoustic	acoustic	chevorle	chevrolet
liyerature	literature	tournemen	tournament
shinngle	shingle	newpape	newspaper
finlad	finland	ccomponet	component
reteive	retrieve	olimpick	olympic

Table 1: Examples of Word Pairs

5.2 Experiments on Accuracy

Two representative methods were used as baselines: the generative model proposed by (Brill and Moore, 2000) referred to as *generative* and the logistic regression model proposed by (Okazaki et al., 2008) referred to as *logistic*. Note that Okazaki et al. (2008)'s model is not particularly for spelling error correction, but it can be employed in the task. When using their method for ranking, we used outputs of the logistic regression model as rank scores.

We compared our method with the two baselines in terms of top k accuracy, which is ratio of the true corrections among the top k candidates generated by a method. All the methods shared the same settings: 973,902 words in the vocabulary, 10,597 rules for correction, and up to two rules used in one transformation. We made use of 100,000 word pairs mined from query sessions for training, and 10,000 word pairs for testing.

The experimental results are shown in Fig. 3. We can see that our method always performs the best when compared with the baselines and the improvements are statistically significant ($p < 0.01$). The *logistic* method works better than *generative*, when k is small, but its performance becomes saturated, when k is large. Usually a discriminative model works better than a generative model, and that seems to be what happens with small k's. However, *logistic* cannot work so well for large k's, because it only allows the use of one rule each time. We observe that there are many word pairs in the data that need to be transformed with multiple rules.

Next, we conducted experiments to investigate how the top k accuracy changes with different sizes of vocabularies, maximum numbers of applicable rules and sizes of rule set for the three methods. The experimental results are shown in Fig. 4, Fig. 5 and Fig. 6.

For the experiment in Fig. 4, we enlarged the vocabulary size from 973,902 (smallVocab) to 2,206,948 (largeVocab) and kept the other settings the same as in the previous experiment. Because more candidates can be generated with a larger vocabulary, the performances of all the methods de-

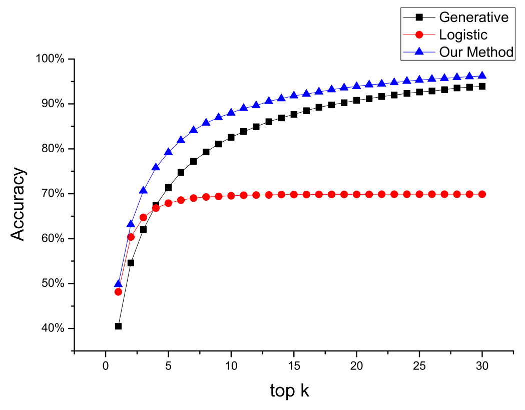


Figure 3: Accuracy Comparison between Our Method and Baselines

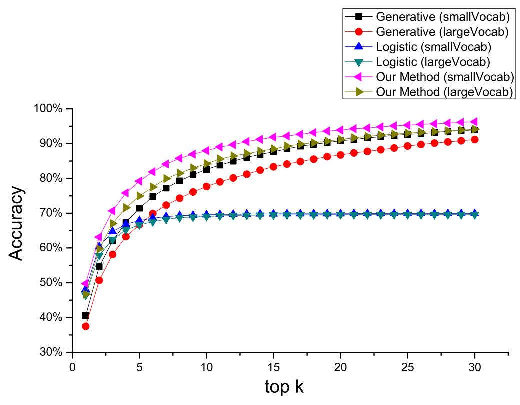


Figure 4: Accuracy Comparisons between Baselines and Our Method with Different Vocabulary Sizes

cline. However, the drop of accuracy by our method is much smaller than that by *generative*, which means our method is more powerful when the vocabulary is large, e.g., for web search. For the experiment in Fig. 5, we changed the maximum number of rules that can be applied to a transformation from 2 to 3. Because *logistic* can only use one rule at a time, it is not included in this experiment. When there are more applicable rules, more candidates can be generated and thus ranking of them becomes more challenging. The accuracies of both methods drop, but our method is constantly better than *generative*. Moreover, the decrease in accuracy by our method is clearly less than that by *generative*. For the experiment in Fig. 6, we enlarged the number of rules from 10,497 (smallRuleNum) to 24,054 (largeRuleNum). The performance of our method and those of the two baselines did not change so much, and our method still visibly outperform the baselines when more rules are exploited.

5.3 Experiments on Efficiency

We have also experimentally evaluated the efficiency of our approach. Because most existing work uses a predefined ranking function, it is not fair to make a comparison with them. Moreover, Okazaki et al.' method does not consider efficiency, and Brill and Moore's method is based a complicated retrieve algorithm which is very hard to implement. Instead of making comparison with the existing methods in terms of efficiency, we evaluated the efficiency of our method by looking at how efficient it becomes with its data structure and pruning technique.

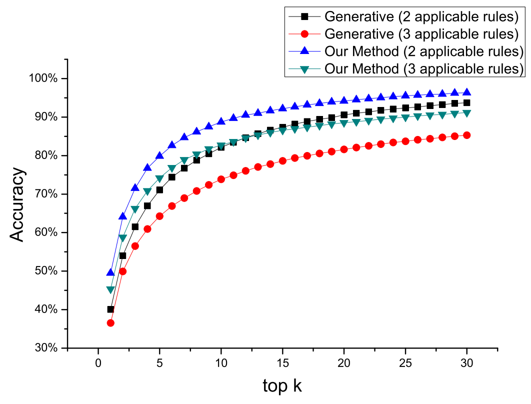


Figure 5: Accuracy Comparison between Generative and Our Method with Different Maximum Numbers of Applicable Rules

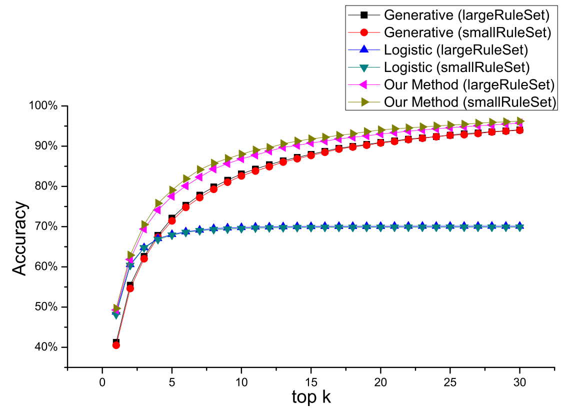


Figure 6: Accuracy Comparison between Baselines and Our Method with Different Numbers of Rules

First, we tested the efficiency of using Aho-Corasick algorithm (the rule index). Because the

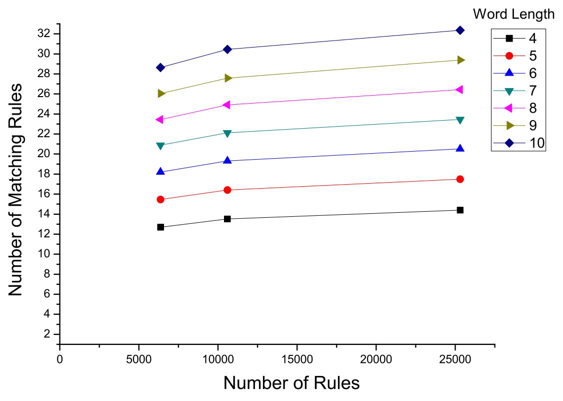


Figure 7: Number of Matching Rules v.s. Number of Rules

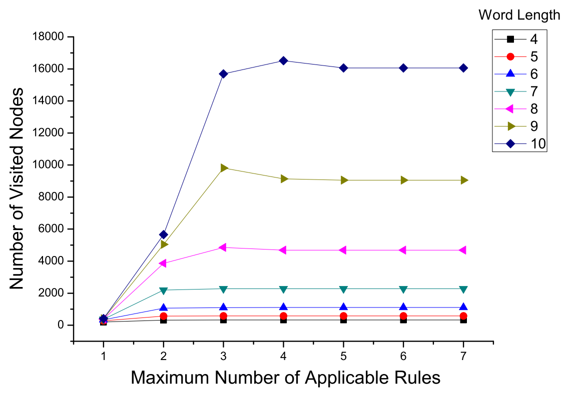


Figure 8: Efficiency Evaluation with Different Maximum Numbers of Applicable Rules

time complexity of Aho-Corasick algorithm is determined by the lengths of query strings and the number of matches, we examined how the number of matches on query strings with different lengths changes when the number of rules increases. The experimental results are shown in Fig. 7. We can see that the number of matches is not largely affected by the number of rules in the rule index. It implies that the time for searching applicable rules is close to a constant and does not change much with different numbers of rules.

Next, since the running time of our method is proportional to the number of visited nodes on the vocabulary trie, we evaluated the efficiency of our method in terms of number of visited nodes. The result reported here is that when k is 10.

Specifically, we tested how the number of visited nodes changes according to three factors: maximum number of applicable rules in a transformation, vocabulary size and rule set size. The experimental results are shown in Fig. 8, Fig. 9 and Fig. 10 respectively. From Fig. 8, with increasing maximum number of applicable rules in a transformation, number of visited nodes increases first and then stabilizes, especially when the words are long. Note that pruning becomes even more effective because number of visited nodes without pruning grows much faster. It demonstrates that our method is very efficient when compared to the non-pruning method. Admittedly, the efficiency of our method also deteriorates somewhat. This would not cause a noticeable issue in real applications, however. In the previous section,

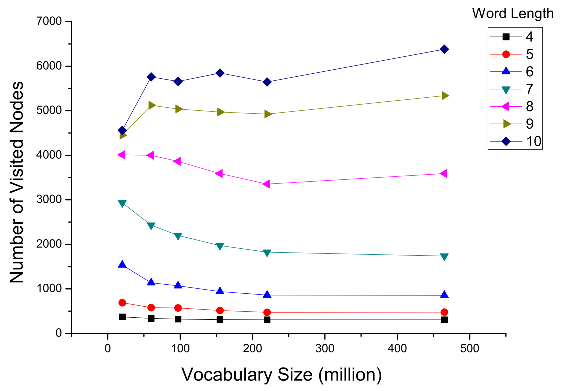


Figure 9: Efficiency Evaluation with Different Sizes of Vocabulary

we have seen that using up to two rules in a transformation can bring a very high accuracy. From Fig. 8 and Fig. 9, we can conclude that the numbers of visited nodes are stable and thus the efficiency of our method keeps high with larger vocabulary size and number of rules. It indicates that our pruning strategy is very effective. From all the figures, we can see that our method is always efficient especially when the words are relatively short.

5.4 Experiments on Model Constraints

In Section 3.1, we introduce the non-positive constraints on the parameters, i.e., $\forall \lambda_r \leq 0$, to enable the pruning technique for efficient top k retrieval. We experimentally verified the impact of the constraints to both the accuracy and efficiency. For ease of reference, we name the model with the non-positive constraints as *bounded*, and the origi-

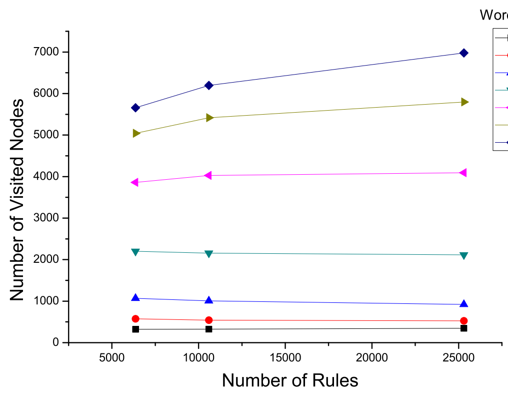


Figure 10: Efficiency Evaluation with Different Number of Rules

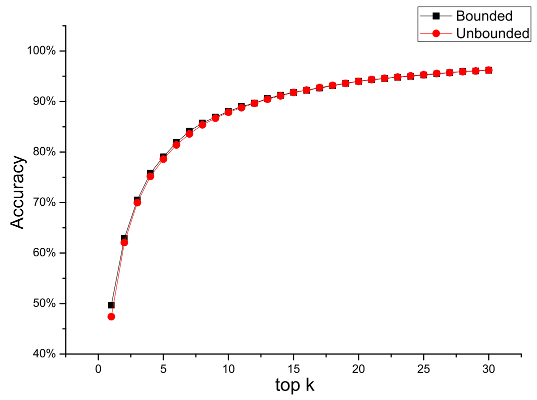


Figure 11: Accuracy Comparison between Bounded and Unbounded Models

nal model as *unbounded*. The experimental results are shown in Fig. 11 and Fig. 12. All the experiments were conducted based on the typical setting of our experiments: 973,902 words in the vocabulary, 10,597 rules, and up to two rules in one transformation. In Fig. 11, we can see that the difference between *bounded* and *unbounded* in terms of accuracy is negligible, and we can draw a conclusion that adding the constraints does not hurt the accuracy. From Fig. 12, it is easy to note that *bounded* is much faster than *unbounded* because our pruning strategy can be applied to *bounded*.

6 Conclusion

In this paper, we have proposed a new method for approximate string search, including spelling error correction, which is both accurate and efficient. Our method is novel and unique in its model, learning

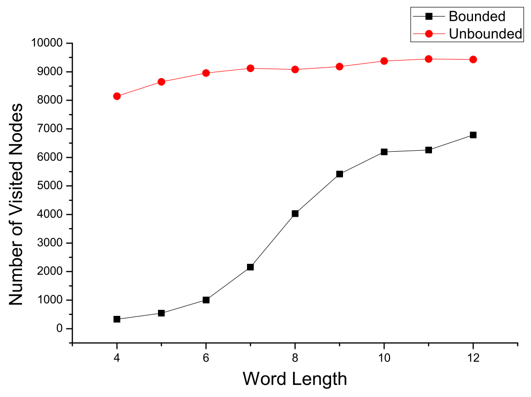


Figure 12: Efficiency Comparison between Bounded and Unbounded Models

algorithm, and retrieval algorithm. Experimental results on a large data set show that our method improves upon existing methods in terms of accuracy, and particularly our method can perform better when the dictionary is large and when there are many rules. Experimental results have also verified the high efficiency of our method. As future work, we plan to add contextual features into the model and apply our method to other data sets in other tasks.

References

Farooq Ahmad and Grzegorz Kondrak. 2005. Learning a spelling error model from search query logs. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 955–962, Morristown, NJ, USA. Association for Computational Linguistics.

Alfred V. Aho and Margaret J. Corasick. 1975. Efficient string matching: an aid to bibliographic search. *Commun. ACM*, 18:333–340, June.

Alexander Behm, Shengyue Ji, Chen Li, and Jiaheng Lu. 2009. Space-constrained gram-based indexing for efficient approximate string search. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 604–615, Washington, DC, USA. IEEE Computer Society.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 286–293, Morristown, NJ, USA. Association for Computational Linguistics.

Qing Chen, Mu Li, and Ming Zhou. 2007. Improving query spelling correction using web search results. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 181–189.

Andrew R. Golding and Dan Roth. 1999. A winnow-based approach to context-sensitive spelling correction. *Mach. Learn.*, 34:107–130, February.

Aminul Islam and Diana Inkpen. 2009. Real-word spelling correction using google web it 3-grams. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1241–1249, Morristown, NJ, USA. Association for Computational Linguistics.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 387–396, New York, NY, USA. ACM.

Mu Li, Yang Zhang, Muhua Zhu, and Ming Zhou. 2006. Exploring distributional similarity based models for query spelling correction. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 1025–1032, Morristown, NJ, USA. Association for Computational Linguistics.

Chen Li, Bin Wang, and Xiaochun Yang. 2007. Vgram: improving performance of approximate queries on string collections using variable-length grams. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 303–314. VLDB Endowment.

Chen Li, Jiaheng Lu, and Yiming Lu. 2008. Efficient merging and filtering algorithms for approximate string searches. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 257–266, Washington, DC, USA. IEEE Computer Society.

Andrew McCallum, Kedar Bellare, and Fernando Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *Conference on Uncertainty in AI (UAI)*.

Stoyan Mihov and Klaus U. Schulz. 2004. Fast approximate search in large dictionaries. *Comput. Linguist.*, 30:451–477, December.

Naoaki Okazaki, Yoshimasa Tsuruoka, Sophia Ananiadou, and Jun'ichi Tsujii. 2008. A discriminative candidate generator for string transformations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 447–456, Morristown, NJ, USA. Association for Computational Linguistics.

Jose Oncina and Marc Sebban. 2005. Learning unbiased stochastic edit distance in the form of a memoryless finite-state transducer. In *International Joint Conference on Machine Learning (2005). Workshop: Grammatical Inference Applications: Successes and Future Challenges*.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string-edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20:522–532, May.

Johannes Schaback and Fang Li. 2007. Multi-level feature extraction for spelling correction. In *IJCAI-2007 Workshop on Analytics for Noisy Unstructured Text Data*, pages 79–86, Hyderabad, India.

Kristina Toutanova and Robert C. Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 144–151, Morristown, NJ, USA. Association for Computational Linguistics.

Rares Vernica and Chen Li. 2009. Efficient top-k algorithms for fuzzy search in string collections. In *Proceedings of the First International Workshop on Keyword Search on Structured Data*, KEYS '09, pages 9–14, New York, NY, USA. ACM.

Casey Whitelaw, Ben Hutchinson, Grace Y. Chung, and Gerard Ellis. 2009. Using the web for language independent spellchecking and autocorrection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 890–899, Morristown, NJ, USA. Association for Computational Linguistics.

Xiaochun Yang, Bin Wang, and Chen Li. 2008. Cost-based variable-length-gram selection for string collections to support approximate queries efficiently. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 353–364, New York, NY, USA. ACM.

Zhenglu Yang, Jianjun Yu, and Masaru Kitsuregawa. 2010. Fast algorithms for top-k approximate string matching. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI '10, pages 1467–1473.