International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
International Conference on Humming Bird (01st March 2014)

RESEARCH ARTICLE OPEN ACCESS

An Efficient Reformulated Model for Transformation of String

Asha Achenkunju [1], V.R. Bhuma[2]

[1]Asha Achenkunju. is currently pursuing M.E. (Software Engineering) at Vins Christian College of Engineering. e-mail: chinjus14@gmail.com,

[2]V.R. Bhuma is currently working as an Assistant Professor, from the department of IT, at VINS Christian College of Engineering.

Abstract:
String Transformation is an essential problem in many applications. Some of which are data mining, natural language processing, and bioinformatics. In string transformation, the input strings are transformed into k most likely output strings by applying a number of operators. The strings can be strings of words, character or any type of tokens and an operator is a transformation rule that defines the replacement of a substring with another substring. This novel and probabilistic approach to string transformation is very accurate and efficient compared to other approaches. String transformation can be conducted at two different settings depending upon the use of dictionary. This approach employs a log linear model, an effective and accurate algorithm and an efficient algorithm for string generation. This log model is defined in terms of an output string and an input string. Here, for the output string, conditional probability distribution is taken into account. Also, for the transformation, a rule set is given as input string. For the purpose of correction of spelling errors in queries and for reformulation of queries by word by word transformation in web search, this proposed method is applicable. This paper evaluates that the aforementioned method is accurate and is particularly powerful when the scale is large.

Keywords— String Transformation, Log Linear Model, Spelling Error Correction, Query Reformulation

I. INTRODUCTION

This is a novel statistical learning approach to string transformation. The main aim of this project is to make string transformation accurate and efficient in spelling error correction of queries and query reformulation. String transformation has many applications in data mining, natural language processing, information retrieval and bio informatics. The major difference between existence work and this work is that, this focuses on enhancement of both accuracy and efficiency of string transformation.

Given an input string and a set of operators, we are able to transform the input string to the *k* most likely output strings by applying a number of operators. Here the strings can be strings of words, characters, or any type of tokens. Each operator is a transformation rule that defines the replacement of a substring with another substring. The likelihood of transformation can represent similarity, relevance, and association between two strings in a specific application. Although certain progress has been made, further investigation of the task is still necessary, particularly from the viewpoint of enhancing both accuracy and efficiency, which is precisely the goal of this work. Misspelling is a common phenomenon among search engine queries. In order to help users effectively express their information needs, mechanisms for automatically correcting misspelled queries are required. This method is a study of a generative model for input queries, based on a noisy channel transformation of the intended queries [1].

Most approaches of string transformation is accurate but not efficient. Aravind Arasu proposed string transformation based on record matching [2]. It is the well-known problem of matching records that represent the same real-world entity and is an important step in the data cleaning process. But at sometime it does not improve record matching quality and it remains unanswered. Other works tried to learn a model with different approaches. The use of edit distance is a typical approach, 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 [3][4]. The approach includes the use of a log linear model, a method for training the model, and an algorithm for generating the top *k* candidates, whether there is or is not a predefined dictionary. The learning method is based on maximum likelihood estimation. Thus, the model is trained toward the objective of generating strings with the largest likelihood given input strings. The generation algorithm efficiently performs the top *k* candidates generation using top *k* pruning. It is guaranteed to find the best *k* candidates without enumerating all the possibilities. An Aho-Corasick tree is employed to index transformation rules in the model. When a dictionary is used in the transformation, a tree is used to efficiently retrieve

the strings in the dictionary.

This paper aims to learn a model for string transformation which can achieve both high accuracy and efficiency. There are three fundamental problems with string transformation: (1) how to define a model which can achieve both high accuracy and efficiency, (2) how to accurately and efficiently train the model from training instances, (3) how to efficiently generate the top *k* output strings given the input string, with or without using a dictionary. The experimental result shows that this method consistently and significantly performs better than the baseline methods of generative model and logistic regression model in terms of accuracy and efficiency. String transformation is about generating one string from another string, such as "TKDE" from "Transactions on Knowledge and Data Engineering". Studies have been conducted on automated learning of a transformation model from data. Eric Brill and Robert C. Moore proposed a method which uses noisy channel model for string transformation. Spelling can be checked by the following equation. Let C be the number of words in the confusion set of d. Then they define the error model.

(1) Add a single letter.
(2) Delete a single letter.
(3) Replace one letter with another.
(4) Transpose two adjacent letters.

$$p(s|d) = \begin{cases} \alpha \text{ if } s = d \\ \dfrac{(1-\alpha)}{(C-1)} \quad \text{otherwise} \end{cases} \qquad (1)$$

There are two possible settings for string transformation. One is to generate strings within a dictionary, and the other is to do so without a dictionary. Eq.1 is used as the former, string transformation becomes approximate string search, which is the problem of identifying strings in a given dictionary that are similar to an input string [5] [6]

Spelling error correction normally consists of candidate generation and candidate selection. The former task is an example of string transformation. Candidate generation is usually only concerned with a single word. For single-word candidate generation, a rule-based approach is commonly used. The use of edit distance is a typical approach, 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. Edit distance does not take context information into consideration. For example, people tend to misspell "c" as "s" or "k" depending on context, and a straightforward use of edit distance cannot deal with the case. To address the challenge, some researchers proposed using a large number of substitution rules containing context information (at character level). Only high frequency pairs can be found from log data, 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 [7].

Query reformulation involves rewriting the original query with its similar queries and enhancing the effectiveness of search. Most existing methods manage to mine transformation rules from pairs of queries in the search logs. One represents an original query and the other represents a similar query (e.g., hotmail sign-on, hotmail sign-up) The weights of the transformation rules are calculated based on log likelihood ratio. A query dictionary is used in this case mined contextual substitution patterns and tried to replace the words in the input query by using the patterns. They created a set of candidates that each differ from the input query in one word [8]. The existing methods mainly focused on how to extract useful patterns and rank the candidates with the patterns, while the models for candidate generation are simple. In this paper, we work on query reformulation as an example of string transformation and we employ a more sophisticated model.

II. MODEL FOR STRING TRANSFORMATION

The string transformation approach is very accurate and efficient improving upon existing methods in terms of accuracy and efficiency in different settings. Here it is assumed that a large number of input string and output string pairs are given as training data. Furthermore, a set of operators for string transformation is provided [9].

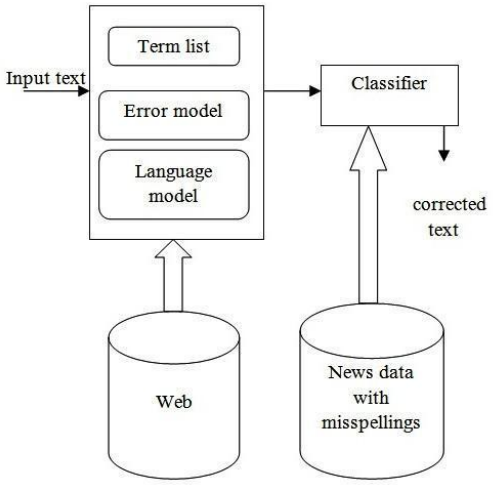

Figure 1: Spelling process

In this paper a log linear model is specified as follows;

$$P(S_0, R(S_i, S_0) \mid S_i) = \frac{\exp\left(\sum_{r \in R} (S_i, S_0)\lambda_r\right)}{\sum_{S' \in R(S_i, S')) \in Z(S_i)} \exp\left(\sum_{0 \in R} (S_i, S_i')\lambda_0\right)} \quad (2)$$

Where r and o denote rules. λ_r and λ_o denote weights. Eq.2 is used in log linear model which actually uses binary features to indicate whether or not rules are applied.

A. Model

The model consists of rules and weights. A rule is formally represented as α->which denotes an operation of replacing substring α in the input string with substring β, where $\alpha, \beta \in \{s \mid s=t, s = \wedge t, s = t\$$, or $s = \wedge t\$$ and $t \in \Sigma^*$ is the set of possible strings over the alphabet, and ^ and $ are the start and end symbols respectively. Sometimes a dictionary is utilized in string transformation in which the output strings must exist in the dictionary, such as spelling error correction, database record matching, and synonym mining. In the setting of using a dictionary, we can further enhance the efficiency. Specifically, we index the dictionary in a trie, such that each string in the dictionary corresponds to the path from the root node to a leaf node. When we expand a path (substring) in candidate generation, we match it against the trie, and see whether the expansions from it are legitimate paths.

The main goal is to find the optimal parameter of the log linear model by solving Eq.3;

$$\lambda^* = \arg\max L(\lambda) = \arg\max \sum_j \log P(s_o^j \mid s_i^j) \quad (3)$$

B .String Generation Algorithm

In this section, it shows how to efficiently generate the top *k* output strings. Top *k* pruning algorithm is performed here, which can guarantee to find the optimal *k* output strings. This also exploits two special data structures to facilitate efficient generation and index the rules with an Aho-Corasick tree. When a dictionary is utilized, we index the dictionary with a tree. The scoring function to rank candidates of output strings s_o given an input string s_i is determined using the eq.4

$$rank(s_o \mid s_i) = \max_{R(s_i, s_o)} \left(\sum_{r \in R(} \right) \quad (4)$$

C. Rule Index

The rule index stores all the rules and their weights using an Aho-Corasick tree (AC tree) , which can make the references of rules very efficient. The AC tree is a trie with "failure links", on which the Aho-Corasick string matching algorithm can be executed. The Aho-Corasick algorithm is a well-known dictionary-matching algorithm which can quickly locate the elements of a finite set of strings within an input string. The time complexity of the algorithm is of linear order in the length of input string plus number of matched entries.

D. Top k Pruning

The string generation problem amounts to that of finding the top *k* output strings given the input string. To further improve the efficiency of pruning algorithm, we need to limit the search space and prune unpromising paths early. In practice, carefully designed beam pruning methods can usually achieve significant improvement in efficiency without causing much loss in accuracy. For absolute pruning, we limit the number of paths to be explored at each position in the target query. With relative pruning, we only explore the paths that have probabilities higher than a certain percentage of the maximum probability at each position. The threshold values are carefully designed to achieve the best efficiency without causing a significant drop in accuracy. In practice we find relative pruning to be generally more effective for pruning unpromising paths. In our system, we make use of both absolute pruning and relative pruning to improve search efficiency and accuracy.

Sometimes a dictionary is utilized in string transformation in which the output strings must exist in the dictionary, such as spelling error correction, database record matching, and synonym mining. In the setting of using a dictionary, we can further enhance the efficiency. Specifically, we index the dictionary in a trie, such that each string in the dictionary corresponds to the path from the root node to a leaf node. When we expand a path (substring) in candidate generation, we match it against the trie, and see whether the expansions from it are legitimate paths. If not, we discard the expansions and avoid generating unlikely candidates. In other words, candidate generation is guided by the traversal of the trie.A search session in web search is comprised of a sequence of queries from the same user within a short time period. Many of search sessions in our data consist of misspelled queries and their corrections. We employed heuristics to automatically mine training pairs from search session data at Bing. First, we segmented the query stream from each user into sessions. If the time period between two queries was more than 5 minutes, then we put a session boundary between them.

Cause	Misspelling	Correction
Typing Quickly	exit misspell	exit misspell
Keyboard adjacency	important	Important
Inconsistent rules	Conceive Concierge	Conceive
Ambiguous word breaking	Silver light	Silverlight

Table 1: Types of Misspellings

Short sessions were used because we observed 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) The two queries have the same number of words.
2) There is only one word difference between the two queries.
3) For the two distinct words, the word in the first query is considered misspelled and the second one its correction.

Finally, we aggregated the identified word pairs across sessions and users and discarded the pairs with low.

III. ALGORITHM DESCRIPTION

Input : rule index I_r, input string s
Output: top k output strings in $S_{topk.}$

1. Begin
2. Find all rules applicable to s from I_r
3. Apply these rules using edit distance method
4. Remove candidate with minimum score
5. Perform spelling error correction
6. Query reformulation by word by word performed using top k pruning
7. Update score and drop the path with smaller score
8. Return S_k

Step 1: Dictionary Creation

String transformation can be conducted at two different settings, depending on the dictionary. When dictionary is used, the output string must exist in the given dictionary, while the size of the dictionary can be very large and the response time is very short. To create a dictionary it follows these steps: add words to the dictionary, arrange the words alphabetically and add description related to that word.

Step 2: Candidate Generation

Misspelling is a common phenomenon in search engine queries. Misspelling occurs for a variety of reasons. When typing quickly users may add or drop letters unintentionally. Accidently hitting an adjacent key on the keyboard make spelling mistakes. In addition to typographical errors, some errors result from the challenge of spelling itself with inconsistent spelling rules, ambiguous word breaking boundaries, and constant introduction of new words. To assist users in expressing information needs, it is important for search engines to automatically generate corrections for misspelled queries. Candidate generation is concerned with a single word and based on rule based approach. Edit method is the typical method which is used here.

Step 3: Candidate Selection

In candidate selection, top k pruning technique is used to select the top k candidates. The string generation problem amounts that of finding top k output strings given the input string. The algorithm uses top k pruning strategy to eliminate unlikely paths and thus improve efficiency. Finally calculate the weight of the generated candidates and thus produces a rank list. Based on this the top k candidates has developed.

Step 4: Query Reformulation

Query reformulation is the process of rewriting the original query with its similar queries and enhancing the effectiveness of search. It involves evaluating a user's input and expanding the search query to match additional documents. Some of the reformulation methods are add or remove words; word substitution, acronym expansion. The existing methods mainly focus on how to extract useful patterns and rank the candidates with the patterns while the models for candidate generation are simple. This method made the string transformation efficient and accurate by word by word transformation.

Step 5: String Mining

The user's has to download its meanings also he/she can download its substrings and reverse etc. Also check the given string which is present in the bunch of strings, if its present the result will be "String Found" otherwise "String Not Found". System generates k most likely output strings based on the input string.

IV. EXPERIMENTAL RESULTS

The difference between the two problems is that string transformation is performed at a character level in the former task and at a word level in the latter task. A dictionary is used in the former problem. The effect of using the Aho-Corasick algorithm is tested here. The time complexity of Aho-Corasick algorithm is determined by the length of query word plus the number of matches. Also examined how the number of matches on query word

changes when the size of the rule set increases. The number of matches is not largely affected when the size of the rule set increases 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. Table 2 shows the sample matching pairs in query reformulation.

Id	Left	Right
1	1st Ave N E	North East First Avenue
2	2nd Ave N	2nd Avenue North
3	3rd Ave E	3rd Avenue East

Table 2: Sample Matching Pairs

In this experiment, it shows a method of logistic in query reformulation. The retrieval part of Brill and Moore's method is based on a hierarchy of tries, which are constructed using dictionary. However, there is no dictionary in the query reformulation task. Although we can compute the score for candidates, we cannot retrieve candidates using the data structure proposed by Brill and Moore. Table 3 shows the example suggestions.

Input Query	Top suggestions
milk shak	milkshake recipes
hwo to tain ur dra	how to train your dragon
alice on wander land	alice in wonderland
mision inpos	mission impossible

Table 3: Examples Suggestions

This method has been applied to two applications, spelling error correction of queries and reformulation of queries in web search. Experiments have been conducted between this method and the baselines methods. The result shows that this method performs consistently better than the baselines in terms of accuracy. Moreover, the accuracy of this method is constantly better than the baselines in different experiment settings, such as size of the rule set, maximum number of applicable rules, and dictionary size. Experiments on efficiency have been conducted with running time as the measure. The running times of our method are smaller than those of the baselines, which indicate that the pruning strategy in this method is very efficient.

V. CONCLUSION

This paper addresses the problem of spelling error correction and query reformulation in string transformation by adopting a discriminative model for string transformation. Spelling error correction performed at character level which is very accurate. Query Reformulation in web search is performed by pruning algorithm. To efficiently retrieve the query corrections with the highest probability according to the discriminative model based on pruning algorithm. Several experiments are conducted here and conclude that the proposed method is effective, accurate and efficient for the task of spelling error correction and query reformulation in web search. This method is novel and unique in its model, learning algorithm, and string generation algorithm. In future work am planning to extend this method by trie based algorithm and instead of showing simple data this shows a graphical representation of the complete derivation of the data in the search engine.

REFERENCES

[1] H.Duan and B.J.P.Hsu, "Online spelling correction for query completion," in Proceedings of the 20th international conference on World wide web, ser. WWW '11. New York, NY, USA: ACM, 2011, pp. 117–126.

[2] A. Arasu, S. Chaudhuri, and R. Kaushik, "Learning string transformations from examples," *Proc.VLDB Endow.*, vol. 2, pp. 514–525, August 2009.

[3] F. Ahmad and G. Kondrak, "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, ser. HLT '05. Morristown, NJ, USA: Association for Computational Linguistics, 2005, pp. 955–962.

[4] E. S. Ristad and P. N. Yianilos, "Learning string-edit distance," IEEE Trans. Pattern Anal. Mach. Intell., vol. 20, pp. 522– 532, May 1998.

[5] A. Islam and D. Inkpen, "Real-word spelling correction using google web it 3-grams," in Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, ser. EMNLP '09.Morristown, NJ, USA: Association for Computational Linguistics, 2009, pp. 1241–1249.

[6] K. Toutanova and R. C. Moore, "Pronunciation modeling for improved spelling correction," in Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ser. ACL '02.Morristown, NJ, USA: Association for Computational Linguistics 2002, pp. 144–151.

[7] A. V. Aho and M. J. Corasick, "Efficient string matching: an aid to bibliographic search," Commun. *ACM*, vol. 18, pp. 333–

340, June 1975.

[8] "Correcting different types of errors in texts," in Proceedings of the 24th Canadian conference on Advances in artificial intelligence, ser. Canadian AI '11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 192–203.

[9] C. Whitelaw, B. Hutchinson, G. Y. Chung, and G. Ellis, "Using the web for language independent spellchecking and auto correction," in Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, ser. EMNLP '09. Morristown, NJ, USA: Association for Computational Linguistics, 2009, pp. 890–899.

[10] R. Jones, B. Rey, O. Madani, and W. Greiner, "Generating query substitutions," in Proceedings of the 15th international conference on World Wide Web, ser. WWW '06. New York, NY, USA: ACM, 2006, pp. 387–39.