A Survey Paper on String Transformation using Probabilistic Approach

[1]Miss. Gayatridevi N.Kotame, [2]Prof. P. N. Kalavadekar
[1]ME-II Scholar, [2]PG Co-ordinator
Computer department, SRES College of Engineering, Savitribai Phule Pune University,Pune.
Kopargaon, Maharashtra, India.

Abstract **- The string is given as an input to the system generates the *k* most likely output strings corresponding to the input string. This system proposes a novel and probabilistic approach to string transformation, 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 generating the top *k* candidates, whether there is or is not a predefined dictionary. The log linear model is defined as a conditional probability distribution of an output string and a rule set for the transformation conditioned on an input string. The learning method employs maximum likelihood estimation for parameter estimation. The string generation algorithm based on pruning is guaranteed to generate the optimal top *k* candidates. The proposed method will apply to correction of spelling errors in queries as well are formulation of queries in web search.**

IndexTerms **-String Transformation, Log Linear Model, Spelling Error Correction, Query Reformulation**

I. INTRODUCTION

In data mining, bioinformatics natural language processing string transformation can include, pronunciation generation, spelling error correction, word transliteration, and word stemming. In query reformulation and query suggestion in search string transformation can also be used. In data mining, string transformation can be employed in the mining of synonyms that is words in similar meaning and database record matching for various applications. As many of the above are consider as an online applications, the transformation must be conducted not only accurately but also efficiently in any of them.

The transformation of one String to other string is a main problem in computational area and natural language processing area. It occurs in tasks with diverse or different names as transliteration, spelling error correction, pronunciation model, word matching. A conditional log linear model for string to string transformation, which attempts overlapping features employs over latent alignment sequences, and which learns latent classes. latent variable and latent string pair regions from training data which is incomplete in nature. This approach on morphological evaluation task and demonstrate that even when it is trained on small data sets, latent variables and latent pair can dramatically improve results.

The strings can be anything as strings of words, string of characters, or any type of tokens from string. A transformation rule is the operator that defines the replacement of a substring with another substring. String transformation can be defined as, Given an input string and a set of operators, to transform the input string to the k most likely output strings by applying number of operators.

II. SYSTEM OVERVIEW

The model consists of rules and weights. A rule is formally represented as $\alpha \rightarrow \beta$ which denotes an operation of replacing substring α in the input string with substring β, where $\alpha,\beta \in \{s \mid s = t; s = \hat{}t, s=t\$, \text{or } s=\hat{}t\$\}$ and $t \in \Sigma^*$ is the set of possible strings over the alphabet, and ^ and $ are the start and end symbols respectively. For different applications, we can consider If a set of rules can be utilized to transform the input string si to the output target string so, then the rule set is said to form a "transformation" for the string pair si and so. Note that for a given string pair, there might be multiple possible transformations.. Without loss of generality, we assume that the maximum number of rules applicable to a string pair is predefined. As a result, the number of possible transformations for a string pair is also limited. This is reasonable because the difference between an input string and output string should not be so large. In spelling error correction, for example, the number of possible spelling errors in a word should be rather small. Let (si; so) denote a string pair, and R(si; so) denote a transformation that can rewrite si to so. We consider that there is a probabilistic mapping between the input string si and output string so with transformation R(si; so).

To improve generation efficiency, we further assume that all the weights are non-positive, i.e., $\forall \lambda_r <= 0$. The assumption is reasonable because the chance of making more errors should be lower than that of making fewer errors.

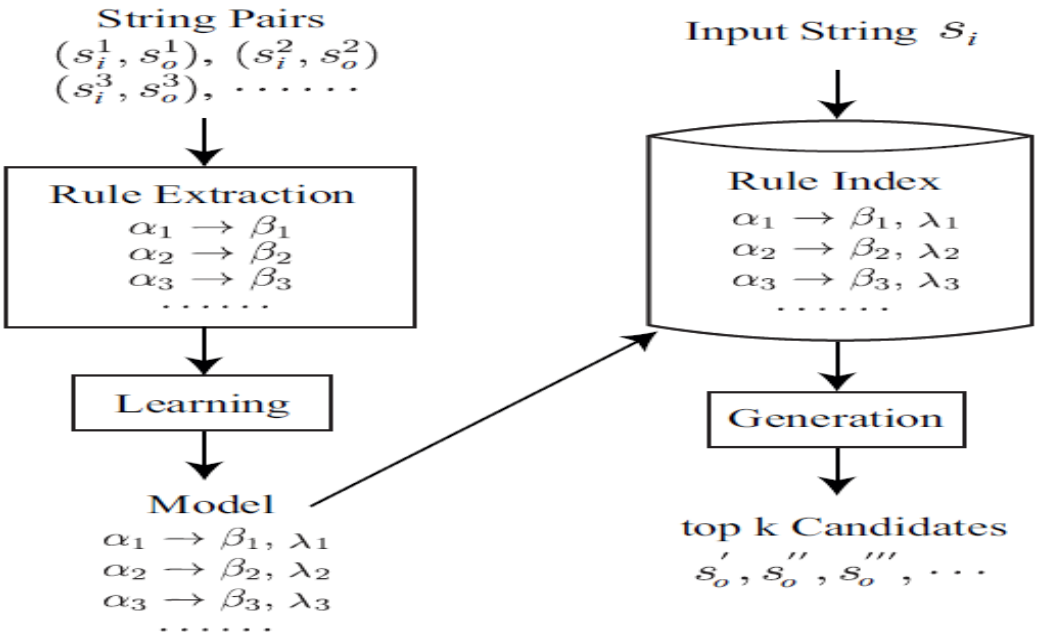

Figure 1. System Overview

III. SYSTEM DESIGN

String transformation can be defined in the following way. 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.

Correcting spelling errors in queries usually consists of two steps: candidate generation and candidate selection. Candidate generation is used to find the most likely corrections of a misspelled word from the dictionary. In such a case, a string of characters is input and the operators represent insertion, deletion, and substitution of characters with or without surrounding characters, for example, “a”->“e” and “lly”->“ly”. Obviously candidate generation is an example of string transformation. Note that candidate generation is concerned with a single word; after candidate generation, the words in the context (i.e., in the query) can be further leveraged to make the final candidate selection, cf., [1], [2].

Query reformulation in search is aimed at dealing with the term mismatch problem. For example, if the query is “NY Times” and the document only contains “New York Times”, then the query and document do not match well and the document will not be ranked high. Query reformulation attempts to transform “NY Times” to “New York Times” and thus make a better matching been the query and document. In the task, given a query (a string of words), one needs to generate all similar queries from the original query (strings of words). The operators are transformations between words in queries such as “tx”->“texas” and “meaning of”->“definition of”, cf., [3]. Previous work on string transformation can be categorized into two groups. Some work mainly considered efficient generation of strings, assuming that the model is given [4]. Other work tried to learn the model with different approaches, such as a generative model, a logistic regression model, Okazaki *et al.* incorporated rules into an *L*1-regularized logistic regression model and utilized the model for string transformation. Dreyer *et al.* also proposed a log linear model for string transformation, with features representing latent alignments between the input and output strings. Finite-state transducers are employed to generate the candidates. Efficiency is not their main consideration since it is used for offline application. This model is different from Dreyer *et al.*’s model in several points. Particularly our model is designed for both accurate and efficient string transformation, with transformation rules as features and non-positive values as feature weights. Okazaki *et al.*’s model is largely different from the model proposed in this system, although both are discriminative models. However, efficiency is not an important factor taken into consideration in these methods. Arasu*et al.* proposed a method which can learn a set of transformation rules that explain most of the given examples. Increasing the coverage of the rule set was the primary focus. Tejada *et al.* proposed an active learning method that can estimate the weights of transformation rules with limited user input.

Query Reformulation

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). For example, the method proposed by Jones *et al.*first identifies phrase-based transformation rules from query pairs, and then segments the input query into phrases, and generates a number of candidates based on substitutions of each phrase using the rules. The weights of the transformation rules are calculated based on log likelihood ratio. A query dictionary is used in this case. Wang and Zhai 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. 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 system, we work on query reformulation as an example of string transformation and we employ a more sophisticated model.

Spelling Error Correction

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 [1], Other methods learn weighted edit distance to enhance the representation power .

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 con-text information (at character level). For example, Brill and Moore developed a generative model including contextual substitution rules. Toutanova and Moore further improved the model by adding pronunciation factors into the model. Duan and Hsu also proposed a generative approach to spelling correction using a noisy channel model. They also considered efficiently generating candidates by using a trie. In this system, we propose using a discriminative model. Both approaches have pros and cons, and normally a discriminative model can work better than a generative model because it is trained for enhancing accuracy.

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 a maximum entropy model and similarity functions have been developed. Only high frequency pairs can be found from log data, however. In this system, we work on candidate generation *at the character level*, which can be applied to spelling error correction for both high and low frequency words.

Learning for String Transformation

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.

Proposed a method which can learn a set of transformation rules that explain most of the given examples. Increasing the coverage of the rule set was the primary focus. Proposed an active learning method that can estimate the weights of transformation rules with limited user input. The types of the transformation rules are predefined such as stemming, prefix, suffix and acronym. Incorporated rules into an L_1-regularized logistic regression model and utilized the model for string transformation. Also proposed a log-linear model for string transformation, with features representing latent alignments between the input and output strings. Finite-state transducers are employed to generate the candidates. Efficiency is not their main consideration since it is used for offline application. Our model is different from Dreyer *et al.*'s model in several points. Particularly our model is designed for both accurate and efficient string transformation, with transformation rules as features and non-positive values as feature weights.

Okazaki *et al.*'s model is largely different from the model proposed in this system, although both are dis-criminative models. Their model is defined as a logistic regression model (classification model) $P(t|s)$, where s and t denote input string and output string respectively,and a feature represents a substitution rule

$$f_k(s,t) = \begin{cases} 1 & \text{rule } r_k \text{ can convert } s \text{ to } t \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Their model utilizes all the rules that can convert s to t and it is assumed only one rule can be applied eachtime.

Approximate string search

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. In 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 . In approximate string search, it is usually assumed that the model (similarity or distance) is fixed and the objective is to efficiently find all the strings in the dictionary. Most existing methods attempt to find all the candidates within a fixed range and employ *n*- gram based algorithms [4], or trie based algorithm There are also methods for finding the top *k* candidates by using *n*-grams . Efficiency is the major focus for these methods and the similarity functions in them are predefined. In contrast, our work in this system aims to learn and utilize a similarity function which can achieve both high accuracy and efficiency.

IV. EXPERIMENTAL RESULT

Table 1. Misspelled word as an Input and Generated Rules Set

Misspelled word	Corrected word	Rules set	Num of Rules
Nycro	micro	n→m y→i	2
Office	office	ϕ→o	1
Office hold	Office word	h→w l→r	2

Documnnt	document	n→d Φ→n	2
Aacoustic	acoustic	a→a	1
Liyeratue	literature	y→t ϕ→r	2
chevorolet	chevrolet	ϕ→o	1
tournament	tournament		0

Table 2. Misspelled word as an Input and Suggested Output Strings with top k Ranking

isspelled word	**rrected word**	**p k**	**Suggested word**
Nycro	Micro	3	Micro
		2	Microsoft
		1	microsoft office
		1	microsoft word
		1	microsoft power point
Office	Office	1	microsoft office
		0	icrosoft office power point
Office hold	Office word	2	Office word
		1	Office
		0	Microsoft word office
		0	office hour
		0	office holder
		0	officer

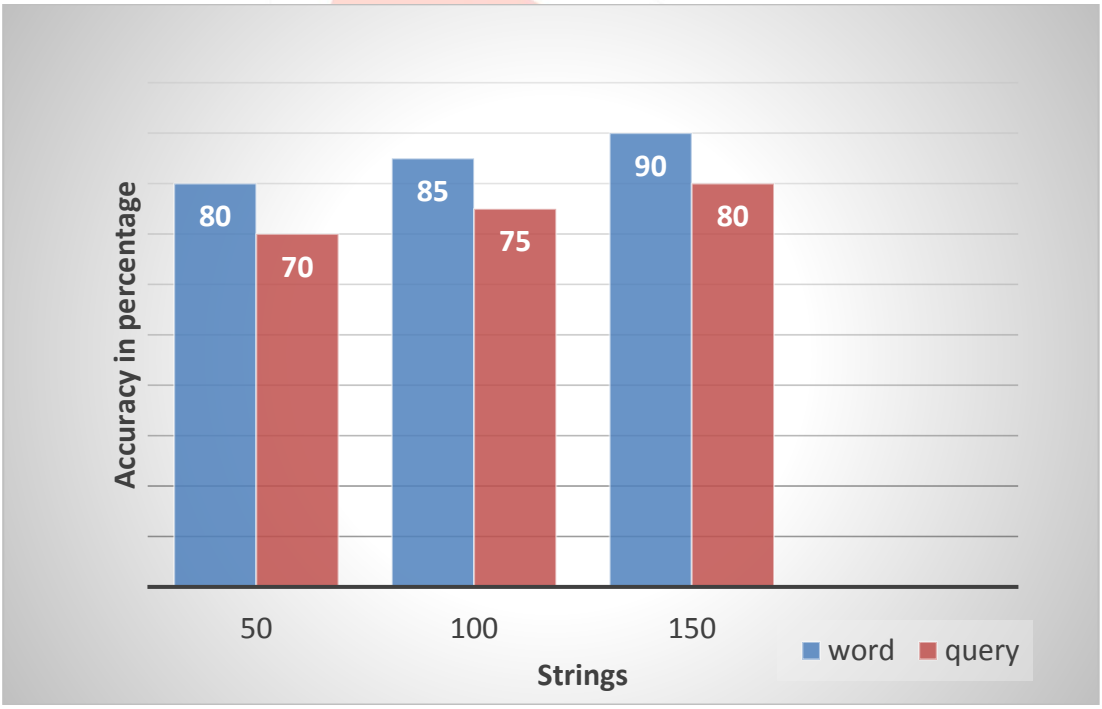

Fig 1. Graph showing the accuracy change with the provided number of input strings

V. ACKNOWLEDGEMENT

It is with the greatest pleasure and pride that I present this report before you. At this moment of triumph, it would be unfair to neglect all those who helped me in the successful completion of this Stage1 seminar. I am very much thankful to my respected project guide Prof. P. N. Kalavadekar, Computer Engineering Department, for his ideas and help proved to be valuable and helpful during the creation of Dissertation Stage1 report and set me in the right path.
I would also like to thank all the faculties who have cleared all the major concepts that were involved in the understanding of techniques behind my seminar. Lastly, I am thankful to my friends who shared their knowledge in this field with me.

VI. REFERENCES

[1] A Probabilistic Approach to String Transformation Ziqi Wang, GuXu,Hang Li, and Ming ZhangA Probabilistic Approach to String Transformation Ziqi Wang, Gu Xu, Hang Li, and Ming Zhang
[2] A. R. Golding and D. Roth, A winnow-based approach to contextsensitive spelling correction, Mach. Learn., vol. 34, pp. 107130, February 1999.

[3] J. Guo, G. Xu, H. Li, and X. Cheng, A unified and discriminative model for query refinement, in Proceedings of the 31st annual internationalACM SIGIR conference on Research and development in informationretrieval, ser. SIGIR 08. New York, NY, USA: ACM, 2008, pp. 379386.
[4] A. Behm, S. Ji, C. Li, and J. Lu, Space-constrained gram-based indexingfor efficient approximate string search, in Proceedings of the 2009IEEE International Conference on Data Engineering, ser. ICDE 09.Washington, DC, USA: IEEE Computer Society, 2009, pp. 604615.
[5] E. Brill and R. C. Moore, An improved error model for noisy channel spelling correction, in Proceedings of the 38th Annual Meeting onAssociation for Computational Linguistics, ser. ACL 00.Morristown, NJ,USA: Association for Computational Linguistics, 2000, pp. 286293.
[6] N. Okazaki, Y. Tsuruoka, S. Ananiadou, and J. Tsujii, A discriminativecandidate generator for string transformations, in Proceedings of theConference on Empirical Methods in Natural Language Processing,ser. EMNLP 08. Morristown, NJ, USA: Association for ComputationalLinguistics, 2008, pp. 447456.
[7] M. Dreyer, J. R. Smith, and J. Eisner, Latent-variable modeling of stringtransductions with finite-state methods, in Proceedings of the Conferenceon Empirical Methods in Natural Language Processing, ser. EMNLP 08.Stroudsburg, PA, USA: Association for Computational Linguistics, 2008,pp. 10801089.
[8] A. Arasu, S. Chaudhuri, and R. Kaushik, Learning string transformationsfrom examples, Proc. VLDB Endow., vol. 2, pp. 514 525, August 2009.
[9] S. Tejada, C. A. Knoblock, and S. Minton, Learning domain independent string transformation weights for high accuracy object identification, in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, ser.
[10] J. Oncina and M. Sebban, Learning unbiased stochastic edit distance in the form of a memoryless finite-state transducer,
[11] C. Li, B. Wang, and X. Yang, Vgram: improving performance of approximate queries on string collections using variable-length grams.
[12]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.