ISSN:2320-0790

COMPUSOFT
An International Journal of Advanced Computer Technology

A Probabilistic Approach to String Transformation

V. Vinothh[1], R.Thaneshwaran[2], K. G. S. Venkatesan[3]
[1,2]UG Student, Department of CSE, Bharath University, Chennai, [3]Asst.Professor, Department of CSE, Bharath University, Chennai

Abstract: The string model has been applied to a wide range of problems, including spelling correction. These models consist of two components: a source model and a channel model. Very little research has gone into improving the channel model for spelling correction. We Describes a new channel model for spelling correction, based on generic string to string edits. Using this model gives significant performance improvements compared to previously proposed models. We propose a novel and probabilistic approach to string transformation, which is both accurate and efficient. In this 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. 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 string generation algorithm based on pruning is guaranteed to generate the optimal top k candidates. The proposed method is applied to correction of spelling errors in queries as well as reformulation of queries in web search. Experimental results on large scale data show that the proposed approach is very accurate and efficient improving upon existing methods in terms of accuracy and efficiency in different settings.

I. Introduction

String transformation can be employed in the mining of synonyms and database record matching. For online applications, the transformation should be conducted accurately and efficiently. String transformation is defined in the following way. Given an input string and a set of operators, we can transform the input string to the k most likely output strings by applying a number of operators. Here the strings could be strings of words, characters, or tokens. Each operator is a transformation rule defines that the replacement of a substring with another substring. The likelihood of transformation can represent similarity, relevance, and association between the strings in a specific application.

String transformation can only be conducted at two different settings, depending on whether a dictionary is used. When a dictionary is used, the output strings will exist in the given dictionary, while the size of the dictionary is very large. Without loss of generality, we specifically study correction of spelling errors in queries as well as reformulation of queries in web search. In the first task, a string consists of characters and in the second task, a string is comprised of words. It needs to exploit a dictionary while the latter does not. Correcting spelling errors in queries 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, then, a string of characters is input and the operators represent insertion, deletion, and substitution of characters with or without surrounding characters.

II. System Architecture

A. Algorithm used

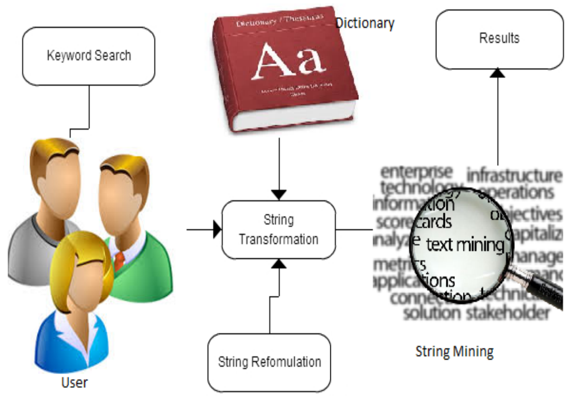
i. Efficient Dictionary Matching Algorithm:

A dictionary matching algorithm is utilized in string transformation in which the output strings must exist in the dictionary, like spelling error correction, database record matching, and synonym mining. Using a dictionary, we can enhance its efficiency. Specifically, we try to index the dictionary in a trie, so 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 look whether the expansions from it are legitimate paths. If not, we delete the expansions and avoid generating unlikely candidates. In other words, the candidate generation is guided by the traversal of the tried.

ii. STRING GENERATION ALGORITHM

The algorithm uses the top k pruning strategy to eliminate unlikely paths and thus improve efficiency. If the score of a path is smaller than the minimum score of the top k list Stopk, then the path is discarded and will be not used further. This strategy works, because the weights of rules are all non-positive and applying additional rules cannot generate a candidate with higher probability. Therefore, it is easy to prove that the best k candidates in terms of the scores can be guaranteed to be found.

B. System Architecture diagram



C. System Modules

- Registration
- Login
- Spelling Error Correction
- Dictionary
- String Transformation
- String Reformulation
- String Mining

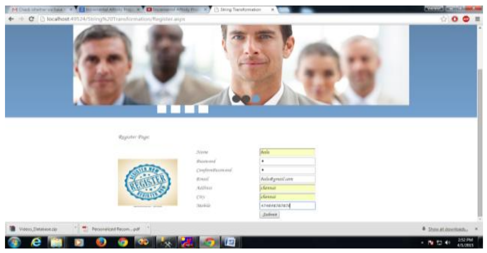
III. System Implementation

We propose a probabilistic approach to the task. This method is novel and unique in the following aspects. It has a log-linear (discriminative) model for string transformation, an effective and an accurate algorithm for model learning, and an efficient algorithm for string generation. The log linear model is a conditional probability distribution of an output string and a rule set for the transformation given an input string. The learning method is based upon maximum likelihood estimation. Thus, the model is trained toward the objective of generating strings with the largest likelihood given input strings. The generation algorithm performs the top k candidate's generation using top k pruning. It is guaranteed to find the best k candidates without enumerating all the possibilities. When a dictionary is used in the transformation, a tire is used to efficiently retrieve the strings in the dictionary. We empirically evaluated our method in spelling error correction of queries and reformulation of queries in web search. The experimental results on the two problems demonstrate that our method consistently and significantly performs better than the baseline methods of generative model and logistic regression model in terms of accuracy and efficiency. We also applied our method to the Microsoft Speller Challenge and found that our method achieves a performance comparable to those of the best performing systems in the challenge.

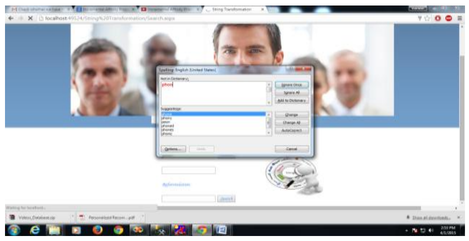
Screenshots:

User Home:

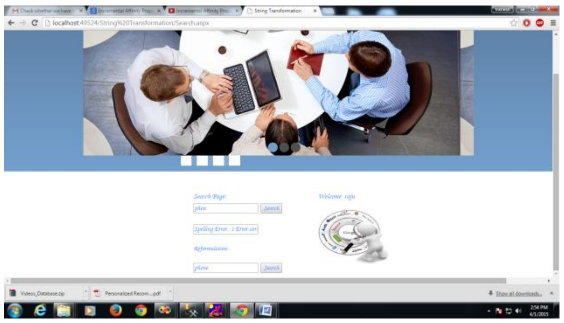
Register:



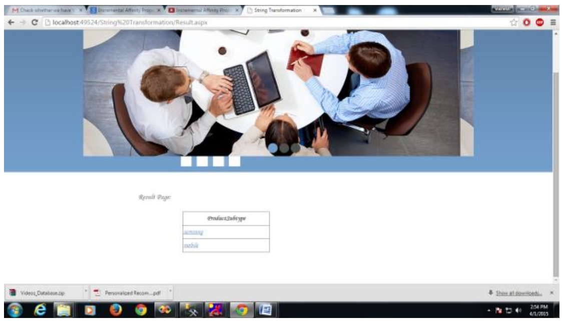
String Transformation:



Transformation:



Result:



IV. Future Enhancements

In the setting of using a dictionary, we can further enhance the efficiency. Using another method to provide efficient string transformation based on user input it matches the string as well as index of the each string also show the reformulation string index number it helps user to get effective output.

V. Related Work

String transformation, which will map a source string into its desirable form t, is related to different applications including stemming, lemmatization and spelling correction. The most important step for string transformation is to make candidates to which the given string s is likely to be transformed. This paper shows a discriminative approach for generating candidate strings. We use substring substitution rules and score them using an L1-regularized logistic regression model. We propose a procedure to generate negative instances that affect the decision boundary of the model. The advantage of this model is that candidate strings can be enumerated by an efficient algorithm because the processes of string transformation are tractable in this approach. We demonstrate the remarkable performance of the proposed method in normalizing inflected words and spelling variations.

String-to-string transduction is a central problem in computational linguistics and natural language processing. It takes place in tasks as diverse as name transliteration, spelling correction, pronunciation modeling and inflectional morphology. We show a conditional log linear model for string-to-string transduction, which has overlapping features over latent alignment sequences, and which studies latent classes and latent string pair regions from incomplete training data. We evaluate our approach on morphological tasks and demonstrate that variables which can dramatically improve results, even when trained on small data sets. While generating morphological forms, we outperform a baseline method reducing the error rate by up to 48%. On a lemmatization task, we reduce the error rates in Wicentowski.

The task of object identification occurs when integrating from multiple websites. The same data objects will exist in inconsistent text formats in different sites, making it difficult to identify matching objects using the exact text match. Previous methods of object identification require manual construction of domain-specific string trans-formations or manual setting of general transformation pa-rameter weights for recognizing the format inconsistencies. This manual process is time consuming and error-prone. We have developed an object identification system called Active Atlas, which will apply a set of domain-independent string transformations so that it can compare the objects' shared attributes in order to identify matching objects. In this paper, we discuss extensions for the Active Atlas system, which will allow it to learn how to tailor the weights of a set of general transformations to a specific application domain through little user input. The experimental results demonstrate that this model achieves higher accuracy and will only require less user involvement than previous methods across various application domains.

Top-k approximate querying on strings is an important data analysis tool for a lot of applications, and it has been exhaustively studied. However, the scale of the problem has been increased dramatically because of the prevalence of the Web. In this paper, we explore the efficient top-k similar string matching problems. Several efficient strategies are introduced, such as length aware and adaptive q-gram selection. We show a general q-gram based framework and propose two efficient algorithms based on the strategies introduced. Our techniques are experimentally evaluated on three real data sets and show a good performance.

Many applications need to solve the following problem of approximate string matching: from a collection of strings, how to handle a given string, or the strings in another (possibly the same) collection of strings. Lot of algorithms are developed using length grams, which are substrings of a string used as signatures to identify similar strings. In this paper we develop a novel technique, called VGRAM, to improve the performance of these algorithms. The main idea is to judiciously choose high-quality grams of variable lengths from a collection of strings to support queries on the collection. We give a full specification of this technique, including how to select high-quality grams from the collection, how to generate variable-length grams for a string based on the preselected grams, and what is the relationship between the similarity of the gram sets of two strings and their edit distance. A primary advantage of the technique is that it can be adopted by a plethora of approximate string algorithms without the need to modify them substantially. We present our extensive experiments on real data sets to evaluate the technique, and show the significant performance improvements on three existing algorithms.

VI. Conclusion

We have presented a new error model for spelling correction based on generic string to string edits, and have been demonstrated that it results in a significant improvement in performance compared to previous approaches. In this method is novel and unique in its model, learning the algorithm, and string generation algorithm. Two specific applications will be addressed with our approach, namely spelling error correction of queries and query reformulation in web search. Experimental results on two large data sets and Microsoft Speller Challenge show that our approach improves upon the baselines in terms of accuracy and efficiency.

VII. References

[1] .N. Okazaki, Y. Tsuruoka, S. Ananiadou, and J. Tsujii, “A discriminative candidate generator for string transformations,” in Proc. Conf. Empirical Methods Natural Language Processing, Morristown, NJ, USA, 2008, pp. 447–456.

[2] M. Dreyer, J. R. Smith, and J. Eisner, “Latent-variable modeling of string transductions with finite-state methods,” in Proc. Conf. Empirical Methods Natural Language Processing, Stroudsburg, PA, USA, 2008, pp. 1080–1089.

[3] A. Arasu, S. Chaudhuri, and R. Kaushik, “Learning string transformations from examples,” Proc. VLDB Endow., vol. 2, no. 1, pp. 514–525, Aug. 2009.

[4] S. Tejada, C. A. Knoblock, and S. Minton, “Learning domainindependent string transformation weights for high accuracy object identification,” in Proc. 8th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, New York, NY, USA, 2002, pp. 350–359.

[5] M. Hadjieleftheriou and C. Li, “Efficient approximate search on string collections,” Proc. VLDB Endow., vol. 2, no. 2, pp. 1660–1661, Aug. 2009.

[6] C. Li, B. Wang, and X. Yang, “VGRAM: Improving performance of approximate queries on string collections using variable-length grams,” in Proc. 33rd Int. Conf. Very Large Data Bases, Vienna, Austria, 2007, pp. 303–314.

[7] X. Yang, B. Wang, and C. Li, “Cost-based variable-length-gram selection for string collections to support approximate queries efficiently,” in Proc. 2008 ACM SIGMOD Int. Conf. Management Data, New York, NY, USA, pp. 353–364. [13] C. Li, J. Lu, and Y. Lu, “Efficient merging and filtering algorithms for approximate string searches,” in Proc. 2008 IEEE 24th Int. Conf. Data Engineering, Washington, DC, USA, pp. 257–266. [14] S. Ji, G. Li, C. Li, and J. Feng, “Efficient interactive fuzzy keyword search,” in Proc. 18th Int. Conf. World Wide Web, New York, NY, USA, 2009, pp. 371–380.

[8] R. Vernica and C. Li, “Efficient top-k algorithms for fuzzy search in string collections,” in Proc. 1st Int. Workshop Keyword Search Structured Data, New York, NY, USA, 2009, pp. 9–14.

[9] Z. Yang, J. Yu, and M. Kitsuregawa, “Fast algorithms for topk approximate string matching,” in Proc. 24th AAAI Conference Artificial Intelligence, 2010, pp. 1467–1473. [17] C. Whitelaw, B. Hutchinson, G. Y. Chung, and G. Ellis, “Using the web for language independent spellchecking and autocorrection,” in Proc. 2009 Conf. Empirical Methods in Natural Language Processing, Morristown, NJ, USA, pp. 890–899.

[10] E. S. Ristad and P. N. Yianilos, “Learning string-edit distance,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 20, no. 5, pp. 522–532, May 1998.