Open access Journal International Journal of Emerging Trends in Science and Technology
Impact Factor: 2.838 DOI: http://dx.doi.org/10.18535/ijetst/v3i06.09

String Transformation for Effective Search

Authors
Ms Sheetal More[1], Prof Manisha Naoghare[2]
SVIT, Chincholi, Nashik Maharashtra, India
Email: *sheetal2406@gmail.com*
SVIT, Chincholi, Nashik Maharashtra, India
Email: *manisha.naoghare@gmail.com*

ABSTRACT
String transformation can be considered as a problem in natural language processing such as data mining, information retrieval, bioinformatics, medical science etc. If we simply consider the medical system there is great need of string transformations in the system. Not only common user but the experts also need the help of string suggestions for accurate results. There are systems which uses different methods of string transformation and generation for giving better results. But these results are accurate and efficient for only small scale datasets. Here a method is proposed which not only give accurate results but also efficient. Thus different string transformation methods are used and queries are reformulated for getting accurate as well as efficient results. The approach includes a model which is used to find the different strings from the databases which match the input string or query on the basis of parameter estimation. An algorithm is used to find the top K matching candidates. The proposed method is used for correction of spelling errors in queries and reformulates the query for better search results.
According to experimental results on large scale datasets the proposed method is accurate and efficient on different string transformation methods.
Keywords- *String Transformation (ST), Spelling Error Correction, Query Reformulation, Top-K pruning, indexing*

INTRODUCTION

String transformation is an essential problem in many applications such as natural language processing, spelling mistakes correction, word transpose, pronunciation generation, and word stemming. Using these string transformations we can reformulate the search query in search system for better search results. String transformation can be done by performing different operations using number of operators on the input strings. Each operator is a transformation rule that defines the replacement of a substring with another substring. This gives top n output strings in results. Among these results the best suited that is candidate having highest score can be selected for better search results. This paper mainly focuses on the important issue of string transformation in many applications. A string can be transformed in many different ways such as spelling mistakes correction, pronunciations generation word translation and word stemming. One of the main important applications of string transformation is query transformation and query suggestions in search system. In many different data mining applications synonym mining and database record matching can be used as a type of string transformation.

String information can be defined as applying different operations on the given input string and obtaining different most likely output strings. Here the operations can be of different types such as removing extra characters, adding the missing characters, adding substring, removing substring, swapping characters, replacing the string bye synonym etc. The operators used for the string transformation rules mostly consist of replacement of substring. There are two ways of string

transformation with and without using dictionary. The approach used in this method is with using the dictionary with large dataset. This transformation of string will help to reformulate the queries for effective search. The main approach of this method is to reformulate the query for effective search. The query can be reformed by applying different string transformation approaches. Many a times it is observed that user enters the short forms of string instead of whole word. So the system should understand whether the entered string is a wrong word or spelling mistake or short form. Sometimes there is need to replace the string with its synonym for better search results.

The main purpose of this method is to achieve both high accuracy and efficiency. The main issue with the string transformation is how to define a model which can achieve both accuracy and efficiency, how to generate the top output results from the large datasets. To overcome these hurdles a log linear model is proposed and an efficient algorithm for string generation. The log linear model is trained with the result set for the given input string to obtain the probable output strings. Then the algorithm is used to select the most likely suitable and accused string. Thus using this method the strings can be transformed and thus query is reformulated for effective search.

LITERATURE SURVEY

String Transformation

Generation of one string from another can be considered as string transformation. For example we can generate three different meanings of HCL as "Hindustan Computer Limited" or "Hindustan Copper Limited" or "Hydro Chloric". So depending upon the query the shortforms HCL will be replaced by the appropriate full string. Similarly if we consider the medical database there are many short forms used for different strings. For example BP in medical terms stands for blood pressure. Thus for the exact and accurate search there is need to replace these short forms by their original meaning. Many researches are made for string transformation such as Arasu el.al proposed a method which focuses on the coverage of rule sets. Tejada el.al proposed an method which estimates the weights of transformation rule with small user input. Okazaki incorporated the predefined rules such as stemming, prefix, suffix, acronym in L1-regularized logistic regression model and utilized it for string regeneration.

Approximate string search

In medical terms there are many strings which are almost similar to each other. There is just a difference of one or two alphabets in these strings. so depending on the query the exact string can be selected. The approximate string can be found by two methods 1) using dictionary and 2) without using dictionary. It is assumed that here the string will be chosen with the help of dictionary only i.e. dataset. It is assumed that in approximate string search the model is fixed and the objective is to efficiently find all the strings in dictionary the existing methods uses N-gram based algorithms or trie based algorithm for finding candidates with a fixed range. There are also methods which uses n-grams for finding the top k candidates.

Spelling Error Corrections

There are different approaches to find the spelling mistakes in a particular word.

a. spelling checking algorithm:-Every query term is checked against dictionary. If a term is not found in a dictionary then those words from the dictionary are shown as spelling suggestions which are most similar to the query terms.

b. similarity/edit distance:- In computer science edit distance Is a way of quantifying how dissimilar two strings (words) are to one another by counting the minimum number of operations required to transform one string into the other. Edit distances find applications in natural processing where automatic spelling correction can determined candidate corrections for the misspelled spelled word by selecting words from a dictionary that have a low distance to the word in question.

c. weighted edit distance:-This is another approach for spelling error correction. With this approach we

can give higher priority to pairs with sounds similar or which are close to each other on the keyboard layout. The naïve approach is used for calculating the edit distance between dictionary terms and query terms but is very expensive. The peter norvig's approach derives all possible terms with an edit distance less than equal to two from the query term and looking them up in the dictionary. it is also expensive .the faroo's approach deletes only the terms with an edit distance less than two both from query term and each dictionary term. Hidden markov model is a statistical alternative to dictionary based spelling error correction.

Reconstruction of Query

There have been different methods proposed for reconstruction of query. Reconstruction of query means rewriting the query with similar meaning. In many of the existing methods query reconstruction means mining some rules from the search logs. In search log there are pairs of queries where one is the original one and another is of similar type. In the method proposed by Jones et al phrase based transformation rules are identified from query pairs and then partitions the input query into segments and generates candidates. Here in this method query dictionary is used. Wang and Zhaihas proposed the method where the patterns are used to substitute the words in input query.

SYSTEM ARCHITECTURE

The main focus of this approach is to achieve accuracy as well as efficiency for large dataset. Initially all input strings and related output strings are provided as training data. Using different rules of string conversion various output string are obtained and scores are assigned to them. Among these output strings, the most accurate matching string is selected as the final output string and results are displayed.

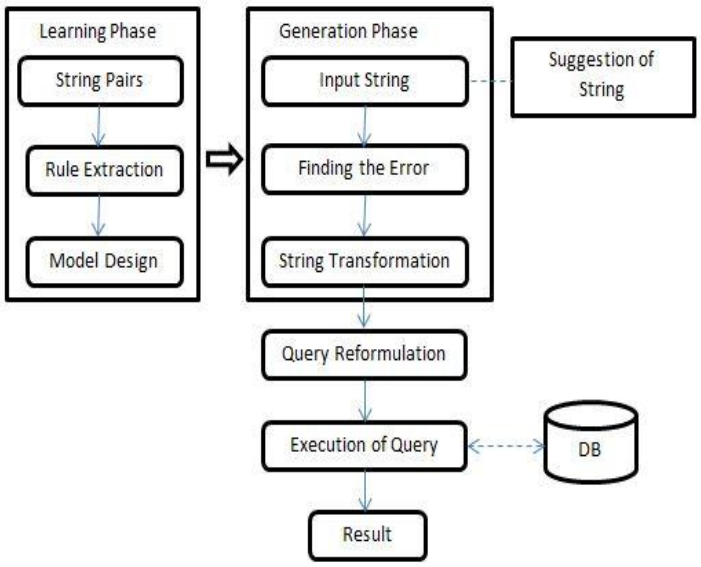

Fig 1. System Architecture

In this method large database/dataset of different strings is provided as input data. The system first accepts the input strings from the user. As soon as the user starts inserting the input string, from the user convenience the system starts giving hint and starts displaying the drop down lists, so that the user will get help to avoid the spelling mistakes. The further work of the system is as follows when the user inputs the query; it is being forwarded to the controller. Here the controller checks whether the inserted word is found in the dictionary. If not, the controller checks the spelling mistakes and if it is so it corrects the spelling errors. Then that correct string and what information is to be displayed related to the input is being forwarded to the model. Suppose spelling mistakes are found then the correct word along with other options i.e. tentative matching words are sent to the model. There the model checks in the database that which string best suits the input string. The string having the highest score (matching accurately) will be selected and displayed to the user in highest matching to lowest matching order. There can be a case where user enters a string which is just a substring of a string. In such cases the strings containing the substring should be displayed according to their ranks.

STRING GENERATION ALGORITHM

Mainly there are two parts of this system. Learning phase and generation phase. Learning phase is at a

background level. Firstly the data is studied i.e database is studied and required string pairs are found out. Depending on the string pairs, different rules are extracted which can be applied for string transformation. Thus the rules are studied and a model is generated which is used for the different string transformation.

The second phase includes the actual execution of the system. When the user enters the string the system starts suggesting the possible related string suggestions. After the user enters they query, the query is distributed in parts into different strings. Then these strings are cheked in the dictionary, if not found then system starts applying different rules on the strings so that based output string can be selected. Thus after selecting the best suitable output string the query is reformulated and passed for further processing. Thus the system executes the fired query and displays the output to the user.

Following are different steps in query reformulation.

Step 1: Indexing the rules.

The Aho-Corasick tree (AC tree) is used to store all the rules and there weights. Indexing the rules will make the references of the rules very efficient. The AC tree is a very well-known dictionary matching algorithm which helps to find quickly finite set of strings related to input.

Step 2: Top k-pruning.

There is a need to select only few output strings which are based suiting the whole query or mainly the input string. The top k pruning algorithm is used to find out top output strings. The input to the algorithm is rule index, input string and candidate number. All applicable rules are found out using AC tree. Then recursive process is applied to apply the rules on the input string to get the top k output stings. Here dictionary is used to obtain the top k output strings.

Step 3: Reformulation of query

Finally the best suited output string obtained by applying the top k pruning algorithm is replaced in the query and thus query is forwarded for execution. Thus query is executed by the system and final results are displayed.

RESULT ANALYSIS

Experiments are performed on different datasets in different conditions. Here each record has 10-15 different more strings. Thus as the number of records increases the number of strings also increases. Some of the results obtained are as follows.

Fig.2 shows the results of records verses matches. Here it can be seen for different word length how the number of matches changes as the size of record increases. As the number of record changes the system starts filtering the strings and only suggests the top mostly suitable strings. Even if the word length changes it shows only limited suitable matches.

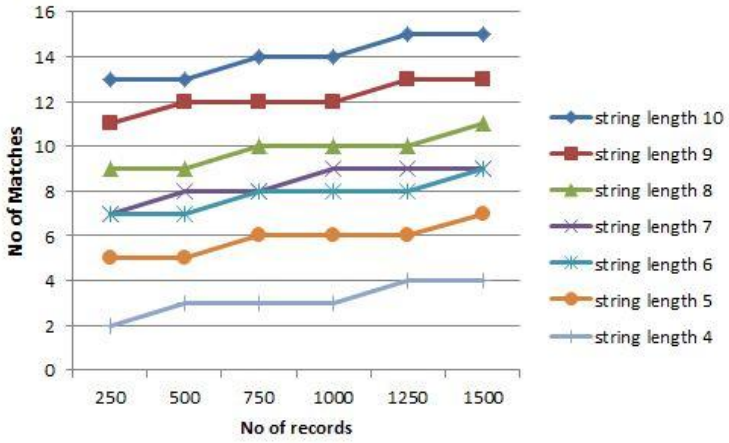

Fig. 2 Efficiency for no of matches

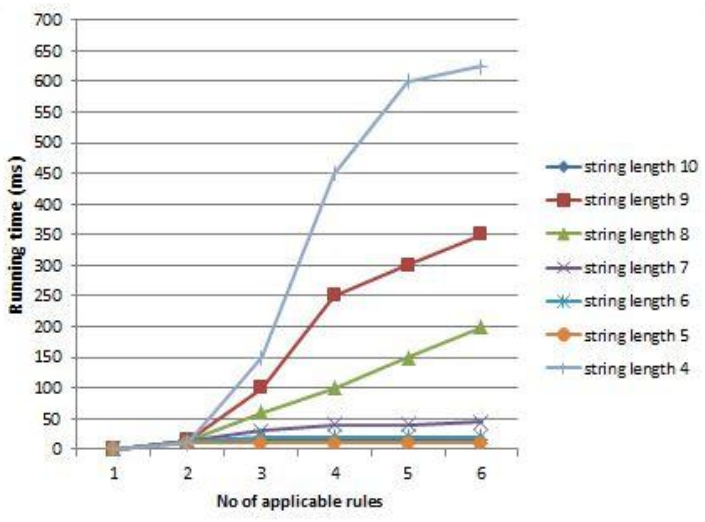

Fig.3 Maximum no of application rules

The above fig.3 shows how many rules are applicable on the input string of different length. It has been found that for the small strings the time required to apply the rules is very less. Sometimes not all the rules are applicable to each and every string, so the execution time remains constant.

The below Fig.4 shows how the performance of the system changes as the no of records increases. It can be seen that the performance of the system does not change as the no of records increases.

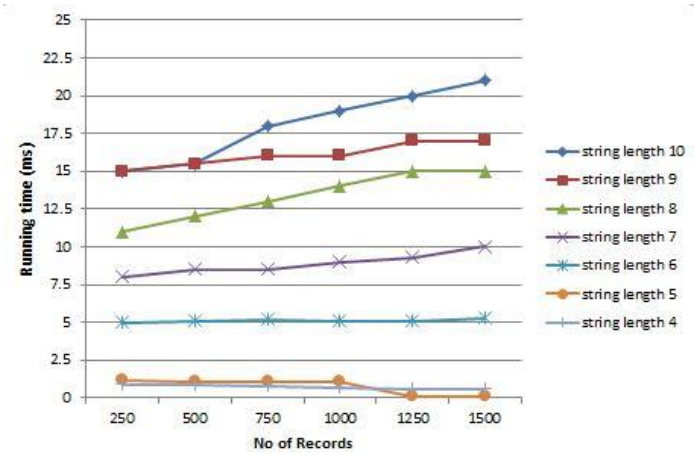

Fig.4 Efficiency of the system

CONCLUSION

As per the need of today's world it has become necessary to increase the search speed which gives not only accurate results but also efficient. This method mainly focuses on both accuracy and efficiency. Thus using the different rules of string transformation helps to regenerate the strings which ultimately improves the search results. Using these string transformation query can be reformulated and thus it also improves the search results. The learning phase helps the system to study the data before only generate the different rules which can be applied for different string transformation. Thus the generation phase takes the input string and applies different string transformation rules extracted in learning phase. Thus because of these phases the system not only give accurate but also efficient results. This approach is both novel and unique in its context.

ACKNOWLEDGEMENT

My sincere thanks go to SVIT CHINCHOLI, NASHIK for providing a strong platform to develop my skills and capabilities. I would like to thank my guide for their constant support and motivation. Also I would like to thanks PG coordinator and HOD of computer department who helped me in presenting this paper. I would also like to thank my friends for their support.

REFERENCES

1. Ziqi Wang, Gu Xu, Hang Li, and Ming Zhang" A Probabilistic Approach to String Transformation" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING VOL:PP NO:99 YEAR 2013.
2. A.Arasu, S. Chaudhuri, and R. Kaushik, "Learning string transformations from examples," Proc. VLDB Endow., vol. 2, pp. 514– 525, August 2009.
3. M. Dreyer, J. R. Smith, and J. Eisner, "Latent-variable modeling of string transductions with finite-state methods," in Proceedings of the Conference on Empirical Methods in Natural Language Processing, ser. EMNLP '08. Stroudsburg, PA, USA: Association for Computational Linguistics, 2008, pp. 1080–1089.
4. S. Tejada, C. A. Knoblock, and S. Minton, "Learning domainindependent string transformation weights for high accuracy object identification," in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, ser. KDD '02. New York, NY, USA: ACM, 2002, pp. 350–359.
5. N. Okazaki, Y. Tsuruoka, S. Ananiadou, and J. Tsujii, "A discriminative candidate generator for string transformations," in Proceedings of the Conference on Empirical Methods in Natural Language Processing, ser. EMNLP '08. Morristown, NJ, USA: Association for Computational Linguistics, 2008, pp. 447–456.

6. M. Li, Y. Zhang, M. Zhu, and M. Zhou, "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, ser. ACL '06. Morristown, NJ, USA: Association for Computational Linguistics, 2006, pp. 1025–1032.
7. A.Behm, S. Ji, C. Li, and J. Lu, "Space-constrained gram-based indexing for efficient approximate string search," in Proceedings of the 2009 IEEE International Conference on Data Engineering, ser. ICDE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 604–615.
8. 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, 02, pp. 144–151.
9. 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. 387–396.
10. .E. Brill and R. C. Moore, "An improved error model for noisy channel spelling correction," in Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, ser. ACL '00 Morristown, NJ, USA: Association for Computational Linguistics,2000, pp. 286–293
11. X. Wang and C. Zhai, "Mining term association patterns from search logs for effective query reformulation," in Proceeding of the 17th ACM conference on Information and knowledge management, ser. CIKM '08. New York, NY, USA: ACM, 2008, pp. 479–488.