A MHR Tree Approach To String Transformation

Mrs. Vinothini R
Department of Computer Science and Engineering, Anna University Regional Center, Madurai, Tamil Nadu

Abstract

String transformation is an important problem in many applications; it must be accurate and efficient. String transformation is a task as follows. An input string is given, the k most likely output strings are generated by the system corresponding to the input string. This project proposes a tree approach to string transformation and to retrieve the data, which is both accurate and efficient. This approach includes the use of a method named log linear model for training the model, and an algorithm for generating the top k candidates, whether it is there or is not a dictionary that is predefined. The log linear model is defined as a conditional probability distribution (CPD) of an output string and a rule set for the conditioned transformation on an input string. The learning method includes maximum calculation for parameter estimation. The string generation algorithm is based on pruning which is guaranteed to generate the top k optimal 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 wide range of data show that the proposed approach is very accurate and efficient improving upon existing methods in terms of accuracy and efficiency in different formats. And the proposed system uses MHR-Tree for string matching process. In existing AC-Tree have related results only, but in MHR-Tree have accurate result of user's query. The MHR-Tree has a root and multiple leaves. The string matching process based on this root and leaves. So finally, user can get expected and accurate result and it provides good performance than existing.

Keywords

MHR Tree, Aho-Corasick Algorithm.

1. Introduction

This project addresses string transformation, in many applications is an essential problem. In natural language processing, generation of pronunciation, spelling error correction, transliteration of word, and word stemming can be formalized as string transformation. String transformation can also be used in query reformulation and query suggestion in string search. In data mining, string transformation can be considered as the mining of synonym sand database record matching. As all of the above said are online applications, the transformation must be conducted accurately and also efficiently.

String transformation is considered as an important problem in many applications; it should be accurate and efficient. String transformation is a generation of one string from another string. Information is retrieved using this, given an input string incorrectly lead to misunderstanding of a query string and expected result is not produced. Spell checking, word stemming, word transliteration are considered as a string transformation. The task followed is that the system generates k most likely output strings corresponding to the input string. The proposed method is applied to correction of spelling errors in queries as well as reformulation of queries in web search.

String transformation can be defined in the following way as when an input string and a set of operators is given, it is ready to transform the input string to the k most likely output strings by applying a number of operators. Here the strings used can be of words, characters, or any type of tokens. Each operator is a transformation rule that defines the replacement of a substring with another substring. This type of string transformation can represent similarity, association, and relevance between two strings in the application specified. Although certain progress has been made, various investigations of the task are still necessary for enhancing both accuracy and efficiency, which is main objective of this work. String transformation can be conducted at two different forms, depending on whether dictionary is used or not used. When a dictionary is used, the output strings produced must be very large. Without generality loss, specifically studied about correction of spelling errors in

queries as well as query reformulation web search in this paper. In the first task, a string consists of characters where dictionary is necessary. In the second task, a string is comprised of words, it does not require dictionary.

Correcting spelling errors in queries comprises usually of two steps, first one is candidate generation and the second one is candidate selection Candidate generation is used to determine the most likely corrected misspelled word from the dictionary. In that case, input is a string of characters and the operators represent insertion, deletion, and substitution of characters with or without characters surrounding, for example, "e"→"a" and "lly"→"ly". Obviously candidate generation is considered to be an example of string transformation. Note that candidate generation is concerned with only single word; after candidate generation, the words in the query can be further undergoing progress to make the final candidate selection, cf.

Query reformulation in web search is aimed at dealing with the problem termed as mismatch. For example, "NY Times" is the query given and if the document only contains "New York Times", then the given query and document do not match well and the document will not be highly ranked. Query reformulation is done to transform "NY Times" to "New York Times" and thus make a matching better between the document and query. In the task given a query which is 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 "Texas" → "Tx" and "definition of" → "meaning of". The proposed approach is very accurate and efficient improving upon existing methods in terms of accuracy and efficiency in different forms. This method is a unique and novel in the following aspects. It involves the use of a log-linear which is a discriminative model for string transformation, an effective and accurate algorithm for model learning, and an efficient algorithm for string generation .The log linear model is defined as a conditional probability distribution of an output string and a rule set for the transformation given an input string. The learning method is based on maximum likelihood estimation. Thus, the model is trained towards achieving the objective of generating strings with the largest similarity of given input strings. The generation algorithm efficiently performs the top k candidate's generation using top k pruning. It is guaranteed to find the k best candidates without trying all the possibilities.

A log-linear model is a mathematical model which takes the form of a function. Its logarithm is a first-degree polynomial function of the parameters of the model that makes it possible to apply linear regression (possibly multivariate). That is, it has the general form

$$\exp\left(c + \sum_i w_i f_i(X)\right),$$

in which the fi(X) are quantities, are functions of the variables X, in general it is considered as a vector of values, where c and the wi stand for the parameters of a model. The term may specifically be used for a log-linear plot or a graph, which is a type of plot. Poisson for contingency tables is a type of model. The specific applications of log-linear models are where the output quantity lies in the range 0 to ∞, for values of the independent variables X, or immediately, the transformed quantities fi(X) in the range −∞ to +∞. This may be little contrasted to logistic models, similar to the logistic function, for which the quantity of the output lies in the range of 0 to 1. Thus the contexts where the models are realistic or useful always depend on the range of the values being modeled.

The proposed approach is a new way of statistical learning approach to string transformation. This method is unique and novel in its learning algorithm, model, and string generation algorithm. Spelling error correction of queries and query reformulation are the two specific applications that are addressed with this method in web search. Experimental results on Microsoft Speller Challenge and two large data sets show that this method improves upon the baselines in terms of efficiency and accuracy. The spatial approximate string (SAS) query is used. In Euclidean space, proposed an approximate solution, the MHR-tree, which embeds min-wise signatures into an R-tree. The min-wise signature for an index node u keeps a concise representation of the union of q-grams from strings under the sub-tree of u. The pruning functionality is analyzed for such signatures based on the resemblance of sets between the q-grams and the query string from the sub-trees of index nodes.

2. Related Works

Z. Yang et al, "Fast algorithms for top-k approximate string matching": Top k similar string match, and fast matching in-gram used to k matching. It helps to real data evaluated, and showing the self-performances. Large query string could not be correct and q-gram dictionary given to top 10k information. The count filtering and length-aware mechanism are adapted to tackle the top-k similar search issue, while the adaptive q-gram selection is employed to improve further the performance of frequency counting. Experimental results show that the proposed algorithms exhibit a superior performance. The demerits of this project includes, fast counting made in mismatching the data, large data could not be read, dictionary should not be used, but if u adding new words it should be added and it is a time consuming process.

Behm et al, "Space-constrained gram-based indexing for efficient approximate string search" In this paper, studied how to reduce the size of such an indexing structure to a given amount of space, while retaining efficient query processing. Two novel approaches for achieving this goal: one is based on discarding gram lists, and one is based on combining, correlated lists. Many information systems need to support approximate string queries: given a collection of textual strings, such as person names, telephone numbers, and addresses, find the strings in the collection that are similar to a given query string. Various functions can be used to measure the similarity between strings, such as edit distance Jaccard similarity, and cosine similarity. The algorithm using the idea of "grams" of strings is a q-gram of a string is a substring of length q that can be used as a signature for the string. The demerits of this includes data base system can only allocate a limited amount of memory for the inverted-list index, accurate and efficiency is low, cost is high and query is not given to correct result (it's given to least result).

J. Guo et al, "A unified and discriminative model for query refinement", Query refinement typically includes a number of tasks such as spelling error correction, word splitting, word merging, phrase segmentation, word stemming, and acronym expansion. Conditional Random Field (CRF) model suitable for the problem, referred to as Conditional Random field for Query Refinement (CRF-QR) is used by which accuracy is enhanced. A sequence of query words, CRF-QR predicts a sequence of refined query words as well as corresponding refinement operations. Two types of CRF-QR models, namely a basic model and an extended model are introduced. Queries may contain misspelled words, mistakenly split words, or mistakenly merged words. Unified and discriminative model in query refinement is effective and it provides better representation for users to find relevant information easily. The demerits of this project includes it's based on cost, query is not given to maximum result, sequence of query word choosing in random, it's not given to accuracy.

M. Li, Y. Zhang, M. Zhu, and M. Zhou, "Exploring distributional similarity based models for query spelling correction", this paper describes novel methods for use of distributional similarity estimated from query logs in learning improved query spelling correction models. The key to methods is the property of distributional similarity between two terms:1) High between a frequently occurring misspelling and its correction,2) Low between two irrelevant terms only with similar spellings.

The methods those are able to take advantage of distributional similarity information are, the first method extends a string edit-based error model with confusion probabilities within a generative source channel model. The second method explores the effectiveness of the approach within a discriminative maximum entropy model framework by integrating distributional Similarity-based features. Experimental results show that both methods can achieve significant improvements over their baseline settings. The demerits of this project includes low frequency terms in query logs are not reliable, spell check is very difficult to correct spelling errors, distance based error model has less accuracy.

C. Whitelaw, B. Hutchinson, G. Y. Chung, and G. Ellis, "Using the web for language independent spellchecking and auto correction", in this paper, an end-to-end system is designed, implemented and evaluated. Spellchecking and auto correction system that does not require any manually annotated training data. Spellchecking and auto correction are widely applicable for tasks such as word processing and post processing Optical Character Recognition. Token n-grams are used to build an LM, which is used to make context-appropriate corrections. Because error model is based on scoring substrings, there is no fixed lexicon of well-spelled words to determine misspellings. Evaluation shows that system performance improves from a system that embeds handcrafted knowledge, yielding a 3.8% total error rate on human typed data that originally had a 10.8% error rate. The demerits of this project includes in this method optimization characteristics is not achieved, spell checking can be performed at a low speed hence time consuming.

3. Models for String Transform

In this existing system probabilistic approach to string transformation is followed where related words are retrieved and which is not accurate and efficient. The string generation algorithm based on pruning method and is guaranteed to generate the optimal top k candidates. It uses LLM (log linear model, that helps in spell checking and query reformation.

Aho-Corasick algorithm is used for index transformation rules. A probabilistic model is then obtained from the training data and the operators, which can assign scores to candidates of output strings given an input string. The best candidates are defined as those having the highest probabilistic scores with respect to the training data.

Even though this project server the purpose it has number of disadvantages, it has some limitations over testing pairs.

4 MHR-Tree

In this proposed approach MHR-Tree (min-wise signature with linear hashing R-tree) is used. It helps to determine the

approximate string match from the database for the given input string. It stores signatures in its nodes. Tree is constructed by using disk block size (B).They first group<=B in a spatial proximity to a MBR, which is stored in a leaf node. Pruning is done to identify candidate strings within small edit distance based on q-grams.

This project works as a query is processed a tree is constructed and retrieve the string that is accurate to the given query and result is displayed. MHR-tree embeds min-wise signatures into an R-tree. The min-wise signature for an index node u keeps a very concise representation of the union of q-grams from strings under the sub-tree of u, suppose the block size of the disk is B. The R-tree and its variants share a similar principle.

The merits of the project includes using MHR-tree the query accuracy is improved. Approximate string is identified and provides fast retrieval of query result as tree strategy is used. Use of pruning algorithm eliminates the unlikely paths from the queue and thus improves efficiency. Accurate expected result is achieved.

5. Spelling Correction

Single word (string) error correction is performed. Edit distance approach is used, which provides similarity indication. Edit distance of two strings s1 and s2 is the minimum number of point of mutations required to change s1 to s2. The point of mutation is a change of one letter, insertion of a letter, deletion of a letter.

String transformation comprises of two process learning and generation. It is a representation of rules and weights. Model is constructed using rules extracted from training pairs. Generation process efficiently produces top k candidates of output strings. The method involves output string based on edit distance alignment. To derive rules, finally expand the derived rules with context. The highest probability is that a better candidate is selected out of similar candidates.

In query reformulation, it involves rewriting the original query with its similar queries and enhancing the effectiveness of search. To incorporate the pruning power of edit distances into the R-tree, the intuition is that if we store the q-grams for all strings in a sub tree rooted at an R-tree node u, denoted as gu, given a query string , extracting the query q-grams and check the size of the intersection between gu and gα, even if u does intersect with the query range r. the combined R-tree with signatures embedded in the nodes as the Min-wise signature with linear Hashing R-tree (MHR-tree). Computing edit distance exactly is a costly operation.

Several techniques have been introduced for fast identifying candidate strings from the given input string within a small edit distance. All of them are based on q-grams approach and a q-gram argument counting. For a string s, its q-grams are produced by sliding a window of length q over the characters of string s. The special case at the beginning and the end of s, that have fewer than q characters can be handled by introducing special characters, such as "#" and "$", which are not in S. This conceptually extends by prefixing q - 1 occurrences of "#" and suffixing q - 1 occurrences of "$", each q-gram for the string s has exactly q characters.

6. References

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

[2] 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 Com-putational Linguistics, 2008, pp. 1080–1089.

[3] J. Guo, G. Xu, H. Li, and X. Cheng, "A unified and discriminative model for query refinement," in Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, ser. SIGIR '08. New York, NY, USA: ACM, 2008, pp. 379–386.

[4] M. Hadjieleftheriou and C. Li, "Efficient approximate search on string collections," Proc. VLDB Endow., vol. 2, pp. 1660–1661, August 2009.

[5] C. Li, B. Wang, and X. Yang, "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, ser. VLDB '07. VLDB Endowment, 2007, pp. 303–314.

[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] Ziqi Wang, Gu Xu, Hang Li, and Ming Zhang, " A Probabilistic Approach to String Transformation" , 2013 IEEE.

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

[9] R. Vernica and C. Li, "Efficient top-k algorithms for fuzzy search in string collections," in Proceedings of the First International Work-shop on Keyword Search on Structured Data, ser. KEYS '09. New York, NY, USA: ACM, 2009, pp. 9–14.

[10] Z. Yang, J. Yu, and M. Kitsuregawa, "Fast algorithms for top-k approximate string matching," in Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, ser. AAAI '10, 2010, pp. 1467–1473.