International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

COPY RIGHT

ELSEVIER SSRN

IJIEMR Transactions, online available on 13th Sept 2019. Link :http://www.ijiemr.org/downloads.php?vol=Volume-08&issue=ISSUE-09

Title **A UNIFIED FRAMEWORK FOR STRING SIMILARITY SEARCH WITH HASH-BASED**

Volume 08, Issue 09, Pages: 589–594.

Paper Authors

P. SUSMITHA VADANA, HARI KRISHNA DEEVI

G.V.R. & S. COLLEGE OF ENGINEERING & TECHNOLOGY (APPROVED BY AICTE, NEW DELHI & AFFILIATED TO JNTUK KAKINADA) NEAR BUDAMPADU, GUNTUR-522007, A.P, INDIA

USE THIS BARCODE TO ACCESS YOUR ONLINE PAPER

To Secure Your Paper As Per UGC Guidelines We Are Providing A Electronic Bar Code

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

A UNIFIED FRAMEWORK FOR STRING SIMILARITY SEARCH WITH HASH-BASED

[1]P. SUSMITHA VADANA, [2]HARI KRISHNA DEEVI
[1]STUDENT, DEPARTMENT OF CSE, G.V.R. & S. COLLEGE OF ENGINEERING & TECHNOLOGY (APPROVED BY AICTE, NEW DELHI & AFFILIATED TO JNTUK KAKINADA) NEAR BUDAMPADU, GUNTUR-522007, A.P, INDIA.
[2]ASSOCIATE PROFESSOR DEPARTMENT OF CSE, G.V.R. & S. COLLEGE OF ENGINEERING & TECHNOLOGY (APPROVED BY AICTE, NEW DELHI & AFFILIATED TO JNTUK KAKINADA) NEAR BUDAMPADU, GUNTUR-522007, A.P, INDIA.
[1]susmithapittala@gmail.com, [2]harikrishnadeevi@gmail.com

ABSTRACT String similarity search is a fundamental query that has been widely used for DNA sequencing, error-tolerant query auto-completion, and data cleaning needed in database, data warehouse and data mining. In this paper, we study string similarity search based on edit distance that is supported by many database management systems such as Oracle and PostgreSQL. Given the edit distance, ed(s, t), between two strings, s and t, the string similarity search is to find every string t in a string database D which is similar to a query string s such that ed(s, t) for a given threshold. In the literature, most existing work take a filter-and-verify approach, where the filter step is introduced to reduce the high verification cost of two strings by utilizing an index built offline for D. The two up-to-date approaches are prefix filtering and local filtering. In this paper, we study string similarity search where strings can be either short or long. Our approach can support long strings, which are not well supported by the existing approaches due to the size of the index built and the time to build such index. We propose two new hash-based labeling techniques, named OX label and XX label, for string similarity search. We assign a hash-label, Hs, to a string s, and prune the dissimilar strings by comparing two hash-labels, Hs and Ht, for two strings s and t in the filter step. The key idea behind is to take the dissimilar bit-patterns between two hash-labels. We discuss our hash-based approaches, address their pruning power, and give the algorithms. Our hash-based approaches achieve high efficiency, and keep its index size and index construction time one order of magnitude smaller than the existing approaches in our experiment at the same time.

1. INTRODUCTION

Strings are widely used to represent a variety of textual data including DNA sequences, messages, emails, product reviews, and documents. And there are a large number of string datasets collected from various data sources in real applications. Due to the fact that string data from different sources may be inconsistent caused by the typing mistakes or the differences in data formats, as one of the most important fundamental tasks, string similarity search has been extensively

studied [1], [5], [7], [18], [26], [32], [33], [34], which checks whether two strings are similar enough for data cleaning purposes in databases, data warehousing, and data mining systems. The applications that need string similarity search include fuzzy search [11], query auto-completion [29], and DNA sequencing [13]. In the literature, there are two categories to measure string similarity. One is token-based similarity metric including overlap, Jaccard, cosine and dice [12], [31], and the other is the character-based similarity metric including edit distance [9]. The edit distance, ed(s, t), between two strings, s and t, is the minimum number of operations (substitution, insertion, and deletion) required to transform one string to another string. In this paper, we focus on string similarity search based on edit distance, which is supported in many database management systems such as Oracle, PostgreSQL and Lucene [7], and is used in detecting spammers [22] and DNA sequence alignment [17]. The challenge of the string similarity search is to design an effective index that can achieve high efficiency for query processing with small overhead in the index size and the indexing time. It becomes more challenge in the age of big data since the string datasets become increasingly large from two aspects, the lengths of the strings and the number of strings insides. The existing approaches need to build a large index when the strings are longer in a dataset. The larger the index is, the more time it requires to process queries. Accordingly, the performance decreases. In addition, such performance will be affected by a large number of strings in a dataset. In this paper, we focus on new hash-based approaches to deal with large short/long string datasets.

2. EXISTING SYSTEM:

Most of the existing string similarity search algorithms take a filter-and-verify approach. The filter step is introduced to reduce the verification cost of two strings, s and t, which is costly when two strings are long. In order to find similar strings in a string dataset D for a given query string s with a threshold, they first prune strings, t, that cannot be possibly similar with s such that ed(s, t) > using an index built offline for D in the filter step, and then verify those strings that are possibly similar one by one in the verification step. The performance of an approach is measured by the query cost and the index cost. The query cost is the sum of the filter cost (the total running time in the filter step) and the verification cost (the total running time in the verification step). The index cost is the index construction time and the index space needed.To efficiently process string similarity search, the existing work attempts to prune strings in D as many as possible based on the index built offline. Almost all the existing work needs to know the edit distance threshold beforehand, in order to construct the index for a string dataset D, except for BitTree. Behm et al. propose a hierarchical structure containing different filters, e.g., the length and charsum filter, in Flamingo package. Gravano et al. propose to partition a string into a set of q-grams and prune a string pair (s, t) that have less than a certain number of common q-grams. The chunk-based approaches share the similar idea but partition the string using disjoint q-grams, called chun. Instead of using fixed-length q-grams, Li et al.

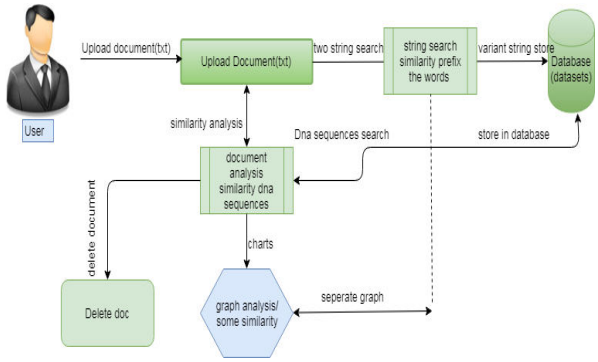
International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

selectively choose high-quality grams of variable length in index construction.

3. PROPOSED SYSTEM:

In this paper, we study string similarity search, when the query string s and the average string t in D can be long. The up-to-date approaches cannot efficiently process long string similarity search for the following main reasons. For the prefix filtering approaches, the main idea is to use a small number of q-grams for filtering. When strings become long, the pruning power of such a small number of q-grams will reduce significantly. In addition, the prefix filtering approaches need to know before the index construction. However, when the average strings become long, users want to use different for string similar search: a small for short strings and a large for long strings. It cannot be easily handled by the prefix filtering approaches. For the local filtering approach, the BitTree index will be extremely large to be stored and it is time consuming to construct such an index. Different from the existing work in the literature, we propose new hash-based labeling for string similar search. Let Hs and Ht be two hash-labels for strings, s and t. We show that s and t are definitely dissimilar for a given using Hs and Ht. We propose two hash-based approaches, namely OX label and XX label. Both are in the scheme of (~, ℵ,}, #). Here, ~ and ℵ are two functions to create a hash-label Hs for a string s, and} and # are two functions to compare two hash-labels, Hs and Ht for two strings, s and t. The key idea behind is to take the dissimilar bit-patterns between two hash-labels. We discuss our hash-based approaches, address their pruning power, and give the algorithms. New optimizations to the verification algorithm are proposed for efficiently verifying whether a candidate string is an answer. We have conducted extensive performance studies and confirm the efficiency of our hash-based approaches in both datasets of long strings and datasets of short strings with much smaller index size.

4. ARCHITECTURE:



5. ALGORITHM:

String Similarity Search: Given a string dataset D of n strings, aquery string s and an edit distance threshold _, the string similarity search problem is to find all strings t ∈D such that ed(s, t) ≤ _. A well-known algorithm to compute the edit distance between two strings s and t is to fill an edit distance matrix of size (|s| + 1) × (|t| + 1) using dynamic programming. However, it requires O(|s| · |t|) time complexity which is costly for long strings. The *filter-and-verify* framework adopted by the existing work builds an index to prune the dissimilar strings of the query
string in the dataset D in the filter step, and verifies the remaining candidates to get the real result in the verification step. The filter step is important to reduce the cost of computing the edit distance between two strings, by pruning the strings that cannot be

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

possibly in the final results as many as possible using the index built offline. There are some simple heuristics that can be applied in the filter step. The length-filter is such an example, which prunes the string t if ||s| − |t|| > _. The index built will further prune strings thatcannot be simply pruned by such simple heuristics.

$|Qs \cap Qt| \geq \max\{|s|, |t|\} + q - 1 - q\_$

6. IMPLEMENTATION

UPLOAD PRODUCT

The registered users are authorized to upload the product. The product owners have ability to change or even delete the product from the application at any point of time. The products can be viewed to other users and product owners can only access the details.

STRING SIMILARITY SEARCH

The uploaded products are listed in the users' view. There are lot of products are listed and in order to avoid congestion, the search can be available to make utilize the products in effective way. The searches have more number of details. In order to avoid the congestions searches can be utilized and give suggestion.

UPLOAD DOCUMET ANALYSIS

According to user search it shows the suggestion of product can be shown to the user. The products are shows to user according to most searches and have different types of search to get the details and better retrieval of product in order to implement and make use of the search.

GRAPH ANALYSIS

Graph analysis of details can be taken from the data which are utilized in flow of project. The graph can be utilized to showcase the products maximum retrieval by users search and how effective to user while they are searching in the system.

7. CONCLUSION:

In this paper, we study two new hash-based approaches, OX label and XX label, for string similarity search based on edit distance, where OX = (~, ∨,⊕,#) and XX = (~,⊕,⊕,#). Both OX and XX label use the same last two functions, ⊕ and #, to compare two hash-labels for pruning. But they take a different way to create the hash-labels. Here, OX label uses two functions, ~ and ∨, to create a hash-label for a string, whereas XX label uses two functions, ~ and ⊕, to create a hash-label for a sting. We prove that both OX and XX label can be used to prune dissimilar strings, s and t, when ed(s, t) > . The index size for OX label and XX label is determined by L, and the hash-label for string of any length has the same L (the number of bits). We analyze the pruning power by OX label and XX label. We show that OX label is effective when L is sufficiently large comparing to the sum of the lengths of two strings, s and t. We also show that the pruning power of XX label only depends on the number of different q-grams between the q-gram set Qs and the q-gram set Qt for s and t, and can be effectively used for both short and long string similarity pruning. We conducted extensive performance studies using 6 real string datasets.

8. FUTURE WORK:

In future, we will show that the index size and index construction time for OX label and XX label can be at least one order of magnitude smaller than the up-to-date approaches, and the hash-based approaches can significantly reduce query time. We also

analyze the impact of index length to the query performance and the improvement of the proposed verification optimizations in our experiments. Scalability study is conducted to confirm the efficiency of our approaches by increasing the number of strings in the dataset and by increasing the string length. XX label with L = 640 (bits) per hash-label is capable of handling any short/long string datasets in our testing.

REFERENCES

[1] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In Proc. of VLDB, 2006.

[2] A. Behm, R. Vernica, S. Alsubaiee, S. Ji, J. Lu, L. Jin, Y. Lu, and C. Li. UCI Flamingo Package 4.1, 2010.

[3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. Commun. ACM, 13(7), 1970.

[4] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Minwise independent permutations. In Proc. of STOC'98, 1998.

[5] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In Proc. of SIGMOD'03, 2003.

[6] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In Proc of ICDE'06, 2006.

[7] D. Deng, G. Li, and J. Feng. A pivotal prefix based filtering algorithm for string similarity search. In Proc. of SIGMOD'14, 2014.

[8] J. Feng, J. Wang, and G. Li. Trie-join: a trie-based method for efficient string similarity joins. VLDB J., 21(4):437–461, 2012.

[9] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In Proc. of VLDB'01, 2001.

[10] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In Proc. of STOC'98, 1998.

[11] S. Ji, G. Li, C. Li, and J. Feng. Efficient interactive fuzzy keyword search. In Proc. of WWW'09, 2009.

[12] Y. Jiang, G. Li, J. Feng, and W. Li. String similarity joins: An experimental evaluation. PVLDB, 7(8), 2014.

[13] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. Genome biology, 10(3), 2009.

[14] D. Lemire and O. Kaser. Recursive n-gram hashing is pairwise independent, at best. Computer Speech & Language, 24(4), 2010.

[15] C. Li, B. Wang, and X. Yang. VGRAM: improving performance of approximate queries on string collections using variable-length grams. In Proc. of VLDB'07, 2007.

[16] G. Li, D. Deng, J. Wang, and J. Feng. PASS-JOIN: A partition-based method for similarity joins. PVLDB, 5(3), 2011.

[17] Y. Li, A. Terrell, and J. M. Patel. WHAM: a high-throughput sequence alignment method. In Proc. of SIGMOD'11, 2011.

[18] W. Lu, X. Du, M. Hadjieleftheriou, and B. C. Ooi. Efficiently supporting edit distance based string similarity search using B+-trees. TKDE, 26(12), 2014.

[19] J. McAuley, R. Pandey, and J. Leskovec. Inferring networks of

substitutable and complementary products. In Proceedings of SIGKDD, 2015.

[20] M. Mitzenmacher, R. Pagh, and N. Pham. Efficient estimation for high similarities using odd sketches. In Proc. of WWW'14, 2014.

[21] M. Mitzenmacher and E. Upfal. Probability and computing: Randomized algorithms and probabilistic analysis. Cambridge University Press, 2005.

[22] V. M. Prieto, M. ´Alvarez, and F. Cacheda. Detecting linkedin spammers and its spam nets. IJACSA, 4(9), 2013.

[23] J. Qin, W. Wang, Y. Lu, C. Xiao, and X. Lin. Efficient exact edit similarity query processing with the asymmetric signature scheme. In Proc. of SIGMOD'11, 2011.

[24] S. Wandelt, D. Deng, S. Gerdjikov, S. Mishra, P. Mitankin, M. Patil, E. Siragusa, A. Tiskin, W. Wang, J. Wang, and U. Leser. State-of-the-art in string similarity search and join. SIGMOD Rec., 2014.

[25] J. Wang, G. Li, and J. Feng. Trie-join: Efficient trie-based string similarity joins with edit-distance constraints. PVLDB, 3(1), 2010.

[26] J.Wang, G. Li, and J. Feng. Can we beat the prefix filtering?: an adaptive framework for similarity join and search. In Proc. of SIGMOD'12, 2012.

[27] P. Wang, C. Xiao, J. Qin, W. Wang, X. Zhang, and Y. Ishikawa. Local similarity search for unstructured text. In Proc. of SIGMOD'16, 2016.

[28] W. Wang, J. Qin, C. Xiao, X. Lin, and H. T. Shen. Vchunkjoin: An efficient algorithm for edit similarity joins. TKDE, 25(8), 2013.

[29] C. Xiao, J. Qin, W. Wang, Y. Ishikawa, K. Tsuda, and K. Sadakane. Efficient error-tolerant query autocompletion. PVLDB, 2013.

[30] C. Xiao, W. Wang, and X. Lin. Ed-join: an efficient algorithm for similarity joins with edit distance constraints. PVLDB, 1(1), 2008.

[31] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang. Efficient similarity joins for near-duplicate detection. TODS, 36(3), 2011.

[32] X. Yang, B. Wang, and C. Li. Cost-based variable-length-gram selection for string collections to support approximate queries efficiently. In Proc. of SIGMOD'08, 2008.

[33] X. Yang, Y. Wang, B. Wang, and W. Wang. Local filtering: Improving the performance of approximate queries on string collections. In Proc. Of SIGMOD'15, 2015.

[34] Z. Zhang, M. Hadjieleftheriou, B. C. Ooi, and D. Srivastava. Bed-tree: an all-purpose index structure for string similarity search based on edit distance. In Proc. of SIGMOD'10, 2010.