International Journal of Advanced Technology in Engineering and Science
Vol. No.4, Issue No. 03, March 2016
www.ijates.com
ijates
ISSN 2348 - 7550

Q-GRAM BASED JOIN FOR STRING SIMILARITY SEARCH

Jeenath Bee[1], Pajany[2]

[2]Associate Professor, [1]Department of Computer Science and Engineering,
IFET college of Engineering, Villupuram

ABSTRACT

Q-gram based Similarity join and FP-Growth algorithm are combined. The Q-gram based similarity join matches both the substrings and the whole one. FP Growth Stands for frequent pattern growth. It divides a known join string predicate into short substrings of length q, called Q-grams and creates the auxiliary table The outcome from this approach supports join string predicates like "name similar to Campbell" with an accepted error rate of ε. This project proposes a general gram filter which is a novel generalization of existing filters. A choose-and-extend framework to efficiently find the high quality grams in the query process is presented.

Index terms: gram filter, string similarity, frequent pattern.

I. INTRODUCTION

Auto completion guides users to type the query correctly and efficiently. Due to the convenience it brings to the users and the server, it has been adopted in many applications [2]. For example, search engines like Google dynamically suggest keywords, and can optionally show top-ranked search results while user is typing a query. Other applications include command shells, desktop search, software development environments (IDE), and mobile applications. **String** similarity search takes as input a set of strings and a query string, and outputs all the strings in the set that are similar to the query string. It is an important operation and has many real applications such as data cleaning, information retrieval, and bio informatics [1]. In this paper q-gram based similarity search are used to find the String similarity. Q-gram based similarity join and FP Growth algorithm are combined. The Q-gram based similarity join matches both the substrings and the whole one. FP Growth Stands for frequent pattern growth. It divides a known join string predicate into short substrings of length q, called Q-grams and creates the auxiliary table. The outcome from this approach supports join string predicates like "name similar to Campbell" with an accepted error rate of ε. A choose-and-extend framework to efficiently find the high quality grams in the query process is presented. There are many similarity functions to quantify the similarity of two strings, such as Jaccard similarity, Cosine similarity, and edit distance. We focus on edit distance of two strings is the minimum number of single-character edit operations (i.e. insertion, deletion, and substitution) needed to transform one string to another string. Large enterprises are increasingly relying on web services as methodology of sharing services within the organization. The growing number of web services available raises a new and challenging search problem. When the user finds out that the web service operation is inappropriate for some reason, would like to find similar operations. Existing literatures usually adopt a gram-based filter and verification framework for string similarity search. After the query string is decomposed into grams, an efficient

and critical type of filter can utilize the grams and inverted index (from grams to strings) of the string collection to generate candidates. Traditional prefix filters focus on minimizing the filtering cost, whereas counter filter aims at reducing the verification cost. In our Project the Gram Filter. It consists of two steps: (a) choosing high quality grams as the base query-gram set; (b) extending the prefix by choosing more grams until no benefit can be gained from the new gram. A series of heuristics are proposed to achieve efficiency and effectiveness of our algorithms. This process is in terms of categorization accuracy or representation efficiency but it might be hard to identify good set of vocabularies. Proved to be NP-hard problem, we give a cost model to measure the filter ability of grams and develop efficient heuristic algorithms to find high-quality grams.

II. RELATED WORKS

2.1 Similarity Join Size Estimation using Locality Sensitive hashing

Similarity joins are important operations with a broad range of applications. In this paper, we study the problem of vector similarity join size estimation (VSJ). It is a generalization of the previously studied set similarity join size estimation (SSJ) problem and can handle more interesting case such as TF-IDF vectors. One of the key challenges in similarity join size estimation is that the join size can change dramatically depending on the input similarity threshold .We propose a sampling based algorithm that uses Locality Sensitive-Hashing (LSH). The proposed algorithm LSH-SS uses an LSH index to enable effective sampling even at high thresholds. We compare the proposed technique with random sampling and the state-of-the-art technique for SSJ (adapted to VSJ) and demonstrate LSH-SS offers more accurate estimates throughout the similarity threshold range and small variance using real-world data sets.

2.2 Approximate String Similarity Join using Hashing Techniques under Edit Distance Constraints.

The string similarity join, which is employed to find similar string pairs from string sets, has received extensive attention in database and information retrieval fields. To this problem, the filter-and-refine framework is usually adopted by the existing research work firstly, and then various filtering methods have been proposed. Recently, tree based index techniques with the edit distance constraint are effectively employed for evaluating the string similarity join. However, they do not scale well with large distance threshold. In this paper, we propose an efficient framework for approximate string similarity join based on Min-Hashing locality sensitive hashing and tree-based index techniques under string edit distance constraints.

2.3 An efficient tree-based string similarity join algorithm

A string similarity join finds all similar pairs between two collections of strings. It is an essential operation in many applications, such as data integration and cleaning, and has attracted significant attention recently. In this paper, we study string similarity joins with edit-distance constraints. Recently, a Tree-based similarity Join framework is proposed. Existing Tree-based Join algorithms have shown that Tree Indexing is more suitable for Similarity Join on short strings. The main problem with current approaches is that they generate and maintain lots of candidate prefixes called active nodes which need to be further removed. With large edit distance, the number of active nodes becomes quite large. In this paper, we propose a new Tree-based Join algorithm called

Pre Join, which improves over current Tree based Join methods. It efficiently finds all similar string pairs using a new active-node set generation method, and a dynamic preorder traversal of the Tree index.

2.4 PASS-JOIN: a partition-based method for similarity joins

As an essential operation in data cleaning, the similarity join has attracted considerable attention from the database community. In this paper, we study string similarity joins with edit-distance constraints, which find similar string pairs from two large sets of strings whose edit distance is within a given threshold. Existing algorithms are efficient either for short strings or for long strings, and there is no algorithm that can efficiently and adaptively support both short strings and long strings. To address this problem, we propose a partition-based method called Pass-Join. Pass-Join partitions a string into a set of segments and creates inverted indices for the segments. Then for each string, Pass-Join selects some of its substrings and uses the selected substrings to find candidate pairs using the inverted indices. We devise efficient techniques to select the substrings and prove that our method can minimize the number of selected substrings

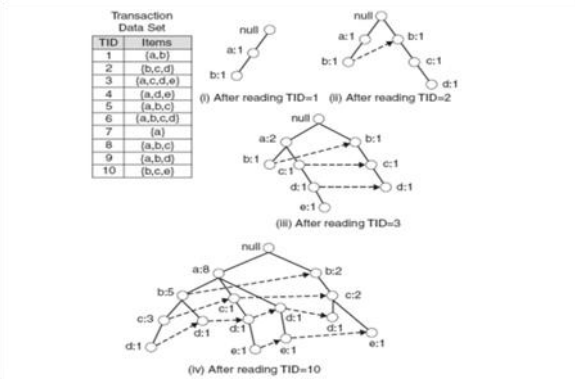
III. PROPOSED SYSTEM

Q-gram based Similarity join and FP-Growth algorithm are combined. The Q-gram based similarity join matches both the substrings and the whole one. FP Growth Stands for frequent pattern growth. It divides a known join string predicate into short substrings of length q, called Q-grams and creates the auxiliary table to store the information about the grams. The outcome from this approach supports join string predicates like “name similar to Campbell” with an accepted error rate of ε. This project proposes a general gram filter which is a novel generalization of existing filters. A choose-and-extend framework to efficiently find the high quality grams in the query process is presented. The advantage of the proposed system is Existing literatures usually adopt a gram-based filter and verification framework for string similarity search Modern search engines often apply similarity search to suggest relevant query keywords to fix the typos in user queries. Traditional prefix filters focus on minimizing the filtering cost, whereas counter filter aims at reducing verification cost.

3.1 Algorithm

In this project Q-gram based Similarity join and FP-Growth algorithm are combined. The Q-gram based similarity join matches both the substrings and the whole one. FP Growth Stands for frequent pattern growth. It divides a known join string predicate into short substrings of length q, called Q-grams and creates the auxiliary table are used to store the information. Fp -tree has been constructed with the use of prefix based dataset from inverted index. FP-Growth about grams with extracts frequent item sets from the FP-tree. In this calculation cost will be low and it will be effective for large volume of data set Fp -tree has been constructed with the use of prefix based dataset from inverted index. FP-Growth simply a two-step procedure

- ✓ Step 1: Build a compact data structure called the FP-tree Built using 2 passes over the data-set.
- ✓ Step 2: Extracts frequent item sets directly from the FP-tree



Advantages of FP-Growth is only 2 passes over data-set "compresses" data-set no candidate generation much faster than Apriori. Disadvantages of FP-Growth FP-Tree may not fit in memory. FP-Tree is expensive to build.

3.2 1Data Set

The data Sets, a model is built based on the characteristics of each category in a pre-classified set of data. The data set should be selected in such a way that it is varying in content and subject. The pre-processing is first performed to extract prefix string and determine the number of occurrences of each words in each category.

3.2.2 Inverted Index

An inverted index is built on the string collection, which takes all the grams as the keys and the strings that contain the gram as the values. Prefix filtering technique is commonly used in the refinement step to further prune false positives of candidate pairs that share a certain number of common tokens. Candidate pairs with no overlap in their corresponding prefixes can be safely pruned. In this manner, the number of candidate pairs can be significantly reduced since popular words or frequent tokens can be set to the end of the global ordering so that they are not probable of lying in the prefixes.

3.2.3 Generation process

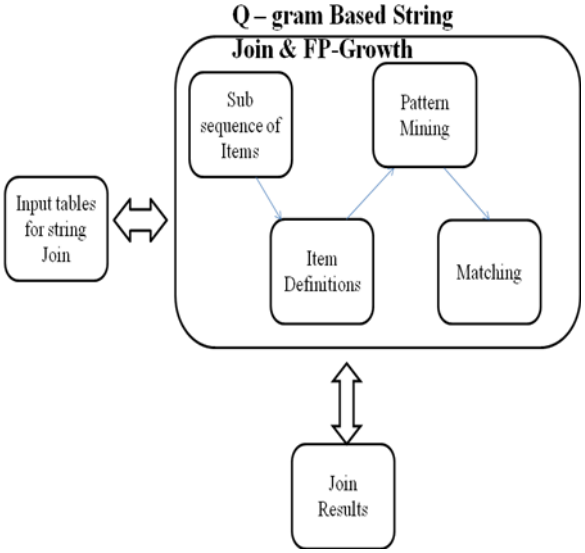
This step probes the inverted index by using the grams in the query string and in the meantime counts the occurrences of the strings in the chosen lists.A string will be a candidate if its occurrences are above a lower bound. Some filter technologies such as length filter can be used in this step to reduce the number of candidates.

3.2.4 Verification process

The actual distance between the query and candidates is calculated and a candidate will be added to the final results, if the distance is no larger than the given threshold t.

IV. SYSTEM ARCHITECTURE

In this q- gram based join the input tables for string can be scanned and it is given as input to the string. It can be splitted into number of sub sequence of items and pattern matching is used to match the correct pattern from the string.The item defintion is used to identity the items in the FP growth algorithm.



REFERENCES

[1] 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. 27th Int. Conf. Very Large Data Bases, 2001, pp. 491–500.

[2] M. Hadjieleftheriou, A. Chandel, N. Koudas, and D. Srivastava, "Fast indexes and algorithms for set similarity selection queries," in Proc. 24th Int. Conf. Data Eng., 2008, pp. 267–276.

[3] H. Lee, R. T. Ng, and K. Shim, "Similarity join size estimation using locality sensitive hashing," in Proc. Int. Conf. Very Large Data Bases, 2011, pp. 338–349.

[4] C. Xiao, W. Wang, and Xuemin Lin, "Ed-join: An efficient algorithm for similarity joins with edit distance constraints," in Proc.Int. Conf. Very Large Data Bases, 2008, pp. 933–944.

[5] S. Chaudhuri, V. Ganti, and R. Kaushik, "A primitive operator for **similarity joins in data cleaning," in Proc. 22nd Int. Conf. Data Eng.,**2006, pp. 795–825.

[6] J. Qin, W. Wang, Y. Lu, C. Xiao, and X. Lin, "Efficient exact edit similarity query processing with the asymmetric signaturescheme," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2011, pp. 1033–1044.

[7] J. Wang, G. Li, and J. Feng, "Can we beat the prefix filtering? An adaptive framework for similarity join and search," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2012, pp. 85–96.

[8] C. Li, B. Wang, and X. Yang, "VGRAM: Improving performance of approximate queries on string collections using variable length grams," in Proc. Int. Conf. Very Large Data Bases, 2007, pp. 303–314.

[9] X. Yang, B. Wang, and C. Li, "Cost-based variable-length-gram selection for string collections to support approximate queries efficiently," in Proc. Int. Conf. Very Large Data Bases, 2008, pp. 353–364.

[10] C. Xiao, W. Wang, X. Lin, and J. X. Yu, "Efficient similarity joins for near duplicate detection," in Proc. 17th Int. Conf. World Wide Web, 2008, pp. 131–140.

[11] A. Z. Broder, "On the resemblance and containment of documents," in Proc. Compression Complexity Sequences, 1997, pp. 21–29 G. Navarro, "A guided tour to approximate string matching," ACM Comput. Surveys, vol. 33, no. 1, pp. 31–88, 2001.

[13] R . J. Bayardo, Y. Ma, and R. Srikant,, "Scaling up all pairs similarity search," in Proc. Int. Conf. World Wide Web, 2007, pp. 131–140.

[14] Z. Zhang, B. C. Ooi, S. Parthasarathy, and A. K. H. Tung, "Similarity search on Bregman divergence: Towards non-metric indexing," VLDB Endowment, vol. 2, no. 1, pp. 13–24, 2009.

[15] J. Xu, Z. Zhang, A. K. H. Tung, and G. Yu, "Efficient and effective similarity search over probabilistic data based on earth mover's distance," VLDB Endowment, vol. 3, no. 1, pp. 758–769, 2010.

[16] A. Arasu, V. Ganti, and R. Kaushik, "Efficient exact set-similarity joins," in Proc. Int. Conf. Very Large Data Bases, 2006, pp. 918–929.

[17] 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. ACM SIGMOD Int. Conf. Manage. Data, 2010, pp. 915–926.

[18] S. Schleimer, D. S. Wilkerson, and A. Aiken, "Winnowing: Local algorithms for document fingerprinting," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2003, pp. 76–85.

[19] W. Wang, J. Qin, C. Xiao, X. Lin, and H. T. Shen, "VChunk Join: An efficient algorithm for edit similarity joins," IEEE Trans. Knowl. Data Eng., vol. 25, no. 8, pp. 1916–1929, Aug. 2012.

[20] T. Bocek and B. Stiller, "Fast similarity search in large dictionaries," Dept. Inf., Univ. Zurich, Zurich, Switzerland, Tech. Rep. ifi-2007.02, 2007.

[21] W. Wang, C. Xiao, X. Lin, and C. Zhang, "Efficient approximate entity extraction with edit constraints," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2009, pp. 759–770.

[22] S. Chaudhuri and R. Kaushik, "Extending auto completion to tolerate errors," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2009,