ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (815-820), Month: April - June 2015, Available at: www.researchpublish.com

Efficient Query Processing System on Web Search Engine

[1]K.Bhuvaneswari, [2]G.Sudhakhar

[1]PG Scholar, Department of Computer Science and Engineering, Ranganathan Engineering College, Coimbatore
[2]Head of the Department, Department of Computer Science and Engineering, Ranganathan Engineering College, Coimbatore

Abstract: **a string similarity measure or similarity function calculation is a main operation in web search engine. The operation of string metric calculation is the main part that can be used to identify similarity between two text strings. For appropriate string similarity calculation, there are various kinds of technologies has been introduced. In recent years, the efficient string similarity measure operations were done with the processing concept of inverted indexes. The filter and refine techniques is the main operation for identifying and verifying similar strings in the given data collection. For large volume of data, the computation cost will be high. A prefix based inverted index has been calculated for relevant query extraction. Based upon the prefix value, the approach of multiple prefix filtering can be used to produce most similar string. In this paper, we propose a Q-gram based edit distance metric for efficient string similarity operation. For approximate string matching, Q-gram distance is a very effective and widely used distance metric. And FP-growth algorithm can be used for user query recommendation which makes effective query processing system.**

Keywords: **prefix filtering, Q-gram edit distance metric, FP-growth algorithm, information retrieval.**

1. INTRODUCTION

In Search engine, the string similarity measure is the basic operation for producing relevant results. There has been lot of approaches was introduced for efficient similarity calculation. Most of the approaches either suffer from high computation cost or poor filtering. The similar strings are produced if it matches with some common tokens. Then the relevant strings are grouped together and filtered again for exact similarity measure. These two processing scheme are done with the step of filter and refine processes.

The inverted index is the one which supports efficient string similarity joins. It works based on filter and refine processes. In this, it first generates candidate pairs based on their signature value and then verifies all the generated candidate pair by computing their similarity. A Prefix based inverted index has been calculated in which some selected words of all the strings are extracted as signatures. Based upon the signature values the given query is processed and grouped together for the next step of refining process. By applying Q-gram edit distance metric, the filtered data are further pruned for relevant queries. This can reduces different number of multiple prefix filtering. So, the processing time will be reduced for large volume of data.

The filtered candidate pair is refined for exact similarity joins processing. The Q-gram edit distance metric is defined with three different properties. By applying these properties on prefix based inverted index, the exact similar candidate pairs will be calculated. The FP growth algorithm can be used for query recommendation. For the recommendation process it generates FP tree which works using prefix valued data. So the calculated prefix based inverted index can be used for defining FP tree. It shows that the processing time will be reduced and this can work well on large volume of data.

2. RELATED WORK

Prefix filtering is one approach in string similarity join that commonly applied in refinement step to further prune false positives of candidate pairs. In [1], multiple prefix filtering approach has been introduced for string similarity join

processing. This can be applied on multiple global ordering, so that the generation of number of candidate pair will be reduced. For each set of data collection, single global ordering has been done based upon prefix filtering. For large of volume of data collection, the outcome of candidate pairs will be high. That will require another set of operation to prune false positive results. By applying multiple global ordering on the calculated results of single global ordered data, the similar candidate pairs will be produced. This was done by using multiple prefix filtering. But for large volume of data, the generation cost will be high. The multiple prefix filtering based on multiple global ordering; perform in the basis of pipelining manner. The pipeline process will be continued until it prunes false positive results. And this will increase the threshold value. And this approach require additional framework to refine the similar string pairs. The exact string similarity measure is the main operation for search engine query processing system. In [2], The notion of similarity is captured numerically using a string-based similarity function, and two records are considered similar if the value returned by the similarity function for these two records is greater than a threshold. The performance of our algorithm is comparable to that of LSH-based approximate algorithms for many scenarios, especially the data cleaning ones we are most interested in, and in many cases they even outperform LSH-based algorithms. One important feature of our algorithms is that for SS-Joins involving jaccard and hamming, they provide a guarantee that two highly dissimilar sets will not appear as a candidate pair with a high probability. One drawback of the LSH-based algorithms is that they can be used only when the SS-Join predicate admits a locality-sensitive hash function; currently, locality-sensitive hash functions are known only for standard similarity measures such as jaccard.

De-duplication is the process of identifying references [4] in data records that refer to the same real-world entity. It is a crucial step in the data cleaning process. Collective de-duplication is a generalization in which one wants to find types of real-world entities in a set of records that are related. We present a declarative framework for collective de-duplication of entity references in the presence of constraints. Constraints occur naturally in many data cleaning domains and can improve the quality of de-duplication. We have proposed a novel language, Dedupalog, to specify de-duplication programs that can be run at large scale. They have validated its practical utility on two datasets, Cora and ACM data. The main algorithmic contribution is an efficient, scalable algorithm for the entire Dedupalog language. Creating such an algorithm is a very challenging problem because clustering optimally is *NP*-hard in many setting. In fact, for some natural variants of the clustering problem it is believed that no algorithm can even give a bounded approximation guarantee. Their algorithms have precise theoretical guarantees for a large subclass of our framework. They show, using a prototype implementation that our algorithms scale to very large datasets. They provide thorough experimental results over real-world data demonstrating the utility of our framework for high-quality and scalable de-duplication.

Search engines often suggest alternate query formulations. One approach to generating such suggestions is to find all pairs [3] of similar queries based on the similarity of the search results for those queries. Since the goal is to offer only high quality suggestions, we only need to find pairs of queries whose similarity score is above a threshold. Near duplicate detection is made possible through similarity search with a very high similarity threshold. Recent work has applied algorithms for finding all similar pairs within an application for identifying coalitions of click fraudsters. We propose a simple algorithm based on novel indexing and optimization strategies that solve this problem without relying on approximation methods or extensive parameter tuning. We show the approach efficiently handles a variety of datasets across a wide setting of similarity thresholds, with large speedups over previous state-of-the-art approaches. This approach efficiently handles a variety of datasets across a wide setting of similarity thresholds, with large speedups over previous state-of-the-art approaches. It aggressively exploited these insights to produce an algorithm, All-Pairs, that is easy to implement and does not require any parameter tuning. Need algorithm for the problem can serve as useful primitives for other mining tasks. Need to focus, exploiting all similar pairs for improving the quality of heuristic clustering approaches, performing deeper social network analysis, or in improving performance of related problems.

Auto completion is a ubiquitous feature [5] found to be useful in several environments. As the user types, a list of appropriate completions is returned by such a feature. The goal is not only to reduce typing effort but also to help guide the user's typing. Auto completion is a useful feature when a user is doing a look up from a table of records. We considered the problem of performing auto completion when a record is being looked up against a database table. We show that a naive approach of invoking an offline edit distance matching algorithm at each step performs poorly and present more efficient algorithms. It indicates both the utility of error-tolerant auto completion and the fact that we can perform error-tolerant auto completion with performance tradeoffs similar to the case of exact auto completion. The issue of performing error-tolerant auto completion for other similarity functions is not to be addressed. One way of viewing

auto completion is as an online method of performing exact matching, since in the absence of auto-completion, a user would have to type out the string in its entirety and then match it against the table of records. In contrast to the online auto completion process, we call the latter an offline lookup. In an offline lookup setting, the data cleaning community has long recognized the need to go beyond exact matching. This is because the input string being typed can contain errors and differences in representation making exact matching inadequate.

Data cleaning is an essential step in populating and maintaining data warehouses and centralized data repositories. A very important data cleaning operation is that of "joining" similar data. Data cleaning based on similarities involves identification of "close" tuples, where closeness is evaluated using a variety of similarity functions chosen to suit the domain and application. A new primitive operator which can be used as a foundation [12], to implement similarity joins according to a variety of popular string similarity functions, and notions of similarity which go beyond textual similarity. They introduce a primitive operator SS-Join for performing similarity joins. They showed that similarity joins based on a variety of textual and non-textual similarity functions can be efficiently implemented using the SS-Join operator. They then developed very efficient physical implementations for this operator mostly using standard SQL operators. It shows that similarity joins based on a variety of textual and non-textual similarity functions can be efficiently implemented using the SS-Join operator. It is very efficient physical implementations for this operator mostly using standard SQL operators. Need to integrate the SS-Join operator with the query optimizer in order to make cost-conscious choices among the basic, prefix-filtered, and inline prefix-filtered implementations.

Rank aggregation [8], plays an important role in our daily lives. Ordered lists are ubiquitous and we, consciously or unconsciously, attempt to make sense of them. The task of ranking a list of several alternatives based on one or more criteria is encountered in many situations. One of the underlying goals of this endeavor is to identify the best alternatives, either to simply declare them to be the best (e.g., in sports) or to employ them for some purpose. When there is just a single criterion (or \judge") for ranking, the task is relatively easy, and is simply a reflection of the judge's opinions and biases. Besides the heuristics, we identify a crucial property of Kemeny optimal solutions that is particularly useful in combatting spam, and provide an efficient algorithm for minimally modifying any initial aggregation so as to enjoy this property. This property is called the \extended Condorcet criterion," and we call the efficient process that is guaranteed to achieve it \local Kemenization." The algorithms for initial aggregation are based on two broad principles. The first principle is to achieve optimality not with respect to the Kemeny guidelines, but with respect to a different, closely related, measure, for which it is possible to find an efficient solution. The second principle is through the use of Markov chains as a means of combining partial comparison information | derived from the individual rankings | into a total ordering.

3. STRING SIMILARITY JOIN

In search engine, the query processing system plays an important role. The efficient and effectiveness of query processing will be determined based on string similarity. The multiple prefix filtering is one of the efficient approaches for exact string similarity measure. The inverted index is used to support multiple prefix filtering. The candidate pairs are generated using the pipelining manner of multiple prefix filtering in multiple global ordering. When the user enters a query to search, it first starts with inverted index that will compare the string with different set of global ordering. A multiple prefix filtering will be applied for the similarity search of the given query.

The similar strings are obtained by using the following criteria:

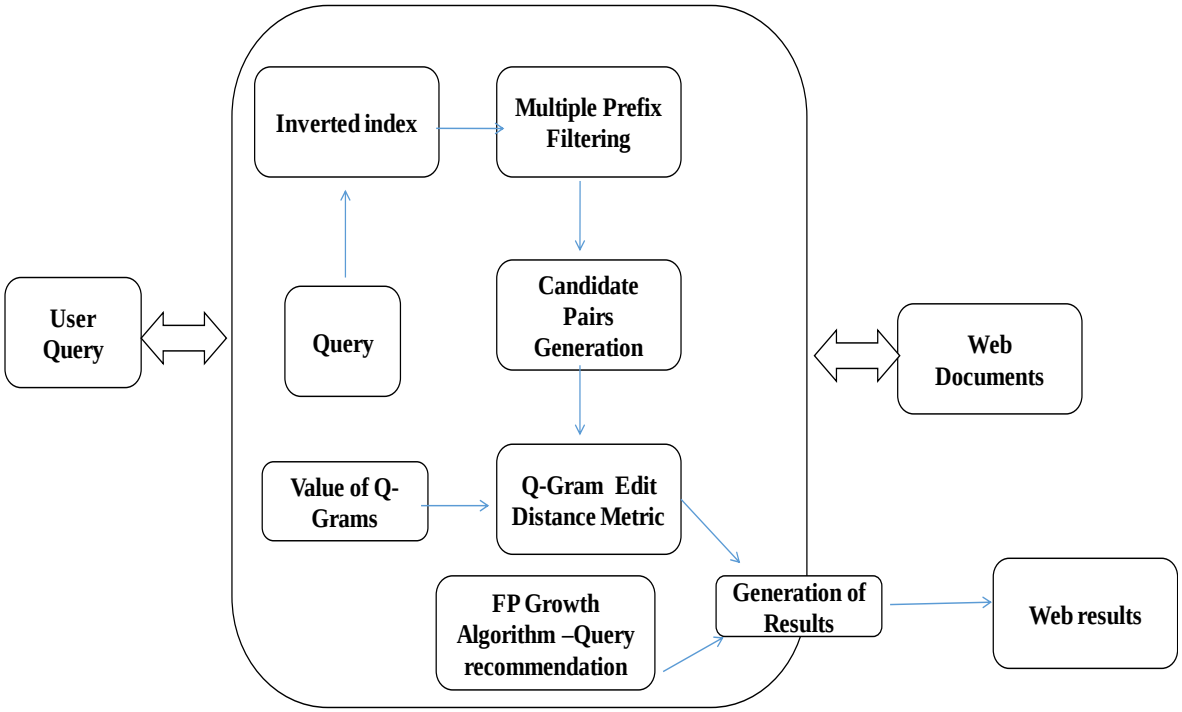
- Inverted Index
- Multiple Prefix Filtering
- Q-Gram Edit Distance Metric

3.1 Inverted Index:

The inverted index is defined with prefix values of associated query or string. Once the user has enters the query, it will be compared with data's in inverted index. The inverted index is defined with different data sets. Based on user query, the similar strings are grouped together for further processing. The inverted index is the best support for multiple prefix filtering on multiple global ordering.

3.2 Multiple Prefix Filtering:

For large volume of dataset the number of candidate pairs will be high. This will be reduced by applying multiple global ordering on different dataset. Each data set should be processed with prefix filtering of given query. To identify all similar string pairs in a collection, the inverted index-based methods follow a filter-and-refine process that first generates candidate string pairs based on their signatures, and then verifies each candidate pair by computing their similarity. By applying different global orderings to sort the words of strings in a given collection, we can derive multiple sets of candidate pairs. The string pairs whose similarity is above the predefined threshold always qualify as candidate pairs as they must share at least one common prefix token under any global ordering.



3.3 Q-Gram Edit Distance Metric:

The multiple prefix filtering method can be used to produce more relevant data's that are matched with the given query. When the size of the data set will be increase, the verification process time will be quite high. This can be reduced by applying Q-Gram edit distance metric. This calculates the similarity based upon the value of Q-Grams. The Q-gram based similarity join, which matches both the substrings and the whole one. Q-gram distance is a very effective and widely used distance metric for approximate string matching. The outcome from this approach supports join string predicates like "name similar to Campbell" with an accepted error rate of ε. The Q-gram approach was derived from N-gram distance metric. An N-gram is defined as a subsequence of n items from a given sequence. The items in the definition can be phonemes, syllables, letters, words or base pairs according to the application. N-gram models are widely used in statistical natural language processing. It can also be used for efficient approximate matching by converting a sequence of items to a set of N-grams. However, the set of N-grams make the approach lose information about the strings. A positional Q-gram was born to compensate some weaknesses of N-grams. This occurs when they share a large number of Q-grams in common.

3.3.1. Properties of Q-gram Edit Distance Metric:

- Count Filtering
- Position Filtering
- Length Filtering

Count Filtering:

Count filtering is an important property in improving the efficiency of approximate string join. The intuition of count filtering is small edit distance of each other shares large number of Q-gram in common.

Position Filtering:

The count filtering performs its operation without considering positions of a string. The position of a string may have various numbers of combination and position. The position filtering restricts in certain k value position to avoid mismatching of Q-gram. That is, it counts the difference value of the same Q-gram from two different sets of data if the calculated difference is not more than k position.

Length Filtering:

If the strings are not within the desired edit distance, the length filtering quickly removes for further processing. This will improve the processing time.

4. QUERY RECOMMENDTION

The process of finding frequent patterns is an important and cost effective approach if the database is large. The FP-growth algorithm is one of the efficient and scalable methods for frequent pattern analysis. It performs by using prefix based tree structure for storing information about frequent patterns named FP Tree. The FP-tree contains item set related information with the concept of divide and conquer strategy. The construction of FP-tree start with root as null and the children's of the root node will be defined based on prefix value of each item set. For large volume of dataset, it's not possible to hold the values of FP-tree. The projected database i.e. the database will be divided into smaller databases. While processing, the FP-growth algorithm does not require constructing of candidate pair. This will increase the processing time.

5. CONCLUSION AND FUTURE WORK

We present an approach for efficient string similarity measure, which performs well on multiple global ordering. The Q-gram edit distance metric plays a vital role to produce similar strings and to prune false positive values. The FP-growth algorithm for query recommendation is an efficient technique. While processing, the FP-growth algorithm does not require constructing of candidate pair. This will increase the processing time. The FP-tree has been constructed based on prefix based data. We have already defined prefix based inverted index for similar join measure. So, the construction coast and time for FP-tree will be low and efficient for query processing. In future work, further processing of based on user profile personalization and location based search result will be carried out.

REFERENCES

[1] Chuitian Rong, Wei Lu, Xiaoli Wang, Xiaoyong Du, Yueguo Chen, and Anthony K.H. Tung "Efficient and scalable processing of string similarity join". 2013.

[2] A. Arasu, V. Ganti, and R. Kaushik, "Efficient Exact Set-SimilarityJoins," Proc. 32nd Int'l Conf. Very Large Data Bases, pp. 918-929,2006.

[3] R. Bayardo, Y. Ma, and R. Srikant, "Scaling up All Pairs SimilaritySearch," Proc. Int'l Conf. World Wide Web, pp. 131-140, 2007.

[4] A. Arasu, C. Re´, and D. Suciu, "Large-Scale Deduplication withConstraints Using Dedupalog," Proc. Int'l Conf. Data Eng. (ICDE),pp. 952-963, 2009.

[5] S. Chaudhuri and R. Kaushik, "Extending Autocompletion toTolerate Errors," Proc. ACM SIGMOD Int'l Conf. Management of data pp. 707-718, 2009.

[6] Geoffrey C. Fox, "Unified Data Access/query over Integrated Data-views for Decision Making in Geographic Information Systems" , in AAAI/IAAI, vol. 1, 1996, pp. 40{47.

[7] X. Dong, A. Halevy, and J. Madhavan, "Reference Reconciliationin Complex Information Spaces," Proc. ACM SIGMOD Int'l Conf.Management of Data, pp. 85-96, 2005.

[8] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "RankAggregation Methods for the Web," Proc. Int'l Conf. World WideWeb, pp. 613-622, 2001.

[9] A. Elmagarmid, P. Ipeirotis, and V. Verykios, "Duplicate RecordDetection: A Survey," IEEE Trans. Knowledge and Data Eng., vol. 19,no. 1, pp. 1-16, Jan. 2007.

[10] S. Chien, C. Dwork, R. Kumar, D.R. Simon, and D. Sivakumar, "Link Evolution: Analysis and Algorithms," Internet Math., vol. 1, no. 3, pp. 277-304, 2003.

[11] Y. Wu and L. Raschid, "Approxrank: Estimating Rank for a Subgraph," Proc. IEEE Int'l Conf. Data Eng. (ICDE '09), pp. 54-65, 2009.

[12] S. Chaudhuri, V. Ganti, and R. Kaushik, "A Primitive Operator forSimilarity Joins in Data Cleaning," Proc. Int'l Conf. Data Eng.(ICDE), pp. 61-72, 2006.

[13] R. Lempel and S. Moran, "Rank-Stability and Rank-Similarity of Link-Based Web Ranking Algorithms in Authority-Connected Graphs," Information Retrieval, vol. 8, no. 2, pp. 245-264, 2005.