Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 12, December 2013, pg.64 – 68

RESEARCH ARTICLE

A STUDY ON HYBRID APPROACH TO AVOID REDUNDANCY IN APPROXIMATE MEMBERSHIP LOCALIZATION

Farzhana.I[1], Akila Rani.M[2]

PG Student[1], Assistant Professor[2], Department of CSE
NPR college of Engineering and Technology, TamilNadu, India
Email: farzu27@gmail.com; akilakamalam@gmail.com

Abstract-Dictionary-based entity extraction identifies predefined entities from a document. A recent trend for improving extraction recall is to support approximate entity extraction, which finds all substrings in the document that approximately match entities in a given dictionary but this causes redundancy and lower its performance. To improve the performance of string matching from a document a technique called Approximate Membership Localization is used. This technique aims at locating non overlapped substring which eliminates redundancy and improves performance, efficiency of searching process. This survey paper provides an overview of a string matching process and their accuracy. The objective is to provide accurate matching of string in the search process.

Index Terms- Approximate Membership Localization (AML); Approximate Membership Extraction (AME)

I. INTRODUCTION

Data mining automates the detection of relevant patterns in a database, using defined approaches and algorithms to look into current and historical data that can then be analyzed to predict future trends. Because data mining tools predict future trends and behaviors by reading through databases for hidden patterns, they allow organizations to make proactive, knowledge-driven decisions and answer questions that were previously too time-consuming to resolve.

Given an input text string consisting of a sequence of tokens, the task of approximate membership checking (or, approximate dictionary lookup) is to identify all sub-strings that approximately match with some string from a potentially large dictionary. This problem has many applications. Most comparison shopping sites (e.g., MSN shopping) have backend databases with product dictionaries. It is important for them to identify product mentions in documents.

Documents are identified and stored by means of key words or index words in typical information search. User's diversified search request has resulted in various search technologies, which have developed into exemplary Boolean search and similarity search, e.g., vector space models. As a result of Internet information search, for example, Google, information search is

generalized, which uses ancillary information, for example, similarity search and link relationship on the basis of weight information.

String matching can be understood as the problem of finding a pattern with a property within a given sequence of symbols. Its application can be used in many fields, such as bioinformatics and computer science. This paper focuses on a string matching technique for computer security, especially exact multi-pattern matching for intrusion-detection systems.

Given a text string of length *n*, a pattern string of length *m*, and a distance *k*, the approximate string matching problem is to find all substrings of the text that are within a distance *k* of the pattern.The problem of string matching is that there are two strings one is text T [1…..n] i.e. is main string given and the other is pattern P [1…….m] i.e. is the given string to be matched with the given main string given m<=n. String matching is in variably used in real word applications like Database schema, Network systems. There are main 2 techniques of string matching one is exact matching Various string matching algorithms are used to solve the given above problems like wide window pattern matching, approximate string matching.

The approximate matching string is reviewed with the aim of surveying technique suitable for finding an item in a database when there may be spelling mistake or other error in the keyword. The method found is classified as their equivalence or similarity problem. The equivalence problem is readily solved by using canonical form. For similarity problem difference measure are surveyed with full description of well established dynamic program, methods related to this approach using probabilities. The particular interest that has attracted the attention of music, database and data mining communities is that of searching a large database to find the most similar objects to a query object

Finding the best matching subsequence to a query has been attracting the attention of both database and data mining communities for the last few decades. The problem of subsequence matching is defined as follows: given a query sequence and a database of sequences, identify the subsequence in the database that best matches the query. Achieving efficient subsequence matching is an important problem in domains where the target sequences are much longer than the queries and where the best subsequence match for a query can start and end at any position in the database.

The task of Entity Recognition is to identify predefined entities such as person names, products, or locations in this document. With a potentially large dictionary, this entity recognition problem transforms into a Dictionary-based Membership Checking problem, which aims at finding all possible substrings from a document that match any reference in the given dictionary. With the growing amount of documents and the deterioration of documents' quality on the web, the membership checking problem is not trivial given the large size of the dictionary and the noisy nature of documents, where the mention of the references can be approximate and there may be mentions of non relevant references.

The dictionary-based approximate membership checking process is now expressed by the Approximate Membership Extraction (AME), finding all substrings in a given document that can approximately match any clean references. The objective of AME guarantees a full coverage of all the true matched substrings within the document, where the true matched substring is a true mention of the clean reference semantically.

On the other hand, it generates many redundant matched substrings, thus rendering AME unsuitable for real-world tasks based on entity extraction. Indeed, redundant pairs are qualified to be part of AME results, but are unlikely to be true matches in real-world situations. AML only aims at locating true mentions of clean references. An important observation is as follows: in real-world situations, one word position within a document generally belongs to only one reference-matched substring, meaning that the true matched substrings should not overlap. Therefore, AML targets at locating non overlapped substrings in a given document that can approximately match any clean reference.

II. LITERATURE SURVEY

2.1 Exploiting Web Search Engines to Search Structured Databases.

The task of efficiently and accurately identifying relevant information in a structured database for a web search query. The relevant structured data items are then returned to the user along with web search results.

Each structured database is searched in isolation. To provide more informative results the web query is federated to one or more structured database. It mainly focus on a web document is related to an entity if the entity occurs the document. Main goal is to be able to establish this relationship. Identify mentions of entities in web documents for a wide class of entities.

This task is an instance of the entity extraction problem. The search in each structured database is "siloed" in that it exclusively uses the information in the specific structured database to find matching entities. That is, it matches the query terms only against the information in its own database. The results from the structured database search are therefore independent of the results from web search.

In contrast, several web documents from product review sites, blogs and discussion forums may mention the relevant products in the context of the query keywords flight-weight, gaming, and laptop. Therefore, the documents returned by a web search engine are likely to mention products that are relevant to the user query.

However, each of the returned web documents may contain many relevant products and even some irrelevant ones. To assemble the relevant products mentioned in documents returned by a web search engine, a user has to read through a potentially large number.

An architecture which is being proposed it effectively leverages existing search engine components in order to efficiently implement the integrated entity search functionality. Specifically this technique add very little space and time overheads to current search engines while returning high quality results from the given structured databases. The drawbacks are it takes high search time, processing time is difficult to adopt and it has high overhead.

2.2 Efficient Exact Set-Similarity Joins

A similarity join is an important operation for reconciling different representation of an entity. Set similarity joint algorithm is used, that is defined as given two input collections of sets identify all pairs of sets, one from each collection, that are highly similar.

A data collection often has various inconsistencies which have to be fixed before the data can be used for accurate data analysis. The process of detecting and correcting such inconsistencies is known as data cleaning.

A common form of inconsistency arises when a real-world entity has more than one representation in the data collection; for example, the same address could be encoded using different strings in different records in the collection. Multiple representations arise due to a variety of reasons such as misspellings caused by typographic errors and different formatting conventions used by data sources.

The notion of similarity is captured numerically using a string based similarity function and two records are considered similar if value returned is greater than threshold. The similarity between two sets is captured using a general class of predicates involving the sizes of the sets and the size of their intersection. Apart from string-based similarity, semantic relationships between entities can be exploited to identify different representations of the same entity. Important feature of our algorithms is that for ssjoins involves jaccard and hamming.

A general-purpose data cleaning system is therefore faced with the daunting task of supporting a large number of similarity joins with different similarity functions The algorithms can be broadly characterized as signature-based algorithms that first generate signatures for input sets, then find all pairs of sets whose signatures overlap, and finally output the subset of these candidate pairs that satisfy the set-similarity predicate.

The performance of the 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. Finally, the algorithms can be implemented on top of a regular DBMS with very little coding effort .The drawback is it does not give exact similarity and it just compare only minimum amount of datasets.

2.3 Improved Fast Similarity Search in Dictionaries

An algorithm to solve the approximate dictionary matching problem. Given a list of words w, maximum distance d fixed at preprocessing time and a query word q, to retrieve all words from w that can be transformed into q with d or less edit operations.

Presenting data structures that support fault tolerant queries by generating an index. On top of that, present a generalization of the method that eases memory consumption and preprocessing time significantly. At the same time, running times of queries are virtually unaffected.

Each word is represented by a string of characters over a finite alphabet Σ.The Levenshtein distance ed (a, b) defines a metric between two words a, $b \in \Sigma *$ and is used in this work to compute the distance between two words. The most trivial algorithm to solve the problem is scanning sequentially through the input list and noting the best match at each entry.

The running time is obvious and consists of a linear number of distance computations and searching an entire directory on a standard desktop computer takes only a few seconds even for dictionaries up to a few hundred thousand or even a million entries. But in many settings this is too much, because queries arrive in a high frequency. For example, a web search engine only has a few milliseconds to process a single request and does not have the time to do exhaustive searching in a large dictionary.

Since these distance computations are rather expensive, it is natural to find an algorithm that does not compare the input to the entire dictionary, but only a few entries. A so-called filter represents a criterion to quickly discard large portions of the search space.

On the other hand, it might be interesting to use data compression techniques to further reduce the storage requirements. The drawback is **i**t takes more time and it performs lower preprocessing.

2.4 A Linear Size Index for Approximate Pattern Matching

The problem of indexing a text to support searching substrings that match a given pattern with at most errors. A naive solution either has a worst-case matching time complexity or requires space.

Devising a solution with better performance has been a challenge for calculating the space index that can support error matching in respect to time, where occ is the number of occurrences.

Motivated by the indexing of dna, here the feasibility of devising a linear-size index that still has a time complexity linear in m. In particular, an o (n)-space index is given that supports k-error matching worst-case time.

Furthermore, the index can be compressed from o(n) words into o(n) bits with a slight increase in the time complexity. The number of errors is measured in terms of either the Hamming distance (number of character substitutions) or the edit distance (number of character substitutions, insertions or deletions).

The major concern is how to achieve efficient matching without using a large amount of space for indexing. Recently, two compressed solutions, namely, compressed suffix arrays and FM-index, have been proposed. they requires O(n) bits only and the matching time is errors, one can perform a brute-force search on an one-error index.

Alternatively, one can improve the matching time by including all possible erroneous substrings into the index; yet this seems to require O (n k) space.. They are able to avoid brute-force matching of a pattern with a moderate increase in the index size. The drawback is that it has space complexity and its time duration is high.

2.5 Exploiting Web Search to Generate Synonyms for Entities

Exploit web search engines in order to define new similarity functions. The entity matching task identifies entity pairs one from a reference entity table and the other from an external entity list. The task is to check whether or not a candidate string matches with member of reference table.

Consider another application. The entity matching task identities entity pairs, one from a reference entity table and the other from an external entity list, matching with each other. An example application is the offer matching system which consolidates offers (e.g., listed price for products) from multiple retailers.

New document-based similarity measures are proposed to quantify the similarity in the context of multiple documents. However, the challenge is that it is quite hard to obtain a large number of documents containing a string unless large portion of the web is crawled and indexed as done by search engines.

A Class of synonyms where each synonym for an entity *e* is an identifying set of tokens, which when mentioned contiguously (or within a small window) refers to *e* with high probability and identifying token set as IDTokenSets.

The approach is used to compute string similarity score between the candidate and the reference strings. Then develop efficient techniques to facilitate approximate matching in the context of proposed similarity functions. In an extensive experimental evaluation, demonstrate the accuracy and efficiency of a technique. The drawback is that it does not match document words and the Quality of id token set is low.

III. CONCLUSION

In this paper we have done literature survey for analyzing and formalizing the AML problem and propose to solve it with an efficient P-Prune algorithm. Prune is proved to be several times faster, sometimes even tens or hundreds of times faster, than simply adapting formerly existing AME methods. To inspect the improvement of AML over AME here apply both approaches within our proposed web-based join framework, which is a typical real-world application that greatly relies on the results of membership checking.

REFERENCES

[1] S. Agrawal, K. Chakrabarti, S. Chaudhuri, V. Ganti, A. Konig, and D.Xin, "Exploiting Web Search Engines to Search Structured Databases," Proc. 18th WWW Int'l Conf. World Wide Web, pp. 501-510, 2009.

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

[3] H. Chan, T. Lam, W. Sung, S. Tam, and S. Wong, "A Linear Size Index for Approximate Pattern Matching," Proc. 17th Ann. Symp. Combinatorial Pattern Matching, pp. 49-59, 2006.

[4] S. Chaudhuri, V. Ganti, and D. Xin, "Exploiting Web Search to Generate Synonyms for Entities," Proc. 18th Int'l Conf. World Wide Web (WWW), pp. 151-160, 2009.

[5] K. Chakrabarti, S. Chaudhuri, V. Ganti, and D. Xin, "An Efficient Filter for Approximate Membership Checking," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 805-818, 2008.

[6] A. Chandel, P. Nagesh, and S. Sarawagi, "Efficient Batch Top-K Search for Dictionary-Based Entity Recognition," Proc. 22nd Int'l Conf. Data Eng., p. 28, 2006.

[7] D. Karch, D. Luxen, and P. Sanders, "Improved Fast Similarity Search in Dictionaries," Proc. 17th Int'l Conf. String Processing and Information Retrieval, pp. 173-178, 2010.

[8] W. Hon, T. Lam, R. Shah, S. Tam, and J. Vitter, "Cache-Oblivious Index for Approximate String Matching," Theoretical Computer Science, vol. 412, pp. 3579-3588, 2011.

[9] L. Gravano, P. Ipeirotis, H. Jag dish, N. Koudas, S. Muthukrishnan, and D. Srivastava, "Approximate String Joins in a Database (Almost) for Free," Proc. 27th VLDB Int'l Conf. Very Large Data Bases, pp. 491-500, 2001.

[10] N. Koudas, S. Sarawagi, and D. Srivastava, "Record linkage: Similarity Measures and Algorithms," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 802-803, 2006.

[11] Z. Li, L. Sitbon, L. Wang, X. Zhou, and X. Du, "Approximate Membership Localization (AML) for Web-Based Join," Proc. 19th CIKM Int'l Conf. Information and Knowledge Management, 2010.

