ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

Top K Fuzzy Search Results over XML Data

[1] Ch.Lavanya Susanna, [2]Dr.D.Rajeshwara Rao, PHD
[1] M Tech, [2]Professor
[1,2]K.L University, Vaddeswaram, Guntur (dt).

Abstract: Efficient query retrieval systems are implemented for RDBMS systems only and not for XML based systems. Uses keyword-search system over XML data. A user composes a keyword query, submits it to the system, and retrieves relevant answers. They are using fuzzy type-ahead search over XML data. Even though this concept is nothing new for RDBMS based systems, this is a new information-access paradigm for XML based systems. Here, the system searches XML data on the fly as the user types in query keywords. Prior Systems Use Minimal-Cost Tree based techniques for producing top-k results. Minimal-Cost Tree based approaches are efficient as long as the query keywords are singular or dual utmost. As the number of attributes in the keyword for fuzzy query increases Minimal-Cost Tree construction is a computationally expensive process. For each criterion, we must assign a weight that describes its relativity importance. The best alternative is obtained by the affecting weights vector on decision matrix. Based on the Fuzzy Multiple Attribute Decision Making (FMADM) algorithm we intend to support multi attribute based queries over xml data with reduced computations.

Index Terms: *Interconnection semantics, keyword search, XML, XML, snippets, type-ahead search, fuzzy search.*

I. INTRODUCTION

The extreme success of web search engines makes keyword search the most popular search model for ordinary users. As XML is becoming a standard in data representation, it is desirable to support keyword search in XML database. It is a user friendly way to query XML databases since it allows users to pose queries without the knowledge of complex query languages and the database schema.

In most systems that incorporate keyword search into relational or XML data [1, 3, 8, 9, 10], the sole criterion is proximity (e.g., Jones is deemed the manager of Harris if these keywords appear in a

small sub tree of the given XML document) In [5], it is argued that in a tree document, the keywords are semantically related if they appear in a uniquely-labeled sub tree of the document. This approach is extended in [4] by incorporating information-retrieval techniques. In the work of [4, 5] is improved by introducing an approach that avoids some cases of incorrect results.

To be self-contained, a result snippet should represent a semantic unit. Suppose Figure 1 shows

the fragments of a result of query Texas apparel. If we choose the fragment between keyword matches in the corresponding XML document as the snippet of this query result, just as what a text search engine does, the users will not be able to see that both matches are nested in the tag retailer, and thus not able to easily understand that this query result is about an apparel retailer in Texas (rather than a book discussing the popular apparel styles in Texas).

The second goal of snippets is to allow users to easily distinguish different query results from each other. To achieve this in text document search, result snippets often include the document titles. Analogously, we propose to select the unique identifier (aka. key) of a query result into its snippet to identify this query result and highlight the fundamental differences among results. However, it is not clear how to identify the key of a query result.

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

The third goal is to design snippets that provide a representative summary of the query result by including the most prominent features in the result. Intuitively, a prominent feature is often reflected by a large number of occurrences of such a feature in the result. Continuing our example, suppose Brook Brothers has 1000 clothes of different styles, among which 600 are for men and 40 for children.

By using these three goals process hierarchy we will define the contribution of our application in the efficient fuzzy search results accurately.

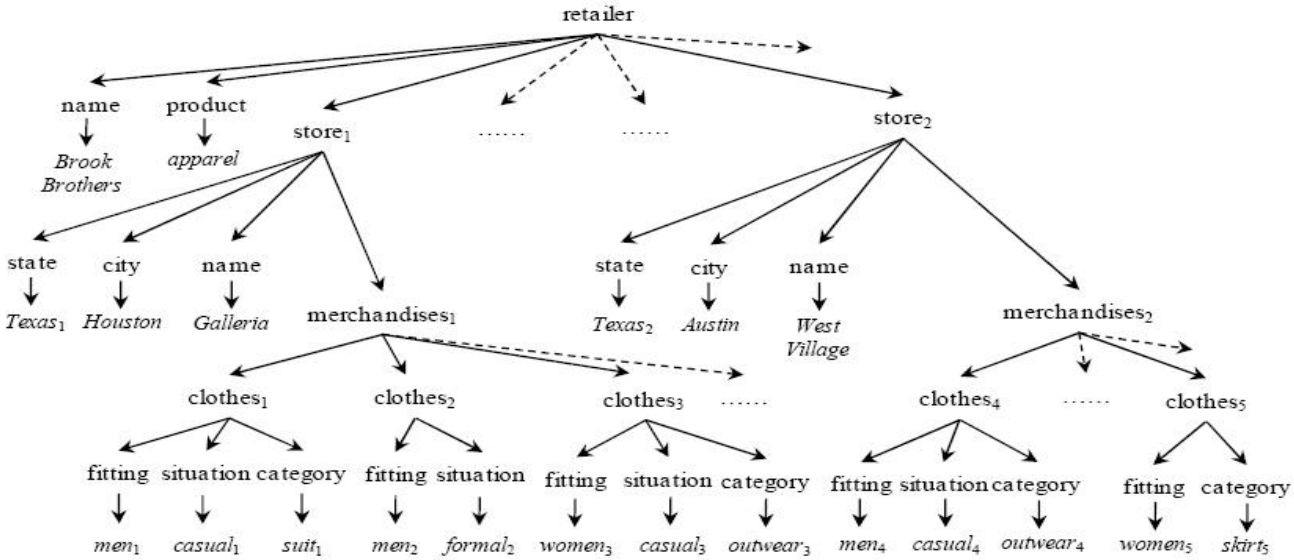

Figure 1: Part of a Query Result of Query Texas apparel retailer and Statistics about Value Occurrences.

The contributions of our work include:

• This is the first work that studies the problem of generating query result snippets for XML search.

• We identify four goals that a good query result snippet should meet in order to help users quickly get the essence of a query result and assess its relevance.

• To address the goals, we identify the significant information in a query result to be selected into the snippet.

• We prove that the decision problem of whether we can construct a snippet of a given size limit that contains all the significant information is NP-complete.

• We design an efficient algorithm to generate snippets that capture the identified significant information given the snippet size limit.

• A system for generating snippet for XML search has been implemented and tested for its efficiency and effectiveness through experimental studies.

II. RELATED WORK

Inspired by the great success of IR approach on web search (especially its distinguished ranking functionality), we aim to achieve similar success on XML keyword search, to solve the above three issues without using any schema knowledge. The main challenge we are going to solve is how to extend the keyword search techniques in text databases (IR) to XML databases, because the two types of databases

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

are different. *First*, the basic data units in text databases searched by users are flat documents. For a given query, IR systems compute a numeric score for each document and rank the document by this score. In XML databases, however, information is stored in hierarchical tree structures.

Statistics is a mathematical science pertaining to the collection, analysis, interpretation or explanation of data; it can be used to objectively *model a pattern* or *draw inferences* about the underlying data being studied. Although keyword search is a subjective problem that different people may have different interpretations on the same keyword query, statistics provides an objective way to distinguish the major search intention(s).

III. FUZZY RESULTS WITH SINGLE QUERY PROCESS

Exact Search: We are submitting query with single keyword then it will display relevant results regarding their requirement. The closeness of a match is measured in terms of the number of primitive operations necessary to convert the string into an exact match. This number is called the between the string and the pattern. Some approximate matchers also treat *transposition*, in which the positions of two letters in the string are swapped, to be a primitive operation. Changing *cost* to *cots* is an example of a transposition. Different approximate matchers impose different constraints. Some matchers use a single global unweighted cost, that is, the total number of primitive operations necessary to convert the match to the pattern. For example, if the pattern is *coil*, *foil* differs by one substitution, *coils* by one insertion, *oil* by one deletion, and *foal* by two substitutions. If all operations count as a single unit of cost and the limit is set to one, *foil*, *coils*, and *oil* will count as matches while *foal* will not.

Other matchers specify the number of operations of each type separately, while still others set a total cost but allow different weights to be assigned to different operations. Some matchers permit separate assignments of limits and weights to individual groups in the pattern

Traditionally, approximate string matching algorithms are classified into two categories: on-line and off-line. With on-line algorithms the pattern can be preprocessed before searching but the text cannot. In other words, on-line techniques do searching without an index. Early algorithms for on-line approximate matching were suggested by Wagner and Fisher and by Sellers. [5] Both algorithms are based on dynamic programming but solve different problems. Sellers' algorithm searches approximately for a substring in a text while the algorithm of Wagner and Fisher calculates Leven shtein distance, being appropriate for dictionary fuzzy search only.

On-line searching techniques have been repeatedly improved. Perhaps the most famous improvement is the bitmap algorithm (also known as the shift-or and shift-and algorithm), which is very efficient for relatively short pattern strings.

Fuzzy Search:

A fuzzy search is a process that locates Web pages that are likely to be relevant to a search argument even when the argument does not exactly correspond to the desired information. A fuzzy search is done by means of a fuzzy matching program, which returns a list of results based on likely relevance even though search argument words and spellings may not exactly match. Exact and highly relevant matches appear near the top of the list. Subjective relevance ratings, usually as percentages, may be given.

A fuzzy matching program can operate like a spell checker and spelling-error corrector. For example, if a user types "Mississippi" into Yahoo or Google (both of which use fuzzy matching), a list of hits is returned along with the question, "Did you mean Mississippi?" Alternative spellings, and words that sound the same but are spelled differently, are given. A fuzzy matching program can compensate for common input typing errors, as well as errors introduced by optical character recognition scanning of printed documents. The program can return hits with content that contains a specified base word along with prefixes and suffixes. For example, if "planet" is entered as a search word, hits occur for sites containing words such as "protoplanet" or "planetary." The program can also find synonyms and related terms, working like an online thesaurus or

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

encyclopedic cross-reference tool. In the Ask Jeeves search engine, if the word "galaxy" is entered, hits are returned such as "Galaxy Photography," "Milky Way," and "The Nine Planets Solar System Tour."

Fuzzy matching programs usually return irrelevant hits as well as relevant ones. Superfluous results are likely to occur for terms with multiple meanings, only one of which is the meaning the user intends. If the user has only a vague or general idea of the topic, or does not know exactly what to look for, the ratio of relevant hits to irrelevant hits tends to be low. (The ratio is even lower, however, when an exact matching program is used in this situation.)

Fuzzy searching is much more powerful than exact searching when used for research and investigation. Fuzzy searching is especially useful when researching unfamiliar, foreign-language, or sophisticated terms, the proper spellings of which are not widely known. Fuzzy searching can also be used to locate individuals based on incomplete or partially inaccurate identifying information.

IV. FUZZY RESULTS WITH SINGLE QUERY PROCESS

For a keystroke that invokes a query, we first tokenize the query string into keywords, k1; k2; . . . ; k‘. For each keyword ki (1 < i< L), we compute its corresponding active nodes, and for each such active node, we retrieve its leaf descendants and corresponding inverted lists.

As consider the above example present in the single query submission we will find the query results related to submitted query. As the same time we will find query results related to multi keywords with single query. For example, we will find the results of “Mango” then submit with multi keywords related to Mango, like Mango features and mango color mango types. In this way we will find the efficient results related query submission.

Assume a user types in a query “db mics” letter by letter. As the user types in the keyword “db,” for each keystroke, we incrementally answer the query as discussed before. We identify predicted word “db” and compute the union list Udb ¼ f13; 16g. When the user types in “db mics,” we find the active nodes micð7Þ; micesð9Þ; michð10Þg, identify the predicted words of the active nodes, “mices” and “mich,” and compute the union list Umics ¼ f14; 18; 26g. Then, we compute the ELCAs on top of the two union lists Udb and Umics, get the ELCAs (XML elements 12 and 15), and return the sub trees rooted at the two ELCAs. Accordingly, we can incrementally answer the keyword query “db mics.”

V. TOP-K FUZZY SEARCH RESULTS

We are consider the top preferable results in the query results using Top-K algorithms present in the data retrieval of data mining applications. Keyword search in XML data has attracted great attention recently. Xu and Papakon stations proposed smallest lowest common ancestor (SLCA) to improve search efficiency. Sun et al. studied multi way SLCA-based keyword search to enhance search performance. They proposed several optimization techniques using mesh to answer keyword queries over streams. Type-ahead search is a new topic to query relational databases.

An instance of an *enumeration problem* consists of an input x and a finite set $A(x)$ of *answers*. An *enumeration algorithm* E prints all the answers of $A(x)$ without repetitions. We assume that there is an underlying *ranking function* that maps answers to positive real numbers. The rank of an answer a is denoted by *rank* (a). Suppose that the algorithm E Enumerates the sequence $a1, \ldots, an$. If *rank* $(ai) \geq$ *rank* (aj) holds for all $1 \leq i \leq j \leq n$, then the enumeration is in *ranked order.* The delay of E is the length of time (i.e., execution cost) between printing two successive answers. There is also a delay at the beginning, i.e., until the first answer is printed, and at the end, i.e., until the algorithm terminates after printing the last answer.

VI. EXPERIMENTAL RESULTS

We employed the data sets DBLP6 and XMark.7 the sizes of DBLP and XMark were 510 and 113 MB, respectively. In this paper we are developing essential hybrid algorithm for XML data based on XRANK conditions. The verb XRANK allocates

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

items to buckets based on value. If the total number of items is evenly divisible by the number of buckets, then each bucket will have the same number of items; otherwise the first buckets have extra items.

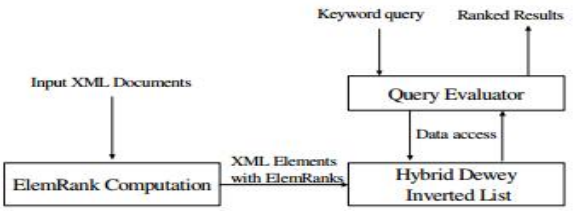

Figure 2: XRANK Architecture

We are giving XML documents as a input with single keyword and multi keyword submission then it will check that XML document can be related to that single and multi keyword results as follows.

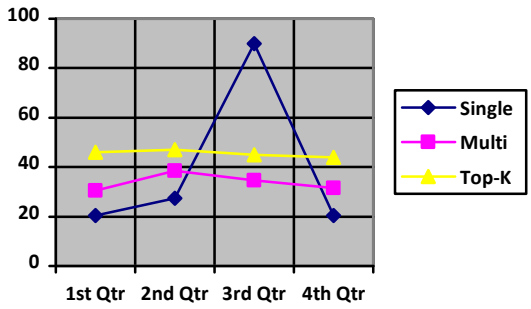

Figure 3: Query results Comparison

As discussed in the above experiment we will define efficient fuzzy results compared with existing approaches in single and multiple keywords Submission.

VII.CONCLUSION

We proposed effective index structures, efficient algorithms, and novel optimization techniques to progressively and efficiently identify the top-k answers. We examined the LCA-based method to interactively identify the predicted answers. By using hybrid algorithm for accessing efficient results in the XML documents. We also define the comparative results with user entry in the XML data with existing approaches to the proposed techniques. Top-K results are also defining the comparative results with user click in the XML documents data retrieval using XML query results.

VIII. REFERENCES

[1] Y. Luo, X. Lin, W. Wang, and X. Zhou, "Spark: Top-k Keyword Query in Relational Databases," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 115-126, 2007.
[2] Y. Luo, W. Wang, and X. Lin, "Spark: A Keyword Search Engine on Relational Databases," Proc. Int'l Conf. Data Eng. (ICDE), pp. 1552-1555, 2008.
[3] A. Markowetz, Y. Yang, and D. Papadias, "Keyword Search on Relational Data Streams," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 605-616, 2007.
[4] L. Qin, J.X. Yu, and L. Chang, "Keyword Search in Databases: The Power of Rdbms," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 681-694, 2009.
[5]Arasu, A., Babu, S., Widom, J. The CQL Continuous Query Language: Semantic Foundations and Query Execution. VLDB Journal, 15(2): 121–142, 2006.
[6] Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: XRANK:
ranked keyword search over XML documents. In: SIGMOD (2003)
[7] Hristidis, V., Gravano, L., Papakonstantinou, Y.: Efficient IR-s.
[8]. Shao, F., Guo, L., Botev, C., Bhaskar, A., Chettiar, M.M.M., Yang,
F., Shanmugasundaram, J.: Efficient keyword search over virtual
XML views. In: VLDB, pp. 1057–1068 (2007).