International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 05 | May-2018 www.irjet.net p-ISSN: 2395-0072

XML SEARCH ENGINE USING DATA MINING

KEERTHANA V.S

S4. M.E Computer Science and Engineering, Noorul Islam University Kanyakumari, Tamilnadu

---

Abstract - T*he increasing popularity of XML for data representations, there is a lot of interest in searching XML data. Due to the structural heterogeneity and textual content's diversity of XML, it is daunting for users to formulate exact queries and search accurate answers. Therefore, approximate matching is introduced to deal with the difficulty in answering users' queries, and this matching could be addressed by first relaxing the structure and content of a given query and, then, looking for answers that match the relaxed queries. Ranking and returning the most relevant results of a query have become the most popular paradigm in XML query processing. However, the existing proposals do not adequately take structures into account, and they, therefore, lack the strength to elegantly combine structures with contents to answer the relaxed queries. To address this problem, we first propose a sophisticated framework of query relaxations for supporting approximate queries over XML data. The answers underlying this framework are not compelled to strictly satisfy the given query formulation; instead, they can be founded on properties inferable from the original query. We, then, develop a novel top-k retrieval approach that can smartly generate the most promising answers in an order correlated with the ranking measure.*

Key Words: **XML**- Extensible Mark-up Language, **XPATH**- XML path language, **XQUERY**- XML query

1. INTRODUCTION

TRADITIONAL methods use query languages such as XPath and XQuery to query XML data. These methods are powerful but unfriendly to non expert users. First, these query languages are hard to comprehend for nondat abase users. For example, XQuery is fairly complicated to grasp. Second, these languages require the queries to be posed against the underlying, sometimes complex, database schemas. Fortunately, keyword search is proposed as an alternative means for querying XML data, which is simple and yet familiar to most Internet users as it only requires the input of keywords. Keyword search is a widely accepted search paradigm for querying document systems and the World Wide Web. Recently, the database research community has been studying challenges related to keyword search in XML data One important advantage of keyword search is that it enables users to search information without knowing a complex query language such as XPath or XQuery, or having prior knowledge about the structure of the underlying data.In a traditional keyword-search system over XML data, a user composes a query, submits it to the system, and retrieves relevant answers from XML data. This information-access paradigm requires the user to have certain knowledge about the structure and content of the underlying data repository. In the case where the user has limited knowledge about the data, often the user feels "left in the dark" when issuing queries, and has to use a try-and-see approach for finding information. He tries a few possible keywords, goes through the returned results, modifies the keywords, and reissues a new query. He needs to repeat this step multiple times to find the information, if lucky enough. This search interface is neither efficient nor user friendly

2. LITERATURE SURVEY

Extensive research studies have been done on structured queries as well as on text search over XML data and graph data. In view of the difficulty of formulating the queries with precise structures over XML data, an IR-style querying, in particular, full-text and keyword search is introduced. This approach has the merit of eliminating structures in the query. It, therefore, lightens the users from the burden of knowing the relationships occurring among XML data. Maio et al. presented an ontology-based retrieval approach, which supports data organization and visualization and provides a friendly navigation model. Built on the availability of a bulk of ontologies (knowledge), existing commercial solutions (e.g., Google knowledge graph, etc.) accomplish the ontology-based information retrieval and question answering (IR&QA) on structured and unstructured data.

2.1 Flexible queries over semi structured data

Flexible queries facilitates in a novel way, easy and concise querying of databases that have varying structures. two different semantics, flexible and semiflexible, are introduced and investigated. the complexity of evaluating queries under the two semantic is analyzed. query evaluation is polynomial in the size of the query the database and the result in two cases. a semiflexible DAG query and a tree database. A flexible tree query and a database that is any graph. For the flexible semantics query equivalence are also investigated. for the semiflexible semantics query equivalence is polynomial for DAG queries and exponential when the queries have cycles. under the semiflexible and flexible semantics. Two databases could be equivalent even they are not isomorphic. Traditional query languages and traditional querying methods are not well suited for semi structured data. The semi structured data model is based on a labeled directed graph.more importantly, traditional query languages are not geared to data having no schema at all, or a schema that may change considerably over time or from one data instance to another. An additional consideration is the size of the schema, which could be quite large compared to the size of schemas of structured data. When the schema is large and complicated, querying the data could be rather difficult. An initial phase of querying the schema might be needed before the query can be formulated. Even with this

additional step, the query could be quite large and hard to phrase, due to the need to cover many structurally similar, but not structurally identical data instance. The size of the schema is not the only source of difficulties. A case in point is an XML repository of document with DTDs designed for machine interchange. Traditional query languages are based on the concept of rigid matching that is, there should be a perfect match between the condition specified in the query and the data.

2.2 An Implementation of Tree Pattern Matching Algorithms for Enhancement of Query Processing Operations in Large XML Trees

XML has become a defacto standard for storing sharing and exchangeing the information across the various domains. interoperability is achieved using XML. Due to the increasing popularity of XML enterprises are generating and exchanging the data across the various domains. there is a wide analysis to identify the efficiency of XML tree pattern matching algorithms. previous years many method have been proposed to match XML tree queries efficiently. In particularly TwigStack, OrderedTJ, TJFast and TreeMatch algorithms. All algorithms to achieve something through these own way like structural relationship including parent-child relationship and ancestor- descendant relationship and more. The result to show that which algorithm is superior to previous approaches in term of the performance.

Data mining is the process of analyzing data from different perspectives and summarizing it into useful information. Mining is applied to gain knowledge for large amount of XML Datasets. XML has become ubiquitous language sharing, storing and exchanging information across various platforms. XML documents can be represented as a tree structure using DOM parser. DOM Parser is mainly used to store, access or manipulate the XML tree. XQuery (XML Query Language) and XPath (XML Path Language) are traditional XML query languages to query the XML Data. Our existing system provides answers to the queries using these query languages. These query languages requires some complex notations to perform query processing. XQuery and XPath are powerful but unfriendly to non-expert users. Existing system uses TwigStack Algorithm to perform query answering. But, TwigStack algorithm provides answers to the queries containing P-C (Parent-Child) and A-D (Ancestor Descendant) relationships. This causes sub-optimality problem in proposed system we are using keyword query to perform query answering. A Holistic XML Tree Pattern Matching Algorithm called TreeMatch is used to overcome the sub-optimality problem faced by the existing system.

2.3 Structural Joins: A Primitive for Efficient XML Query PatternMatching

XML queries typically specifies patterns of selection predicates on multiple elements that have some specified tree structured relationships. the primitive treestructured relationships are parent-child and ancestor-descendant and finding all occurances of these relationships in an XML database is a core operation for XML query processing. The tree-merge algorithms are a natural extension of traditional merge joins, while the stack tree algorithm have no counterpart in traditional relational join processing. a range of data and queries using the timber native XML query engine built on top SHORE. tree merge algorithm can have performance comparable to the stack algorithm, they are considerably worse. the stack tree algorithm have worst-case input output and CPU complexities linear in the sum of the size of input and output while the tree- merge algorithms do not have the same guarantee. XML employs a tree-structured model for representing data. Quite naturally, queries in XML query languages typically specify patterns of selection predicates on multiple elements that have some specified tree structured relationships. For example, the XQuery path expression book[title = 'XML']//author[. = 'jane'] matches author elements that (i) have as content the string value "jane", and (ii) are descendants of book elements that have a child title element whose content is the string value "XML". This XQuery path expression can be represented as a node-labeled tree pattern with elements and string values as node labels. Such a complex query tree pattern can be naturally decomposed into a set of basic parent-child and ancestor-descendant relationships between pairs of nodes

2.4 Fuzzy XPath Queries in XQuery

A fuzzy extension of the Xpath language which provides ranked answers to flexible queries taking profit of fuzzy variants of and, or and avg operators for XPathconditions, as well as two structural constraints called down and deep, for which a certain degree of relevance is associated. the fuzzy XPath with the XQuery language. it have been defined an XQuery library able to fuzzily handle XPath expression in such a way that our fuzzy XPath can be encoded as XQuery expressions. XQuery processors can handle a fuzzy version of XPath by using the library that had been implemented. Information retrival requires the requires the design of query languages able to adapt to user's references and providing rnked sets of answers. the degree of satisfaction of the users with respect to an answers can be measured in several ways. A fuzzy extension of XPath whose main aim is to provide mechanism to assign priority to queries. The XPath language [6] has been proposed as standard for XML querying and it is based on the description of the path in the XML tree to be retrieved. XPath allows to specify the name of nodes (i.e., tags) and attributes to be present in the XML tree together with Boolean conditions about the content of nodes and attributes. XPath querying mechanism is based on a Boolean logic: the nodes retrieved from an XPath expression are those matching the path of the XML tree, according to Boolean conditions. Information retrieval requires the design of query languages able to adapt to user's preferences and providing ranked sets of answers. The degree of satisfaction of the user with respect to an answer can be measured in several ways. XPath lacks on mechanisms for giving priority to queries and ranking answers. In an XPath-based query, the main criteria to provide a certain degree of

satisfaction are the hierarchical deepness and document order. Moreover, conditions on XPath expressions are usually of varying importance for a user, that is, the user gives a higher degree of importance to certain requirements when satisfying his (her) wishes. With this aim we have recently designed a fuzzy extension of XPath whose main aim is to provide mechanisms to assign priority to queries and to rank answers.

2.5 A Fuzzy Extension for the XPath Query Language

XML has become a wide spread format for data exchange over the internet. the current state of the art in querying XML data is represented by XPath and XQuery, both of which rely on Boolean conditions for node selection. Boolean Selection is too restrictive when users do not use or even know the data structure precisely. When queries are written based on a summary rather than on a precise representation of the schema.XML querying framework called fuzzy XPath, based on Fuzzy set Theory, relying on fuzzy condition for the definition of flexible constraints on stored data. A function called deep similar which aims an substituting XPath's typical deep-equal function. it provides a degree of similar both structure-wise and content-wise. XPath allows selecting XML node sets via tree traversal expressions. although XPath is not fully-fledged query language. The need for expressing in a declarative form complex manipulations of XML trees has raised a lot of interest on XML query languages. The World Wide Web Consortium (W3C) has defined two standard languages for querying XML data: XPath and XQuery. XPath allows selecting XML node sets via tree traversal expressions. Although XPath is not a fully-fledged query language, it is expressive enough for many practical tasks, and has been adopted within XQuery for expressing selection conditions. XQuery adds to XPath the capability of working on multiple XML documents, of joining results, and of transforming and creating XML structures. XPath selection, used by both languages, is Boolean in nature: it partitions XML nodes into those which fully satisfy the selection condition, and those which do not. However, Boolean conditions can be, in some scenarios, not flexible enough for effectively querying XML data. A few considerations can be made to justify this claim. The first is about prescriptive schemas defining the tree structure of individual documents. Even when XML schemas do exist, they may be not available to users. Moreover document trees with the same schema may be widely different (both in used tags and nesting), and hence the schema will allow for diverse instantiations, making it difficult to predict a particular document structure from the schema. As a consequence, users often end up defining blind queries (queries written without a precise knowledge of the schema).

This happens either because users do not know the XML schema in detail, or because they do not know exactly what they are looking for. Finally, the same XML tree can be sometimes described using different schemas.

2.6 Automated Ranking Of Database Query Result

Ranking and returning the most relevant results of a query is a popular paradigm in information retrival the challanges and investigate several approaches to enable ranking in database, including adaptation of known techniques from information retrival. Automated ranking of the results of a query is a popular aspects of the query model in information retrieval that had grown to depend on the database system supports only a Boolean query model. the relational database to adapt ranking function from IR for handling the database ranking problem. when each attribute in the relation is a categorical attribute which can mimic the IR solution by applying the TF-IDF idea that is based on the frequency of occurance of attribute values in the database. the database contains numeric as well as categorical information that need to extends the concepts to numerical domains.

Empty answers: When the query is too selective, theanswer may be empty. In that case, it is desirable to have the option of requesting a ranked list of approximately matching tuples without having to specify the ranking function that captures "proximity" to the query. An FBI agent or an analyst involved in data exploration will find such functionality appealing.

2. Many answers: When the query is not too selective, too many tuples may be in the answer. In such a case, it will be desirable to have the option of ordering the matches automatically that ranks more "globally important" answer tuples higher and returning only the best matches. A customer browsing a product catalog will find such functionality attractive. Conceptually, the automated ranking of query results problem is really that of taking a user query (say, a conjunctive selection query) and mapping it to a Top-Kquery with a ranking function

2.7 An intelligent fuzzy agent approach for realizing ambient intelligence in intelligent inhabited environments

A novel life long learning approach for intelligent agent that are embedded in intelligent environments. the agent aims to realize the vision of Ambient Intelligence in Intelligent Inhabited Environment (IIE) by providing ubiquitous computing intelligence in the environment supporting the activities of the users. An unsupervised, data-driven, fuzzy, technique is there for extracting fuzzy membership functions and rules that represent the user's particularized behavior in the environment. the user's learnt behaviours can then be adapted online in a life long mode to satisfy the different user and system objectives. During a stay of five consecutive days in the intelligent dormitory which is real ubiquitous computing environment test bed. both online and offline experimental results are presented. they effectively can physically disappear. With the addition of communication capability to these artefacts and the use of pervasive networking, such artefacts can be associated together and remotely accessed in both familiar and novel arrangements to make highly personalized systems. The challenge however

is how to manage, program or direct such systems, a task that could quickly become prohibitive and an obstacle to the achievement of the pervasive computing. The vision of ambient intelligence can help to address this challenge [9, 19]. Ambient Intelligence is a new information paradigm where people are empowered through a digital environment that is "aware" of their presence and context, and is sensitive, adaptive and responsive to their needs [9]. Ambient intelligence improves the quality of life through creating desired environmental conditions and functionality via intelligent, personalised interconnected systems and services [9]. An environment with ambient intelligence can be characterised by its ubiquity, transparency and intelligence [9]. It is ubiquitous because the user is surrounded by a multitude of inter-connected embedded systems which form a pervasive infrastructure. Its transparency is due to the invisible nature of the computing based artefacts being seamlessly integrated into the surrounding environment. Its intelligence spawns from the fact that the technology is able to recognise the users and program itself to their needs by learning from their behavior. In addition, environments constructed in such a way would be adaptive to changing conditions and user preferences.

2.8 A TOOL FOR APPROPRIATE LARGE GRAPH MATCHING

Large graph datasets are common in many emerging database application, and most notably in large scale scientific application. to fully exploit the wealth of information encoded in graphs, effective and efficient graph matching tools are critical. due to the noisy and incomplete nature of real graph datasets, approximate, rather than exact, graph matching required. Many more modern applications need to query large graphs, each of which has hundred to thousand of nodes and edges and it present a novel technique for approximate matching of large graph queries. A novel indexing method that is incorporate graph structural information in a hybrid index structure. The indexing technique achieves high pruning power and the index size scales linearly with the database size. An innovating matching paradigm to query large graph. The technique distinguishes nodes by their importance in the graph structure.Graphs provide a natural way to model data in a wide variety of applications, such as social networks, road networks, network topology, protein interaction networks and protein structures. Many graph databases are growing rapidly in size. The growth is both in the number of graphs and the sizes of graphs (the number of nodes and the number of edges). There is a critical need for efficient and effective graph querying tools for querying and mining these growing graph databases. The database community has had a long-standing interest in querying graph databases [6], [9], [13], [15], [17]–[22]. These previous studies have mostly been carried out within the context of precise graph data, and have focused on exact graph or subgraph matching queries. However, many real graph datasets are noisy and incomplete in nature. Moreover, the discovered interactions only represent a small fraction of the true network. As a result, exact graph or subgraph matching often fails to produce useful results. In contrast, approximate graph or subgraph matching plays a critical role in these applications. Approximate matching allows node/edge insertions and deletions, and node/edge mismatches. Furthermore, many new graph applications prefer approximate matching results rather than exact ones as they can provide more information such as what might be missing or spurious in a query or a database graph. In addition, most existing graph matching methods areapplicable to databases that contain graphs with small sizes, i.e. each graph has a small number (tens) of nodes and edges. Moreover, the query graphs allowed in these methods are also small in size. However, in many new applications, both the query and database graphs are "large

2.9 EFECTIVE XML KEYWORD SEARCH WITH RELEVANCE ORIENTED RANKING

Information Retrieval style keyword search on web, keyword search on the web, keyword search on XML has emerged recently. The extreme success of web search engine 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 database since it allows user to pose queries without the knowledge of complex query language and the database schema. Designing a novel formulae to identify the search for nodes and search via nodes of thee query and present a novel XML ranking strategy to rank the individual matches of all possible search intentions. the techniques are implemented in an XML keyword search engine called XReal, and extensive experiments show the effectiveness of that approach.

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. Effectiveness in term of result relevance is the most crucial part in keyword search, which can be summarized as the following three issues in XML field.

Issue 1: It should be able to effectively identify the type of target node(s) that a keyword query intends to search for. WE call such target node as a search for node. **Issue 2**: It should be able to effectively infer the types of condition nodes that a keyword query intends to search via. We call such condition nodes as search via nodes. **Issue 3**: It should be able to rank each query result inconsideration of the above two issues.

2.10 ANSWERING IMPRECISE QUERIES OVER AUTONOMOUS WEB DATABASE

Answering queries with imprecise constraints requires user-specific distance metrics and importance measure for

attributes of interest- metrics that are hard to elict from lay users. a domain and user independent approach for answering imprecise queries over autonomous web database and method for query Database query processing models have always assumed that the user knows what she wants and is able to formulate a query that accurately expresses her needs. But with the rapid expansion of the World Wide Web, a large number of databases like bibliographies, scientific databases etc. are becoming accessible to lay users demanding "instant gratification". Often, these users may not know how to precisely express their needs and may formulate queries that lead to unsatisfactory results. Although users may not know how to phrase their queries, when presented with a mixed set of results having varying degrees of relevance to the query they can often tell which tuples are of interest to them. Thus database query processing models must embrace the IR systems' notion that user only has vague ideas of what she wants, is unable to formulate queries capturing her needs and would prefer getting a ranked set of answers. This shift in paradigm would necessitate supporting imprecise queries. This sentiment is also reflected by several database researchers in a recent database research assessment.

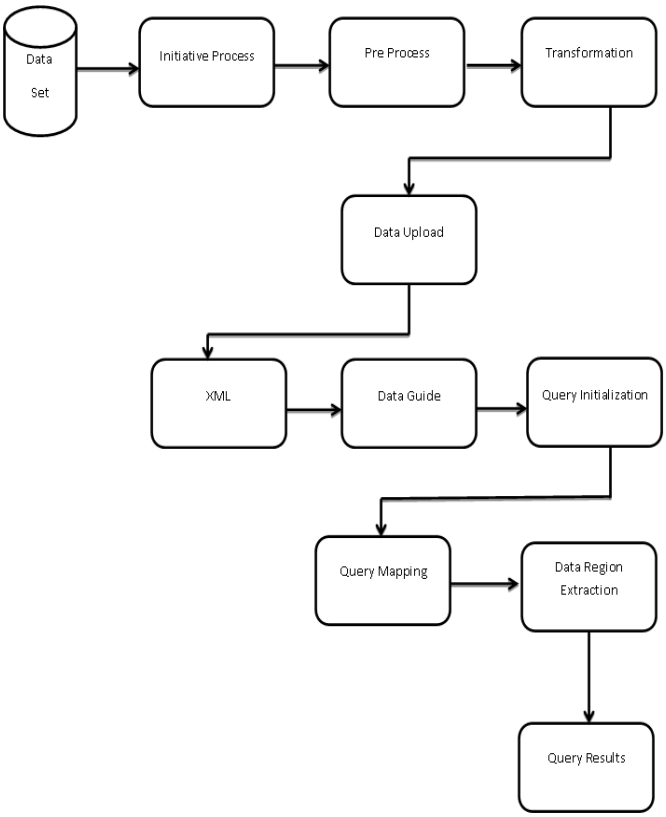
3 PROPOSED SYSTEM

A sophisticated framework of query relaxations for supporting approximate queries over XML data in this paper. Our approach adequately takes structures and the surmise of users' concerns into account, and it, therefore, has the ability to elegantly combine structures with contents to answer approximate queries. The answers underlying our proposed framework are not compelled to strictly satisfy the given query formulation; instead, they can be founded on properties inferable from the original query. Then, develop a novel top-k retrieval approach that can smartly generate the most promising answers in an order correlated with the ranking measures. In particular, rather than shifting the burden of providing the similarity functions to the users, our approach can effectively extract the semantics inherently presented in the XML data sources and automatically rank the results satisfying the approximate queries.

ADVANTAGES OF PROPOSED SYSTEM

A query relaxation method incorporating structures and contents, as well as the factors that users are more concerned about, for supporting approximate queries over XML data. In particular, our method surmises the factors that users are more concerned about based on the analysis of user's original query for supporting query relaxations. In addition, our approach differentiates the relaxation ordering instead of giving an equal importance to each node to be relaxed. In particular, the first relaxed structure to be considered is the one that has the highest similarity coefficient with original query, and the first node to be relaxed is the least important node. That customize the similarity relation assessment by analyzing the inherent semantics presented in XML data sources. In our solution, we classify nodes into two groups: categorical attribute nodes and numerical attribute nodes, and design the corresponding approaches on the similarity relation assessments of categorical attribute nodes and numerical attribute nodes. In addition, we design a clue-based directed acyclic graph (CDAG) to generate and organize structure relaxations and develop an effective assessment coefficient for the similarity relation assessment on structures. Based on the proposed similarity assessment and the degrees of importance, we complement the query relaxations with an automatic retrieval approach that can efficiently generate the most promising top-k We present an extensive experimental evaluation, which proves the effectiveness of our proposal on real-world data.

3.3 PROPOSED ARCHITECTURE



4. MODULES

4.1 System Construction Module

In the first module, we develop our proposed system with the entities, to show the performance of our contribution model. We consider a data model for XML where information is represented as a series of data trees. Essentially, a data tree represents a portion of the real world through entities (usually contains a set of attributes), values, and relationships among them. A simple XML data instance, which contains a heterogeneous collection of used cars. We develop the system for the application of user car sales system. The entities available in our system are admin and users. A organizes cars based on the model of a car, B organizes cars according to the selling location, and C

includes cars that are organized by model and year. The approximate queries can be achieved by introducing substitutes having the approximate query intents with the original query, which we call similar substitutes.

4.2 Query Relaxations

In this module Framework of query relaxations for supporting approximate queries over XML data. The answers underlying this framework are not compelled to strictly satisfy the given query formulation; instead, they can be founded on properties inferable from the original query**.** A query relaxation method incorporating not only structures and contents, but also the factors that users are more concerned about (we infer these factors by first analyzing the original query and then identifying relaxation ordering of structures and nodes), to answer approximate XML queries. Our method surmises the factors that users are more concerned about based on the analysis of user's original query for supporting query relaxations. In addition, our approach differentiates the relaxation ordering instead of giving an equal importance to each node to be relaxed. In particular, the first relaxed structure to be considered is the one that has the highest similarity coefficient with original query, and the first node to be relaxed is the least important node. Query relaxation enables systems to weaken the query constraints to a less restricted form to accommodate users' needs. Traditionally, queries submitted by users are modified in various aspects and ways to cope with different situations. The importance of such techniques that enable automatic query modification stems from the fact that this behavior is a very common activity in human discourse.

4.3 Approximate Query Processing

In this module Approximate query processing (AQP) is an alternative way that returns approximate answer using information which is similar to the one from which the query would be answered. A sophisticated framework of query relaxations for supporting approximate queries over XML data. The answers underlying this framework are not compelled to strictly satisfy the given query formulation; instead, they can be founded on properties inferable from the original query. The approximate queries can be achieved by introducing substitutes having the approximate query intents with the original query, which we call similar substitutes. Approximate query is a retrieval technique, which finds matches that are likely to be relevant to a search argument even when the argument does not exactly correspond to the desired information. An approximate query is done by means of an approximate matching strategy, which returns a list of results based on likely relevance even though search argument may not exactly match.

4.4 Top-k Retrieval Approach:

In this module a novel top-k retrieval approach that can smartly generate the most promising answers in an order correlated with the ranking measure. The proposed similarity assessment and the degrees of importance we complement the query relaxations with an automatic retrieval approach that can efficiently generate the most promising top-k The answer score of an answer (a match) measures the relevance of that answer to the user's query. For a given parameter k,the top-k problem is searching the best top-k answers (matches) ordered from best (highest answer score) to the worst.

5. Algorithm

1valuePartition

Input: numerical attribute node Ai, all the values val of Ai and number of partition n

Output: the set of each partition's upper limit s

01 min=getMinValue(val) //get the minimal value

02 max=getMaxValue(val)□1 //get the maximal value

03 total=getSize(val) //get the values' number

04 avg=total/n

05 low=min, up=max //set the lower limit and upper limit

06 while (low□up)

07 c=query(low□Ai□up)//execute query low□Ai□up andreturn

the number of results

08 if(c□avg)

09 add(up, s) // add up to set s

10 low=up, up=max

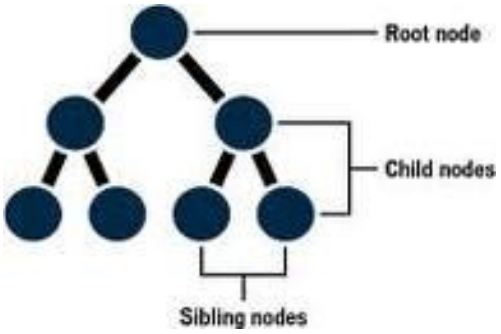
11 else up=low□ (up□low)/2

12 end while

13 return s

An ANV-pair can be visualized as a selection query that binds only a single attribute node. By issuing an ANV-pair query (e.g., model = 'Sedan') over XML data trees, a set of sub-trees rooting at the entity nodes (e.g., usedcar) which contain the ANV-pair can be identified. We represent the answerset containing each ANV-pair as a tree structure called the SemanticTree. The SemanticTree contains a set of keywords for each attribute node in the data trees not bound by the ANV-pair. Figure 5 shows SemanticTrees for make='Ford' and make='Dodge'. To represent the set of keywords, we extend the semantics of a set of keywords by associating an occurrence count for each member of the set. For example, for attribute node model in Figure 5(A), we see SUV with an

occurrence count of one hundred, suggesting that there are one hundred SUV modeled Ford cars in the data trees that satisfy the ANV-pair query.



XML TREE

In view of the continuous character of numerical values, we use a continuous domain (i.e., a partition), which is considered as a keyword, instead of each numerical value, to efficientlyevaluate the similarity in numerical attribute nodes. Algorithm1 is used to partition the numerical values of the numerical attribute node labeled t, starting with a value associating with t applying the method of equi-height histogram partition [36], partitioning all the values associating with t on the fly. In particular, when constructing histograms of certain numerical attribute node labeled (interpreted as) t, values associated with other numerical attribute nodes will not participate in the construction of t's histograms). In line 07, the count c records the number of the results when executing the query low□Ai□up, if the condition in line 08 is not satisfied, then Algorithm 1 will narrow the range by reducing the value of current "up" at line

11. Through the narrowing work, Algorithm 1 will iteratively determine the upper limits of all the partitions (line 13). As introduced in [34], the computation of equi-height histograms could be optimized and done in a constant time complexity per data item. Additionally, the evaluated results can be further computed during the offline processing step. Therefore, the equi-height histogram partition is feasible in practice. Since the construction of equi-height histogram is simple, the cost of maintaining equi-height histogram is low, and it is more effective than other partition approaches such as equi-width histogram[36], many commercial vendors have switched to equi-height histogram to develop their products. That is also the reason why we choose equi-height histogram to partition numerical values. For example, for attribute node "price", the partition can be done as follows: (i) we first issue simple queries on "price" to obtain all values, depicted as val, including the minimal value, maximal value and the total size of val, of "price", (ii) according to the number of partition n, we apply Algorithm 1to partition val to make the number of values ineach partition (roughly) the same.

Now let us introduce how to build the SemanticTree. The Semantic Tree of a given categorical value ai associating with an attribute node Ai could be built in two phases. In the first phase, the sets of keywords of all the interconnected attribute nodes of Ai are generated. If the interconnected attribute node Aj of Ai is a categorical attribute node, then we will record each value of Aj, associating with an occurrence count, as the set of a keyword. Otherwise, if the interconnected attribute node Aj of Ai is a numerical attribute node, then we will store each partition inferred by using Algorithm 1, associating with an occurrence count, as the set of a keyword. In the second phase, we will merge the generated sets combining their corresponding attribute nodes to generate the Semantic Tree.

6. CONCLUSION

Driven by real-world applications and key industrial stakeholders and initialized by national funding agencies, managing and mining Big Data have shown to be a challenging yet very compelling task. While the term Big Data literally concerns about data volumes, our HACE theorem suggests that the key characteristics of the Big Data are 1) huge with heterogeneous and diverse data sources, 2) autonomous with distributed and decentralized control, and 3) complex and evolving in data and knowledge associations. Such combined characteristics suggest that Big Data require a "big mind" to consolidate data for maximum values [27].

To explore Big Data, we have analyzed several challenges at the data, model, and system levels. To support Big Data mining, high-performance computing platforms are required, which impose systematic designs to unleash the full power of the Big Data. At the data level, the autonomous information sources and the variety of the data collection environments, often result in data with complicated conditions, such as missing/uncertain values. In other situations, privacy concerns, noise, and errors can be introduced into the data, to produce altered data copies. Developing a safe and sound information sharing protocol is a major challenge. At the model level, the key challenge is to generate global models by combining locally discovered patterns to form a unifying view.

7. FUTURE ENHANCEMENT

Big Data as an emerging trend and the need for Big Data mining is arising in all science and engineering domains. With Big Data technologies, we will hopefully be able to provide most relevant and most accurate social sensing feedback to better understand our society at real time. We can further stimulate the participation of the public audiences in the data production circle for societal and economical events. The era of Big Data has arrived.

8. REFERENCES

[1] sanjayagarwal , surajitchaudhari, gautamdas" Automated Ranking Of Database Query Results" on proceedings of the 2003 CIDR conference

[2]Faiyas doctor, hani hagras, victor Callaghan "An intelligent fuzzy agent approach for realizing ambient intelligence in intelligent inhabited environments

[3] zhifeng bao, tok wang ling, Bo Chen"effective XML keyword search with relevance oriented ranking

[4] ullas Nambiar and Subbarao Kambhampati" Answeing imprecise queries over autonomous Web Databases

[5] Jesus. M ,Almendros-Jimenez, Alejandro Luna, and Gines Moreno "Fuzzy XPath Queries in XQuery

[6]Shurug Al-Khalifa, nick koudas, Divesh srivastava" structural joins: Aprimitive for efficient XML query pattern matching, proceedings of the 18th International conference on data engineering 2002 IEEE

[7] N.Murugesan and R.Santhosh"an implementation of tree pattern matching algorithms for enhancement of query processing operations in large XML trees

[8]Yaron Kanza and Yehoshua sagiv"flexible queries over semi structured data

[9] Alessandro Campi Ernesto Damiani "A fuzzy extension for the XPath query language

[10]Yuanyuan Tian, Jignesh M.Patel" A tool for approximate large graph matching.