Fuzzy querying of XML documents
The minimum spanning tree

Abdeslame ALILAOUAR*, Florence SEDES*
*I.R.I.T , 118 route de Norbonne, 31062
Toulouse Cedex, France
{laouar, sedes}@irit.fr

Abstract

In this work we propose a fuzzy model to perform queries adapted to semi structured documents and human qualitative reasoning, by taking into account not only the contents of the documents, but also their structure. The fuzzy logic framework is used to represent flexible constraints through weighted formulas representing goals purchased, with their level of importance or priority. Furthermore, according to the user's knowledge level of the documents structure, we can distinguish several types of queries, therefore several ways to perform queries.

1. Introduction

The idea to make a flexible framework for querying XML documents collection is very intuitive and natural way to deal with heterogeneous nature of these documents and user preferences or interests. Indeed, a query with strict criteria can lead to empty or incomplete answers set. Consequently the exact matching technique becomes less appropriate or for querying these documents. Thus the inexact pattern matching along with returning a ranked list of answers become increasingly important to provide such solution of querying-answering problem for XML documents. Fuzzy evaluation of queries introduces a flexibility that, as compared to classical evaluation (crisp), is more appropriate for human dialog and/or natural language. It is mainly dependent on the similarity measures used, for both content and structure matching.

In this paper, we propose within the fuzzy logic framework and the minimum spanning tree problem [15] a flexible model for querying semi-structured document in general and XML documents in particular, initially based on the works of Dubois, Sedes and Prade for expressing user preferences and querying multimedia database [8, 11].

2. Background on XML model

A semi-structured document like XML document is a textual representation of a semi-structured data. Whereas classical IR models treat documents as atomic units, XML suggests a tree-like view on documents. Indeed, a well-formed XML document can be seen as an ordered, labelled tree [13], such that each node in this tree corresponds to an element in document is labelled with element tag name, and the leafs node correspond to the pure content (data). Each edge in this tree represents inclusion of the element corresponding to the child node under the element corresponding to parent node in the XML document file. To begin exploring the problem of flexible querying of XML documents, let us note that these documents can be classified in two categories: *data-centric* and *document-centric*. Data-centric documents are characterized by little or no mixed content, fairly regular structure, and a fine granularity. Essentially, these documents are designed for the data exchange between software. On the contrary, document-centric documents are characterized by less regular or irregular structure, larger granularity of data units (i.e. the smallest independent unit of data might be at the level of an element with mixed content or the entire document itself), and many mixed content.

3. Motivation example

Let us take an extract from the collection of cottages used as a base for PRETI[1] platform in the form of a heterogeneous XML document (i.e. each cottage has a *specific structure* which may be different from that of the others cottage) such as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<cotglist>
<cottage identifier = "23" >
    <character>
      <room> <nbeds> 4 </nbeds></room>
      <room> <nbeds> 2 </nbeds></room>
    </character>
    <price> 1700 </ price> </cottage>
<cottage identifier = "40" >
     <character>
       <nbeds > 4 </nbeds>
       <price>
              <summer>1300</summer>
              <winter>1100</winter> </price >
      </character>
</cottage >
</cotglist >
```

Let's assume that one user search for all cottages with 4 beds. This can be expressed with XQuery such as follows:

[1] http:// www.irit.fr/PRETI

```
FOR  $gt IN document("cottages.xml")//cotglist
WHERE $gt//cottage/nbeds = 4
RETURN <cottage>$gt@identifier</cottage>
```

This query will lead to a failure or in other words, it makes an empty set of answers, owing to the incomplete knowledge of the documents structure. In other words because the user made a mistake on the relationship between the "cottage" and "nbeds" tags, instead of ancestor/descendant relationship, he has given parent/child relationship, or in addition, he forget to put "character" tag between "cottage" and "nbeds" in a query.

Indeed, querying semi-structured data can be lead to compare two structures, that of the query and that of the documents. However, due to irregularity of the XML document structure, the query techniques based on the exact matching are typically inadequate, and are likely to lead to an empty or incomplete set of answers. Inside this sense, several techniques were proposed to deal with this problem, such as, the fuzzy logic based methods.

2. Fuzzy querying of XML documents

Fuzzy logic has the aim of studying the problem of representation of imprecision and uncertainty as well to design and develop a fuzzy reasoning mechanism, able to use and take into account the fuzzy and uncertain knowledge, such as the human being is able to do. In other part, when querying the semi-structured data (particularly XML data), we sometimes doesn't want to define an exact adjacent structure of data (i.e. a classical hard criteria over adjacent structure), especially when the adjacent structure is unknown a priori, or when querying heterogeneous XML documents, in which the same content can be available in different structures from a document to another. Therefore, the search process often leads to failure (even if feasible answers exist) or incomplete list of answers. To resolve this weakness, we assume that the user prefers exact matchers but also interested in similar results.

In fact, a fuzzy logic framework can be applied here in two cases, firstly: the expression of the user's preferences and the relative levels of importance. Secondly: the evaluation of queries in order to rank the answers according to the degree to which they satisfy the user's preferences. Inside this framework, one model has been proposed by Damiani and his work-group [5]. This model is based on the notion of the fuzzy labelled graphs [12]. These latter are obtained from the XML documents tree by evaluating the importance (weighting) of the information associated to each node/arcs. The new arcs can be added between two nodes if they are connected via a path of any length in the original document tree. The weight of new arc (i.e. the arc of closure) is computed by aggregating via a t-norme[6] the weights of the arcs belonging to the path it corresponds to in the original graph[5]. Hence, the principal aim of this approach is to pass from a tree matching problem into a fuzzy graph matching problem. While this greatly increases the flexibility, it also increases immensely the time and space complexity of the problem, because after calculate fuzzy closure, it shall compute a degree of matching between the tailored graph and the query graph.

Nevertheless, in order to not increase the complexity degree of this problem which is already high (graphs matching is among the most complex programming problems). We have focused our researchers on another method based on the works of Dubois, Sedes, Prade [8, 11] on the flexible querying of multimedia database and semi-structured data.

In fact, the nature of XML documents is not the only reason of the insufficiency of querying methods based on exact matching, it is sometimes necessary to permit the user to express queries with a minimum of knowledge about adjacent structure. In other words, the queries with high structural abstraction, such as the queries can contain obligatory elements, the elements more or less preferred, weighted criteria, then gradual property or requirement. This can be more or less satisfied by pieces of documents or documents entire. In fuzzy logic framework this can be represented by a fuzzy predicates [6] like for example: expensive, old, rare, good, etc. However, before we start describing our approach, according to the knowledge of data structure that the user has, we will be distinguishing between three sorts of queries then three ways to perform its, such as follows:

3. Querying in case of a known structure

Supposing that the user has an a priori idea on the document structure in term of tagging, i.e. the names of the tags, attributes and their hierarchical structures. We can find the case of the usual databases querying. In this case, the flexibility concerns mainly the contents of the documents (purely data). It can be used to translate the users' preferences on the values of attributes or elements. It is also possible to use flexibility to indicate the importance of the query elements or to indicate the set of priorities between the different search criteria. Nevertheless, this form of flexibility is already treated in the work of Dubois, Prade, Bosc and Pivert [4, 8, 9] on the databases flexible querying. Therefore, we will focus only on the remaining cases, i.e. the querying with an unknown structure and a partial knowledge of the structure.

4. Querying with unknown structures

Without a priori knowledge of documents structures the user may express his queries in different ways. The user has the possibility to express queries in a free-form, i.e. a set of keywords referring to tags names, attribute names or value and content. In general, any information may be useful in search process. In this sense, an approach has been presented in [11] assume that a data

guide to be available. It provides a set of keywords refereeing to tags and attributes names and/or key expressions of the domain. In such a case, the tags or attributes elicited from the regularity of the descriptions can be displayed according to their embedding level. The data guide might be also based on ontology's describing the considered domain of knowledge. This data guide is a tool aiming at helping the user for expressing his query, but the user is only supposed to have a general knowledge of the domain (he understands the keywords), but not of the underlying structures of documents [11].

However, in our approach the presence of any data guides in order to help the user to choose the appropriate terms existed in target documents, it has not excluded the new. But, in a general way, the user will express queries by indicating a set of preferences (i.e. fuzzy criteria) $\{\omega_i : i = 1, ..., m\}$ with their levels $w_{i : i=1,...,m}$ of importance or priority one w.r.t the other. A priority level w_i for criterion ω_i can be modelled in the framework of possibility theory [10] by a degree of necessity estimating how imperative ω_i is. Therefore $N(\omega_i) = w_i$.

Indeed, the formula (ω_i, w_i), can be represented by a special kind of crisp constraints [7] using degrees of membership $\mu_{\omega i*}$ as follows:

$$\mu_{\omega_i *}(\alpha) = 1 \text{ if } \alpha \text{ satisfies } \omega_i,$$
$$= 1 - w_i \text{ if } \alpha \text{ violates } \omega_i.$$

In this way w_i represents to what extent it is necessary to satisfy ω_i, $1\text{-}w_i$ indicates to what extent it possible to violate it. In other words, any potential solution that violates ω_i satisfies (ω_i, w_i) to degree equal to $1\text{-}w_i$. When possibility degrees are interpreted in term feasibility, the necessity degrees are thus levels of priority. If ω_i is it self a vague criteria, thus we can write :

$$\mu_{\omega_i *}(\alpha) = \max\left(1 - w_i, \mu_{\omega_i}(\alpha)\right)$$

$\mu_{\omega i}(\alpha)$ is feasibility level of potential answer α by ωi criterion. Such as : $\mu_{\omega i}(\alpha) = 1$ indicates that a answer α is totally satisfied ω_i, while $\mu_{\omega i}(\alpha) = 0$ indicates that it is totally violated ω_i , $0 < \mu_{\omega i}(\alpha) < 1$ when α satisfies only partially ω_i, and $\mu_{\omega i}(\alpha) > \mu_{\omega i}(\beta)$ indicates that ω_i is more satisfied by α than by β, and also the formula $N(\omega_i) > N(\omega_j)$ expressing that the satisfaction of criterion ω_i is preferred to the satisfaction of ω_j. Hence, the queries are evaluated progressively by evaluating sequentially the criteria from the highest-priority to the lowest-priority, therefore the system starts by ordering the query elements according to their importance level in order to increasingly evaluate them and to start by handling the fully important elements at before. Then, we take the elements according to their importance level and we start searching for all available corresponding elements (tags, attributes, PCDATA) in the documents which can be considered as sufficiently similar (with respect to a given threshold). However, for domains with a natural order it is straight-forward to define similar values for a given value; these are the values closest to the given value according to the order. The problems arise when the domain in question is without ordering. In that case the similarity can be defined explicitly in some kind of network relating similar elements, i.e., the common language of thesauri, or defined implicitly by means of the string edit distance [14]. In fact, the feasibility degree is considered as a similarity level. As a result, the answers will be created gradually, such that, the answers set will be initialized by set of partial answers satisfied the highest-important criteria. Each time that a feasible element is found, a partial answer is created and we add the corresponding node in the document tree to this answer. Afterwards for every criterion, each time a feasible element is found, it will be added to one of already partial answer created, such that a minimal spanning tree is produced.

The minimum spanning tree [14] of a graph is the tree of minimum length connecting the all nodes, where by 'length' I mean the sum of the weights of the connecting links in the tree. In our context the minimum spanning tree is defined as a tree which contains a query elements and a less number of nodes/arcs. Hence, the evaluation of query leads to calculate two degrees for each answer, a degree of possibility and a degree of necessity, by means of aggregation of the possibility and necessity degrees of elementary criteria, such that queries formed by conjunctions, the degree of possibility is calculated by the minimum and the queries formed by disjunctions, the degree of possibility is calculated by the maximum as follows:

$$\mu_{\Pi_{\text{conj}}}(\varpi_{\varpi \subset \wp}) = \inf_{i=1..m} \max_{\alpha \in \varpi}\left(1 - w_i, \mu_{\omega_i}(\alpha)\right)$$
$$\mu_{\Pi_{\text{disj}}}(\varpi_{\varpi \subset \wp}) = \sup_{i=1...m} \min_{\alpha \in \varpi}\left(w_i, \mu_{\omega_i}(\alpha)\right)$$

Such that α be an element of the query, ω_i a criterion on α and w_i the necessity degree of ω_i. ϖ be a fragment of document $\wp$.

5. Case of partial knowledge on the structure

In this case, the queries are modelled by weighted tree patterns. A tree pattern is a labelled tree whose nodes are labelled by variable names, constants. The nodes labelled by variable names are called variable nodes and those labelled by element names and data values are called constant nodes. The tree also has a distinguished set of edges called descendant edges. The weighted tree pattern is a tree pattern in which approximate and imprecise weights are associated with the nodes and edges of the tree pattern. These weights represent the importance level of the information associated to the nodes and the edges. But in this work for the moment, we do not introduce the weight notion on the edges. Nevertheless, we suppose that the user does not have a complete knowledge and has some uncertainty about hierarchical structure of the documents because of their heterogeneity and versatile nature.

Consequently, the queries formulated by this user are structurally imprecise, uncertain and approximate. This hypothesis, we allow and justify the fact that we do not content ourselves with the answers simply with exact hierarchical structure. For instance, let us suppose that one user does not know precisely the structure of this document search to rent a cottage with four beds and the price equals to 1400€, with an importance equal to "0,8" attributed to number of beds and "0,6" attributed to price. This query can be represented in the by a weighted tree pattern such as in Figure1. So, we can say that this query has an approximate hierarchical structure or in other words is structurally ambiguous. This is due to the fact that we have several price of the same cottage in the given document. In the context of labelled graphs matching this problem is called the problem of the burst-node. In the context querying XML documents, it represents the fact that such sub-tree of target document is substitute by a single node in tree pattern query. In fact, there is several ways to deal with this problem. For example, in the case of the price, we can use the *average* for aggregate the nodes "price" in to one node "price". In other cases (case of nbeds) we can use the *sum*, etc. and that depending on the context. But, for a better interoperability of our search algorithm, for now, we will content ourselves to give back to user the whole sub-tree corresponding to the burst node.

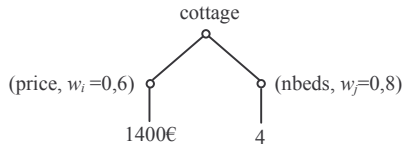

Figure 1: An example of a weighted tree pattern

Our search algorithm can be separated into two steps: The first consists to localize in the document all similar leaf-nodes of the tree pattern query nodes regardless of their hierarchical structure. This step is closely equivalent to the querying with unknown structures. The second is concern the sorting-out of the localized nodes and the validation of their hierarchical structure with tree pattern query. It is based neither on the matching tree nor query relaxation, but minimum spanning tree. The answers list with initially consist of all similar nodes to more important node in the tree pattern query. In every step of the algorithm, one node is added to each previous intermediate answer to obtain a new answers list, taking into account their hierarchical structure such as defined tree pattern query by user, In the contrary case, i.e. when the hierarchical structure of tree pattern query not respected, we use the minimum spanning tree criteria to choose the node to be added. A button-up approach is used to construct the tree answer progressively from the leaf-nodes to root-node and from the highest-priority to the lowest-priority.

6. Conclusion and perspective

In this paper, we have tackled the problem of querying XML documents. We have made a state of the art on the suggested methods, without claiming to make an exhaustive and very detailed study. In the fuzzy logic framework, we have proposed a method allowing to progressively building the answers by evaluating the possibility of finding the most relevant answer in a particular branch of the document tree. For that, we have used the minimum spanning tree.

In the continuation of this work, we will endeavour to initially make the necessary experiments within the PRETI–platform and INEX (INitiative for the Evaluation of XML Retrieval) in order to make a comparative study with the others proposed approaches.

References

[1] Amer-Yahia S., Cho S., Srivastava. D. Tree Pattern Relaxation, In EDBT, Czech Republic, 2002.

[2] Bengoetxea E.,Inexact Graph Matching Using Estimation of Distribution Algorithms, Ph.D Thesis, E.N.S.T Paris, 2002

[3] Chan K., Cheung Y., Fuzzy Attribute Graph with Applications to Character Recognition, IEEE Trans. Systems, Man and Cybernetics, 1992

[4] Bosc P., Prade H., An introduction to fuzzy set and possibility theory to the treatment of uncertainty and imprecision in database, In UMIS,1997, p. 285-324.

[5] Damiani E., Tanca L., Blind queries to XML data, Database and expert systems applications 2000, p.266-279.

[6] Klir G., Yuan B., Fuzzy Sets and Fuzzy Logic : Theory and Applications, edition: Paperback 1993

[7] Dubois D., Fargier H., Prade H.: Possibility Theory in Constraint Satisfaction Problems. Applied Intelligence, Vol. 6, 1996, p. 287-309

[8] Dubois D., Prade H., Sedes F., Fuzzy Logic Techniques in Multimedia Database Querying: A Preliminary Investigation of the Potentials, IEEE Trans. Knowl. Data Eng. 13(3) , 2001, p. 383-392.

[9] Debois D., Prade H., Using fuzzy sets in flexible querying: why and how, *Kluwer Academic Publishers* , 1997, P. 45 - 60

[10] Dubois D., Lang J., Prade H.. Possibilistic logic. In Handbook of Logic in AI and Logic Programming, (D. Gabbay et al., Oxford University Press 1994, p. 439–513.

[11] Prade H., Sèdes F.. Flexible querying of semi-structured data or documents. In 10th IFSA, 2003, p. 392-395.

[12] Mordeson J., Nair P.,Fuzzy Graphs and Fuzzy Hypergraphs, (Studies in Fuzziness and Soft Computing) Physica- Verlag, Heidelberg 2000

[13] Nierman A., Jagadish H., Evaluating Structural Similarity in XML Documents, in 5th International Workshop on the Web and Databases, 2002.

[14] Levennshtein V., Binary codes capable of correcting deletions, insertions, and reversals. Sov. Phys Dokl. 6, 1966, p.707-710.

[15] Graham R.L., Hell P., On the history of the spanning tree problem", IEEE Annals of the History of Computing, 7, 1985, pp. 8-23.