5J-04 Similarity Search of XML Documents Based on Tag Structures and Contents

Said Mirza Pahlevi † and Hiroyuki Kitagawa ‡

† Doctoral Program in Engineering, University of Tsukuba

‡ Institute of Information Sciences and Electronics, University of Tsukuba

1 Introduction

XML is a standard for data representation and exchange on the Internet. It is widely believed that in the near future Internet will be populated with a huge number of Web-accessible XML files – an "Internet Database". Therefore, searching for XML files will become as important as searching for HTML files like today. XML makes no official provision for constructing DTDs. For presenting one specific topic, each DTD author possibly will define his/her own DTD. As a result, there may be many XML documents with the same ontology, but with different DTDs. Furthermore, not all XML documents have DTDs. On the other hand, each document author probably has a different textual description for the same objects. For example, an author name could be expressed as "John Doe", "J. Doe" or "John Doe, et al.".

Recently, there are several XML query languages that have been proposed. One of the languages is XML-QL [1]. By using tag variables and regular expressions, this language can query even when the DTD is not fully known. But still, it is very hard to write such query that can cover many different DTDs. Furthermore, since well-formed XML documents have no DTDs, there is no way to write a meaningful query against that documents.

In this paper we describe a query evaluation method based on the structures and contents of XML documents. We combine the approximate tree matching algorithm [2] with the information retrieval (IR) similarity metrics [3] in order to rank XML documents based on how similar their structures and contents are to a given query structure and content. In Section 2, we describe query construction method. In Section 3, we describe query evaluation mechanism combining similarity metrics. We give our conclusion and future work in Section 4.

2 Querying XML Documents

As in conventional web search engines, to query the Internet Database a user has some *keywords* related to the information he needed. Since DTDs are not given, existing XML documents do not have DTDs or so many DTDs exist, he *does not know exactly* in what element names the keywords may exist. In this situation, he could decide some tag names that may be related to the keywords and organize the tags based on the concept/part-of relationship between them. For example, if a user wants to find books with keywords "White", "XML", and "Database", he probably will construct a query like this:

```
<book><author>White</author>
        <title>XML Database</title>
</book>
```

XML-QL will do exact pattern matching againts the structure and contents of the query. All documents matching the query should contain `<book>` element that has at least one `<author>` and `<title>` element. Therefore, documents with a slightly different structure/content from the query pattern will be considered as *not* relevant to it.

Just as the keyword-based search engines that rank Web pages based on how related the given keywords are to their contents, rather than exact matching, our query evaluation mechanism ranks XML documents based on how similar the structures and contents of the documents to the given query. While some situations demand "precise" query results, searching on online databases, such as the Internet, enable users to interactively browse results and submit refined queries. In these situations, approximate matching based on the structures and contents of XML documents can be very useful for focusing a search.

3 Query Evaluation Mechanism

The *edit distance* from tree T and T' is the minimum

cost sequence of edit operations needed to transform T into T'. There are three kinds of edit operations considered here: *changing* a node label to another, *deleting* and *inserting* a node. An edit operation is represented by a pair $n_i \rightarrow n_j$, where each of n_i and n_j is either a node or null. Let r be a cost function that assigns to each edit operation $n_i \rightarrow n_j$ a nonnegative real number $r(n_i \rightarrow n_j)$. Function r is constrained as a distance metric:(a) $r(n_i \rightarrow n_j) \geq 0$; $r(n_i \rightarrow n_i)=0$; (b) $r(n_i \rightarrow n_j)=r(n_j \rightarrow n_i)$; (c) $r(n_i \rightarrow n_j) \leq r(n_i \rightarrow n_k)+r(nk \rightarrow n_j)$. A mapping is a description of how a sequence of edit operations transform tree T into T' [2]. A dotted line from node n_i in T to node n_j in T' indicates that n_i should be changed to n_j if $n_i \neq n_j$, or that n_i remains unchanged if $n_i = n_j$. Nodes of T not touched by dotted lines are to be deleted and nodes of T' not touched are to be inserted.

Query evaluation is done by comparing structure and content of a query with the structures and contents of documents. The evaluation result is XML documents ranked by their similarity score to the query. We use *approximate tree matching algorithm* combining with similarity metrics to rank the documents. For doing this, query and documents are modeled by the labeled ordered tree data model. In the model, intermediate and leaf nodes are labeled by *element names* and *XML data/keywords*, respectively. The similarity score between query q and document d is defined as the minimum cost sequence of edit operations needed to transform q into d. We modified the algorithm such that *Change* cost between nodes is a function of *similarity distance* between node labels.

To calculate the *similarity distance*, we use the similarity metrics from IR. The label of node n_i is represented by a term vectors of the form $l_i=(w_{i1},w_{i2},\ldots,w_{it})$. Weight w_{ij} represents the values of term j in the label of node i, and given by the formula $w_{ij}=tf_{ij}\log(N/df_j)$, where tf_{ij} is the number of occurrence of term j in the label of node i, N is the total number of nodes in a collection of XML documents, and df_j is the number of nodes in a collection of N nodes in which term j occurs. Let W_{ij} be a normalized w_{ij}, $simdist(n_i, n_j)$ be a *similarity distance* between nodes n_i and n_j, and $r_{change}(n_i \rightarrow n_j)$ be a cost for changing node n_i to n_j:

$$W_{ij}=w_{ij}/(\Sigma_{j=1\sim t}(w_{ij})^2)^{1/2} \quad (1)$$

$$simdist(n_i, n_j)=(\Sigma_{k=1\sim t}|W_{ik}-W_{jk}|^2)^{1/2}/(t)^{1/2} \quad (2)$$

$$r_{change}(n_i \rightarrow n_j)= simdist(n_i, n_j) \quad (3)$$

Let nodes n_r and n_s be in the same path from root of a tree T. A *node path* with the first node n_r and the last node n_s is a set of nodes, $\{m_1, m_2,\ldots, m_k\}$, such that we can traverse from n_r to n_s a path of k-1 edges $(e_1.e_2\ldots e_{k-1})$ through k nodes $(m_1=n_r, m_2,\ldots, m_k=n_s)$ where edge e_i connects nodes m_i and m_{i+1}.

Let M be a mapping from query q to document d, and let $p_1,\ldots, p_r$ be *node paths* such that the first and the last node of each path are touched by a line of M. Let d' be a subtree of d constructed by $p_1,\ldots, p_r$. Let I and J be the node sets of q and d', respectively, not touched by any line in M. The similarity score between q and d is given by:

$$SimScore(q, d) = \Sigma_{(ni, nj)\in M} r_{change}(n_i \rightarrow n_j) + \Sigma_{ni\in I} r_{delete}(n_i) + \Sigma_{nj\in J} r_{insert}(n_j) \quad (4)$$

The first right term corresponds to the total of *similarity distances* between nodes mapped by M. The second term corresponds to the deletion of a query parts that cannot be mapped to a document. The third right term corresponds to the insertion of the d' part not exist in q.

4 Conclusions and Future Work

In this paper we propose a query evaluation method that ranks XML documents based on their similarity to a query. In order to match keywords given by users to the contents of XML documents, we put a similarity metrics from IR into the algorithm. For future work, we are going to evaluate the effectiveness of the evaluation mechanism.

References

[1] A. Deutsch et al., "XML-QL: A Query Language for XML", Proc. of the Eighth Int. W3C, Toronto, 1999.

[2] K. Zhang et al., "Approximate Tree matching in the Presence of VLDC", J. Algorithms, 16(1):33-66, 1994.

[3] G. Salton, "Automatic Text Processing", Addison-Wesley, 1988.