Chapter 9

XTEngine:
A Twin Search Engine for XML

Kamal Taha
The University of Texas at Arlington, USA

ABSTRACT

There has been extensive research in XML Keyword-based and Loosely Structured querying. Some frameworks work well for certain types of XML data models while fail in others. The reason is that the proposed techniques overlook the context of elements when building relationships between the elements. The context of a data element is determined by its parent, because a data element is generally a characteristic of its parent. Overlooking the contexts of elements may result in relationships between the elements that are semantically disconnected, which lead to erroneous results. We present in this chapter a context-driven search engine called XTEngine for answering XML Keyword-based and Loosely Structured queries. XTEngine treats each set of elements consisting of a parent and its children data elements as one unified entity, and then uses context-driven search techniques for determining the relationships between the different unified entities. We evaluated XTEngine experimentally and compared it with three other search engines. The results showed marked improvement.

INTRODUCTION

Extensive research has been done in keyword querying using relational data (Agrawal & Chaudhuri & Das, 2002; Aditya & Bhalotia & Sudarshan, 2002; Hristidis & Papakonstantinou, 2002). Research in XML querying has significant boost with the emergence of World Wide Web, online businesses, and the concept of ubiquitous computing. Some of these works model XML data as a rooted tree (Liu & Chen, 2007; Xu & Papa-konstantinou, 2005; Li & Yu & Jagadish, 2004; Cohen & Mamou & Sagiv, 2003). Others, model it as a graph (Cohen & Kanza, 2005; Balmin &
Hristidis & Papakonstantinou, 2003; Balmin & Hristidis & Papakonstantinou, 2004; Botev & Shao & Guo, 2003). Most of these works target either: (1) naïve users (such as business’ customers) by proposing Keyword-based search engines, or (2) sophisticated users, by proposing fully structured search engines.

Business’ customers are most likely not aware of the exact structure of the underlying data. On the other hand, business’ employees are likely to be aware of some labels (or attributes) of elements containing data, but they are unlikely to be fully aware of the underlying data structure. Thus, business’ customers need a pure Keyword-based search engine, while business’ employees need a Loosely Structured search engine for answering their queries. A Loosely Structured query combines keywords and element names. We propose in this chapter: (1) an XML Keyword-based search engine called XTEngine-K for answering business’ customers, and (2) an XML Loosely Structured search engine called XTEngine-L for answering business’ employees. Consider that the user wants to know the data \( D \), which is contained in an element labeled \( E \). If the user knows only the keywords \( k_1, k_2, ..., k_n \), which are relevant to \( D \), he/she can submit a Keyword-based query to XTEngine-K in the form: \( Q(\ "k_1", \ "k_2", \ ..., \ "k_n\" ) \). If, however, the user knows the label \( E \) and the labels \( l_1, l_2, ..., l_n \) (which are the labels of the elements containing the keywords \( k_1, k_2, ..., k_n \) respectively), but this user is unaware of the structure of the data, he/she can submit a Loosely Structured query to XTEngine-L in the form: \( Q(\ l_1 = \ "k_1\" , \ ..., \ l_n = \ "k_n\" , \ E\ ) \). XTEngine is built on top of XQuery search engine (Katz, 2005).

A few works (Cohen & Mamou & Sagiv, 2003; Li & Yu & Jagadish, 2004) have proposed XML search engines that answer both Keyword-based and Loosely Structured queries. Other works (Liu & Chen, 2007; Xu & Papakonstantinou, 2005) have proposed XML search engines that answer only Keyword-based queries. Computing the Lowest Common Ancestor (LCA) of elements containing keywords is the common denominator among these engines. Despite their success, they suffer recall and precision limitations. As we will show in the coming sections, the reason for these limitations stems from the fact that these engines employ mechanisms for building relationships between data elements based solely on their labels and proximity to one another, while overlooking the contexts of these elements. In our work, the context of a data element is determined by its parent, because a data element is generally a characteristic of its parent element. If for example a data element is labeled “title”, we cannot determine whether it refers to a book title or a job title without referring to its parent. Consider as another example that an XML document containing two elements labeled “name”, one of them refers to the name of a student, while the other refers to the student’s school name. Building a relationship between these two “name” elements without consideration of their parents may lead to the incorrect conclusion that the two elements belong to the same type. Building relationships between data elements while overlooking their contexts may lead to relationships that are semantically disconnected. Consequently, the results generated by non context-driven systems are susceptible to errors, especially if the XML document contains more than one element having the same label but representing different types of information or having different labels but representing the same type. XTEngine-L and XTEngine-K avoid the pitfalls cited above of non context-driven search engines by employing novel context-driven search techniques. These techniques consider each set of elements in an XML tree consisting of a parent and children data elements as one unified entity.

**Example 1**: Consider that an XML tree containing interior elements labeled book, job, student, and school. Consider that each of elements book and job has a child data element labeled title, and each of elements student and school has a child data element labeled name. XTEngine-L and