XML Schema Search Engine
Zhishuang Zhang
Faculty Advisor: Val Tannen, Zackary Ives

Abstract:

- XML has become important as a data storage and interchange format, because of its properties of being able to describe structured data, platform independence, human readable. This project is aiming to build a search engine for XML Schema, in order to help people find XML data.
- This project does not contain a web crawler. In other words, this project is focusing on how to maintain enough information about the structure of XSD in the index database and search against the database, instead of how to find XSD on the internet by a crawler.
- Apache Lucene (http://lucene.apache.org) is a high-performance, full-featured text search engine library written entirely in Java. Lucene’s indexing data structure and the searcher are used in this project.

Query Format:

- The query format used in this project is a little more complicated than keywords query format. The search engine also allows user to specify the relation between keywords in the query string. Take the following XML Schema as a example. If the query string is “city/province/name” or “city/name”, the XML Schema below will be considered as match. “/” is specifying the parent-child relation, and “//” is specifying the ancestor-descendents.

```
<xs:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xs:complexType>
    <xs:attribute name="id" type="xsd:ID" use="required" />
    <xs:attribute name="longitude" type="xsd:double" use="optional" />
    <xs:attribute name="latitude" type="xsd:double" use="optional" />
    <xs:attribute name="country" type="xsd:IDREF" use="optional" />
  </xs:complexType>
  <xs:element name="city">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="id" type="xsd:ID" use="required" />
        <xs:attribute name="name" type="xsd:string" use="required" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
  depth  start  end
  city        0     0
  id          0     1
  longitude   0     2
  latitude    0     3
  country     0     4
  province    0     8
  id          1     2
  longitude   1     3
  latitude    1     4
  id          2     7
  name        2     8
```

- The whole .xsd files will be displayed with highlighted keywords including exact matched keywords and fuzzy matched keywords by clicking on the links in the search result.
- The whole .xsd files will be displayed with highlighted keywords including exact matched keywords and fuzzy matched keywords by clicking on the links in the search result.

Parse and Index XML Schemas:

- XML Schema has the same format at XML file. Therefore, the parse used in this project uses the XML parser in “org.xml.sax” package which is a event driven parser and traversing XML files from top to bottom.
- The indexing data structure Lucene provided is a tab based structure. It will be very easy to index keywords and search by keywords. We can just put “tag:keyword” pairs into index. However, the challenge here is to preserve enough information about the structure of the XML Schema and support structure search.
- XML Schema has the same structure as XML which is basically a tree structure. The schema mentioned before can be represented by the following tree structure.

```
  <city>
    <id></id>
    <longitude></longitude>
    <latitude></latitude>
    <country></country>
    <province></province>
  </city>
```

- Next step is how to store the information about this tree structure into tag based structure. The information stored should be sufficient to determine “/” and “//” relations between nodes. The method used in this project is, first numbers each node and stores the depth information; then when the parser gets to the end of node, stores the range of all descendents. The result can be represented by the graph below.

```
  Depth:0  Start:1  End:1
  Depth:1  Start:2  End:2
  Depth:1  Start:3  End:3
  Depth:1  Start:4  End:4
  Depth:1  Start:5  End:5
  Depth:2  Start:6  End:6
  Depth:2  Start:7  End:7
  Depth:2  Start:8  End:8
```

- According to the graph above, for any given two nodes, if one node’s id is in the range of (start, end), these two nodes are in “/” relation. Furthermore, if the depth of the two nodes are differed by one, these two nodes in “//” relation.

Scoring:

- All documents either match the whole query string, sub-query, some keywords or fuzzy match some keywords, will be returned. Documents will be sorted by their scores. Scores are based no term frequency and weight of the term. A structure term is weighted more than a keyword term.

Experimental Setup:

- All XML Schemas for are downloaded from some XML database websites. Schemas are across different subjects, including geometry, geography, biology, online treading, and so on.

Experimental Results:

- The whole .xsd files will be displayed with highlighted keywords including exact matched keywords and fuzzy matched keywords by clicking on the links in the search result.

---

Senior Project Poster Day 2006, CIS Dept. University of Pennsylvania