Tutorial 5: XML and XPath

Informatics 1 Data & Analysis

Week 7, Semester 2, 2016/17

This worksheet has three parts: tutorial *Questions*, followed by some *Examples* and their *Solutions*.

- Before your tutorial, work through and attempt all of the Questions in the first section. If you get stuck or need help then ask a question on *Piazza*.
- The Examples are there for additional preparation, practice, and revision.
- Use the Solutions to check your answers, and read about possible alternatives.

You must bring your answers to the main questions along to your tutorial. You will need to be able to show these to your tutor, and may be exchanging them with other students, so it is best to have them printed out on paper.

If you cannot do some questions, write down what it is that you find challenging and use this to ask your tutor in the meeting.

Tutorials will not usually cover the Examples, but if you have any questions about those then write them down and ask your tutor, or post a question on *Piazza*.

It's important both for your learning and other students in the group that you come to tutorials properly prepared. Students who have not attempted the main tutorial questions will be sent away from the tutorial to do them elsewhere and return later.

Some exercise sheets contain material marked with a star $\star$. These are optional extensions.

Data & Analysis tutorials are not formally assessed, but the content is examinable and they are an important part of the course. If you do not do the exercises then you are unlikely to pass the exam.

Attendance at tutorials is obligatory: if you are ill or otherwise unable to attend one week then email your tutor, and if possible attend another tutorial group in the same week.

Please send any corrections and suggestions to Ian.Stark@ed.ac.uk

Introduction

This tutorial exercise is about XML, using material covered in Lectures 9–11. There are several stages, during which you will write some expressions (DTD, XPath) and run some command-line tools (validating a document against a DTD, executing XPath expressions). Please start on this exercise in good time to complete it before the tutorial. If you need assistance then post on Piazza, send me email, or ask after a lecture.

You will need to download two sample files from the course web page, `restaurants.xml` and `restaurants-queries.txt`, and you will also need access to a DICE command line. The

simplest way to get this is in the computing labs, but it's also possible to connect remotely. For more information about remote working see the relevant pages from Informatics Computing Support.

http://computing.help.inf.ed.ac.uk/connecting-home-overview

Question 1: Trees and XML

Draw the tree structure of the XPath data model for the following document:

```
<restaurants>
   <restaurant name="La Pasteria">
      <address>13 Alonisou Street, Patras, 261 35</address>
      <cuisine>Italian</cuisine>
      <phoneno>2610325833</phoneno>
   </restaurant>
   <restaurant name="Kalamarakia">
      <address>21 Poseidonos Street, Patras, 264 45</address>
      <cuisine>Greek</cuisine>
      <phoneno>2610428066</phoneno>
   </restaurant>
</restaurants>
```

Question 2: DTD and XML

Design a DTD for an XML database of restaurants. This should be written to validate the small XML example above, and also the longer document `restaurants.xml` available from the course web page.

Question 3: Validating an XML file with a DTD schema

Validate the given `restaurants.xml` file against your DTD. You can do this using the `xmllint` command-line tool. This is installed as standard on DICE machines, and is readily available for other platforms. First make sure that your XML and DTD files are linked, either by adding your DTD directly to the `restaurants.xml` file with an inline DOCTYPE declaration or by creating a separate DTD file and adding a DOCTYPE reference line to `restaurants.xml`. Then run the following command in a terminal window on a DICE machine.

```
xmllint --noout --valid restaurants.xml
```

This will check that the XML file is well-formed and that it validates against the DTD you have given. If the document is valid, you get no response; if invalid, you get an error message. Try changing parts of the XML and the DTD to see how `xmllint` responds.

Question 4: Writing XPath expressions

Write path expressions to extract the following textual information from any XML document that matches the restaurants DTD you gave for Question 2.

(a) Retrieve the addresses of all restaurants.

(b) Retrieve the names of all restaurants.

(c) Retrieve the address of the *Kalpna* restaurant.

⋆ **(d)** Retrieve the telephone numbers of all restaurants serving Indian food.

⋆ **(e)** Retrieve the names of all restaurants which serve Italian food.

Question 5: Executing XPath expressions

In this part you will execute your XPath expressions against the `restaurants.xml` file provided using the `cat` command of `xmllint`. To do this, you first need to enter the shell mode of `xmllint` by running the following command in a terminal window on a DICE machine.

```
xmllint --shell restaurants.xml
```

This will leave you at a prompt reading "`/ >` ". The slash "`/`" shows that you are at the root node of the document. To leave this shell, type `quit`, `bye` or `exit`. However, before you do that, test the XPath expressions that you wrote for Question 4. You can do this by typing

`cat` ⟨*path expression*⟩

into the `xmllint` shell. This will print out nodes matching the path expression, and the subtrees beneath them.

Finally, copy-and-paste your queries and solutions into the file `restaurants-queries.txt`. Print this file and bring it to the tutorial.

Question 6: XML and relational databases

How would you represent this data in a relational database? Use the SQL data definition language (DDL) to define an SQL schema for representing the same data.

Question 7: Discussion

(a) Suggest some strengths and weaknesses of the relational approach compared to XML.

(b) Given these strengths and weaknesses, name a context in which you would prefer to use XML.

(c) Given these strengths and weaknesses, name a context in which you would prefer to use a relational database.

⋆ Question 8: Extension

Write a Java program to carry out the queries from Question 4. There are at least two different ways you could approach this.

- Use an XML parser to read in the file as a tree and then navigate that tree to find the information required.
- Use an XPath API to directly execute your answers from Question 4.

In either cases you will need to choose among an alphabet soup of existing libraries that connect Java to XML and XPath — each with their own advocates and enthusiasts — such as JAXP, JDOM, SAX, StAX, XMLBeans, JiBX, ... One possible route is to start with `javax.xml.xpath`, which is a standard Java package, and read the documentation about it from Oracle.

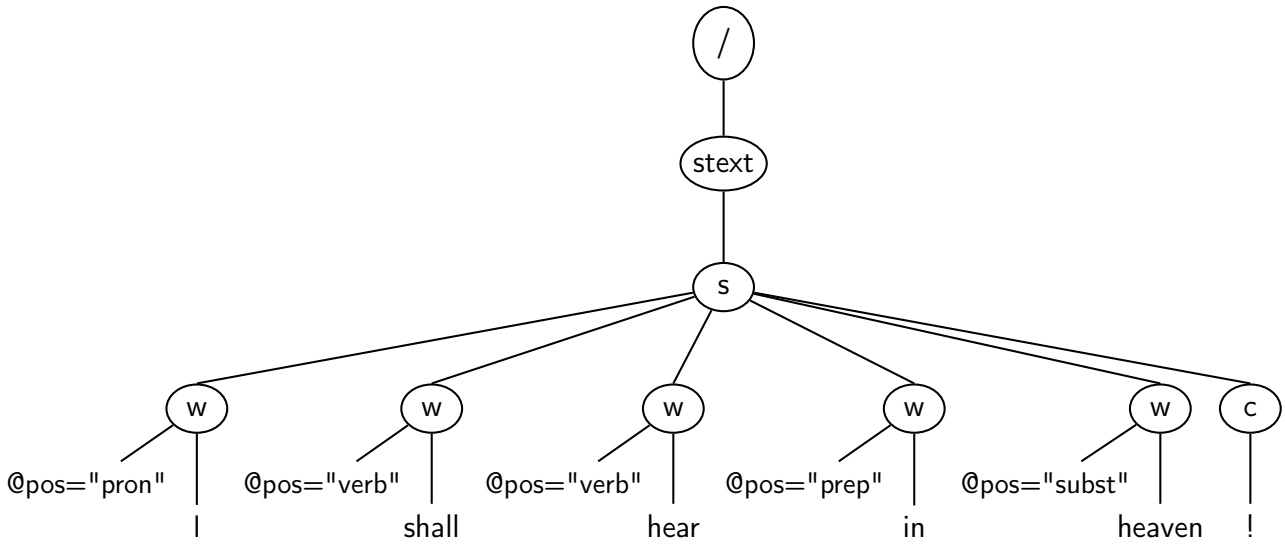

Figure 1: An XML tree

Examples

This section contains further exercises on XML. These examples are similar to the main tutorial questions: they involve writing an XML document based on an XPath data tree structure, specifying its schema with a DTD and writing XPath expressions to formulate queries of interest. You will need two files from the course web page, `stext.xml` and `stext-queries.txt`, and you will also need access to a DICE command line.

The tree in Figure 1 shows an XML document, drawn according to the XPath data model. It represents a passage of spoken text annotated according to (a simplification of) the mark-up scheme of the *British National Corpus*. This annotates text by grouping it into sentences, indicating individual words and punctuation, along with "part of speech" (pos) information — whether a word is a verb, a substantive (a noun), preposition, and so forth.

Example 1: Trees and XML

Write out the tree in Figure 1 as an XML document.

Example 2: DTD and XML

Write a DTD to specify the XML structure of XML documents for spoken text in the format of Figure 1. You should allow for a passage of spoken text to possibly consist of several sentences.

Example 3: Validating an XML file with a DTD schema

Validate the given `stext.xml` file against your DTD. You can do this using `xmllint` on a DICE machine, following the instructions given in Question 3.

Example 4: Writing XPath expressions

Write XPath expressions to return the following lists of text strings from any XML document that is valid with respect to the DTD you wrote for Example 2.

(a) All punctuation marks.

(b) All substantives. The pos tag for substantives is "SUBST".

(c) All words appearing in sentences that contain an exclamation mark "!".

(d) All substantives appearing in sentences that contain an exclamation mark "!".

(e) All punctuation marks of sentences that contain the word "face".

Example 5: Executing XPath expressions

Use `xmllint` to execute your XPath expressions on the `stext.xml` file provided, by following the same procedure as described in Question 5. Fill out the file `stext-queries.txt` by recording both query and response.

Solutions to Examples

These are not entirely "model" answers; instead, they indicate a possible solution. Remember that not all of these questions will have a single "right" answer. If you have difficulties with a particular example, or have trouble following through the solution, please raise this as a question in your tutorial.

Solution 1

The XML document is:

```
<stext>
   <s>
      <w pos="pron">I</w>
      <w pos="verb">shall</w>
      <w pos="verb">hear</w>
      <w pos="prep">in</w>
      <w pos="subst">heaven</w>
      <c>!</c>
   </s>
</stext>
```

Solution 2

The DTD below is one possibility.

```
<!ELEMENT stext (s+)>
<!ELEMENT s ((w|c)+)>
<!ELEMENT w (#PCDATA)>
<!ELEMENT c (#PCDATA)>
<!ATTLIST w pos CDATA #REQUIRED>
```

Variations, such as using "$*$" instead of "+" would also be reasonable here. Another alternative would be to give a specific list of allowed values for "pos".

Solution 3

There are two ways to link XML documents and DTDs. The first is to include the complete document type declaration within the XML file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE stext [
<!ELEMENT stext (s*)>
...
<!ATTLIST w pos CDATA #REQUIRED>
]>
<stext>
...
</stext>
```

This solution is used in the file `stext-sol.xml`. The second method is to put the DTD in a separate file, say `stext.dtd`, and then reference it from the XML.

```
<?xml version="1.0" encoding="UTF−8"?>
<!DOCTYPE stext SYSTEM "stext.dtd">
<stext>
...
</stext>
```

Here the SYSTEM keyword indicates that the DTD can be found at the address named — in this case, a local file.

Solution 4

XPath expressions:

(a) All punctuation marks.

```
//c/text()
```

(b) All substantives.

```
//w[@pos = "SUBST"]/text()
```

(c) All words appearing in sentences that contain an exclamation mark "!".

```
//s[c="!"]/w/text()
```

Alternatively, with some rather roundabout navigation:

```
//w[../c="!"]/text()
```

(d) All substantives appearing in sentences that contain an exclamation mark "!".

```
//s[c="!"]/w[@pos="SUBST"]/text()
```

Again, written to navigate everywhere from the chosen word:

```
//w[../c="!"][@pos="SUBST"]/text()
```

(e) All punctuation marks of sentences that contain the word "face".

```
//s[w="face"]/c/text()
```

Solution 5

The file `stext-queries-sol.txt` available from the course web page contains all the queries given as answers to Example 4.