

IM:XMLIR Assignment

miniGoogle

The aim of this assignment is to build a mini Google-like ranking system for a set of XHTML files that are provided. The assignment also tests your skills in the use of [XML](#) technology as a means to manipulate the source files.

Test Data:

About 500 XHTML files, contained in [testdata.zip](#).

Recommended Tools:

Xerces for [XML](#) DOM parsing and validation, Xalan for XSL Transformation. Contained in [jars.zip](#).

Sample files:

TEST [XML](#)/XSD/XSL/JAVA/CLASS to demonstrate how to bootstrap the DOM parser and provide a fully working, validifiable and transformable dataset. “validate”, “transform” and “domtest” shell scripts. Contained in [script_tools.zip](#).

Submit:

A single tar.gzip archive file containing all required files as indicated below.

Question 1

You are required to do the following:

1. Write a program to use a DOM parser to extract and print
 - the list of *href* attributes associated with every anchor (A) tag in an XHTML file,
 - the list of H1-H3 headings in the document in the order in which they appear, and
 - the title of the document, as found within the head element.
2. Write an XSLT stylesheet to convert an XHTML file into a list of whitespace-separated words.
3. Write an XSLT stylesheet to convert an XHTML file into a simple Google-like XHTML fragment summary with the title of the document as a link followed by a summary of the document. You should ideally strip out all structural markup, thereby flattening the text as much as possible. Enclose the entire summary in a P tag.
4. Write an [XML](#) Schema corresponding to your simplified XHTML summary, to enforce that each [XML](#) document has no structural markup except what you put in explicitly.

Question 2

You are required to do the following:

Write a program to parse the XHTML files, extract links and apply the naive PageRank algorithm to rank pages. Output a simple HTML file with each document listed as its summary (using the stylesheet from Q1.3), in order from highest to lowest rank.

- To deal with rank leaks, set $N=0.0000001$ in cases where $N=0$.
- Ignore links that start with *http* or contain ':' or '#'.
- You may modify the previous stylesheet if you wish to include rank scores for each item.

Question 3 (Optional)

Write a program to generate inverted indices from the lists of words from Q1. Then write a program to execute simple boolean queries using these inverted indices, utilising the pre-computed ranks from Q2.

Submission

Submit a gzipped/tarred file containing at least the following structure:

- README
- q1.1
 - all files used in DOM parsing application
- q1.2
 - wordlist.xml
- q1.3
 - summary.xml
- q1.4
 - summary.xsd
- q2
 - index.html (generated as output from program)
 - all files that comprise ranking program

Write a short report on the algorithms and design decisions you made, as well as the different modules/programs of your solution and how to use them. Include sample output for 3 files used as input to each of the 4 parts of Question 1. Excluding the output, your report should be at most 3 pages long (12pt Times New Roman body text, single line spacing, 2.5cm margins all around).

General Notes

You may use Java, C++ and/or Perl, and associated [XML](#) processing tools if you wish. (Talk to the TA first if you would like to use a different language/set of tools)

Your assignment will be marked according to the following criteria:

- Correctness – Q1.1-Q1.4 x 6 + Q2 x 16 (40%)
- Output – presence in report and reasonableness (10%)
- Documentation – comments within programs, descriptive report (20%)
- Efficiency (10%)
- Stress – tutors testing your code with different test data (10%)
- Creativity – extra effort or optional part (10%)

Final Note: Please check your results to make sure they make sense!