XML and Structured Information: Coursework 1

Deadline: 14th of November at 11:59 pm (UK time)

This coursework involves writing some XQuery expressions to extract useful information from some real XML data. The coursework must be performed individually.

Download the example files `cw2files.zip`. These are real datafiles from the British National Corpus, a collection of English language data widely used by linguists and computational linguists, together with a schema. There is a full explanation of the schema[1], but the important parts here are:

- `<s>` represents a sentence, containing possibly many children `<w>` representing words in linear order;
- `<w>YYY</w>` represents word `YYY`.

A 12.5 marks

Produce a `extract.xquery` file containing a XQuery FLWOR expression which returns all the occurrences of the word *'has'* in the collection of files, together with the word which comes next in the sentence in each case. The resulting list should be formatted as a HTML table, with each row containing the two words in their own cells, e.g.:

Target	*Successor*
has	there
has	n't
has	n't
...	...

Hints You can use the `collection()` function to load all the corpus files together. The XML files often contain capital letters and stray whitespace within the `<w>` element text values, so the `normalize-space()` and `lower-case()` XPath functions may be useful.

B 12.5 marks

The results from A will contain many duplicates. Produce another version (`freqs.xquey`) which returns only unique results, together with the number of times this combination of words occurred, sorted in descending order of frequency:

Target	*Successor*	*Frequency*
has	been	39
has	n't	15
has	a	15
...	...	...

[1] `http://www.natcorp.ox.ac.uk/docs/URG.xml?ID=bnctags`

Hints You can use the `distinct-values()` function to obtain only unique values from a sequence. You can intersperse `let` and `for` clauses in XQuery, and the right-hand-side of either can itself be a FLWOR expression.

C 12.5 marks

Corpus linguists are often interested in finding whether pairs of words occur together more often than would be expected by chance (these are called *collocations*). Produce a modified version of your query from B (`probs.xquery`) which replaces the simple frequency count with the probability that the successor word occurs after the target. You can calculate this probability as the ratio: (number of times successor word appears after target) / (number of times successor word appears overall).

Target	*Successor*	*Probability*
has	begun	1
...	...	...
has	gained	0.67
...	...	...

Hints To divide one number by another in XPath, use the `div` operator, not the / symbol.

D 12.5 marks

You'll notice that the results from C have a long *tail* - a large number of words with low probabilities of co-occurrence (mostly words which are just quite common, so they appear often after the target word *'has'*, but only because they appear quite often anyway). Produce a final version (`top20.xquery`) which limits the number of results to the top 20 (i.e. the first 20 results returned in C).

E Submission

Your submission has to include the following files:

- `extract.xquery`
- `freqs.xquery`
- `probs.xquery`
- `top20.xquery`