

XML CW2 Report

Zhanelya Subebayeva

List of files:

1. xml-files folder – contains XML input files provided for the assignment
2. bncxml.xsd – schema for XML input files
3. cw1.xquery, cw2.xquery, cw3.xquery and cw4.xquery contain the solution for CW questions
4. style.css – contains table styles
5. output1.html, output2.html, output3.html and output3.html contain result of applying XQuery transformations from files cw(1-4).xquery
6. XML_CW2_Report_Zhanelya_Subebayeva.pdf – report

To implement the solution for this assignment, I used Oxygen XML editor v.16. The solution was developed under Windows7 operating system and tested to an extent on Linux. To run any of the XQuery transformations I created a new transformation scenario called “cw”, which uses “\${currentFileURL}” as XML URL and “\${currentFileURL}” as XQuery URL. It also uses Saxon-PE XQuery 9.5.1.5 transformer. To run the transformation, launch it from any of the XQuery files provided, and it will dynamically link the input XML files from the xml-files folder.

All source files provided have detailed comments within the code, so, only general implementation decisions will be covered in this report.

A) cw1.xquery

To return all the occurrences of the word 'has' in the collection of files and the word following it in the sentence in each case, first, I needed to find a way to load multiple XML file into transformation operation. This was done by using a relative path to direct XQuery transformation into xml-files folder and an asterisk character to get all files with .xml extension:

```
for $s in collection('xml-files?select=*.xml')/s
```

In order to find all the 'has' words within the input, I had to normalize the result to avoid data loss:

```
(lower-case(normalize-space($w/text()))) eq 'has'
```

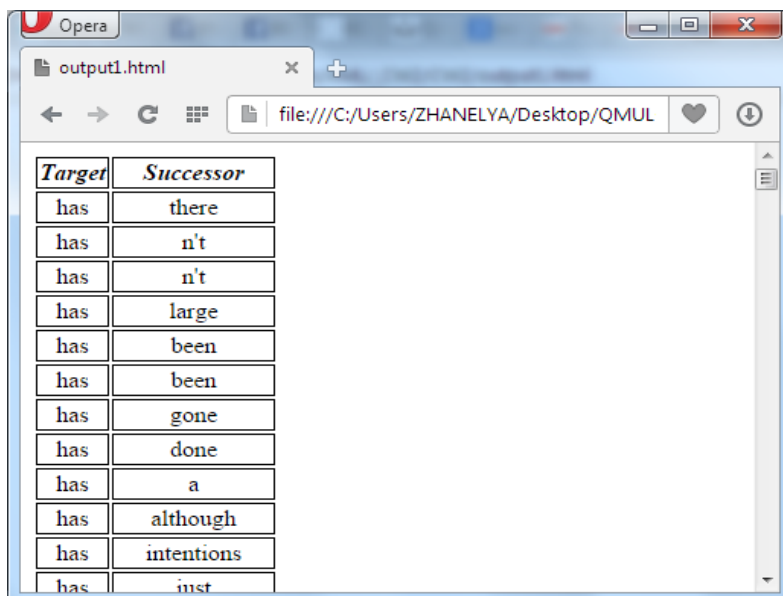
To find the successor of the word 'has' I used “.>> \$w”, which signifies “any word that has position larger than the word 'has’”, and to take only its first successor, I used [1]:

```
(data($s/w[.>> $w][1]))
```

Finally, to output the result in a form of table row, the following return statement was used:

```
return
  <tr>
    <td> {data($w)} </td>
    <td> {$succ} </td>
  </tr>
```

The result for this question looks as following:



Target	Successor
has	there
has	n't
has	n't
has	large
has	been
has	been
has	gone
has	done
has	a
has	although
has	intentions
has	inst

The code for Part A of this assignment was then reused in all of the following questions.

B) cw2.xquery

As there were many duplicates in the result from Part A, I produced another version that produces only unique pairs and the number of occurrences of each combination of the words. First, I formed a list of tuples containing 'has' and its successor – **\$has_succ_tuple**, which essentially reuses the code from Part A, but instead of returning table rows, forms tuples:

```
<tuple> {lower-case(normalize-space($w/text()))} {''} {lower-case(normalize-space($succ))} </tuple>
```

Then, I filtered it through by using `distinct-values()` function:

```
distinct-values($has_succ_tuple/text())
```

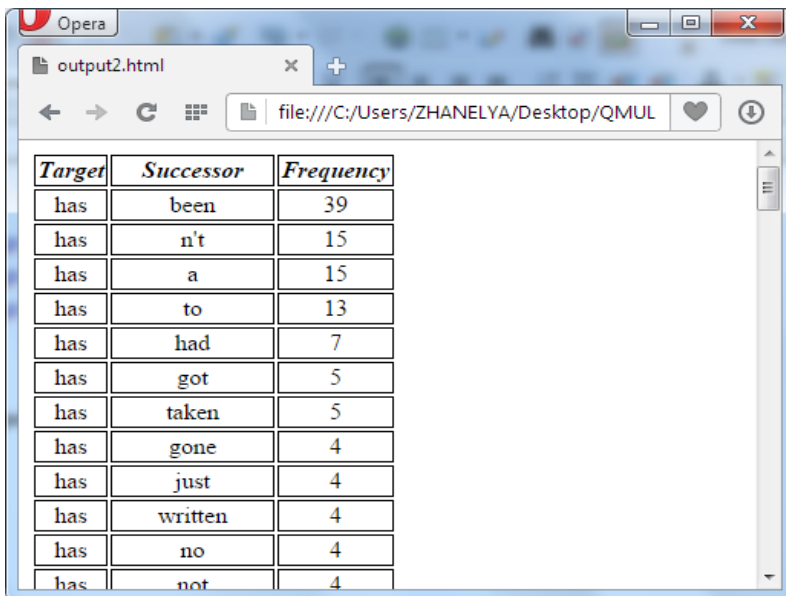
And counted all the occurrences of each tuple:

```
let $count := count($has_succ_tuple[text() = $distinct])
```

Finally, the result was sorted in descending order by the number of occurrences (frequency) and returned in a form of table:

```
return  
  <tr>  
    <td>{$arr[1]}</td>  
    <td>{$arr[2]}</td>  
    <td>{$count}</td>  
  </tr>
```

The result for this question looks as following:



The screenshot shows a web browser window with a single tab titled 'output2.html'. The address bar shows the file path 'file:///C:/Users/ZHANELYA/Desktop/QMUL'. The main content area displays a table with three columns: 'Target', 'Successor', and 'Frequency'. The table contains 14 rows of data, sorted by frequency in descending order.

Target	Successor	Frequency
has	been	39
has	n't	15
has	a	15
has	to	13
has	had	7
has	got	5
has	taken	5
has	gone	4
has	just	4
has	written	4
has	no	4
has	not	4

C) cw3.xquery

For this question I produced a new version of the transformation and replaced frequency with probability that the successor word occurs. To calculate the probability, I needed to divide the

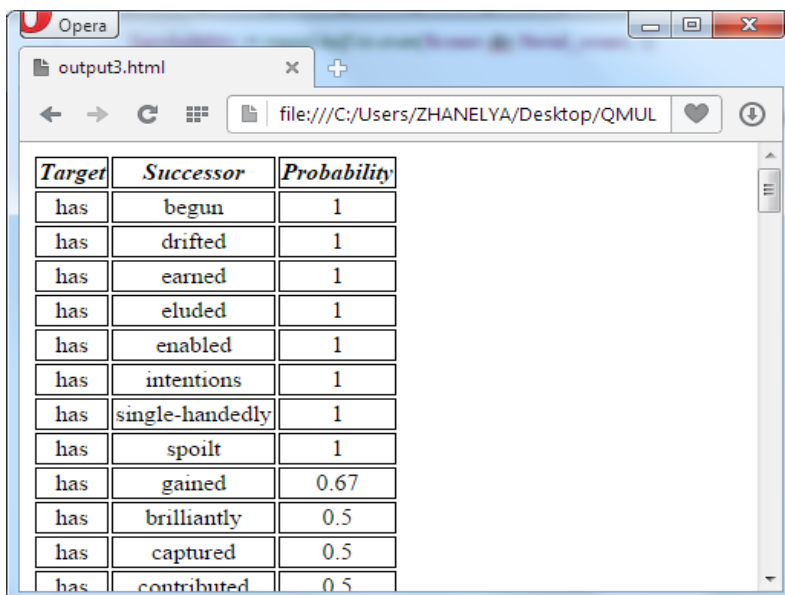
number of times successor word appears after target (word 'has') by the number of times successor word appears overall in input. To achieve this, I used `$total_count`, which calculates the number of occurrences of the successor within the files in a similar manner to calculating `$total_count`, but by a use of `$all_words` that stores all the words from the input. Then, I rounded probability result to 2 decimal points:

`$probability := round-half-to-even($count div $total_count, 2)`

Ordered it in descending order by probability and in ascending order by successor, and returned the result in a form of table:

```
order by $probability descending, $sarr[2] ascending
return
    <tr>
        <td>{$sarr[1]}</td>
        <td>{$sarr[2]}</td>
        <td>{$probability}</td>
    </tr>
```

The result for this question looks as following:



The screenshot shows a web browser window with a single tab titled 'output3.html'. The address bar shows the file path 'file:///C:/Users/ZHANELYA/Desktop/QMUL'. The main content area displays a table with three columns: 'Target', 'Successor', and 'Probability'. The table contains 14 rows of data, sorted by probability in descending order. The first 8 rows have a probability of 1, the next row has 0.67, and the last four rows have 0.5.

Target	Successor	Probability
has	begun	1
has	drifted	1
has	earned	1
has	eluded	1
has	enabled	1
has	intentions	1
has	single-handedly	1
has	spoilt	1
has	gained	0.67
has	brilliantly	0.5
has	captured	0.5
has	contributed	0.5

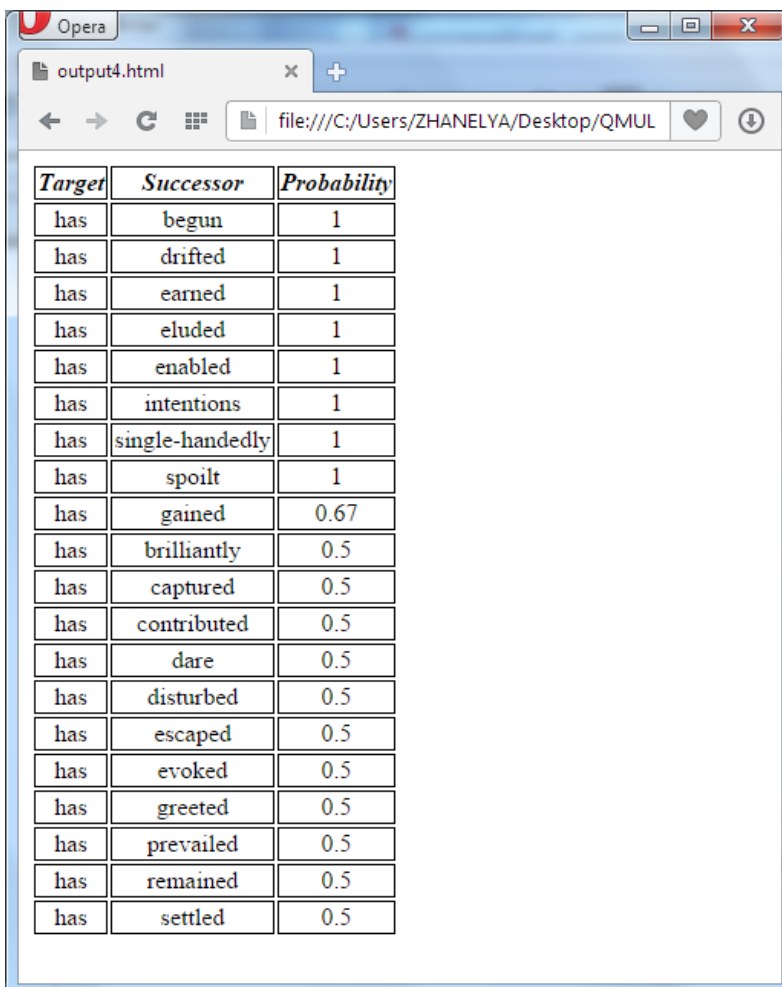
D) cw4.xquery

I noticed that the results from Part C are too long – and in the end there is a large number of words that have low, almost negligible probabilities of co-occurrence. Mostly, these are the words that are

very commonly used, for instance “i”, “the”, “an”. That is why, the final version of transformation that I produced, limits the number of results to the top 20. To do this, I assigned the value of the probability calculation to **Scale_probability** variable and then used `subsequence(var, start_point, length)` function to return only 20 rows:

```
return
  for $x in subsequence($Scale_probability, 1, 20)
    return $x
```

The result for this question looks as following:



The screenshot shows a web browser window with the address bar displaying 'file:///C:/Users/ZHANELYA/Desktop/QMUL'. The main content area displays a table with three columns: 'Target', 'Successor', and 'Probability'. The table contains 20 rows of data, sorted by probability in descending order. The first 9 rows have a probability of 1, the 10th row has 0.67, and the remaining 11 rows have 0.5.

Target	Successor	Probability
has	begun	1
has	drifted	1
has	earned	1
has	eluded	1
has	enabled	1
has	intentions	1
has	single-handedly	1
has	spoilt	1
has	gained	0.67
has	brilliantly	0.5
has	captured	0.5
has	contributed	0.5
has	dare	0.5
has	disturbed	0.5
has	escaped	0.5
has	evoked	0.5
has	greeted	0.5
has	prevailed	0.5
has	remained	0.5
has	settled	0.5

Style

Finally, I added some styling to make the output tables look more natural. Within the `style.css`, which is included in every document that is formed through XQuery, I added style definition for table header, rows and columns. Now, it is even easier to see the results of the processing.