Guide to querying the XML produced by XTranscript

1. Required resources

Text or XML editor: e.g. Notepad++: https://notepad-plus-plus.org/
or TextWrangler: http://www.barebones.com/products/TextWrangler/
or oXygen: download "XML Editor" from https://www.oxygenxml.com/ (30-day trial)

XPath/XQuery platform: e.g. BaseX: download "Core Package" from http://basex.org/
or oXygen: download "XML Editor" from https://www.oxygenxml.com/ (30-day trial)

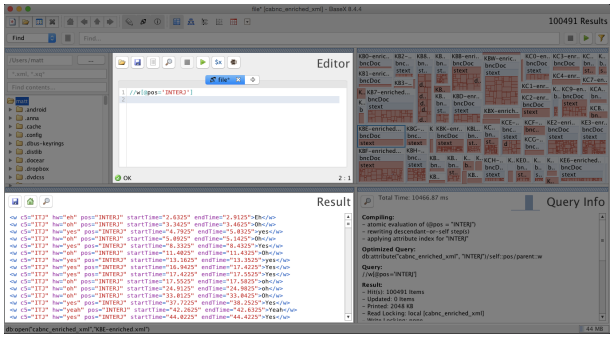
2. Preparing XML output for XPath/XQuery analysis

2.1. Create your XML file(s)

- On the XTranscript website (http://rdues.bcu.ac.uk/xtranscript), go to the conversion page
- Select the CA transcript file to upload (preferred formats are .doc, .docx or .odt)
- Click *Upload and Convert*
- Wait a little while, large files take time to process…
- Copy and paste the XML output into your Text or XML editor
- Save it as a .xml file
- Open your XPath/XQuery platform

You can also upload a zip file containing your transcripts, which will generate a new zip file containing the XML versions.

If using BaseX



2.2.1 Open your XML file(s)

- In the menu, choose: *Database* → *New*
- Choose your XML file(s) as the *Input File or Directory*
- Give your database a name and click OK

(Next time, you can choose: *Database -> Open and Manage* to open the database you created)

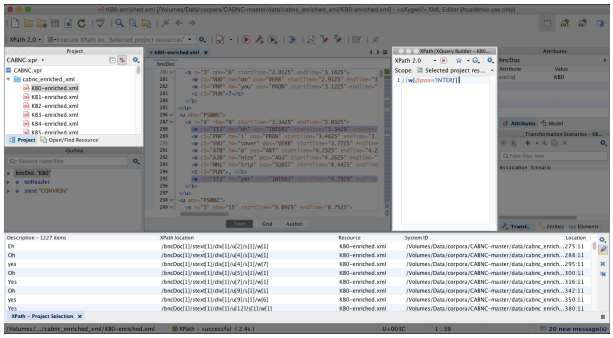
2.2.2 Check the BaseX settings

- In the menu, choose: *Options* → *Preferences*
- Under *Limits* move both sliders to the right until they say *All*.

2.2.3 Run XPath/XQuery

- Enter your query in the *Editor* panel
- Click the 'play' ▶ icon to run the query
- Results are displayed in the *Result* panel. (If the panel is named *Chopped Result*, you are not viewing all the results and must do the steps in 2.2.2 above.

If using oXygen



2.3.1 Open your XML file(s)

- In the menu, choose: *Project* → *New Project*
- Give the project file a name and save it in the directory which contains your XML file(s)

(Next time, you can choose: *Project* → *Open Project*, and open the project file you saved above)

2.3.2 Run XPath/XQuery

- In the menu, choose: *Window* → *Show View* → *XPath/XQuery Builder*
- Enter your query in the *XPath/XQuery Builder* panel
- Click on the 'play' ▶ icon to run your query
- The results will be displayed in a table at the bottom of the window. (You can also double-click on a result in this table to view it in the XML file).

3. Extracting data from XML output using XPath

3.1. Extracting and counting elements

Query	Description
//u	extracts all turns as elements
count(//u)	counts all turns
//voice	extracts all features of vocal delivery as elements
count(//voice)	counts all features of vocal delivery
//sequence	extracts all sequential features as elements
count(//sequence)	counts all sequential features

3.2. Extracting and counting elements via attribute values

Query	Description
//u[@n > 5]	extracts all turns except the first five turns
count(//u[@n > 5])	counts all turns occurring after the first five turns
//voice[@pitch="up"]	extracts all pitch rises
count(//voice[@pitch="up"])	counts all pitch rises
//sequence[@type="overlap"]	extracts all overlaps
count(//sequence[@type="overlap"])	counts all overlaps

3.3. Extracting attribute values

Query	Description
//gaze/string(@duration)	extracts all durations of gazes
//gaze/string(@to)	extracts all gaze directions
//timing[@type="pause"]/string(@duration)	extracts all durations of pauses

3.4. Extracting and counting elements via axes

Query	Description
//u[descendant::voice]	extracts all turns that contain a voice element
//u[descendant::voice[@intonation="rise"]]	extracts all turns that contain a question-like rise in intonation
count(//u[descendant::voice[@intonation="rise"]])	counts the number of turns that contain a question-like rise in intonation
//u[@n > 10 and descendant::voice]	extracts all turns that contain a voice element, except the first ten turns

4. Extracting data from XML output using XQuery

Query	Description
for $turn in //u return string-join($turn//text(),' ')	extracts all turns as text
for $overlap in //sequence[@type="overlap"] return string-join($overlap//text(),' ')	extracts all overlapped text
for $overlap in //sequence[@type="overlap"] return $overlap//timing[@type="pause"]/string(@duration)	extracts durations of pauses within overlap
for $pitch_change in //voice[@pitch="up"] return string-join($pitch_change//text(),' ')	extracts all text spoken with sudden pitch rise
for $turn in //u return concat($turn[@n]/@n,' ', count($turn//gaze),' ', string-join($turn//gaze/@duration,';'),' ', sum($turn//gaze/@duration),' ', string-join($turn//text(),' '))	extracts for all turns: (i) the turn's sequential number (ii) the number of gazes occurring in the turn (iii) the durations of gazes in the turn (iv) the total duration of gazes in the turn (v) the turn's text