**EB Project Log 2018-19**

19-4-4

* **Long-s**: Gary is making changes to the script. We had an hour-long meeting about it. Now, new words need to be added to the “clean” files, not unclean.
* **Entry files**: I converted eb03-s01 successfully, with one error where it inserted part of the TEI header after a <pb> again. I fixed it manually.
  + I finished prepping eb03-s02 and Python threw an error on the first file that contained Hebrew. I removed that file, reran, and it did the same thing on the second file that contained Hebrew. Need Luling to look into a fix for bidirection languages. There is a Python library called “bidi” for displaying text, but I don’t know if that will help.

19-4-1

* **Python <note> problems:** Luling has changed the “move-notes” function to include paragraphs. I tested it, and it works correctly. We are now adding a <p> wrapper to all note text. Two old problems have reappeared and I’ve asked him to correct them. In the meantime, I can add these to the cleanup routine.
* Script is duplicating character entities again: &amp;amp;
* Script is including ending punctuation after an entry term within the <label> tag again: <label>SCOTISTS,</label>.

19-3-29

* **Long-s:** Gary solved the memory problem: script has to be run on 64-bit Python. It runs very slowly (10 files=5 minutes). But it doesn’t crash. Probably will run quicker with smaller batches of files; need to experiment. I asked him to try cleaning up the “R” files, while I start on “S.”

19-3-26

* Transformed docx2tei for all “S” sections in eb03, eb07, eb09.
  + Next: finish eb11 “S”
  + Then start the cleanup procedure on all of S.

19-3-25

* **XSLT Problem** fixed. I looked at the version of docxtoteip5 that came with Oxygen. It has changed the libraries, so I copied them into my existing docx2tei and that fixed it.

19-3-24

* **Python <note> problem**. After Luling’s fix, Python keeps all of the text together, but it strips all of the <p> codes from multi-paragraph footnotes. I’ve sent him a note. Examples in:
  + eb03-16-s01-0665-09.xml
  + eb03-16-s01-0722-02.xml
* **XSLT Problem:** Running docx2tei at home is giving me an error message:
  + “Failed to load net.sf.saxon.TransformerFactoryImpl via the configured classpath.”
  + This may be a result of the upgrade from Oxygen 20 to 21? Or perhaps it’s a problem on the home set up, but not in DSC? Need to check on it next time I’m in DSC.
* **AFR List Problem**: Spent several hours on this. Basic procedure was to search directories of TEI-P files for <list type="ordered">. Then look at the OCR-Project file.
  + There’s not that many of these:
    - **eb03-s02** none
    - **eb03s03** none
    - **eb07-s01** none
    - **eb07-s02** none
    - **eb07-s03** many (35 total).
  + With ABBYY’s recommended preprocessing and resolution optimization, there are fewer lists. Where they do appear is where there is an actual list on the print page; the false lists seem to disappear. For an example of a group of numbered paragraphs, use eb03-s01-0037. A false list that was corrected is eb03-s01-0060. Also eb07-s03-0201 (Smollett entry).
  + There were also examples of list formatting appearing within a table (!). But that seems to be a problem of not analyzing the table structure adequately. See eb07-s03-0072. and eb07-s03-0023.

19-3-22

* **Python Long-S:** The problem was that the script is overwriting the UC files instead of appending the data to the bottom. He corrected this, and it seems to be working, but now he is getting a “memory” error. There’s probably a loop in there somewhere, but it means the script still won’t run.

19-3-21

* **Long-s:** the script identifies word pairs first, and then moves onto single words. That means if something is corrected in the word pairs routine, then changes in single words routine will not have any effect.
  + When adding (or correcting) words, always use the uncleaned (uc) files. There are only four files that I can manipulate:
    - after\_words\_main\_uc.txt
    - before\_words\_main\_uc.txt
    - single\_words\_main\_uc.txt
    - AmbiguousPairs\_Single\_column.txt
  + Always delete the previous output files before running the script again.

19-3-14

* **Project Gutenberg** files: downloaded samples of HTML and epub. Need to look at them more carefully, but HTML is 4.01 Transitional. Epub is XHTML. Basically, I need to try transforms using TEI’s HTML-to-TEI script.

19-3-12

* **Project Gutenberg** files are available in HTML and epub, with bold and italics formatting, and page break information. I need to download some of them and think about the workflow for conversion into our format.

19-3-8

* **Long-s:** Gary claims to have fixed the problem of the script not retaining additions to its dictionaries. I went into his new file for keeping additions and removed all duplicates from the original list, so these are all unique words that supplement single\_words\_main.txt. Should be ready to test.

19-3-3

* **Python <sup/sub>**: Remember to change <sup> to <superscript> and <sub> to <subscript> in Python script, after I get it back from Luling, who is modifying it to accommodate multiple paragraphs in notes.
* **~~Page numbers~~**~~:~~~~change the guidelines and the script to eliminate @n from <pb> for main text.~~

19-2-24

* **Python <note> problem**: The script seems unable to handle note text of multiple paragraphs. In eb03-s01-0089 (gri\_33125011196389\_0673.tif), there’s one note of multiple paragraphs that runs over to the next page. The runover text was added to the tei-p file correctly. However, Python closed the note tag at the end of the first paragraph, the continued with the body text to the end of the page before adding the rest of the note text as regular body text. This led to a <pb n=”nn”> in the middle of body text.
  + Same problem in eb03-16-s01-0722-02.xml (multiple times)
  + Script also eliminated the </p> at the end of that block of note text it thought was body text, inserted the full <pb> and continued with body text as if it were the same paragraph.
  + Script has to be modified to allow for multiple-paragraph notes. This means wrapping all note text in a <p> element, including the first paragraph of every note.
* **Marginal notes**: Use the first regex (find/replace) for anchored marginal notes, and the second for unanchored marginal notes:
  + **find:** <note\s+anchored="true"\s+n="(\d{1,3})"\s+place="bottom"\s+type="authorial">\s\*([\\*|\†|‡|§|‖|¶|※|⁑|⁂]\*)\s\*([<|\s|a-z|=|"|>]\*)\s\*([\\*|\†|‡|§|‖|¶|※|⁑|⁂]\*)\s\*([<|\s|a-z|=|"|>]\*)\[m\]\s\*
  + **replace:** <note anchored="true" n="\1" place="margin" type="authorial">\2\3\4\5
  + **find:** <note\s+anchored="true"\s+n="(\d{1,3})"\s+place="bottom"\s+type="authorial">\s\*([\\*|\†|‡|§|‖|※|⁑|⁂]\*)\s\*([<|\s|a-z|=|"|>]\*)\s\*([\\*|\†|‡|§|‖|※|⁑|⁂]\*)\s\*([<|\s|a-z|=|"|>]\*)\[mu\]\s\*
  + **replace:** <note anchored="false" n="\1" place="margin" type="authorial">\2\3\4\5

19-2-23

* **Python <note> problem**: In eb03-16-s01-0722-02, script terminated a note in the middle, inserted some regular text, then returned to note text outside of the note format … needs more investigation, and I have to check to see if this was the only instance. I found it by tracking down <pb n= and noticed that it was not within a <note> tag.
* **CLEAN-UP ROUTINE:**
  + eb03-s01-entry files clean-up steps
  + In the tei-p files:
    1. replace <cell/> with <cell> </cell>
    2. replace <pb n="(\d{1,4})"\/> with <pb n="\1"></pb>
    3. rerun Python to generate entry files.
  + In the entry files:
    1. Replace all &amp; with &amp;
    2. to find all lists that run onto a new page, use regex and do manual correction
    3. \<\/item\>\s\*\<\/list\>\s\*\<pb
    4. Replace all “,</label> with </label>,”
    5. delete all instances of <name> unknown author </name>.
    6. Run validation routine.
  + NOTE: There was still an error in 1 out of about 350 files, where it inserted this immediately the following immediately after a full page break. It did the same thing in 7 other files, where it inserted a full <pb> at the end of the file, so the insertion did not cause an error:
    - <name> unknown author </name> </p> </div> </body>
    - Script also eliminated a legitimate </p> before the <pb>. I corrected everything in eb03-16-s01-0629-03.xml. It finally validates.
* **Summary**:
  + Two Python conversion problems, with <pb/> and <cell/>.
  + One conversion problem, with <list>
  + One character entity problem with &c.
* **python <pb/>:** when the script encounters a short self-closing <pb> in a note, it eliminates the closing slash, just as it was doing with the <cell> tags; and adds a second space after the closing bracket; and it changes the closing </p> tag for the paragraph to a </pb>, causing an error (also, just like the action with <cell>).
  + input: “adultery at <pb n="668"/>the bar” – converts to:
  + output: “adultery at <pb n="668"> the bar” – causing another error, as you cannot include a text string within the <pb> element.
  + The normal page breaks are also self-closing, but they are fine, presumably because they are original to Python and not conversions.
    - <pb facs="gri\_33125011196389\_0674.tif" xml:id="eb03-16-s01-0668"/>
* **python <list>**: When a list is interrupted with a page break, Python captures the </item></list> tags at the end of the first page, inserts the <pb/> correctly, and eliminates the <p> at the start of the second page. This sequence creates an error, because you have text without a containing element. The text should be part of the previous item, so we want to eliminate the </item></list> tags before the <pb/>, and to add an </item> tag at the close of the first paragraph on page two.
  + But there is still a problem, because the tei-p file for page two does not treat the run-over item text as part of a list. Instead, it interprets it as a free-standing <p>, and then begins a NEW list for the next item that follows.
    - see eb03-s01-0134.xml and -0135.xml for an example.
  + Since this is going to be uncommon, the easiest solution is to clean up the tei-entry file, where we can find all run-over lists with the following regex and then do a manual cleanup:
    - \<\/item\>\s\*\<\/list\>\s\*\<pb
* **Python entry term problem:** Where an entry term is followed by a comma, the comma should be outside the <label> tabs, but it is currently included within them.
* **New <list> problem**: Apparently, ABBYY is too aggressive in interpreting text as a list. In eb03-16-s01-0652-01.xml, I have some geographic coordinates in a paragraph. They should read
  + text … on the Cape Aguer. W. Long. [linebreak]  
    10. 7. N. Lat. 30. 38.
  + But ABBYY interprets the line break after “Long.” as a paragraph break, and since the next line begins with a number, it interprets it as a list. And since docx2tei doesn’t save the start number for new lists, the “10” is lost, and we get this garbage:
    - on the Cape Aguer. W. Long.</p><list type="ordered"> <item> 7. N. Lat. 30. 38. </item> </list>
    - Files: image: gri\_33125011196389\_0658

eb03-s01-0074.docx

eb03-16-s01-0652-01.xml

gri\_33125011196389\_0644

eb03-s01-0060.docx

eb03-16-s01-0638-01.xml

* **Python character entity problem**: It changes input &amp;c. to output &amp;amp;c.
* input: eb03-s01-0001.xml
* output: eb03-16-s01-0579-01.xml

19-2-20

* Next: continue working on validation problems in entry files for eb03-s01.
* **eb03-s01**: cleaned tei-p output and made it ready for python conversion. A few notes:
  + many pages have note text that runs over to the next page. I am inserting <pb n=”nnn”/> at the break within the note text, but WWP has good guidelines for how to encode page breaks in note text by correlating the <pb> to the main page break @id and @corresp values. I can’t do this at the tei-p stage, however, so will have to add a task to clean up the tei-entry file. See <https://www.wwp.northeastern.edu/research/publications/guide/html/referencing_noteBreaks.html>
  + I encountered a note that ran across three pages and includes a note within the note. I encoded it as <note> with no attributes and placed it in the text flow. There are not very many of these, but I need to think about the encoding scheme for them.
* **eb-docx2tei**: discovered bookmarks in the TEI-p, encoded as <anchor xmil:id=”bookmark0”/>. There were 63 of them in eb03-s01 alone. The numeral varies, depending on how many are on the page. Traced it to the docx, where it appears as:
  + <w:bookmarkStart w:id=”0” w:name=”bookmark0”/>, followed by <w:bookmarkEnd w:id=”0”/>
  + I removed them with regex: \<anchor xml:id\=\"bookmark\d\"\/\>
  + Then modified docxtotei.xsl to hide the two bookmark templates. *NOTE: None of the other sections have bookmarks, as far as I can tell, and I wasn’t able to test this change yet.*
* **python 8.05 <cell/>**: when the script encounters an empty table cell in the tei-p input, <cell/>, it changes it to <cell> and adds a closing </cell> after the next element (usually another <cell>. That creates an error and messes up the table formatting. I was able to force it to convert by replacing all of the empty cell elements with a cell of one space:
  + <cell> </cell>
  + But this is not a good solution and is adding another step to preparing the tei-p files.
* **eb-docx2tei**: **LISTS.** In docx output, numbered paragraphs correctly reflect the numbers on the printed page. The transformation to TEI converts these numbered paragraphs to an <ol>, removes the given numbers, and automatically numbers them. When a list runs across two pages that are combined by python, all of the sequencing is screwed up. I don’t want to use automated numbering here, so I need to go into the xsl files and stop it from creating ordered lists. That’s probably also necessary for unordered lists, since I’m trying to match the printed output and not second guess it.
  + Examples files: eb03-s01-0050.docx and 0051

eb03-s01-0050.xml and 0051

eb03-16-s01-0619-01.xml

see also: eb03-16-s01-0712-05.xml

* The problem isn’t that my xslt is creating lists; it’s that the docx file formats numbered paragraphs as a list. And the OOXML standards on this are complex. The default for numbered lists is to start at val=1, but another starting value can be stored in “numbering.xml” within the docx archive.
  + <w:abstractNum w:abstractNumId="0">  
     <w:multiLevelType w:val="multilevel"/>  
     <w:lvl w:ilvl="0">  
     <w:start w:val="3"/>  
     <w:numFmt w:val="decimal"/>  
     <w:lvlText w:val="%1."/>
* w:start records the start value for the first number. In my example, 0051 starts with “3”, because it’s a continuation of a list that begins on 0050. So w:start here is “3”. The docx file displays it properly in Word. I do not think that eb-xml2tei has a routine for looking up that value; instead, it creates a generic list beginning with val=1.
  + It may be necessary to have operators mark pages with numbered lists that continue onto the next page, and manually correct the starting value. Otherwise, there’s the option to manually check every page with an <ol>, or to devise a clean-up routine to convert lists into hard-coded values, which is what I want.
* Finally, eb-docx2tei formats the list as:
  + <list type="ordered">
* But TEI says that is the older way, and they now recommend using @rend, and reserving @type for descriptive information on the function of the list, i.e., index, gloss, instructions, etc. In Oxygen, @rend does not display, though, while @type does. But I think I should modify eb-docx2tei to code lists appropriately, if I cannot get away from them.

19-2-18

* The error below (<hi style=”color:red …>) is caused by outputting from AFR as “editable copy” instead of “formatted” text. Rachel cleaned up the tables and reoutput, and everything ran fine and validates in the transform to TEI. That’s keeping the modifications to textruns.xsl below.
* Transformed eb07-s01, 02, 03 to TEI page files. They look good, with no obvious coding errors or problems.
  + I discovered an entry, “Sabbath-breaking,” where the first term is tagged as small caps; in the original, it’s all sc-italics, and is a sub-entry of SABBATH. Right now, we have no way to tag these …

19-2-17

* Looked into transformation error for **eb03-s03-0200**.docx.
  + Error message: “Required item type of first operand of '/' is node(); supplied value has item type xs:string”
  + Found a reference on stackoverflow which suggested adding @select to the variable declaration: <xsl:variable name="patterns" select="()"/>
  + I modified textruns.xsl and added @select to the two variable causing the problem, $extrarow and $extracolumn. That eliminated the error, but there is a very strange new error in the output file, at the beginning.
    - <p><hi style="color:red; font-size: 14pt; font-weight:bold;" rend="ERROR">�</hi><note  
       place="margin" type="conversion" resp="#teitodocx"><hi rend="docxError">unable  
       to handle picture here, no embed or link</hi></note></p>  
       <p><hi style="color:red; font-size: 14pt; font-weight:bold;" rend="ERROR">�</hi><note  
       place="margin" type="conversion" resp="#teitodocx"><hi rend="docxError">unable  
       to handle picture here, no embed or link</hi></note>
  + Need to look at this further.

19-2-14

* Transformed eb03-s01, 2, and 3. One file causes an error: **eb03-s03-0200**.docx.

19-2-12

* Transformed eb09-s02, s03. That’s a good base to work with for Python conversion. For eb03, redox xsl on s01-3, to fix the @color problem. Then test OCR\_Cleaner on them.

19-2-10

* **docx2tei:** Removed test for text color in textruns.xsl. That gets rid of all color styles.
* eb09s01 transformed to tei and proofed. Ready for entry conversion.
* docx files are also capturing lists, which it wasn’t doing before: <list type=”ordered”>

19-2-8

* Luling has revised Python again. Now it includes TEI header, he has gotten rid of the extra space around <label>, and the small-caps portion of entry terms is included in entry-stats file. But there is a problem with table formatting that shows up in text/entries/eb09-22-s06-0460-03.xml. It is not closing the <row> tags appropriately, or the <table> itself, even though everything is valid in the source file eb09-s06-0010.xml.
  + I tried a test by adding a simple table to another source file, and python converted it without problem.
  + Also I notice that the <pb> is giving us the @facs first, followed by @xml:id. It should be the other way around.

19-2-5

* TO DO: Update user files for eb09 and eb07

19-2-4

* TO DO: For SAM: develop Python script to print word counts for plain text files, after transformation.

19-2-1

* Luling modified Python to capture smallcaps. It does that successfully, but we have lost all of the TEI header, and it adds a space between <label> and the text, both before and after the entry term. And the smallcaps portion of the entry term is not being included in the entry-stats.txt file.
* **eb03-s03**: converted to TEI, but have to check inventory notes and proof footnotes.

19-1-28

* **eb-docx2tei**: the “sub-“ and “superscript” values come from docx, and textruns.xsl copies them directly, using a test for w:vertAlign, which is the OOXML element that defines sub- and superscripts. Allowed values are:
  + baseline - regular vertical positioning
  + subscript - lowers the text below the baseline and changes it to a small size
  + superscript - raises the text above the baseline and changes it to a smaller size
* Changing this means writing xsl to incorporate an if/then routine here with choices for these three options. Easier to update the Python script.

19-1-25

* **eb-docx2tei**: modified textruns.xsl to eliminate all “normalweight” rends in inline formatting. Tried to change “superscript” to “sup”, but it didn’t work and I’m not sure why. Easier to just go with it and change the KP.

19-1-24

* **eb-docx2tei**: modified textruns.xsl to eliminate all bolds in inline formatting. Works perfectly, even on compound attributes. This is especially a problem on eb03, where deterioration and ink spread becomes a false “bold” format.
  + <hi rend=”bold smallcaps”> becomes <hi rend=”smallcaps”>
* Gary made some changes to Python to eliminate the extra xml declarations and other problems. He traces a lot of them to problems in the <label> coding, which is requiring that number 1 becomes I, and changes / to ,--it’s causing all of the problems after the <pb>. He said the script tends to insert the xml decl when it can’t find an exact match, somehow ….
* I output everything ready in docx. Started with eb03 on the transform to tei-page, and finished **eb03-s01 and s02**. Both still need to have their footnotes proofed.

19-1-22

* Worked on fixing Python all day, to no success, so turned it over to Luling. Asked him to:
  + Fix the problem of inserting XML decl and TEI header material after every <pb>
  + Remove <p> codes around <pb>
  + Change definition of an entry term to include smallcaps following all caps.
* Removed all of the .xpr files from sections. I’ll just keep one for each edition.

19-1-21

* Updated manual page “Check Footnote coding 2” with new regex.
* Proofed @@ codes for eb09-s06
* PYTHON: script is still inserting the entire xml declaration and tei header following every <pb>. This is using v8.03.
  + Also need to change the findEntry function to include small caps following all caps, and we want to remove the <p> codes around <pb>.

19-1-20

* TO DO: Proof @@ codes again for eb09-s06.
* Changed eb-docx2tei to eliminate the references in the output to Word styles, like the one mentioned below:  
   <hi rend="Body\_text\_"><seg rend="smallcaps">w. ho</seg></hi>  
  There’s two issues in here: (1) the empty reference to the Word style name; (2) the use of the <seg> tag.
  + <seg> tag is generated by eb-docx2tei whenever there is inline styling (Word calls it a “run,” w:r) within a parent inline style based on a Word style name. In that case, the transformer uses a <seg> instead of <hi> tag.
  + Use of Word style names was difficult to trace. Within the docx archive, the document.xml codes such cases as:  
     <w:p> <w:r> <w:rPr> <w:rStyle w:val=”CharStyle4”/>  
    In Pass0.xsl, the transform creates a function “tei:getWordStyleName” that looks up the value of w:rStyle in Word’s styleDoc, which is the styles.xml file in the docx archive. There, it finds that CharStyle4 is named something like “Body style 4\_”. It uses that value (substituting underscores for spaces) to generate the TEI output <hi rend=”Body\_text\_(5)\_”>.
  + I solved the problem by leaving the condition in tei:getWordStyleName that does the look up, but commenting out the instruction, so that the condition outputs nothing. Without that parent inline style, the transform never generates the <seg> entity, instead using <hi> so that now the above line is rendered:  
     <hi rend="smallcaps">w. ho</hi>
  + I tested it on about 10 files with this problem in them, and in every case it ONLY makes this one change, leaving everything else untouched. It’s a big improvement!! So I will rerun the new transform on eb09-s06.
* This began with an effort to modify my regex for identifying two footnotes on the same line. I got frustrated with the useless double sequence or <hi> <seg>@@@, which made it very difficult. I still have to solve the regex update, but it should be easier now.

19-1-18

* TODO: update production.xlsx as docx files are output. And ask Tyler to add in the necessary data about page spreads for the sections of image collections he created.
* TODO: tell all operators to use fraction characters instead of building fractions with sup/sub, if the fractions are available.
* Continued updating manual; finished “Check Footnote Coding 1.”

19-1-17

* Eb03: Gary’s OCR cleaner uses the TEI file for input, not the old HTML, so the change to DOCX will not affect it.
* Updated the manual with instructions for converting DOCX to TEI.

19-1-1

* Further modified eb-docx2tei.xsl. There’s a problem in its routine for identifying table and figure captions, causing table heads to appear at the end of the table, as <head>, preceding </table>, which is invalid. Evidently, the idea was to have it appear at the beginning, as <head> immediately following <table>, which would be legal. But it applies it to the w:p (Word paragraph) preceding the table, which is not always a table head, so I disabled the whole mechanism. But it wasn’t easy.
  + The <head> tags are created by pass2.xsl, in:  
    <xsl:template match="tei:p[tei:match(@rend,'caption') or tei:match(@rend,'Figure title')]" mode="pass2">  
     <head>  
     <xsl:apply-templates mode="pass2"/>  
     </head>  
     </xsl:template>
  + When I commented out that template, the output changed from “<head>Table A.</head>” to <p rend=”Table captions”>Table A.</p>. So an earlier pass is adding @rend to the line with a “caption” value, Which the template in pass2.xsl then converts to a <head>tag. This has no effect on the position of the title.
  + In functions.xsl, the function “tei:is-caption” defines the w:p as a “caption.” It’s a binary test, and I changed it from T to F. The <p> preceding a table now appears in the proper place, preceding <table>.
* I ran the revised transform on 225 files in eb09-s06, and everything validates now.
* There are other squirrely issues in there to look at.
  + In eb09-s06-0181.xml, the signature at the end of the article, “W. HO,” is all small caps, but it’s formatted as:  
    (<hi rend="Body\_text\_"><seg rend="smallcaps">w. ho</seg></hi>.)
* **Next:** keep testing the transformation and refining it. But in the meantime, get students to output all of “S” as docx, and see if Gary can help with conversions to tei-page files.

18-12-30

* Finished modifying DOCX TEI P5 and created eb-docx2tei.xsl and added my initials to all changes. It uses the local copy of the provided transformation, but with the following alterations:
  + **docxtotei.xsl:** 
    - changed variable preserveFontSizeChanges from T to F
      * removes all instances of <hi style=”font-size:*x*pt;>
    - changed variable preserveEffects from T to F
      * removes @style=”text-align:*x*”; @style=”font-family:*x*”; and even <cell rend=”left”>
      * Note that paragraphs.xsl uses this parameter to decide whether or not to apply Word paragraph styles to the <p> tag.
  + **paragraphs.xsl:**
    - comment out the part of the paragraph-wp template dealing with style attributes for default paragraphs. That eliminates the <p rend=”Body Text”> problem.
    - I changed the “retrieve-styles” template to ignore the test for bold. This may only effect para-styles, though, and not catch inline uses. If so, I need to correct them in textruns.xsl.
  + **tables.xsl:**
    - comment out part of the Table-rendition template, to eliminate @rend from <table rend=”rules”>
    - comment out <xsl:otherwise> routine that set the background color for table cells.
  + **pass2.xsl:**
    - Changed the condition <xsl:when test=”$preserveSpace=’true’”> from T to F. That eliminates the @xsl:space-“preserve” problem. The condition refers to a parameter in docxtotei.xsl, which by default is set to F. Changing it to T made no difference, and I suspected (correctly) that the pass2.xsl default is an error, since there is no condition for when the parameter is set to F.
  + **textruns.xsl:**
    - This stylesheet controls inline formatting. I commented out the condition for converting Word w:b to TEI bold. That eliminated @rend=”bold” in the output. (See also paragraphs.xsl)
* Output looks very clean now, and I think it should run properly in Python without further modification.

18-12-28

* Finished file reorganization.
* Created new Oxygen Project files for each edition and imported all transforms.
* In Oxygen Frameworks, “OTA” is Oxford Text Archive.
* Updated instruction manual for new folder structure, and DOCX through the OCR section—need to continue checking for HTML references thereafter.

18-12-27

* TO DO: Update the instructions manual for (1) the new folder structures; (2) outputting from ABBYY to DOCX; (3) transforming DOCS to TEI.
* Finally have reorganized folders to accommodate a full copy of DOCXtoTEIP5 and all library dependencies. It means centralizing all xslt files and essentially reproducing the OxygenFramework for TEI. The key is maintaining the folder nesting structures, and then learning how to modify a copy of the transformation scenario so it will output files in the correct directory.
  + That meant learning how to navigate within archive files, like DOCX, and for now tying the output location to the project file location.
* The larger reorganization is creating a new top-level folder, “code”, and moving all XSLT, Python, and Regex materials in there. When working in either the Production or Digital-Edition folders, I can add any folders I need from “code,” into the project, so it’s easy to work with.
* Learned how to store transformation scenarios and make them shareable—instead of creating a new file for the docx2tei scenario (since it’s not an .xsl file), I edit the scenario and select the Project Options storage scenario, instead of Global. This saves it to the .xpr file, which get automatically backed up to Box.
  + To export a transformation scenario to another Oxygen project file, open the Configure Transformation Scenarios window. Then use the settings gear icon to export it to a .scenario file.
  + Open the new project, go back to the same Configure Transformation Scenarios window, and use the gear icon to import the .scenario file.

18-12-24

* Tests on alternate output methods in ABBYY:
  + EPUB—does not retain c/sc formatting consistently; seems to follow HTML pattern.
  + RTF—retains c/sc formatting and is identical to Word output
  + WORD-formatted text—ignores layout but captures all word formatting; identical to RTF. Removes column break and joins run-on paragraph into single semantic unit.
  + WORD-editable copy—retains two-column format but uses output page length instead of copying the original. Otherwise like “formatted text”.
  + WORD-exact copy—reproduces page layout and all word formatting. Captures the column break.
* TEI does not have an RTF to TEI XSLT file. Others out there claim to do RTF to XML. Writers note thought that RTF has about as many implementations as HTML, which is why its problematic for conversion.
* Can we use Word “exact copy” and retain the column break? Need to transform it to TEI and see what it looks like.
  + A: NO. The converter removes the <cb> and substitutes a standard <p> sequence for a new paragraph, even when it’s a run-on paragraph.
* COMPARING **WORD-EXACT-COPY** AND **WORD-FORMATTED-TEXT** FORMATS, after conversion to TEI:
  + Headers are identical. Bodies are identical except for the following to differences:
    - Formatted-text adds “style="text-align:left;” to the <p> element.
    - Exact-Copy adds </p><p> at the column break. Formatted-text joins the paragraph seamlessly.
    - Both add @rend="Body text" to every <p> tag.
    - Both add @ xml:space="preserve" unpredictably (but identically) to <p> with @rend for italics or sc.
  + SOLUTION: Use Formatted-text, with *all options turned off*. Modify XSLT to remove @style="text-align:left;” and the above @rend and @xml:space attributes from the <p> tag.
    - Need to try a sample page with tables, to see what else needs adjusting.

18-12-5

* Missing files from the DSC server—about 75% of everything. Was able to replace it, but we lost a few student shifts, and I don’t know what caused it.

18-11-12

* ~~Need to create a routine for ugrads to create a collection of image files for the next letters.~~
* Need to modify DITA to explain how to generate Word output, instead of html.
* DOCX to TEI: Conversion is working well. Needs some tweaking in the XSLT:
  + Generates a filename with –TEI-P5 at the end, and Python doesn’t like that.
  + Add a variety of @rend attributes to <p> tags, which interferes with the script identifying entry terms. Script only recognizes <p>[A-Z](2,).
  + It also adds <hi> with @rend in between <p> and the entry term.
  + I want to look at eliminating the “bold” values.
  + Adds a lot of metadata in the head section. I don’t think that does any harm, but I can probably refine it to better serve our needs.
* PYTHON: When I removed the unnecessary @rend codes, Python ran fine and identified all entry terms appropriately. However, it is adding a <label> tag and the DTD+header metadata following every <pb> again. Luling is looking into that for me.
* HTML: Ever since the last AFR update, it is generating HTML using an outdated doctype, HTML 4.0 Transitional. Current is 4.01. This causes a problem because the HTML 4.0 Transitional DT has an error in its header, so these files will not pass the well-formed test in Oxygen. I have to update the DT in the HTML. And I have to close the two <meta> elements, which are left open. Additionally, I have to find and replace <br clear="all"> with <br clear="all"/>. Here’s the DTD:

FIND:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"  
 "http://www.w3.org/TR/1998/REC-html40-19980424/loose.dtd">  
<html>  
<head><meta http-equiv="content-type" content="text/html; charset=UTF-8"><meta name="generator" content="ABBYY FineReader 14">

REPLACE:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
"http://www.w3.org/TR/html4/loose.dtd">  
<html>  
<head>  
<meta http-equiv="content-type" content="text/html; charset=UTF-8"/>  
<meta name="generator" content="ABBYY FineReader 14"/>

18-11-10

### PYTHON: Transform DOCX to TEI

1. In Oxygen, right-click the Word file and configure transformation scenario. Should see DOCX TEI P5 – Copy in the Global category. Or DOCX TEI P5 in the OOXML category. If you don’t see it, click “settings” next to the search bar of the dialog window, and “show all scenarios.”
2. The transformation produces a new TEI file in the same directory.
3. Move the files to another directory and add a copy of the inventory spreadsheet to the parent directory. Add filenames to the tei-fn.xml column of the inventory spreadsheet before running Python.

18-11-7

* PYTHON: script is not adapted for eb03 marginal notes yet, and it leaves in the @@@[mu] and [m] codes. Need to replace @place=”bottom” with “margin”. It also needs to eliminate the note symbols, since they are not numbers.
* PYTHON: script is still generating the XML declaration after every page break, at least when converting eb03-s01.
* ABBYY: the html output with background color options checked is generating errors for italics, like <hi rend="italic"><hi rend="italic">
* Docx: need to find my notes for transforming docx to tei and try it out on the sample.
* Eb03: has multiple cases where note text runs many paragraphs. This is legitimate in TEI, but it means that all note text needs to be contained within a <p>, and </p> within <note>. I think the first section of the note also has to have a <p> container, but need to run more checks on that before doing anything, since it would mead changing Python script to add <p> to every note.

18-11-5

* PYTHON: script fixed by Luling. Tested on eb11-s01 successfully. All files validate.
* Maddie output 20 pages to docx for me. I saved them in test/word
* Userfiles: I’m not having any luck finding out where AFR is saving the userfiles. I redid eb03, named it eb03a, but it’s not in the userfile directory. Where is it?
  + Solution: cannot determine where to save a userfile in the pattern editor. But after editing, I can save it from the Options menu anywhere I want. Setting up the current system is correct according to AFR’s advice for setting it up for multiple users.

18-10-24

* Updated user pattern for eb07.
* ABBYY support has reproduced the c/sc problem. I gave them further information.
* PYTHON: New script is inserting “<?xml version="1.0" encoding="utf-8"?><p>” again after every page break. It seems to have corrected the added comma problem, but I can’t check it thoroughly with these errors. Need to fix that problem, and then try again. I am using eb11-s01, because eb11 does not have the c/sc problem in the entry terms that the other editions have.
  + Two problems, depending on whether I use the version in the “gary” subfolder, or the one I moved to the root folder.

18-10-23

* Production log updated.
* TO DO: Create sections for “T-Z” and add to file system.
* Quick test on transforming new html files (with background and colors box checked). XSLT script seems to clean it fine. Look same as old TEI files.

18-10-22

* Updated user file for eb11.
* Tested c/sc problem—the changed method does no better than the previous one. I focused on one page in eb09-s05, saved it as an ocr project, output in html and in Word. The Word doc has the c/sc, but not the html. I’m entering a new help request with ABBYY and will upload these files.
* Tyler finished eb07-s01
* I need to retrain the user pattern for eb07, since Bethany’s file has somehow eliminated all soft hyphens.

18-10-11

* Checked html for eb09s02. The output now includes background colors, but it is still not catching the small caps.

18-10-10

* Updated py.bat to call the updated script.
* TO DO:
  + Rerun HTML output:
    - Eb03 s01-s03 done
    - Eb07 s01-s02 done
    - Eb09 s01-s07 done
    - Eb11 s01-s05
  + Transform all reruns to TEI
  + Convert all S-files in Python.
  + ~~Eb09s01 is missing the inventory file—it’s not in any of my backup sites~~.

18-09-21

* c/sc problem in output is solved by changing setting in AFR Options/Format Settings to check “**Keep text and background colors”.** That also adds a lot more css formatting to the html, so I will have to run a test to make sure the current xslt script eliminates everything.
* Luling has fixed the problem of Python eliminating the punctuation after </label>, in 8v02. Still has to upload it to the server.

18-08-20

* Reconciled all entry terms with new style in **eb11-s01, s02, s03**.
* **entry terms:**
  + Consistency across editions is good, as long as it does not result in any loss of original data.
  + **For eb11**
    - There appears to be no difference between parens and brackets in entry terms, so treat them identically.
    - Rewrite Python’s regex to capture any group of words contained within parens or brackets followed by a string of 2 or more caps; or any group of c/sc or c/lc beginning with “or” and followed by all caps.
    - Revise instructions to ugrads reading entry terms to include alternate English spellings or pseudonyms, but leave out references to original language spellings. Original language spelling should follow the full entry term, including first and last name.
  + Strategy: correct all terms manually in “S” before going back to “R”, while revising Python regex to make it easier going forward.
  + See list of entry term formats.

18-08-18

* Talk with Matt about **entry terms**.
  + Follow style of the edition, or make them consistent across editions?
  + Difficulties in eb11:
    - **SCHAFARIK** (Czech, *Šafařík*), **PAVEL JOSEF**
    - **SCHANDORPH** [or Skamdrup], **SOPHUS CHRISTIAN FREDERICK**
* **eb11-s03:** segmented article fixed, entry terms corrected.

18-08-17

* **Notes for ugrads**: (Maddy) is using a bullet instead of a middle dot; Rachel needs to know slight change in coding marginal notes.
* **eb11-s02**: entry-term corrections made; segmentation okay.
* **Metadata:** TEI uses <keywords> with @scheme in the header to identify the controlled vocabulary, using a list of <terms> within it. You can use multiple vocabularies. So that’s the field we want to create in each file.
* **Index trial:** used eb07 v22 from gb. Pretty poor image quality, and I’d need a better image. But it automatically identifies the columns and will output to a spreadsheet, saving each line of the index as a separate row. All in one column. There’s potential there.

18-08-16

* **eb11-s01:** segmented article fixed, entry-term check correction made.
* **Python:** script is adding a comma after the entry term whether there should be one there or not. Stop converting to entry files until I can get Luling to look at it.
* **Metadata:** Sam tells me that the state of automated metadata generation is still primitive, and that a 35% accuracy rate is considered good. She ran some tests with a domain vocabulary for agriculture and found the results for many entries were more accurate that LCSH. Jane wants to add the 1st edition of LCSH (1910-14) to HIVE and try it; that would eliminate a lot of anachronistic results. Sam thinks we might run multiple domain vocabularies and get better results. She’ll try this out over the remaining time this summer. Meanwhile, Joan is tweaking the algorithm parameters in HIVE to improve results for EB entries.

18-08-15

* **eb03:** entry term problem. This edition uses all caps for the entry term, but then it uses c/sc in two cases:
  + alternate meanings of the same term (like different Penelopes—in later editions, these are treated as new entries; in eb03, they are a subsection of the first meaning, even when the first meaning is the least important.
    - Think about a way to bracket these with something like our @@ codes, and adding a line to the script to treat anything between these codes as an entry term.
  + The 2nd word of a 2-word entry term: PATRIARCHIAL cross.
* **eb03:** marginal notes problem: tell Rachel about the change in coding. And think through how this will work in the script. Gary is considering a second parallel system for identifying notes, but the goal is merely to add the “margin” value to @position in the <note> element, so there must be a simpler way of doing it.
* **eb11-s01, s02, s03:** created entry files for all. Still need to address the segmented articles.
* **eb09-s01, s02, s03, s04, s05, s06:** all new entry file output. Segmented articles need correcting.

18-08-14

* **eb03:** Python will not remove a non-numeral character following @@ codes. So we have a problem for both “(a)” and “\* † ‡.” Plan is for me to develop a regex to strip them out in Oxygen, and once I have it refined, then we’ll add a module to do this in the Python script.

18-08-13

* Updated information in manual on eb03 marginal notes.
* **eb03:**
  + Rachel coded subheads as marginal notes in the beginning of the section, and then stopped, but she left them in. Ask her to go back, remove them, and re-output.
  + This edition sometimes uses parenthetical letters for note anchors:
    - … to have been the largeſt then in the world @@(m). He worked with …
    - <p><span class="font0" style="font-variant:small-caps;">@@@(m) Of this ſhip we have the …
  + What will Python do when faced with this?
* **eb11-s03:** ready for python

18-08-11

* **eb09:** converted s05, s06 to tei-p; ready for python.
* **eb09:** s01, s02 ready for python
* **eb11:** s01, s02 ready for python
* Downloaded all **eb07** volumes from Hathi Trust.

18-08-08

* **eb11:** converted s02 to tei-p.
* **eb09:** converted s01, s02 to tei-p.

18-08-07

* updated production spreadsheet with current information.
* TO DO
  + eb09-s01, s02 are verified in AFR. They need to be converted to TEI, then reprocessed as entries. Check new entry-stats list against the previous one to make sure they are identical.
  + students finished entry verification for eb09-s01, s02, s03, s04. Corrections need to be made to entry files for all.
  + entries need to be generated and verified for:
    - eb03-s01, s02
    - eb09-s05, s06
    - eb11-s01, s02, s03

18-07-03

* **eb09**-s01-s02 were never run through the AFR verify process, and I’m seeing too many errors, so I’ve asked Tyler to go back to them and run it. Then, when he’s done, I’ll reconvert everything.
* **eb09**-s01, s02, s03, s04: First entry “S” term fixed. All partial entries corrected.