Re-processing Abbyy outputs

Lou Burnard Consulting *2019-09-15*

1 Choosing your text

Please begin by selecting a text you would like to work on, in a language which you are comfortable with handling. The following items are available in your Work/Novels folder:

Identifier	Title	PDF filename	Other formats	Wordcount
ENG18410	Cecil, or, The adventures of a coxcomb	ENG18410_Gore	abbyy.xml	
ENG18470	The Castle of Ehrenstein	ENG18470_James	abbyy.xml	
ENG19150	Pointed Roofs	ENG19150_Richardson	abbyy.xml, docx	
FRA19120	Vers le pole en aeroplane	FRA19120_Gilbert	abbyy.xml, docx	
HUN18460	Meghasonlott Kedely	HUN18460_Kelmenfy	abbyy.xml, docx	
HUN19090	Végzetes Tévedésh	-	HUN19090_Beniczkyne.docx	
HUN19180	A három galamb	-	HUN19180_Kehel.html	
ROM18860	Teodorei: copila domnului din bucuresti	ROM18860_Macri	abbyy.xml, docx	
ROM19060	Mara	ROM19060_Slavici	abbyy.xml, docx	
ROM19170	Domnul Badina	ROM19170_Smara	abbyy.xml	
SRP18860	Omer Čelebija	SRP18860_Milicevic	abbyy.xml	

Each file with extension .pdf contains a digitized representation of an original printed source, which has been processed by Abbyy OCR. Take a look at this to determine what metadata you can extract for use in the header of your file, and also to familiarize yourself with the basic structure of the text – whether it has prefaces etc.

Use online resources (WorldCat, Wikipedia, etc.) to collect other required metadata, such as the full name of the author, their dates, sex, etc. When you have this information, proceed to create a new document and a TEI Header The procedure to follow is described in detail in the Header Tutorial.

2 Reprocessing abbyy XML format

Each file with extension .abbyy.xml contains an XML representation of the results of the OCR procedure carried out by Abbyy. This representation is quite verbose and contains much information we don't need, but it is easy to translate it to a basic TEI form using an XSLT stylesheet. We won't be teaching you XSLT (a standard language for converting one XML document to another) but we have provided a suitable stylesheet for this task. In oXygen, the easiest way to use it is to use a named 'transformation scenario'. We have created such a scenario for you: to use it,

- Open the abby.xml file of your choice in oXygen

- With this file open in the editor, click the big red triangle icon on the toolbar, type CTRL-SHIFT-T, or select Document -> Transformation -> Apply Transformation Scenario from the menu

- If you have set a default transformation, it will be applied; if not the Transform with dialog box appears. Select the abby2tei transformation to associate it and then click the Apply Associated button.
- After a brief delay, you should see a much less richly tagged version of the text.

Comparing this version with the PDF file you opened earlier, you can see that the tag `<pb/>` indicates the start of a new page in the original. Each line of the output corresponds with a typographic line in the source. Blocks of text are tagged with a `<p>` element, though not all of them are paragraphs. Where there was a non-textual image in the original, the `<gap>` element appears. And of course there are probably quite a few character recognition errors, especially if the original printed text was unclear or damaged.

There is no automatically generated tagging beyond this; we will therefore have to add it by hand. Refresh your memory about the basic structure of an ELTeC-0 text: any front matter, such as the titlepage or a preface, should be wrapped in a `<front>`, while the text itself should be contained by a `<body>` element, grouping containing `<div>` elements.

It's up to you how you do this. Here's a suggestion:

- Type CTRL-A to select the whole text
- Type CTRL-E and surround the whole text with a `<text>` element.
- If you have front matter:
  - insert a `<front>` element at the start of your `<text>`
  - Select the paragraphs making up the titlepage, correct them, and wrap them in a `<div type="titlepage">` inside the `<front>`
  - Decide whether any other prefatory matter is authorial: if it is, wrap it in a `<div type="liminal">`, also inside the `<front>`. If it isn't, remove it.
- Add a `<body>` element to contain the body of the text, and within it at least one `<div type="chapter">`

Your next challenge is to identify the chapter divisions. You may be able to detect them using a regexp to search for specific words, as in the Polish texts, but this is not guaranteed. A text may not have any chapter divisions at all, in which case our schema requires you to wrap the whole thing in one `<div type="chapter">`. Or they may not have been correctly identified by the OCR, in which case you may have to split the current `<div>` element at the right place, using ALT-SHIFT-D for example. Use the Outline view (Document->Show View->Outline) to see the emerging structure of your document.

At some point you will want to tidy up the text a little: some things that would be easy to fix are:

- end-of-line hyphenation (where a word is broken across a line) is indicated by the special character `¬`. This makes it easy to join words up again: use a regexp to remove all occurrences of `¬\n`
- The `<pb/>` tag is followed by any running header or page number in the original source, tagged as a separate `<p>` element. You may think it worth writing another regexp to remove paragraphs containing only digits and spaces or uppercase letters to catch these, though be careful you don't remove real parts of the text.
- You may catch some recurrent OCR errors using the spell checker built into oXygen (remember to set the correct language : you may need to copy the appropriate spelling files from your Work/Spell directory to ~/.com.oxygenxml/spell)

When you're reasonably happy with the transcription, type CTRL-A to select all of it and CTRL-C to copy it. Then return to the file in which you prepared your header, move the cursor to the appropriate point inside the `<text>` element, and type CTRL-V to paste it. Is the completed document valid? If not, fix it!

3 Reprocessing docx files

Abbyy also has the useful capability to export its results in Word DOCX format. Here's a quick guide to converting this to TEI:

- In oXygen, use the usual Open dialog to select the docx file
- A window labelled Archive Browser opens to the left of the main screen. It shows the file structure of the docx archive.
- Click the blue key next to the folder called word to see the contents of this folder
- Select the file called document.xml and double click to open it (may take a few moments if the file is large)

As you see, this is an XML document, with lots of XML tags, using many different namespaces, most of them defined by Microsoft.

- With the document.xml file open in your main editing window, select Transformation -> Configure Transformation Scenario(s) from the Document menu. Or type `CTRL-SHIFT-C`. Or click the little spanner icon
- For the default DOCX to TEI conversion, check the little box next to DOCX TEI P5 and press the Apply Associated button
- A new window opens at the bottom of the screen in which oXygen displays messages relating to the conversion process ; you may safely ignore these.
- In the main editing window, you should now see a TEI-conformant document. (Choose Document -> Source -> Format and Indent from the menu, or type CTRL-MAJ-P or click the Indent button if it is displayed as a single line on the screen) Scroll down to see how the Word styles have been converted.
- Note that many of the paragraphs have a *@rend* attribute. The values of this attribute correspond with the Word style used in the original docx file. We could use these values to make a more intelligent conversion.

We will begin by using oXygen's built in xPath browser to investigate the tagging more closely.

- Type `//p` into the XPath 2.0 box at top left and press return
- Scroll down the list of results which opens below the document editing window
- You can see that most of the `<p>` elements have similar attributes, usually specifying that they contain `body text`, though there are some other formatting values
- Type //p[anchor] into the XPath box and press return again. Abbyy helpfully adds an anchor element at the start of each section of the document
- Up to you to see how other elements are used... for example, try `//hi` to see the`<hi>` elements

oXygen also provides a useful way of automating changes to the markup without the trouble of writing a stylesheet. We will use it to transform all `<p>` elements containing an `<anchor>` into `<head>` elements.

- Open the refactoring tool (select Refactoring from the Tools menu)
- Scroll down to the Elements section; select rename element
- In the dialogue box which opens, enter `//p[anchor]` next to Target elements (XPATH) and `<head>` next to New local name
- Press the Finish button: by default the changes are applied throughout the whole file
- Inside your new `<head>` elements you probably now have some redundant tags such as `<anchor>` or `<hi.>`. You can use the Refactoring Tool to get rid of these too: just select Delete elements rather than rename elements and repeat.

We now have `<head>` elements identifying the titles of chapters, but no `<div>` tags. We suggest using the find and replace command as a simple way of adding these. The search string should be `<head`, and the replacement string should be `</div><div><div>`. Test that this works as you expect before applying it to the whole file. When you have applied it, don't forget to scroll up to the beginning of the text and remove the redundant `</div>`

You can also use find and replace to get rid of redundant *@rend* values: simply use (for example) `rend="Body Text (2)"` as search string and nothing as replacement for it.

You should now be able to work through the file chapter by chapter, comparing it with the original PDF as suggested above. Check that each chapter break has been correctly detected and recorded and remember to save your work!