## Handling plain text files

*2019-09-24*

# 1 Quick ELTeC Corpus Building

We've found a great resource on the internet: 100 transcribed Polish novels, all ready to be used. Just a few problems:

- No metadata (other than filenames), so we need to build a proper ElTeC header for each one
- No markup of any sort, so we need to identify chapters, headings, and paragraphs automatically if possible

This tutorial steps you through the process of going from a plain text file to a level-0 encoded ElTeC conformant version. You should be able to carry this out without any knowledge of Polish, though we do have a Polish participant who has kindly agreed to provide advice if you get stuck.

The complete process looks like this :

- choose the text and assemble metadata about it
- create a new empty ELTeC-0 document
- complete its TEI Header
- insert the text of the corresponding file into the `<body>` of the document
- identify chapter divisions and chapter titles
- identify paragraphs

# 2 choose the text and create a document

Take a look at the POL folder. It contains 18 files we've pre-selected for you, along with a tiny bit of metadata in a file called titles.txt which supplies you with an identifier, a filename, a wordcount, and a magic incantation (see below) for each title. Choose one to work on (or we'll give you one)!

Each filename contains the surname of the author, a word from the title, and the date of first publication. You'll need to enter the name and the title in some suitable search engine (Wikipedia, Worldcat ...) to get the metadata necessary to complete your TEI Header, notably the author's full name, dates, and sex, as well as the full title.

We also know that each text is derived from a digital collection called *100 Polish Novels* transcribed by the Computational Stylistics Group in Krakow, and that it is freely available from their GitHub repository at https://github.com/computationalstylistics/100_polish_novels. We do not however know much about the print sources for these texts: hopefully your researches should throw up some information at least about their first edition in book form. Armed with that additional information for your chosen text, proceed to create a new file and a valid TEI header for the file.

This process is described in detail in the Header Tutorial.

## 3 add some text

Now we are ready to add some text to our document.

- put the cursor inside the solitary `<p>` contained by the `<body>` element.
- select Document-> File -> Insert File from the menu, and navigate to the text file you are planning to convert
- Press Open, and the content of the file is inserted into your document.

Note that the error message `The body of a text must contain at least one chapter` is still present. We will need to introduce more markup. We could do this slowly and painfully, one step at a time, but computers are supposed to make it easier to automate tasks which are slow and painful. In the next section, we'll see how you can take advantage of any systematic patterns in the format of a non-marked-up text to introduce explicit XML markup. To do this we'll use the sophisticated Find/Replace tools built into oXygen.

## 4 identify the chapters and headings

To begin, open the Find/Replace dialogue and make sure that the check box Regular expression is selected. A *regular expression* is a kind of pattern: the find and replace command usually searches for specific character strings; when regular expressions are enabled, it can also search for complex patterns of characters.

For example, in some of our Polish texts, every new chapter begins with a roman number on a line of its own. In other texts the chapter number is preceded by the word `Rozdział`; in yet others the chapter has a title given in uppercase letters only. We can write regular expressions to cater for all of these possibilities. Without regular expression matching, we could seek lines containing the explicit strings I, II, III, IV etc. But it is much simpler to seek all such lines by means of a regular expression matching any sequence of one or more of the letters I V or X followed by a new line.

- In file POL001 for example, each chapter is prefixed by a line containing an uppercase Roman number, followed by a line containing the chapter title, all in uppercase. We can use this regular expression to find these: `\n[IXV]+\n[\s\p{Lu}]+\S\n`.
- In file POL007, however, each chapter is prefixed by the word ROZDZIAŁ, followed by one or more spaces, followed by a roman number and a full stop, and a new line. Here's a regular expression which will find these : `ROZDZIAŁ\s+[IVX]+\.`
- In file POL014, chapters always begin with a lines containing a sequence of uppercase letters and spaces. Here's a regular expression which will find them: `[\p{Lu}\s]+\n`
- Of course, in some cases, for example POL004 or POL006, we can't easily identify where new chapters begin, or they may not be marked at all. Where we can however we've included a suggested regular expression in the titles.txt file (that's what the mysterious incantantion after the word count is)

The real power of regular expression matching is not just that it enables us to find quickly particular strings in the text. It also allows us to specify replacements for those strings. In our case, wherever we identify something which is the heading of a chapter, we need to tag it as a `<head>`, and also show that it begins a new chapter by inserting a `<div type="chapter">` tag, and (to ensure our document remains well-formed XML) preceding it with a `</div>` tag to close the preceding chapter. The syntax for doing this is simple enough: the replacement part of the find/replace dialogue can include numbered references (like this : `\1`) which are to be expanded by a part of the matched string, specifically the part enclosed in parentheses.

This is easier to understand with an example. Let's suppose you are working on POL001.

- Open the Find/Replace dialogue by typing CTRL-F or selecting Find->Find/Replace from the main menu, and type the incantation `\n([IVX]+)\n([\s\p{Lu}]+)\S\n` carefully into the Find box. Press the Find button to check that it is finding what you expect: you should see that the lines `I` and `ALKHADAR` have been selected.
- Each time you press Find, oXygen will step through the file selecting any line containing just a sequence of one or more I X V characters followed by a line containing any sequence of uppercase letters. Try it, just to check. When you're satisfied the regexp is working correctly, move the cursor back to the start of the file and find the first chapter again.
- The parentheses in your regexp allow us to pick out the two character strings that have been matched, for use in the replacement value. Type the following into the Replace With dialogue `\n</div><div type="chapter"><head>\1</head>\n<head>\2</head>\n` Press the Replace button. You should now see something like this at the start of your file:

```
<body>
        <p>Michał Bałucki
            PRZEBUDZENI
            </div><div type="chapter"><head>I</head>
                <head>ALKHADAR</head>
                Warszawo! niewolnico! kiedyś ....
```

- (The `</div>` is underlined in red since there's no chapter to close here. Never mind, we'll fix that later)
- When you are sure that your regexp is working correctly, press the Replace to End button to repeat this manoeuvre for the whole of the rest of the file. You should find 23 matches, and the Outline view (Select Window -> Show view -> Outline if this is not open) should show that you now have 23 chapters.

## 5 identify the paragraphs

In this file, every line other than those beginning with a < character now contains a single paragraph. We can use a simple regexp to mark them all up as `<p>` elements :

- Type `\n([^<].+)` into the Find box, and `\n<p>\1</p>` into the Replace with: box.
- Move the cursor to the start of the file and check that this regexp does what you expect
- Click the Replace to End button as before. There should be about 1994 matches and your task is almost complete!
- Finally, go back to the start of the document and delete the lines before `<div type="chapter">`, including the redundant `</div>`. Your file is valid!

Of course, there's still work to be done... we have not tried to deal with the occasional piece of verse, nor have we tried to markup the trailing headings at the end of some divisions, so these are all appearing as spurious `<p>` elements. Feel free to continue perfecting this text – or maybe you'd rather try a different one? Congratulations on getting this far!