On the end-user editor and XML scheme to store documents for Diogenet

J. Dávila (May, 10th, 2020)

Here I want to start the discussion about two issues which I think are closely related in this project. 1) How to store annotated documents, assuming that it will be in some form of XML which would produced and processed by Python script, as well a for humans, and 2) which editing software to recommend for humans to do the annotating or correcting work on those documents.

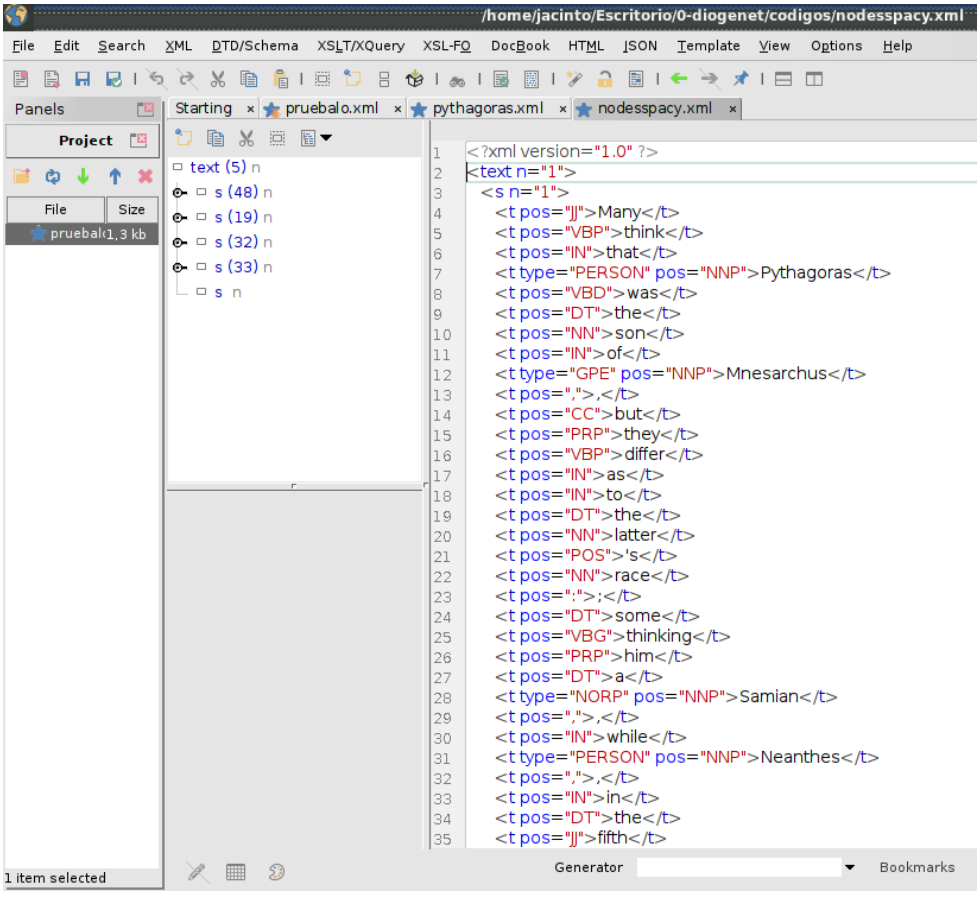
The problems are related because the editing software has to read and produce the same XML than the python scripts. Otherwise, we could not profit from this human-in-the-loop relation extraction processing we are aiming at.

The editor has to be easy enough for humans to identify examples of relations in the documents and, when possible, annotate them for automatic extraction. But, also, it should not be too complex to process, neither should relax the XML strict markup structure, which makes the documents easier for computer consumption.

With all that in mind, I tried three pieces of **free** software that, I think, illustrate the range of possiblities. Namely:

Editix, a piece of Java software, can be downloaded here https://www.editix.com/download.html

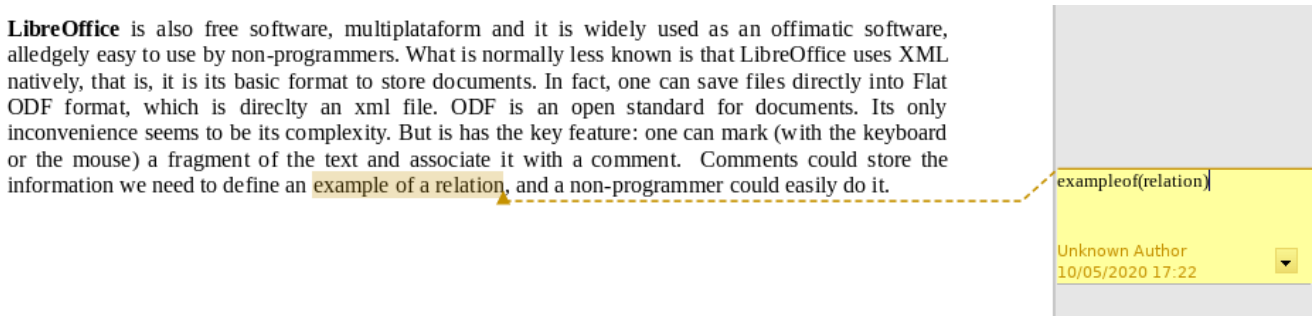
It is a typical XML editor for programmers, able to read any xml file. We could use it to open and process any xml file created by the scripts. This is an screenshot with a file we created earlier in the project:



But writing directly into XML format is a difficult work. Programmers are used to work with complex source codes and therefore, see no special difficulty with an editor that even display tags with colours and check the syntax. But with non-programmers, it is very different and frustation can lead to mistakes and resistance to do the work.

On the other extreme, one finds tools as the one I am using to write this report: LibreOffice.

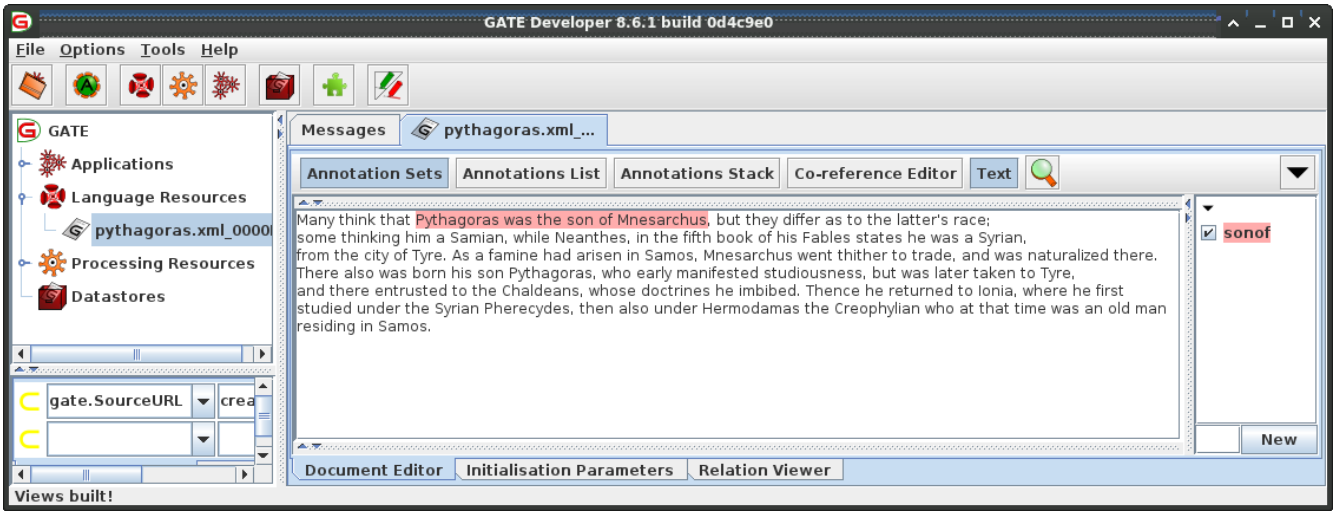
LibreOffice is also free software, multiplataform and it is widely used as an offimatic software, alledgely easy to use by non-programmers. What is normally less known is that LibreOffice uses XML natively, that is, it is its basic format to store documents. In fact, one can save files directly into Flat ODF format, which is direclty an xml file. ODF is an open standard for documents. Its only inconvenience seems to be its complexity. But is has the key feature: one can mark (with the keyboard or the mouse) a fragment of the text and associate it with a comment. Comments could store the information we need to define an example of a relation, and a non-programmer could easily do it.



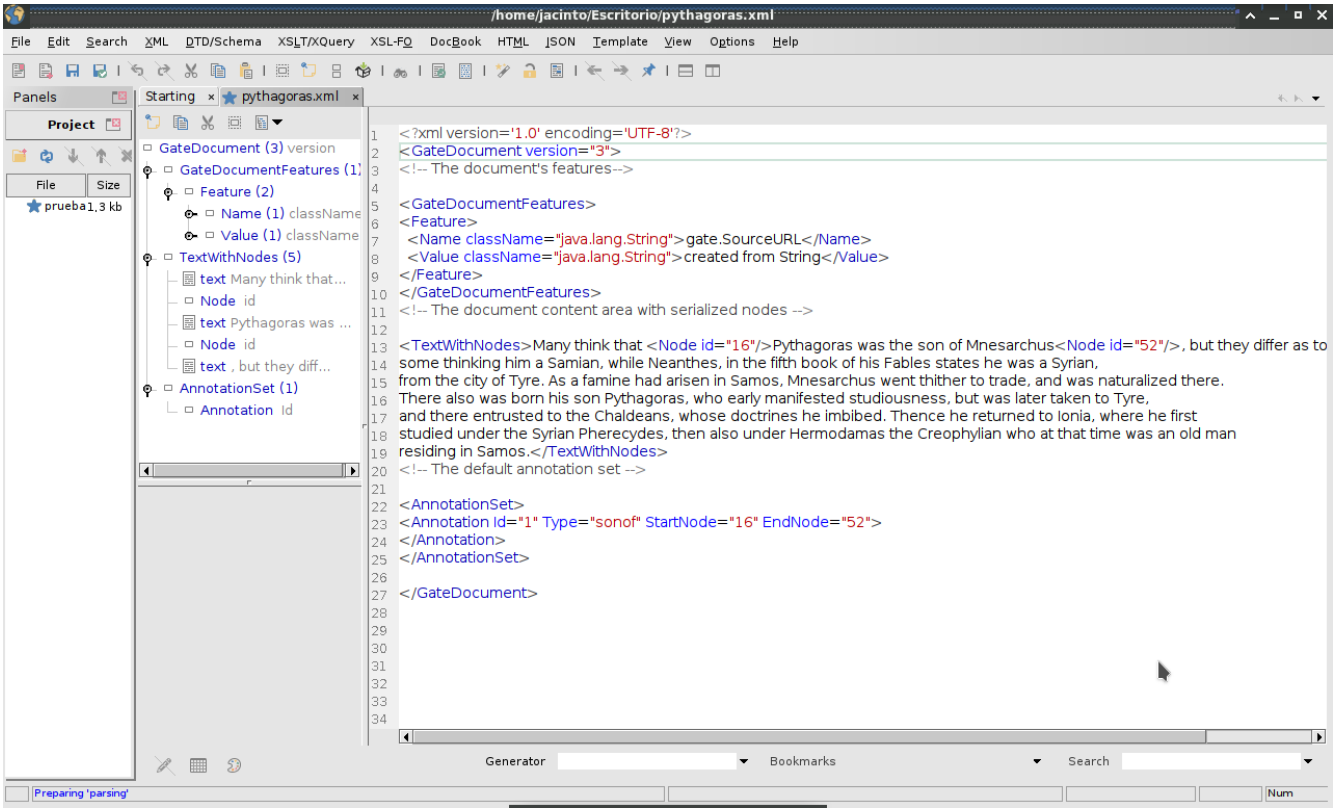
On the downside, the underlying XML format is very complex:

```
name="T4">LibreOffice</text:span> is also free software, <text:span text:style-
name="T6">multiplataform</text:span> and it is widely used as an offimatic software, alledgely
<text:span text:style-name="T6">easy to use by non-programmers. What is normally less known is that
LibreOffice uses XML natively, that is, it is its basic format to store documents. In fact, one can
save files directly into Flat ODF format, which is direclty an xml file. ODF is an open standard for
documents. Its only inconvenience seems to be its complexity. But is has the key feature: one can
mark (with the keyboard or the mouse) a fragment of the text and associate it with a comment.
<text:s/>Comments could store the information we need to define an </text:span><text:span text:style-
name="T6"><office:annotation office:name="__Annotation__6_419715826"
loext:resolved="false"><dc:creator>Unknown Author</
dc:creator><dc:date>2020-05-10T17:25:58.500450420</dc:date><text:p text:style-name="P13"><text:span
text:style-name="T7">exampleof(relation)</text:span></text:p></office:annotation></
text:span><text:span text:style-name="T6">example of a relation</text:span><office:annotation-end
office:name="__Annotation__6_419715826"/><text:span text:style-name="T6">, and a non-programmer
could easily do it. </text:span></text:p><text:p text:style-name="P7"/><text:p text:style-
```

Trying to reach a balance, I went back to a very mature project in computational linguistics: GATE. It stands for General Architecture for text engineering, and it is a two decades effort to produce systems to collect and manage corpuses for computational linguistics. The project has developed (in Java) a very useful XML editor which falls between the two previouly mentioned in our space of conditions. Users do not need special training in computing and they can easily mark (again with the keyboard or with a mouse) fragments of text to associate them with some marginal tag, just like the comments in LibreOffice.



And this is how the underlying XML looks like (I opened it with Editix):



The GATE Editor can be downloaded here: https://gate.ac.uk/download/

My preliminary recomendation is to text these three and perhaps others editors to get a documented impression from users. We could discusss the technical details of the underlying XML after that.