LIPN Annotator manual

F. Lévy, A. Guissé, S. Szulman

Sept. 2012

The LIPN Annotator marks the occurrences of given terms in a text with concepts and individuals of an ontology. It outputs a project which can be directly opened by SemEx, the LIPN semantic explorer[1], to explore the annotations, mark and transform rules, etc. The user can alternatively choose to produce plain result files and to work them with her/his own programs. The output format is textual (.html and .txt) and self explaining.

The output format is language independent, as are the algorithms, so the application can in principle be used for any language where its input makes sense – namely where lemmatizing and POS tagging are possible and not too ambiguous.

The Annotator is included as a plugin in SemEx and in Terminae and can be used from them if preferred. Only the installation differs.

1 Installation

The LIPN Annotator is an autonomous java Eclipse RCP application with a graphical interface, so it need only be copied on your machine. It is provided for Linux, Mac OS X and Windows. Choose the convenient compressed version on the site, download it and uncompress it where you keep applications. To launch it, double-click in the newly created directory on the icon of the application.

Linux specific: Eclipse's browser calls native browsing libraries to do its work. Under Linux, you may have to install specific ones: the present version of the annotator relies on Eclipse 3.7, which browser needs a proper installation of one of Mozilla 1.4 GTK2 - 1.7.x GTK2, XULRunner 1.8.x

[1] The Annotator and SemEx can be found from `http://www-lipn.univ-paris13.fr/~szulman/Annotator/annotator.html` or `http://www-lipn.univ-paris13.fr/fr/rcln` or from `http://www-lipn.univ-paris13.fr/fr/rcln-logiciels`

- 1.9.x and 3.6.x (but not 2.x), WebKitGTK+ 1.2.x and newer. If your installed browser is either too old, or too recent, you can install also XULRunner (the autonomous heart of Mozilla, Firefox and Thunderbird), to enable Eclipse browser. In this case, you have to specify where XULRunner is: modify the annotator.ini file in the executable's directory, to initialize org.eclipse.swt.browser.XULRunnerPath, e.g.
-Dorg.eclipse.swt.browser.XULRunnerPath=/home/szulman/outils/xulrunner-sdk/bin
(Of course, hou must replace /home/szulman/outils/xulrunner-sdk/bin with your own location);

2 Input files

To annotate a document, you need 4 inputs :

- The document itself, in a single text (.txt) file ;
- The output of a morphological analyzer and POS tagger, in three tab-separated columns (word, POS, lemma) ;
- A lexicalization file following the SKOS standard, such as provided by TERMINAE when it builds an ontology. This file can also be created or modified with a plain text editor ; Its DTD is at the end of this document.
- One or several ontologies in OWL format.

3 How to proceed

The ontologies and their lexicalization can generally be reused for several documents. The POS file is of course document dependent, and must be generated before annotating.

When the Annotator is launched, choose the directory of your project (if you don't agree the default value, a browser will ask for another one) and the input encoding (if you don't agree UTF-8, you'll have to choose another one). The output will be encoded in UTF-8.

Then a window opens (see fig. 1) with four fields in the left pane ("Used resources") and with a blank right pane entitled "Annotated text view". Browse in the four left pane fields for the files which have been prepared. Then run by clicking on the button with a triangle down this pane.

The annotated text appears in the right pane. You can check it and, if satisfied, save it : two buttons up the right pane allow to save either a project

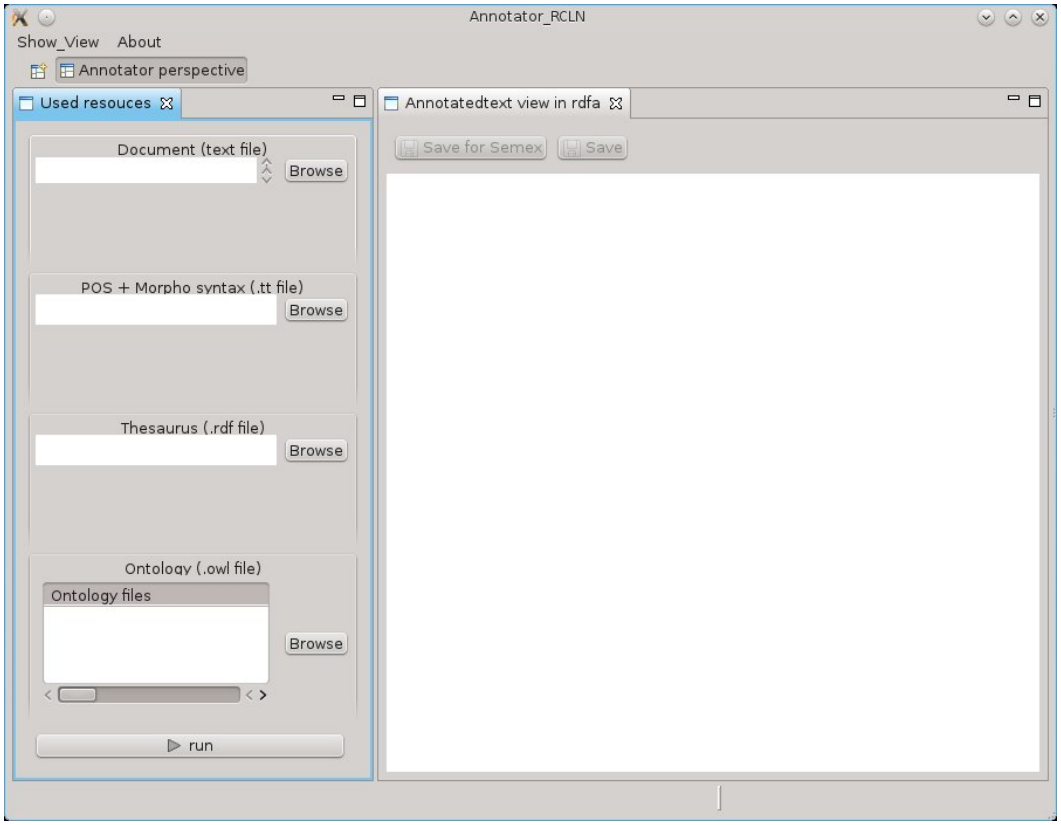

Figure 1: The Annotator window

which SemEx can use, or only two files describing the annotations according two different formats. Then, if you continue annotating some more files for SemEx, you can store the new results in the same project or create a fresh one.

4 Some caveats

The document must be in text format, so pdf and other elaborated files have to be converted. It is required to use the same encoding in the three files where non-ascii characters may appear (text, POS and SKOS). UTF-8 is proposed by default, but other encodings can work too. Due to OS and source files diversity, encoding may need some care. When debugging anomalies, the text and POS file being non homogeneous results in scope

errors and misses of the annotations. The SKOS and POS file being non homogeneous results in misses.

Sentence splitting and word splitting are provided by the POS tagger. Depending on it, sentence boarders may happen to be internally incorrect, e.g. because titles have no end point. But the output exactly preserves the appearance of the input (white space, line length, blank lines). Some typography may be ambiguous w.r.t. word splitting (e.g; "the upper/middle class"), and we have had a version of a POS tagger which blows out ÿ - with some poor effects on the annotation.

The lexicalization of the ontology described in the SKOS file associates several lexical forms to a single labeling entity. Each lexical form stores the lemmatized form of words (don't forget it if you create your own SKOS). As this form is also computed by the morphosyntactic parser, lexicalizations are recognized independently of morphological variants. Note that the technique is a bit over-productive, due to ambiguity of lemmas. We plan to improve it by using the POS category. On the other hand, before annotating according to the SKOS file, the labeling entity is checked against the ontology ; if it is not present there, the annotation is skipped. Discrepancies between SKOS and OWL files are logged in annotator.log in the result directory, and it can be wise to check the content of this file;

Here is the DTD of the SKOS file:

```
<!Element rdf:Description (skos:prefLabel, skos:altLabel*, rdf:type)>
<!ATTLIST rdf:Description rdf:about CDATA>
<!Element prefLabel (#PCDATA)>
<!Element altLabel (#PCDATA)>
<!Element rdf:type EMPTY>
<!ATTLIST rdf:type rdf:resource CDATA>
```