SAPIENZA
UNIVERSITÀ DI ROMA

Setting Up The GATE System for OSINT Project

Authors:

Giulio Ganino
Domenico Lembo
Massimo Mecella
Federico Scafoglieri

1. SETTING UP THE SYSTEM

In this Section we show all the steps needed to set up our solution. We remark that what we describe in the following is not a simple collection of pieces of documentation available for GATE and the processing resources we use. Indeed, some of these resources are lacking of complete or adequate documentation, and their combination in our pipeline requires special configurations. Therefore, the instructions we list below are crucial to properly set up the environment for our aims.

1.1. Downloading and Installing GATE

To download GATE, point your web browser at `http://gate.ac.uk/download` and download the installer for GATE Developer 8.2 (∼570MB) through the selection of Generic installer for any platform. This is an executable JAR file that should run when you double click it. If this fails, run it from the command line using `java -jar gate-8.2-build5482-installer.jar`. GATE runs anywhere that supports Java 7 or later, including Solaris, Linux, Mac OS X ,and Windows platforms.

1.2. Loading Processing Resources

In GATE, processing resources are used to automatically create and manipulate annotations on documents. The CREOLE* plugins allow to use the Processing Resources (and certain language resources) provided by GATE. In the CREOLE plugin menu we can choose the plugin that contains the processing resource that we want to use. Notice that a single plugin may contain various resources, e.g., in Figure 1, we show the resources that are in the ANNIE plugin.

The CREOLE plugins can be managed through the graphical user interface which can be activated by selecting "Manage CREOLE Plugins" from the "File" menu. This will bring up a window listing all the known plugins. For each plugin there are two check-boxes, one labelled "Load Now" , which will load the plugin, and the other labelled "Load Always" which will add the plugin to the list of auto-loadable plugins. A "Delete" button is also provided, which will remove the plugin from the list of loaded plugins. This operation does not delete the actual plugin directory, so that it can be reloaded in future usages.. Installed plugins are found automatically when GATE is started; if an installed plugin is deleted from the list, it will re-appear next time GATE is launched.

In order to set up our pipeline we need to select the Load Now box in the CREOLE Plugin Manager for the following plugins:

- *ANNIE*, from which we take the following processing resources:
  - *Document Reset*
  - *GATE Unicode Tokeniser*
  - *RegEx Sentence Splitter*
  - *ANNIE Gazetteer*
  - *Jape Transducer*
- *Tagger_Framework*, which contains the *Generic Tagger Processing Resource*
- *OwlExporter*. To load this processing resource we need to download the creole.zip file from `http://www.semanticsoftware.info/owlexporter`, unzip the downloaded file, after that push the button with the symbol + in the upper-left corner of the tab "Install Plugins" (cf. Figure 1), and specify the path of the unzipped folder containing the OwlExporer resource. Finally we can flag the Load Now box in correspondence of OwlExporter in the CREOLE Plugin Manager.

1.3. Configure Processing Resources

To set and run CREOLE resources, we need to go back to the GATE home page (Figure 2). By a right click on the menu "Processing Resources", the list of the resources loaded in the previous step

*`https://gate.ac.uk/sale/tao/splitch4.html`

Prepared using speauth.cls
DOI: 10.1002/spe

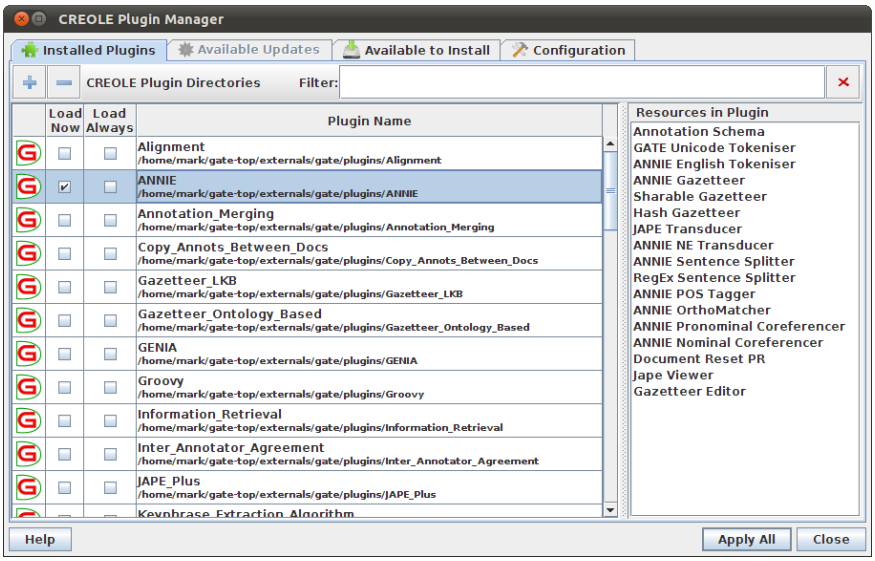

Figure 1. The CREOLE menu.

will be shown, so that it is possible to select the resources to be configured (in the GATE terminology this selection is called resource creation). When a specific processing resource is created, it appears as a sub-item of the menu “Processing Resources”, as we can observe in Figure 2 for the GATE Unicode Tokeniser.

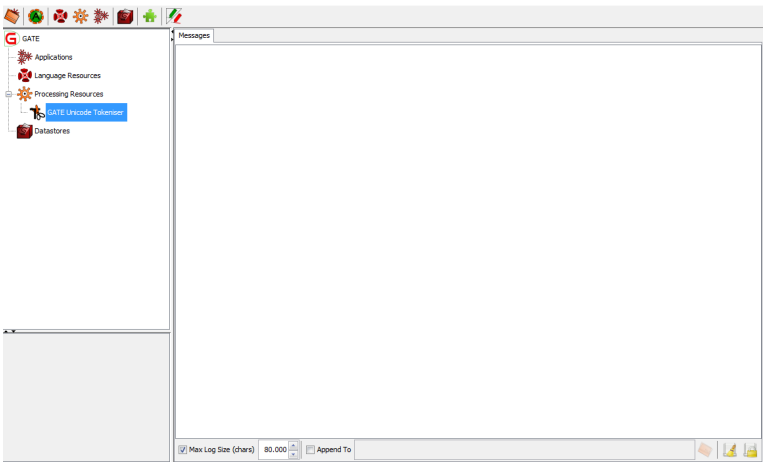

Figure 2. The GATE menu.

Notice that it is possible to select the same processing resource more than once, and thus create more than one instance of the same resource, which however has to be assigned with a different name. This is particularly useful if the same resource will be used several times with different parameters. Below we describe how to set the initialisation parameters for each resource (run-time parameters will be set later).

Document Reset. For this processing resource the only parameter that can be modified is the name. In our solution we have used the default name assigned to such resource (as shown in Figure 3).

Gate Unicode Tokeniser. Besides the name, two other parameters can be set: rulesURL and encoding. The former contains the URL for the ruleset file, which is the file containing the rules for

DOI: 10.1002/spe

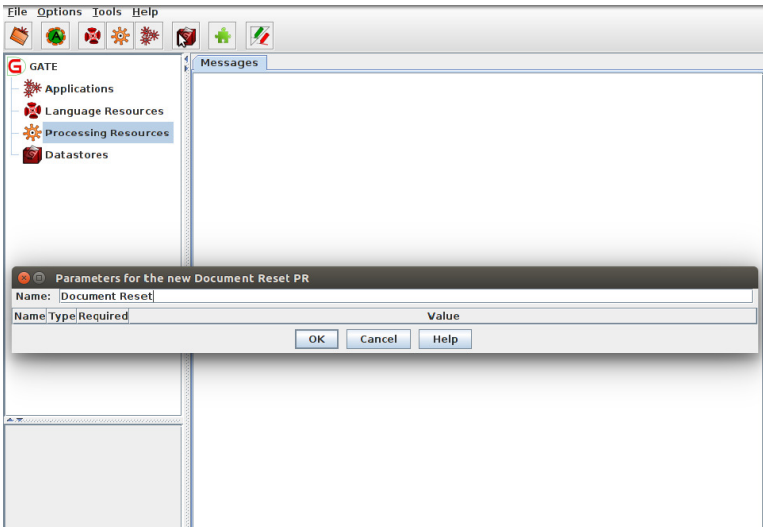

Figure 3. The initialisation parameters of Document Reset.

the tokenization process. The latter indicates the encoding used for text documents to be analyzed. In our solution, we use the standard name for the processing resource, the standard path for rulesURL, whereas we set the value of the encoding to UTF-8, as shown in Figure 4.

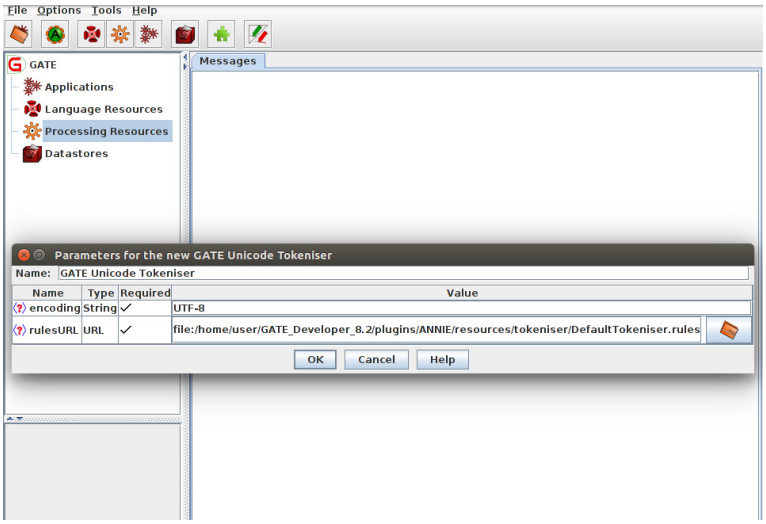

Figure 4. The initialisation parameters of GATE Unicode Tokeniser.

RegEx Sentence Splitter. As before, for the encoding parameter we select UTF-8. For the remaining parameters we use the standard initialisation values. In particular, we do not need to change the value of the following attributes:

- *externalSplitListURL*, which is the URL for the file containing the list of external split patterns;
- *internalSplitListURL*, which is the URL for the file containing the list of internal split patterns;
- *nonSplitListURL*, which is the URL for the file containing the list of non split patterns.

In Figure 5 we show the initialisation parameters of RegEx Sentence Splitter.

DOI: 10.1002/spe

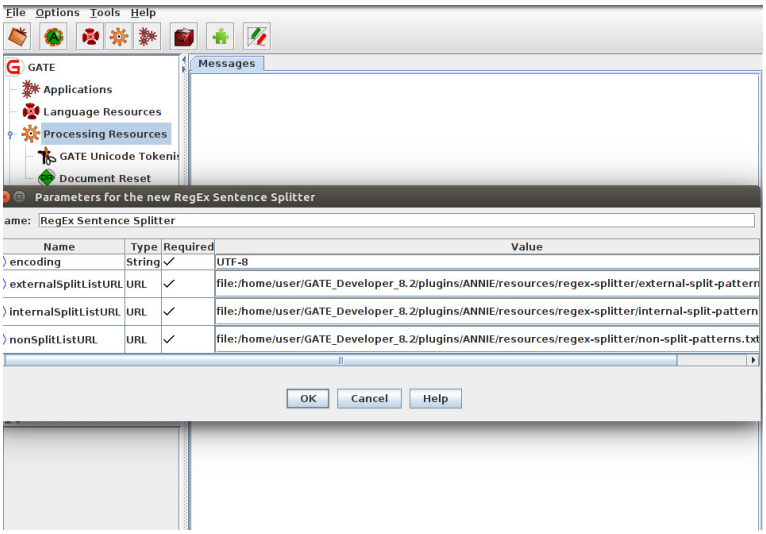

Figure 5. The initialisation parameters of RegEx Sentence Splitter.

TreeTagger Part Of Speech. We first select the item Generic Tagger from the menu Processing Resources and set its parameters. The preProcessURL parameter has to point to a JAPE grammar. This is used to check whether the annotations created by the Sentence Splitter used in the previous phase are in fact compatible with the analysis that the tagger has to perform. This step is the same, whatever tagging resource is used. More precisely, we set preProcessURL to the path of the file `sentencestring.jape` (cf. Figure 6).

The postProcessURL parameter is instead used for adding special annotations, which we do not use in our solution and thus leave this parameter blank.

Since we want to use a third-party POS tagger (i.e., TreeTagger POS), we further need to execute some downloads at the link `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`. More precisely, we need to download: the *tagger package* for the specific operating system we are working with, the *tagging scripts*, the *installation script*, and the *parameter file* for the language that we want to process. We put all the downloaded files in the same directory and unzip the compressed archives. Since we want to use the Italian version of TreeTagger, we need to modify the default settings in the file `tree-tagger-italian` present in the folder `tagger-scripts/cmd`. To this aim, we open `tree-tagger-italian` with a text editor and set the following parameters:

- *TOKENIZER*, which has to point to the file `utf8-tokenize.perl` present in the folder `tagger-scripts/cmd`;
- *TAGGER*, which has point to the file `tree-tagger` present in `bin` folder of the tagger package;
- *ABBR_LIST*, which has to point to the file `italian-abbreviations` present in the folder in the folder `tagger-scripts/lib`;
- *PARFILE*, which has to point to the file `italian-utf8.par`, which is parameter file downloaded for the Italian language.

Notice that the file `tree-tagger-italian` will be pointed by run-time parameters of the Generic Tagger, so that it will in fact run the Tree Tagger POS.

Gazetteer. For this Processing Resource, we set the parameter listsURL so that it points to the `.def` file containing the indexes of the Gazetteer lists we want to use. Furthermore, we set the caseSensitive parameter to true, the encoding parameter to UTF-8, and the gazetteerFeatureSeparator parameter to ":", which is the special character used to separate the name of the list from the major and minor types in the `.def` file.

DOI: 10.1002/spe

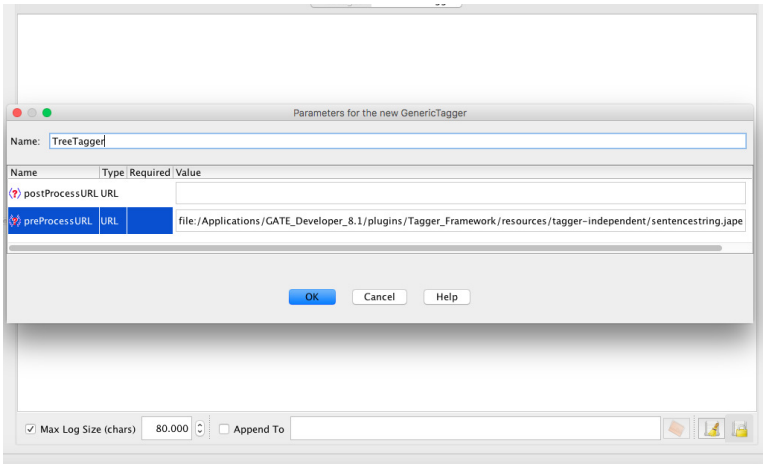

Figure 6. The initialisation procedure to load Italian TreeTagger POS.

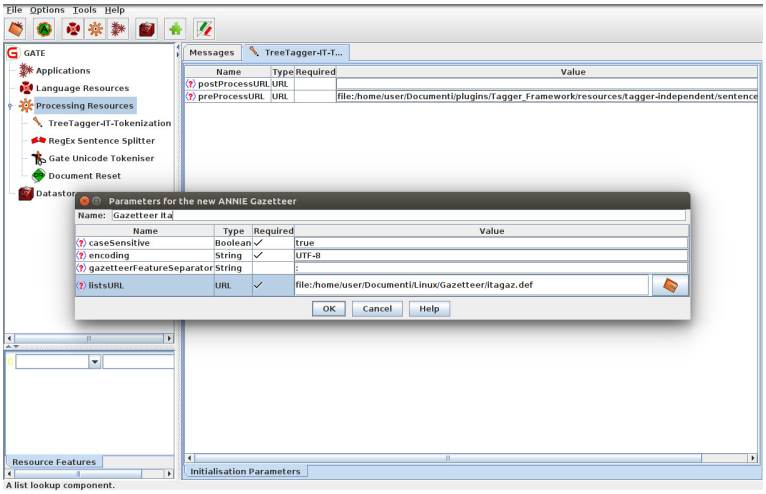

Figure 7. The initialisation parameters of Gazeteer.

Jape Transducer Semantic. For this Processing Resource in grammarURL parameter we have to point the file `main.jape` that contains the identifiers of Jape Semantic rules we want to use (in other terms this file acts as an index of the rules). Furthermore, as usual, we set UTF-8 for the encoding parameter, whereas for the annotationAccessors and operators parameters we adopt the default values. Indeed, such parameters are used to add custom operators and meta-property abilities to the JAPE language, which are features that are not used in our solution. In Figure 8 we show the initialisation parameters of Jape Transducer Semantic.

Jape Transducer MunPEX and Jape Transducer Mapping. Parameters of these processing resources have the same meaning as the ones of the Jape Transducer Semantic, and are set similarly. The only difference is in the value of grammarURL, which refers to a different files of JAPE rules.

OwlExporter. For this Processing Resource we have used the default initialisation parameters. Notice also that changes to the exportFormat parameter are not in fact possible in the practice, since the only format supported for the export is OWL (in the XML/RDF syntax), and thus the only thing we can do is to change the name of the processing resource (which is not necessary).

Prepared using speauth.cls
DOI: 10.1002/spe

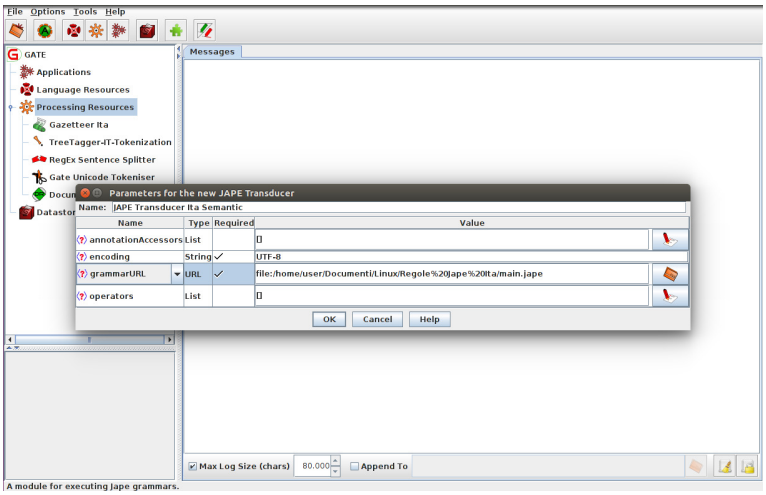

Figure 8. The initialisation parameters of Jape Transducer Semantic.

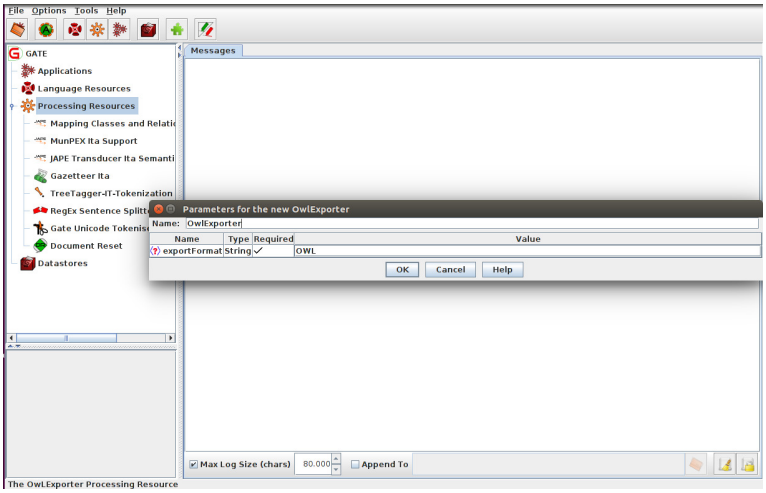

Figure 9. The initialisation parameters of OwlExporter.

1.4. Loading Language Resources

To import the document to be analyzed, the steps below have to be followed:

1. Right-click on “Language Resources” and choose “New”, then “GATE Document”.
2. In the dialog box, choose the file you want to open in GATE or type a URL.
3. Set “markupAware” to true, so that GATE detects a proper reader for the document. If no reader is found, the content of the document is loaded and presented to the user, but the document formatting may be loose.
4. Provide a document name or leave blank to use an automatically generated name, and select “OK”.
5. The document will appear under the list of Language Resources loaded in the system.
6. To view its content, double click on its name.

Prepared using speauth.cls
DOI: 10.1002/spe

Gate supports a variety of formats - HTML, XML, SGML, plain text, emails, etc. For our tests we have used the XML format, which has shown the fastest performance compared to the other formats.

After the creation of new documents, we can merge them in a corpus. To create a corpus and populate it with documents, proceed as follows:

1. Right-click on “Language Resources” and choose “New”, then “GATE corpus”.
2. Provide the corpus name (optional).
3. Choose which of the currently loaded documents have to be added to the corpus, by clicking on the list editing button (optional).

Alternatively, to populate a corpus from a directory, follow the steps below:

1. Right-click on a corpus and choose “Populate”.
2. Choose the directory where your files are located.
3. Specify the file extension, so that only files with this extension are loaded (optional).
4. Specify the encoding to be used (optional). If left blank, then the default platform encoding will be used.
5. To view the corpus, double click on its name, and to view documents displayed inside a corpus, do the same on their names. From a corpus view it is possible to add/remove documents using dedicated buttons.

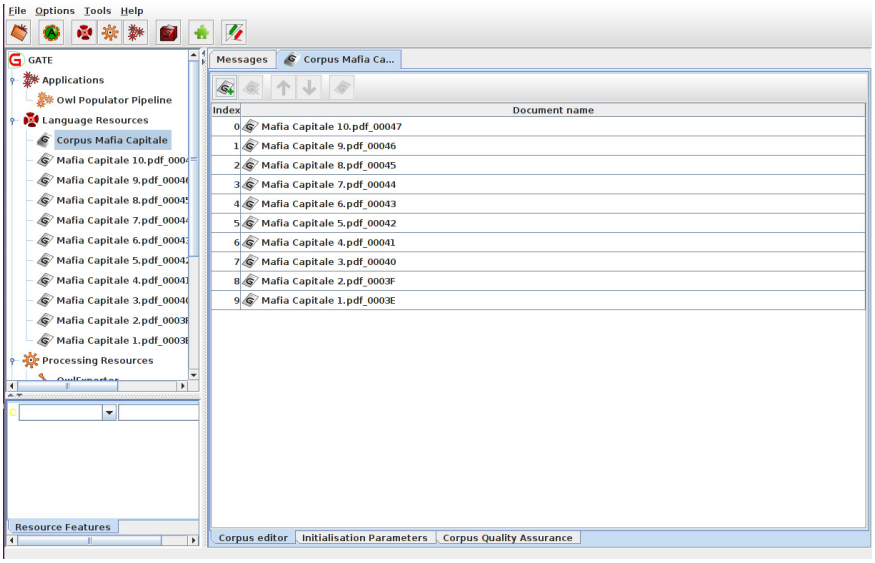

Figure 10. An example of documents collected in a corpus.

1.5. Creating and Setting a new Application

Once all the resources you need have been loaded, an application (set of processing resources in a specific order) can be created from them, and run on your text corpus. Right click on “Applications” and select “New” and then either “Corpus Pipeline” or “Pipeline”. A pipeline application can only be run over a single document, while a corpus pipeline can be run over a whole corpus. To build the pipeline, double click on it, and select the resources needed to run the application (you may not

Prepared using speauth.cls
DOI: 10.1002/spe

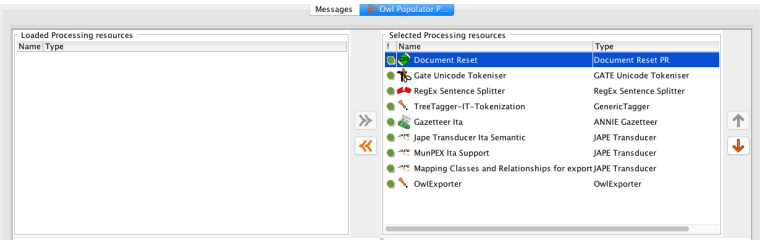

Figure 11. The pipeline of Processing Resources for Ontology Population from Text.

necessarily wish to use all those that have been loaded), i.e., move the selected resource from the set of "loaded components" (left hand side of the main window) to the set of "selected components" (right hand side of the main window) (cf. Figure 11). To order the components in the pipeline, use the up/down arrows. In our solution, we use the the following order:

1. Document Reset
2. Gate Unicode Tokeniser
3. RegEx Sentence Splitter
4. TreeTagger IT
5. Gazetteer
6. Jape Transducer Semantic
7. Jape MunPEX
8. Jape Mapping Classes and Relationships
9. OwlExporter

Now it remains to set up the run-time parameters for the processing resources we have selected in the pipeline. They can be changed as follows:

1. Click on the application name, and
2. Click on the Process Resource into the application menu that contains the runtime parameters that you want to change.

We need to change runtime parameters for only three processing resources, whereas the others maintain their default values:

- *TreeTagger*, for which we set five parameters:
  - *endcoding* to UTF-8;
  - *inputAnnotationType* to Sentence, which is the annotation produced by the RegEx Sentence Splitter component;
  - *outputAnnotationType* to SemanticToken, which is the annotation produced by the TreeTagger component;
  - *taggerBinary* to the path of the file *tree-tagger-italian*, present in the `cmd` folder of the tagging scripts;
  - *updateAnnotations* to false.
- *Gazetteer*, for which we need to chance the parameter *longestMatchOnly* from true (default) to false;
- *Owl Exporter*, for which we set four parameters:

DOI: 10.1002/spe

- *exportDomainOntology* to the location and file name of the domain ontology in output;
- *exportNLPOntology* to the location and file name of the NLP ontology in output;
- *importDomainOntology* to the path of the file that contains the pre-existing domain ontology that the OwlExporter uses to populate and create the output ontology (i.e., the one set in the exportDomainOntology parameter);
- *importNLPOntology* to the path of the file that contains the pre-existing NLP ontology that the OwlExporter will use to populate and create the output NLP ontology (i.e., the one set in the exportNLPOntology parameter).

We thus are finally able to run the application by selecting the corpus that we want to analyze in the "Corpus" box, and by clicking "run this application".

DOI: 10.1002/spe