# TextPro

## By:

Mohammed Qwaider
Christian Cirardi

04-2013

TextPro:
is a NLP pipeline tool which provide processing on a given text, on a specific way, using some modules attached to the system.
Text: should be provided as a raw text in a file, or a formated text in tabler way, providing a header specified as:
“#FIELDS columnName1 columnName2 .... columnNameN” between each columnName and the another is a tab.
Between “#FIELDS” and the first columnName is a space.
and that was the standard header which we use for all the processes in the system.

The advantage on this tool that, it is quite dynamic to improve the flexibility of running the modules, so we could run all the modules together, ignore running a module, or delete a module, add new parameters to run a module, add new column needed to run a module without affecting the other procedures in the system.

The complexity on the system is on building the modules.xml file, which have all the information for each module:

for each module we should have the following:

1- “<module name="CleanPro" lang="it">”

a name and a language to that module.

2- “<descr>FBK HTML relevant content extractor</descr>”

a description of that module.

3- “<cmd type="javaclass">fbk.htmcleaner.CleanPro</cmd>”

The class path to run the module.

4- “<input channel="PIPE" encoding="UTF8"/>” - No input “raw text is the input”

“<input channel="PIPE" encoding="UTF8">

<field name="token"></field>

</input>” - “token” column is the input for that module, so the tool will filter the input file to have intermediate file with just one column.

Note: the intermediate file (could/Not) have a header, see below.

The expected text format to run the module, e.g.:

1)A raw text without header “#FIELDS ...”. so we should not provide any field under “input” tag.

2)A file with a specific columns as input to the module.

so we should specify those required columns to filter the input file and provide just the required columns to the module.”Note: if the module require a header, you should specify that in the “parameters” tag, by adding “-h i””.

5- “<output channel="PIPE" encoding="UTF8">

<field name="sentence"></field>
</output>” - The expected output columns added to the intermediate file which passed to the module.
so we expect to have a new column called “sentence” with the value produced by the module.
Note: if the module produce a header, then you should specify that on the “parameters” tag by adding a field “-h o”

if there will be a header in the output of the module, then the merge will be depending on the index order of that header, otherwise, will handle the ordering depending on the ordering in the modules.xml

6- “<params>
<param name="language">-l $language</param>
<param name="disable">-d sentence</param>
<param name="eos">-eos</param>
<param name="headerInputModule">-h i</param>
</params>” - Those are the parameters which will passed to the module, while initiating the object of the module by the a method call init(Object[] args).
* “$language”: is a reserved string represents the value of the language, in this sense, if the value of the $language = “italian” then the passed parameter will be “-l italian”.
* “$inputFile”: represents the input file path and the file

name.
* “$outputFile”: represents the output file which we will get from the system “ambiguous usage ****Mohammed”.
* “-h i”: means that the module should get an input with a header represents the information of the input.
this field value “-h i” will not passed to the system as a parameter, but it will help on building the intermediate file passed to the module.

* “-h o”: means that the module will give a header with the output represents the information of the module output.
this field value “-h o” will not passed to the system as a parameter, but it will help on merging the module output file with the original input file, and using the module output header, we merge the files.

Note: if we have space in the parameters,we split from the space to tokens, each token will be sent in an index in the array of objects “Object[]”, and the module itself should manage to know the passed parameters.

Running the pipeline:
We would explain the procedure from an example of a user process:

A user need to produce the lemmatization through our system, starting from a raw text file.
the module.xml should be set up with the required modules, in our example:
LemmaPro, TAGPro, MorphoPro, TokenPro modules should be filled in the modules.xml with their dependences and information, as thus modules will lead us to have the lemmatization for the text.

The system will calculate and know the required modules by itself in a dynamic way!
if one modules fails to run or not found the process will fail.

The system will order the calls of the modules dynamically, depending on the required input column needed to call that module.
So to call lemmapro, we need to have “token, pos and full_morpho” columns and to have those, we need the TokenPro, MorphoPro and TAGPro to be active and produced their output correctly.

How to define the order of the Modules “Which to run first”:
From the “Input” tag, we could know who donot need any column to run, in our example “TextPro” won!

Then we would find TAGPro and MorphoPro have the same order, so we donot have any problem to who will run first, so we keep it to the system which one it read first and run!
and finally after having all the dependencies of the system there would be the lemmaPro to run!

In each time we run a module, that module would have some specification to run, for example:
header or specific columns, so we filter the file which will be passed to the module as the specification which we find in modules.xml.
after finish the process of that module, an output is expected to be added to the original file, so we filter the module output to get just the output of that module, then merge it with the original file, then complete to the next module.

As a final stage, and after running all the modules in the system, the tool will filter the final result as the user asks, in other words, to give the user what he asked!

***** important ,

if a module does not have an output to give, it is value should be NULL