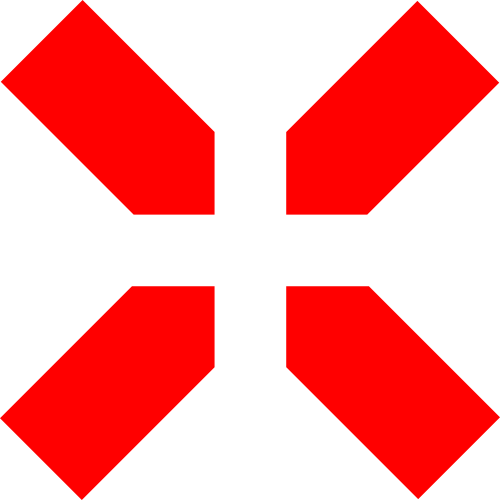


SEMANTIC EXTRACTOR



User Guide

Document date: 2015-12-22

Revision 1.3

Author: Maxim Olivier-Adlhoch

The semantic extractor is a stand-alone utility which allows you to leverage the Coginov Semantics Engine outside of a web environment. It can be useful in order to quickly use semantics within another application.

The current extractor application is very simple and has few options, since most of the work is being done from within the XML used by the application.

How does it work?

You simply create an xml file which contains the source text to process, a list of semantic extraction functions you wish to perform on this text, and parameters you wish to use for each of those functions.

You decide where you want the result of that processing to end up, and you launch the application with the path to the source XML and the destination.

When the application is done, it will save out its result. Simply launch the Extractor synchronously and read the result file when it's done.

The command line

(NOTE AS-OF extractor r172, all command-line switches use the slash '/' instead of the dash '-'.

coginov-semantic-extractor [options] /i | /input {path} /o | /output {path}

Required arguments:

[ /i | /input ] {path} Provide path to input XML specification

[ /o | /output ] {path} Provide path to XML result file.

Options:

[ /v | /verbose ] open the console and show some feedback

[ /p | /prompt ] open the console and request user's prompt before quitting. This can be useful as a simple break-point for debugging.

Ex:

C:\prtbl\_apps\coginov-semantic-extractor /v /p /i {C:\tmp\sem\_req.xml} /o {C:\tmp\sem\_rslt.xml}

Paths must be included within a pair of curly braces, be careful not to put spaces between braces and path. Spaces are allowed in paths.

Note: all file paths should be absolute.

The XML input file

The XML request used by this application is the same one as used by the Coginov Web API.

Most of the API's XML related documentation applies to the extractor directly. Notable changes are listed below.

Here is an example of a simple XML request file:

<soapenv:Envelope xmlns:soapenv=<http://schemas.xmlsoap.org/soap/envelope/> xmlns:api="http://www.coginov.com/api">

<soapenv:Header/>

<soapenv:Body>

<api:coginovApiRequest>

<api:document>

<api:type>T</api:type>

<api:source>Pablo Picasso was a great painter.</api:source>

</api:document>

<serviceCall>

<serviceName>ExtEntity</serviceName>

</serviceCall>

<serviceCall>

<serviceName>ExtConcept</serviceName>

</serviceCall>

</api:coginovApiRequest>

</soapenv:Body>

</soapenv:Envelope>

This request would return a list of entities found in the source text (Pablo Picasso, in this case).

For a list of service names, refer to the API reference guide.

# Notes about the XML specification:

* Tag namespaces are optional, the extractor ignores them.
* The Web API <authkey> tag is now ignored, it is replaced by a dedicated license file.
* You CANNOT use any other document type than text ( <api:type>T</api:type> )
* Not all WebAPI services are available within the extractor.
* The <api:source> is where you put the text you want to analyse. You can put it in plain-text or you can also Base64 encode it. This is preferred since it will prevent any text from interfering with XML syntax.
* As of r172 you can specify the document language manually, by adding a <lang></lang> tag in the <document> tag. In some circumstances this may increase processing speed. You can use any of : <lang>fr</lang>, <lang>en</lang>, <lang>es</lang> .

Ex:

<api:document><

api:type>T</api:type>

<api:source>Pablo Picasso was a great painter.</api:source>

<lang>en</lang>

</api:document>

Supported services

These are the Services (specified in the <serviceName> tag) which are explicitly supported by the extractor:

# ExtConcept

This is the general concept extractor. It has many parameters, you can refer to the technical documentation for its various <serviceParam> tags.

# ExtEntity

This is a more specific version of concept extraction which attempts to find Entities within the text.

# ExtCategory

This will return the general category of a file. It is possible to ask for a category file to use, allowing you to find more specific categories within a knowledge domain.

# ExtSummary

This will return a summary of the text by detecting which concepts are the most prevalent and choosing text based on those concepts. You can ask for more or less text in the result to get longer or shorter summaries.

# ExtractCV (new in r172)

This will analyse a complete CV/Resume, and returns useful information it can gather from the text. Because CVs are so varied in style and format, it is a bit forgiving. This is still in active development and will continue to evolve and improve for a few releases. Note that this service only reports information it finds. Thus, if no phone number is found in the CV header, for example, it will not include an empty phone list, **it simply doesn't return any** phone related xml tag in the result. Note that the ExtractCV service is only enabled in French language in the current version of the extractor.

# DTETime

This will detect all kinds of time and date related patterns. This goes far beyond date notations and includes passing references to time and delays which includes things like : a few years, early years, after tomorrow, etc.

# DTEDate

This service returns numerically encoded dates using a few patterns it can recognise. If a pattern you need isn't currently supported, simply send us a description of the pattern and we will be happy to add it to our collection and ship you a new version of the extractor which includes it.

# DTEPhone

Detects North-American and some European telephone numbers. When possible, it returns extended information about the number such as state, country, and city related to the area code of the number.

# DTEEmail

Returns email patterns in finds in the text. Note that some weird and rarely used patterns may not be detected or are voluntarily ignored because they would collide with other extraction patterns.

# DTEWebsite

Returns urls and references to websites in a text.

# DTESocialTags

Returns often used social media patterns like #hashtags and @reference. These are categorized so you can react on what kind of social tags are in the text.

#### \*\* A more detailed documentation with in-depth reference for each service is being drafted, it should be ready for the next release of the tool. \*\*

Known issues and limitations

#### No piping or stdout

If you try to use piping or try to use the stdout or stderr, you will likely get silence or a seemingly hanging application. You may get the result, but once all the application is done. This app doesn't currently explicitly support the DOS shell. It may work and then stop working from one version to another.

#### Error console window

If a processing error occurs, an error reporting console window may pop up. This is a temporary issue which will be addressed in an ulterior release. These will eventually, optionally, be saved out in a log file.

#### When launching, the full path to the .exe must be used, or must be set as current working directory

When launching the extractor, you must use the full path to the .exe, or set the current working directory of the launched process, explicitly.

You cannot use a PATH environment shortcut, using the application's executable name directly. If you do so, the application will launch and hang with an error.