SemLinker 1.0.0

User Guide

https://code.google.com/p/semlinker/

Eric Charton

Marie-Jean Meurs

Ludovic Jean-Louis

Michel Gagnon

February 2014

Contents

Support note: SemLinker is a powerful but complex software, documented according to normal standards (Java doc, this doc, academic papers), but not easy to use or understand. There are many NLP concepts and a complex evaluation framework to handle for deploying this package. The authors will do their best to answer questions according to their role in the development (see section 4), but no guarantee can be given that it will be enough to help ones to deploy this system.

1 Overview

SemLinker is a Java library designed to experiment various applications of text mining and annotations. Using an external annotator (currently `www.wikimeta.com`), SemLinker can apply various annotation layers and information extraction processes on a text document. SemLinker is divided in two main components:

1. the **Semkit** component
2. the **KBP** component

1.1 Semkit

Semkit is a set of classes and demos dedicated to apply annotation layers on a text document. Those annotation layers can be Part Of Speech tags, Named Entity labels, Semantic Links. Semkit also includes a Natural Language Processing set of classes intended to manage specific linguistic phenomenons like co-references. Semkit produces an annotation object from a text document using an annotation engine. Semkit is used in the KBP experimental component.

1.2 KBP component

This component is an experimental platform compatible with the NIST TAC-KBP[1] entity linking task. The goal of TAC Knowledge Base Population (KBP) is to develop and evaluate technologies for building and populating knowledge bases (KBs) about named entities from unstructured text. KBP systems must either populate an existing reference KB, or else build a KB from scratch.

1.3 Configuration

1.3.1 Software requirements

You need:

- a true Java 7 compliant VM. We recommand the Oracle one: `http://www.oracle.com/technetwork/java/javase/downloads/index.html`
- a version of the NLGbAse metadata (one is in the metadata folder of the SemLinker library): `http://www.nlgbase.org`
- a semantic annotation API key (free for academics): `http://wikimeta.org/product_api.html`

1.3.2 Optional requirements

- You need to install the Redis Base, available for Linux and Windows: `http://redis.io/download`
- You might need various resources to reproduce the KBP task (see KBP section)

[1] `http://www.nist.gov/tac/2013/KBP/`

1.3.3 System requirements

Most of the calculation requirements are handled by the external API. A classical computer with the following properties will be suitable:

- Centrino / AMD FX / Athlon
- 2 cores
- 8 GB RAM
- 500 GB Hard Drive
- Internet access (classical ADSL bW)

Note: the memory size will be the most important part of a SemLinker workstation. Less than 8 GB can lead to swap activation and loss of PC usability.

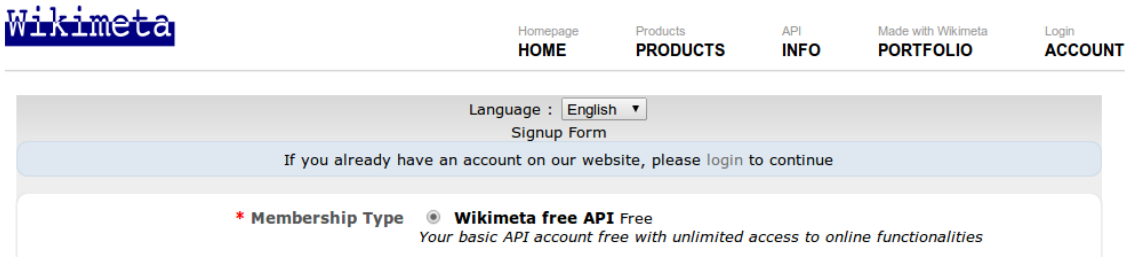
1.3.4 Configuration file

When everything is installed you need to configure *config.cfg* file according to the folders of your system. The *config.cfg* file must be in the same folder as the SemLinker JAR file or in the root folder of SemLinker if you use eclipse to launch it.

For SemLinker global configuration please set:

wmresturi=uri_of_your_API_rest_interface
apiaccount=key_numer_of_your_API_Account

To obtain those information, please open a free API account on Wikimeta:



Then collect the information of the API on your user interface. See next page illustration to view how to collect your API information in the Wikimeta user Interface.

By default, SemLinker is configured with an anonymous Demo account, we cannot guarantee that this account will remain open. Please note that unlimited API accounts will be provided to academic users for experimental purposes. To activate your unlimited API account, just send a mail to contact@wikimeta.com using your academic email address and giving your Wikimeta account reference.

What's new ?

[16 september 2013] Weather is clear no on our calculation grid ! Enjoy semantic annotations.
Tweet

[13 september 2013] It's possible that we limit some free academic unlimited account today and this week-end. Please contact us if you have specific urgent need.
Tweet

[13 september 2013] We add to make some maintenance on our grid this morning and have caused slowdown last night (US time). Speed should be strongly improved now (x10).
Tweet

[10 september 2013] There was a litle mix from google on our Google+ account (not posts). Its solved now by google. Pay us a visit !
Tweet

[4 September 2013] NISKT KBP 2013 campaign end next week. This will help to reduce some slowing encountered on our servers during the past 6 weeks. We plan to double (again) the capacity of our calculation grid along September.
Tweet

[4 September 2013] We found that a few API Free account were not reseted properly (users couldn't get back their 100 calls a day). This is fixed now, sorry for the inconvenience. Please do not hesitate to use support forms and contact us when you encounter such problem.
Tweet

[30 August] Heavy load today on all accounts due to an excess of annotation traffic. The problem is fixed and we apologize for the inconvenience.
Tweet

[6 August] Heavy load currently on our servers. Free accounts may see some delays.
Tweet

We expect heavy load during all summer. Only free accounts may see some delays.
Tweet

[18 July]Heavy load currently on our server due to NIST involved labs works. API Basic account and up have priority, do not hesitate to open a ticket in case of problem.
Tweet

API (?)

Your API key is : 752538502038
Available API server are : http://www.wikimeta.com/wapi/service
Download the sample API at : click to download

API stats (?)

Allowed k words a day : -1 (*1024 words)
Allowed API call a day : 1024
Used calls : 0 Used kw labelling : 0

Metadatas available for download

Download English Metadata.
Download French Metadata.

For KBP module configuration, please check *config.file*, and set:

```
HOME_DIR=/YOUR_PROJECT_HOME_PATH/SemLinker/
INDEX=/YOUR_PATH_TO_LUCENE_INDEX_OF_KBP_CORPORA/index_kbp_folder
INDEX_WIKIPEDIA=/YOUR_PATH_TO_INDEXED_WIKIPEDIA/indexwiki_folder
NLGBASE_PATH=YOUR_PATH_TO_METADATA/metadata/EN.data.csv
```

If you mount SemLinker as an eclipse project, **YOUR_PATH** will be usually to your eclipse workspace. Information on those parameters and the related resources are given in section 3.1.

1.3.5 Loading a Redis Base of NLGbAse metadata

Some applications of SemLinker need to access to a Redis Base (NoSQL database) to collect the NLGbAse metadata.
Follow this process to prepare this Redis Base:

1. Download and install a Redis Server on your system (Redis is free)
2. Launch the server (usually ./redis-server in the folder src after compilation)
3. Verify that the config file section nlgbasepath of SemLinker is pointing on semlinker_folder/metadata/EN.data.csv
4. Load the base with the command `java -cp semlinker.jar Tool`

Watch the base loading. After a long process (many hours) Thats all.
Next time you do not need to reload the Redis Base, but just to start the server.

2 Semkit

There is no specific requirements for usage of the Semkit application for text annotation using the demo classes.

2.1 Demos

The demo classes are intended to present some short sequences of code that demonstrate a possibility of the SemLinker library. Please refer to the JavaDoc for all the specific commands of Semkit demos.

2.1.1 ApiCallSample

Make a call to the API according to the configuration file and return the annotation.

```
java -cp semlinker.jar semkit.demos.ApiCallSample -text text.txt
```

The options are as follows:

```
-h help
-apikey key / define your own API key
-text filename / annotate a text file given in command line
-distrib / display the word distribution
-decodejson / display the JSON decoding
-language {EN|FR} / force a language of annotation
```

2.1.2 BuildHTMLSample demo

A class that demonstrates how to annotate a document, refine it, and then transform it in HTML code for a website.

```
java -cp semlinker.jar semkit.demos.BuildHtmlSample -text text.txt
```

2.1.3 SemanticExtractorSample

Give it a text with name of political people and it returns their affiliation, birthdate and birthplace likethis:

```
java -cp semlinker.jar semkit.demos.SemanticExtractorSample -text text.
   txt
```

The output is as follows:

```
EN: Pierre Moscovici (LOD :http://www.dbpedia.org/resource/
   Pierre_Moscovici  )
-Document returned ok
      ->Political affiliation of Pierre Moscovici is http://dbpedia.org/
         resource/Socialist_Party_(France)
      ->Birthdate of Pierre Moscovici is 1957-09-16"^^<http://www.w3.org
         /2001/XMLSchema#date
```

2.2 How to build and manage an annotation object

The principle of annotation objects allows you to annotate a document and then to apply various processes to the object representing this document.

2.2.1 Create an annotation object

There is a full example of all those processes in the BuildHTMLSample class:

First annotate a document using the API:

```
// Wikimeta Extractor
result = WikiMetaExtractor.getResult(  SemanticConstants.APIAccount ,
   WikiMetaExtractor.Format.XML, toannotate , ln);
```

String result is the resulting annotation in RAW XML of the *String toannotate*.

Build an annotation object:

```
// Wikimeta decoder
WikiMetaXMLDecoder annotations = new WikiMetaXMLDecoder(result, 15);
```

annotations is the annotation object resulting of the *String result* raw XML file.

You can now access this object properties as follows:

- **int annotations.size()** is the number of lines in the object. Each line contain a unique word or symbol (commas, sent separator and so on)
- **String word = annotations.getwordatpos(int x)**: get the word at a x given position.
- **String namedentity = annotations.getENLabel(int x)**: get a named entity tag at given position if exists or null

See the Javadoc for WikimetaXMLDecoder for all the available method (collecting surface form, part of speech tag and so on)

2.2.2 Normalize annotation objects

It is possible to **improve the accuracy of annotation** using the whole document content. This is the normalization process using co-reference and named entity normalization (see the KBP 2013 system paper for more information).

Apply co-reference normalization to an annotation object.

```
 // re-apply coreferences corrections after NE Normalizer
annotations = SimpleCoreferenceDetector.applyCoreferenceCorrection(
   annotations);
```

Apply Named Entity normalization to an annotation object.

```
NormalizeNE NEnorm = new NormalizeNE();

// apply NE normalizer
annotations = NEnorm.rerankNE(annotations);
```

Apply mutual disambiguisation to an annotation object.

```
MutualDisambiguation mdiz = new MutualDisambiguation();

// Disambiguate using document mutual relations
annotations = mdiz.disambig(annotations, content);
```

String Content is the original text document.

3 KBP

KBP is an international evaluation campaign organized by the NIST administration in the context of TAC conferences. KBP can be decomposed into two complementary tracks: entity linking, in which entity mentions must be aligned with entities in the reference Knowledge Base (KB) or new entities discovered in the document collection; and slot filling, which involves finding predefined attributes about target entities in unstructured text. A third major KBP track is Cold Start KBP, which combines entity linking and slot filling to populate an empty KB using the predefined schema from slot filling. We include in the SemLinker toolkit a complete platform to reproduce a KBP entity linking experiment.

3.1 KBP component requirements

The SemLinker KBP component needs a set of resources to work properly. Some of those resources are free to use, others are subject to specific copyright and distribution restrictions (like the evaluation corpora from LDC). The following components are mandatory to build a complete NIST KBP experiment:

1. Lucene Indexed KBP corpora
2. KB correspondence table
3. Lucene indexed Wikipedia XML dump (English)
4. Optional: Lucene Search for Wiki (this can be replaced by a direct access to Wikipedia search engine)

3.1.1 KBP corpora

As explained on the NIST KBP website access to KBP corpora is restricted to KBP task participants:
"Data that are required for the TAC KBP 2013 tracks are distributed at no cost to track participants. Whenever possible, data are distributed by NIST or the Linguistic Data Consortium via Web download; data are mailed as physical disks only if they cannot be made available for download. Access to TAC KBP 2013 data is restricted to registered TAC KBP 2013 participants who have submitted all the required User Agreement forms. Each participating team will provide a TAC KBP 2013 Team ID and will receive a Team Password uponregistration. Teams that have participated in past TAC cycles must register and obtain a new TAC KBP 2013 Team ID and password for 2013."[2]

Once the corpus obtained from LDC as KBP participant or ordered, you can index it for SemLinker with Lucene using the class (**if config file is properly configured**):

```
kbp2013.index.indexSourceCorpus
```

Or with the command line :

```
java -cp semlinker.jar kbp2013.index.indexSourceCorpus
```

3.1.2 Reference KB and correspondence table

The reference KB for 2013 is based on a snapshot of English Wikipedia from October 2008 and is the same reference KB that has been used in TAC KBP since 2009. Each node in the reference KB corresponds to a Wikipedia page for a person (PER), organization (ORG), or geopolitical entity (GPE), and consists of predefined attributes (a.k.a "slots") derived from Wikipedia infoboxes. Unstructured text from the Wikipedia page is also available (as "wiki_text") in the reference KB.

As SemLinker uses external annotations engine to solve the entity linking task, the Wikipedia links provided have to be compliant with the KB nodes numbering system. A correspondence table is generated using NLGbAse (see TAC KBP conference paper for details) with this class :

```
kbp2013.tools.buildKBTable
```

[2]Text from http://www.nist.gov/tac/2013/KBP/data.html

Caution: to use this class, you need to have a Redis Base loaded with the NLGbAse corpus on your computer (see 1.c.iv).

Note: You do not need to regenerate the correspondence table until you want to use the latest NLGbAse version. A pre-calculated sample of the correspondence table is present in folder: *resources/kbp2013/wikimeta_table*

3.1.3 Wikipedia dump indexation

To use the mutual disambiguation capabilities of SemLinker, you need to index a full dump of an English Wikipedia. This is done using the class (**if config file is properly configured**):

```
kbp2013.index.indexWikipediaCorpus
```

Or with the command line:

```
java -cp semlinker.jar kbp2013.index.indexWikipediaCorpus
```

Note: if you do not build a Wikipedia Dump index with Lucene, SemLinker will try to collect the Wikipedia page directly online using the Wikipedia API. This has a strong impact on the speed of the experimental process (over 10 times longer to process all the experiments of KBP2013 test queries). We do not recommend you to use this option.

3.1.4 Lucene-Search for Wiki

Note: installing Lucene-Search for Wiki can be complex and difficult, and involves a lot computing resources. By default, SemLinker includes a hack that access directly to the *"Did You Mean"* page of Wikipedia (see Figure 1) rather than invoke a local copy of Lucene-Search for Wiki.

This solution avoids the need of time consuming configuration, and offers nearly the same results as these obtained with a local copy.

3.2 Experiment

Once the complete system is configured and ready to operate, configure properly the path in the config file to the query file and launch:

```
kbp2013.LinkEntities
```

That's all! You see all the experiments rolling for each query.

The generated file is output in your `TEST_DIR` folder defined in config file and is directly compliant with the KBP evaluation script.

We included this script (el_scorer.py)in the folder:
`resources/kbp2013/entitylinkingeval/scripts`

We also included a modified version of this script (`el_scorer_seg.py`)in the folder to allow calculation of score for each component of the KBP task (for example a specific score per named entity, per sub corpus...).

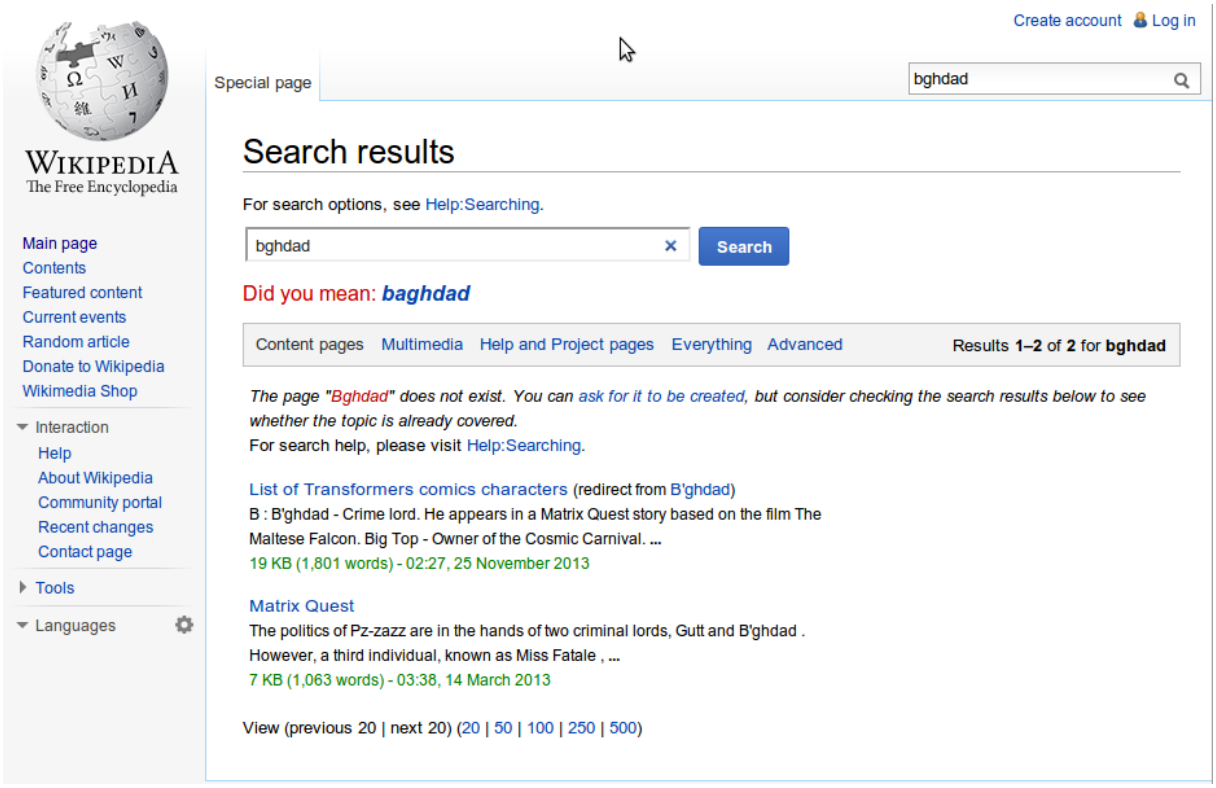


Figure 1: Wikipedia "Did you mean" page

4 Credits

Redaction of this version of the document

Eric Charton, École Polytechnique de Montréal [eric.charton@polymtl.ca]

Contributors on the SemLinker code V1.0.0

Eric Charton, [eric.charton@polymtl.ca]
Main Architect, worked on almost all classes :-)

Marie-Jean Meurs, [marie-jean.meurs@concordia.ca]
Normalization, query reformulation, worked on various classes, cleaning, and refactoring.

Ludovic Jean-Louis, [ludovic.jean-louis@polymtl.ca]
Indexation, worked on various classes and main author of KBP corpus and Wikipedia indexation classes.

Michel Gagnon, [michel.gagnon@polymtl.ca]
Engineering, normalization, and refactoring

5 Complementary informations

The technical description of the SemLinker system presented in KBP 2013 evaluation campaign is in this paper:

Eric Charton, Marie-Jean Meurs, Ludovic Jean-Louis, Michel Gagnon,
SemLinker system for KBP2013: A disambiguation algorithm based on mutual relations of semantic annotations inside a document.
Proceedings of TAC KBP 2013

Javadoc: