Sapienza Ontology-based Search of Genomic Metadata

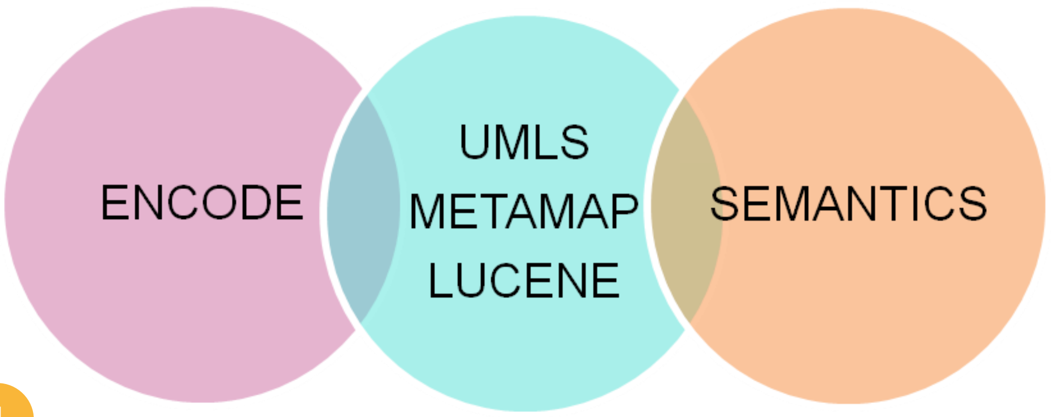
by

Intallation Guide

October 2015

Javier D. Fernández
Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti
Sapienza Università di Roma, Italy

TABLE OF CONTENTS



GENERAL INFORMATION

Sapienza Ontology-based Search of Genomic Metadata (S.O.S. GeM) provides an intuitive Web interface for effectively searching available ENCODE[1] experiments by using natural language or keyword-based queries, browsing the list of detected experiments and their data files whose metadata satisfy the query and downloading the found data files and their complete metadata, with support for direct comprehensive processing of genomic feature data.

S.O.S. GeM enriches the ENCODE database with semantic concepts inferred from solid and curated biological ontologies, creating a Semantic Knowledge Base of ENCODE metadata that enables ontology-based advanced searches.

> *S.O.S. GeM creates and leverages a Semantic Knowledge Base (SKB) of ENCODE metadata.*

In an offline process, S.O.S. GeM first retrieves and indexes available ENCODE metadata. Then, it retrieves the semantic information from very well-known Biological and Biomedical ontologies. In concrete, S.O.S. GeM makes use of MetaMap[2], a tool for recognizing concepts in text. By means of MetaMap, S.O.S. GeM maps each concept recognized in ENCODE metadata to Unified Medical Language System (UMLS[3]) concepts.

S.O.S. GeM materializes all the concepts that can be inferred from the metadata concepts previously recognized. To do so, we make use of the UMLS Metathesaurus, which includes intra-source “distance 1” IS_A hierarchical relationships (immediate parents and children). This means that one concept can be navigated to get its more general concepts or parents (e.g. in the AOD (Alcohol and Other Drug) vocabulary, heart IS_A cardiovascular system, mammal IS_A vertebrate) and its more specific terms or children (e.g. in AOD, primate IS_A mammal). S.O.S. GeM makes use of a forward chaining method and completes the Knowledge Base allowing semantic queries. For instance, an ENCODE experiment with metadata including the concept “primate” is now retrieved when the user searches for “mammal”, given that we have inferred that primate IS_A mammal.

S.O.S. GeM stores and retrieves the number of IS_A steps and the set of ontologies that have been used to reach each fact. For example, the fact “primate IS_A vertebrate” is obtained in 2 steps, using AOD or NCI (National Cancer Institute) vocabularies (since both vocabularies contain the relations primate IS_A mammal and mammal IS_A vertebrate).

[1] https://www.encodeproject.org/
[2] http://metamap.nlm.nih.gov/
[3] http://www.nlm.nih.gov/research/umls/

GENERAL ARCHITECTURE

Figure 1 shows the general architecture and workflow to create the S.O.S. GeM Semantic Knowledge Base of ENCODE metadata.

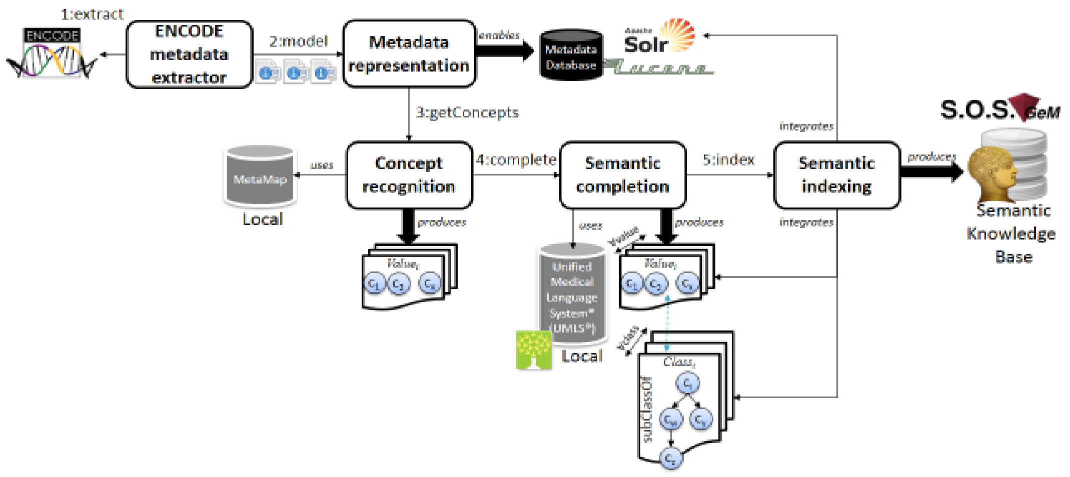


Figure 1. S.O.S. GeM architecture and workflow to create the Semantic Knowledge Base of ENCODE metadata.

We briefly detail the involved process. More information can be found in (Fernández, Lenzerini, Masseroli, Venco, & Ceri, 2015).

ENCODE METADATA EXTRACTION

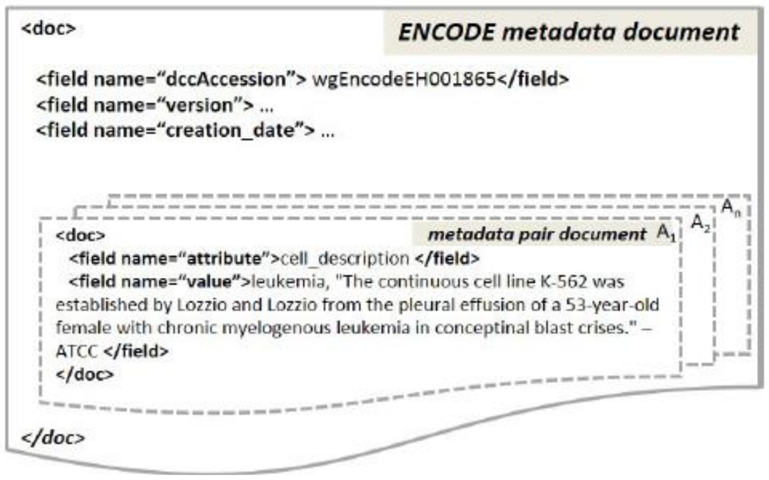
This first step aims at producing the metadata set by extracting the freely-available ENCODE metadata. **This process is out of the scope of this intallation manual**: we assume we run our ENCODE extractor, a Python script that crawls the ENCODE site and collects all metadata for human and mouse ENCODE data. The crawling process is performed sequentially: our metadata extractor explores each provided URL and navigates the subdirectories that group the ENCODE experiments; for each of them, the extractor retrieves a list of files and their metadata, typically named in a files.txt text file. Then, metadata associated with each listed data file in FASTQ, BAM, bigWig, bigBed, BED, broadPeak, narrowPeak or GTF data format are extracted and completed with related metadata from the ENCODE controlled vocabulary[4]. The final metadata are saved in a single tab delimited text file, including all the metadata attribute-value pairs (e.g. ”antibody target CTCF”) for each available ENCODE data file.

[4] http://hgdownload.cse.ucsc.edu/goldenPath/encodeDCC/cv.ra

METADATA REPRESENTATION

As S.O.S. GeM proposes an incremental Knowledge Base creation, the first objective is to allow a syntactic search functionality of the extracted ENCODE metadata. Intuitively, S.O.S. GeM builds a database hosting the extracted metadata, associating each experiment Sample data file with its set of extracted metadata attributes and values.

To do so, we make use of the well-known Apache Lucene/Solr 5.0 framework[5], which proves to be a feasible and scalable solution to tokenize and index text documents. Figure 2 shows a brief example of an ENCODE metadata document indexed by Lucene. More information on the scheme of the documents can be found in (Fernández, Lenzerini, Masseroli, Venco, & Ceri, 2015).


ENCODE metadata document
metadata pair document
A_1
A_2
A_n

```
<doc>
  <field name="dccAccession"> wgEncodeEH001865</field>
  <field name="version"> ...
  <field name="creation_date"> ...

  <doc>
    <field name="attribute">cell_description </field>
    <field name="value">leukemia, "The continuous cell line K-562 was
  established by Lozzio and Lozzio from the pleural effusion of a 53-year-old
  female with chronic myelogenous leukemia in conceptinal blast crises." –
  ATCC </field>
  </doc>
</doc>
```

Figure 2. A first ENCODE metadata document.

See LUCENE Section for a step by step installation guide.

CONCEPT RECOGNITION

This step focuses on inspecting ENCODE metadata values and recognizing concepts that they may include from the biomedical ontologies in the Unified Medical Language System.

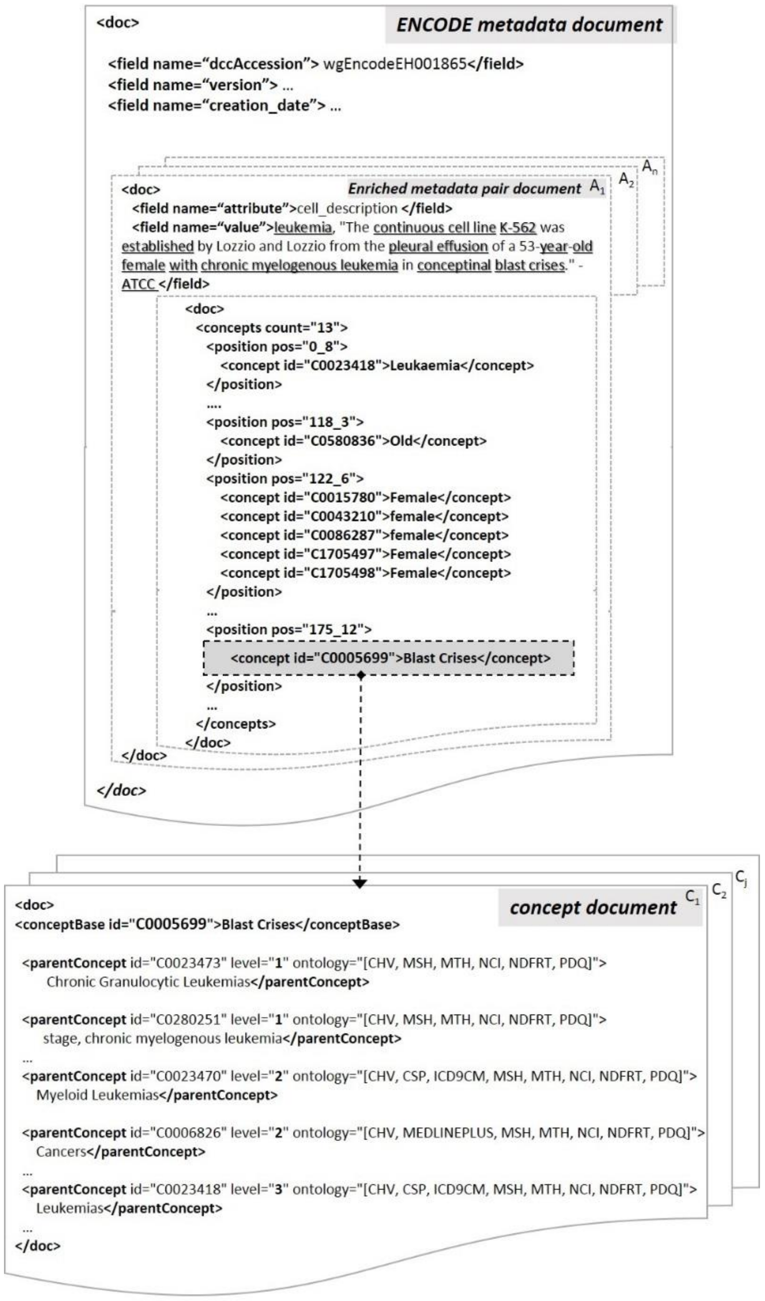
S.O.S. GeM performs this task on the basis of MetaMap (Aronson & Lang, 2010), a tool specifically designed to discover concepts in the UMLS Metathesaurus that are referred to in a given text. MetaMap relies on a knowledge intensive approach based on symbolic, natural language processing and computational linguistic techniques; its efficiency has been largely shown for this particular task. S.O.S. GeM manages a local installation of MetaMap (version 2014) to boost the performance, by processing all the information requests through the provided

[5] http://lucene.apache.org/solr/

MetaMap Java API (release 2014), and obtain the set of UMLS concepts found in each metadata value.**MetaMap local installation is detailed in the**

metamap local **Section.**

The recognized concepts per value are then stored in an enriched metadata pair document in the lucene system, as shown in Figure 3.



```
<doc>
  <field name="dccAccession"> wgEncodeEH001865</field>
  <field name="version"> ...
  <field name="creation_date"> ...

  <doc>
    <field name="attribute">cell_description </field>
    <field name="value">leukemia, "The continuous cell line K-562 was
  established by Lozzio and Lozzio from the pleural effusion of a 53-year-old
  female with chronic myelogenous leukemia in conceptinal blast crises." -
  ATCC </field>
      <doc>
        <concepts count="13">
          <position pos="0_8">
            <concept id="C0023418">Leukaemia</concept>
          </position>
          ....
          <position pos="118_3">
            <concept id="C0580836">Old</concept>
          </position>
          <position pos="122_6">
            <concept id="C0015780">Female</concept>
            <concept id="C0043210">female</concept>
            <concept id="C0086287">female</concept>
            <concept id="C1705497">Female</concept>
            <concept id="C1705498">Female</concept>
          </position>
          ...
          <position pos="175_12">
              <concept id="C0005699">Blast Crises</concept>
          </position>
          ...
        </concepts>
      </doc>
  </doc>

</doc>
```

```
<doc>
<conceptBase id="C0005699">Blast Crises</conceptBase>

 <parentConcept id="C0023473" level="1" ontology="[CHV, MSH, MTH, NCI, NDFRT, PDQ]">
     Chronic Granulocytic Leukemias</parentConcept>

 <parentConcept id="C0280251" level="1" ontology="[CHV, MSH, MTH, NCI, NDFRT, PDQ]">
     stage, chronic myelogenous leukemia</parentConcept>
 ...
 <parentConcept id="C0023470" level="2" ontology="[CHV, CSP, ICD9CM, MSH, MTH, NCI, NDFRT, PDQ]">
   Myeloid Leukemias</parentConcept>

 <parentConcept id="C0006826" level="2" ontology="[CHV, MEDLINEPLUS, MSH, MTH, NCI, NDFRT, PDQ]">
   Cancers</parentConcept>
 ...
 <parentConcept id="C0023418" level="3" ontology="[CHV, CSP, ICD9CM, MSH, MTH, NCI, NDFRT, PDQ]">
   Leukemias</parentConcept>
 ...
</doc>
```

Figure 3. Enriched ENCODE metadata and concept documents.

SEMANTIC COMPLETION COMPUTATION

After the phase of concept recognition, our SKB computes the semantic completion of the various concepts, by taking care of the class assertions SubsetOf. For this purpose, S.O.S. GeM makes use of the UMLS Metathesaurus intra-source “distance 1” IS_A hierarchical relationships (immediate parents and children). For instance, in the AOD vocabulary, heart IS_A cardiovascular system, and mammal IS_A vertebrate, which formally state SubsetOf (heart cardiovascularsystem), and SubsetOf (mammal vertebrate).

In practice, S.O.S. GeM makes use of forward chaining to materialize all the UMLS concepts that can be inferred from the recognized metadata concepts in the enriched metadata pair documents. Thus, for every single recognized concept, we retrieve all its atoms and, for every atom, we navigate its hierarchies by iteratively retrieving the SubsetOf relationships up to the top level concept of the specific UMLS ontology. Note that, at every step, each retrieved atom Aj is again inspected, i.e. its associated UMLS concepts are retrieved and the process is repeated. To avoid processing each concept more than once, we maintain an ad-hoc data structure and, to make the computation efficient, we use a local installation of UMLS (ver. 2014AA on a MySQL database) to access both concept/atom data and hierarchies. **UMLS local installation is detailed in the UMLS LOCAL Section.**

We store the semantic closure also by means of the LUCENE system, as shown in Figure 3.

SEMANTIC INDEXING

Thanks to established Apache LUCENE document schema, this process is straightforward: in practice, S.O.S. GeM loads and indexes the aforementioned documents from previous processes in our Lucene/Solr backend system which, all together, constitutes the S.O.S. GeM SKB.

INSTALLATION

In the following, we detail the installation of the Lucene, UMLS local* and MetaMap local systems.

We distribute all the necessary content in a **sosgem.tgz** file that comprises the following directories and files:

- **lucene_sosgem** -- *directory with all the information to install LUCENCE*
    - run.sh -- *script to run lucene*
    - example-schemaless -- *subfolder with all the S.O.S. GeM configuration*
    - *start.jar, ...* -- *other lucene installation files and folders*
- **umls*** -- *directory with the 2014AA UMLS installation files*
    - mmsys.zip -- *main installation file*
    - 2014AA.CHK, … -- *other umls installation files and folders*
- **metamap** -- *directory with the installation files and run scripts*
    - public_mm_linux_main_2014.tar.gz -- *main installation file*
    - public_mm_linux_javaapi_november_2014.tar.gz --*API installation file*
    - run_server.sh -- *script to run the metamap server*
    - stop_server.sh -- *script to stop the metamap server*
- **webapp** **--** *directory with the Web UI war file and source code*
    - ontologyBroker-restful-webapp.war --*war to deploy the web*
    - ontologyBroker-restful-webapp -- *source code of the web*
- **ontologyBroker** -- *directory with the source code of the system*
- **tools** -- *directory with the tools to import .samplings*

* Note UMLS installation is just required to perform the semantic closure.

LUCENE

1) Install Lucene

a) Installation of a fresh version of Lucene from scratch

To install Apache Lucene from scratch, the simplest way is to download and unzip the binary files. We use the 4.9 version: http://apache.fastbull.org/lucene/solr/4.9.0/

b) Installation from the S.O.S. GeM source code (**Recommended**)

S.O.S. GeM makes use of a particular Lucene configuration called schemaless[6], which allows to define new fields on demand. Thus, we distribute the Lucene 4.9 binary code together with a minor change in the configuration to allow this option. To install it, just copy the **lucene_sosgem** subfolder.

2) Apply the S.O.S. GeM configuration

As stated, S.O.S. GeM makes use of a particular Lucene configuration called schemaless. If you have performed the installation from the S.O.S GeM source code, i.e. the previous 1)b) point, this configuration is already included in the code. Otherwise, please copy (or replace) the **lucene_sosgem/example-schemaless** folder to your Lucene installation folder and make sure this subfolder is accessible.

3) Run the system.

S.O.S. GeM includes a run.sh script to lunch Lucene, which calls the following command:

```
$> java -Dsolr.solr.home=example-schemaless/solr -jar start.jar &
```

Note that the system is running in the default port 8983, which is usually closed externally. This is not a problem once the applications run locally. In any case, to see the Solr web interface one could temporary change the port to :80 in solr/etc/jetty.xml (possibly in this latter case Solr has to be run with root permission). If this is required often, one can install a proxy: e.g. see http://lucene.472066.n3.nabble.com/How-to-work-with-remote-solr-savely-td4102612.html

To stop the server, search for the Solr process (e.g. by: ps -xa | grep solr) and kill it (kill -9 PID).

A simple Lucene tutorial is also available in the official web page[7].

UMLS LOCAL

Note that we strictly follow the procedure in the official documentation[8]. Note also that this installation is just required to perform the semantic closure.

[6] https://cwiki.apache.org/confluence/display/solr/Schemaless+Mode
[7] http://lucene.apache.org/solr/4_9_0/tutorial.html

1) (If needed in a remote server) Install X.
   - See X documentation, for instance: http://forums.debian.net/viewtopic.php?t=12256

2) Download UMLS. We currently use the 2014AA UMLS Active Release Files[9]. We include a local copy of the required files in the **umls** folder, given that the need a UTS user to download these files.

3) After downloaded, unzip mmsys.zip and run (in ssh -X connection)
   - ./run_linux.sh
   - If run_linux fails, you may have to erase a temp file, rm /home/lab/.mmsys/mmsys.prop
   - In the installation, we selected ALL level 0 vocabularies+snomed us (this means all which no requires additional licenses).

4) Install mysql server 5.5 and import the scripts. We follow the official documentation[10]
   - apt-get install mysql-server-5.5
   - Tune the mysql parameters, as pointed in the documentation, in the config file (can be found with: mysql --help | grep "Default options" -A 1)
   - Restart the server (/etc/init.d/mysql restart).
   - Verify that the parameters are well set by: mysqld --verbose –help
   - Set required variables in the populate_mysql_db.sh script and run it.

Known problems:

1) We had one problem ("ERROR 1148 (42000) at line 15: The used command is not allowed with this MySQL version"), but it was solved removing the word "local" from mysql_tables.sql (as referred here: http://dev.mysql.com/doc/refman/5.0/en/loading-tables.html)
2) We had also to include the full path of the files in mysql_tables.sql

METAMAP LOCAL

It consists of two parts: the server (MetaMap Full Download[11]) and the Java API[12] to access the server.

[8] http://www.ncbi.nlm.nih.gov/books/NBK9683/ and http://www.nlm.nih.gov/research/umls/new_users/online_learning/UMLST_004.html
[9] http://www.nlm.nih.gov/research/umls/licensedcontent/umlsknowledgesources.html
[10] http://www.nlm.nih.gov/research/umls/implementation_resources/scripts/README_RRF_MySQL_Output_Stream.html
[11] http://metamap.nlm.nih.gov/MainDownload.shtml
[12] http://metamap.nlm.nih.gov/JavaApi.shtml

1. First, download and install the MetaMap Full Download (~2.3 GB in bzip2). We include a local copy of the files in the **metamap** folder. An installation guide can be found at http://metamap.nlm.nih.gov/Installation.shtml, which basically consists of:
   1.1. Uncompress the file (bunzip2)
   1.2. (If JAVA_HOME is not set) Set the environment variable JAVA_HOME, e.g. in linux:
      - export JAVA_HOME=/usr/local/jre1.6

   1.3. Run the installation script: ./bin/install.sh
      - In the required information, press enter to install in the same directory, and then give the directory provided with the command **which java**
      - (Only If the installation script is not working) add the <parent dir>/public_mm/bin directory to your program path, e.g. in linux: export PATH=<parent dir>/public_mm/bin:$PATH
2. Then, download the JavaAPI and extract in the same previous directory and execute again the same ./bin/install.sh. We include a local copy of the files in the **metamap** folder
3. Run MetaMap server
   3.1. Probably, to run the mmserver in the server machine one has to install the gcc multilib:
      - apt-get install gcc-multilib

   3.2. Run our script: run_server.sh
   3.3. Run the server
      - cd public_mm
      - ./bin/mmserver14 &

To stop the server, just run our script ./stop_server.sh

S.O.S. GEM WEBPAGE

S.O.S. GeM is a WAR application; so it can be deployed in any Java Web Server such as Tomcat. We currently use Tomcat 7. In our case, the steps are the following:

1) (If something has changed) Compile the source code into a war
   i) project OntologyBroker: mvn install (or in Eclipse, Run as/Maven install) [this is required in order to create the ontologyBroker-0.0.1-SNAPSHOT-jar-with-dependencies.jar]
   - project ontologyBroker-restful-webapp:
     i) mvn clean (or in Eclipse, Run as/Maven clean) [this is required in order to update the ontologybroker jar]
     ii) In eclipse, Project/Clean
     iii) mvn install (or in Eclipse, Run as/Maven install).
2) Copy sosgem.war to /var/lib/tomcat/webapp

3) if needed, restart the tomcat server in /etc/init.d/tomcat restart

MAINTENANCE

In the following, we detail the process of creating Lucene documents from ENCODE samples, recognizing the concepts, expanding them and loading all into Lucene. The **tools** directory includes the required scripts.

Note that the main software package is provided, ontologyBroker-0.0.1-SNAPSHOT-jar-with-dependencies.jar. If the source code has been updated, one should create again this jar package. To do so, in project OntologyBroker: mvn install (or in Eclipse, Run as/Maven install) [this is required in order to create the]

LOAD THE ENCODE SAMPLES INTO LUCENE

This is the first step to be done once all the system has been installed.

1) Inside tools directory, run the script loadAllSamplings.sh

CREATE DOCUMENTS FROM ENCODE SAMPLES

This process is just required when new samplings are added.

FORMAT OF THE SAMPLES

By default, we expect files named "datasetName.samplings" where "datasetName" is the name of the dataset. For instance, typical names are HG19_ENCODE_BROAD.samplings, MM9_ENCODE_BAM.samplings, etc.

Each dataset contains one or more samples. The format of the file is:

<numerOfSample> \t property \t value

For instance:

```
1    dataVersion    ENCODE Jul 2012 Freeze
1    readType    2x99D
1    readType_tag    R2X99D
(…)
2    control_tag    STD
2    control_type    control
2    control_description    Standard input signal for most experiments.
(…)
```

Note that it is often the case that we download one file per sample. In order to merge all the samples of a dataset into a datasetName.samplings, place all the samples in a directory, e.g "HG19_ENCODE_BROAD" and execute the script:

- perl tools/concatenate.pl HG19_ENCODE_BROAD

This will create the file HG19_ENCODE_BROAD.samplings

CREATE AND LOAD DOCUMENTS

In the following we describe the subsequent steps to create and load documents.

1) Check that there is a ".mmsys" directory in the home folder of the user that is running the script. Otherwise, create a .mmsys directory. For instance, for user lab:
   (1) mkdir /home/lab/.mmsys

2) Place all the *.sampling files in a folder, e.g. tools/metadata/allFiles/luceneConversion

3) Run the script (in tools folder): "./createLuceneSamplings.sh metadata/allFiles/luceneConversion"
   - This creates a .xml file per each *.sampling file, in which concepts are recognized
   - The file AllConcepts.txt lists all the recognized concepts. As this could include repetitions from different files, do a "sort AllConcepts.txt | uniq > AllConceptsUniq.txt"

   Note:
   - The createLuceneSamplings.sh script considers "encode" samplings by default. To create samplings from other collections, provide the name of the collection as the second argument of the script. For instance, to index a TCGA collection:
     i) ./createLuceneSamplings.sh metadata/tcga/luceneConversion TCGA

4) Create the expansion of terms, running: "./createExpansion.sh AllConceptsUniq.txt". This creates the AllConceptsUniq_expansion.xml.
   Note:
   - The createExpansion.sh script considers "encode" samplings by default. To create expansion from other collections, provide the name of the collection as the second argument of the script. For instance, to index a TCGA collection:
     i) ./ createExpansion.sh AllConceptsUniq.tx TCGA

5) Load all into Lucene, with: " ./loadLucene.sh tools/metadata/allFiles/basicExpansion" and " ./loadLucene.sh tools/metadata/allFiles/concepts"

DELETE ALL INFORMATION IN LUCENE

1) Run the script ./delete_lucene.sh and confirm the action ('Y')

REFERENCES

Aronson, A. R., & Lang, F. M. (2010). An overview of MetaMap: Historical perspective and recent advances. *J. Am. Med. Inform.*, *17*(3), 229-263.

Fernández, J. D., Lenzerini, M., Masseroli, M., Venco, F., & Ceri, S. (2015). Ontology-Based Search of Genomic Metadata. *IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS*, Under review.