README for SemRep Processing

Dongwook Shin, Ph.D. National Library of Medicine  
Date: August 15 2019

Java package: gov.nih.nlm.skr.semrepProcess

**Data format:**

Input: XML format compliant with MEDLINE BASELINE converts XML files into the corresponding ASCII format (a separate directory) and generating input ASCII file for semrep processing. The input file format for Semrep is:

First line: PMID- pmid\_number

Second line: TI - title

Third line: AB - abstract

EXAMPLE:

PMID- 30

TI - Lysosomal hydrolases of the epidermis. I. Glycosidases.

AB - Seven distinct glycosidases (EC 3.2) have been characterized in guinea-pig epidermis. Their properties indicate them to be of lysosomal origin. The 'profile' of

the epidermal glycosidases is significantly different from that reported for whole skin, the activities of beta-galactosidase and beta-acetylglucosaminidase being very

high and those of the remaining enzymes relatively low in epidermis.

If the abstract of a citation is empty, then the third line starting with “AB -“ will not exist.

**Process**

1. Create two databases; one for PREPROCESS and the other for SemMedDatabase
   1. The database schema for PREPROCESS database is:

CREATE TABLE FACT\_DATA (

`FACT\_DATA\_ID` int(10) unsigned NOT NULL auto\_increment,

PMID varchar(20) NOT NULL,

EXIST\_XML tinyint(1) default 0,

EXIST\_SEMREP tinyint(1) default 0,

XML\_DATA MEDIUMTEXT,

SEMREP\_DATA MEDIUMTEXT,

TITLE varchar(999),

ABSTRACT MEDIUMTEXT,

PRIMARY KEY (`FACT\_DATA\_ID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 ;

create index pmid\_index\_btree using btree on FACT\_DATA (PMID);

* 1. The database schema for SemMedDatabase is described in appendix A.

This database information can be configured in ROOT/semrep.properties file as follows:

connectionString=jdbc:mysql://domain name of the machine

preddatabase=PREPROCESS

semmeddatabase=semmedVER50

dbusername=user name for the database

dbpassword=password for the database

Connection string is the connection string for the MySQL database.

Please note that the user needs to have the read/write privilege for the database.

* 1. Insert generic terms from one of the existing SemMed database (for example semmedVER40) using the following SQL statement

Insert into GENERIC\_CONCEPT select \* from semmedVER40.GENERIC\_CONCEPT

\* This SQL statement will copy 259 generic terms (as of September 2019) from GENERIC\_CONCEPT table of semmedVER40 database into GENERIC\_CONCEPT table.

\* Terms in GENERIC\_CONCEPT are used to determine the novelty column of PREDICATION table. That is, if a CUI (either subject or object) does not appear in GENERIC\_CONCEPT, it is determined as novel and its novelty value is 1. Otherwise, it is not novel and its value is 0.

1. Check if there is a jar file ./lib/semrepProcess.jar. If not, run the following command and generate it.
   1. % ant main –f build\_semrepProcess.xml

If a Java program in ./src directory is modified for some reason, make sure to run the “ant” command and have the latest semrepProcess.jar file.

1. Set the parameters appropriately in the file ROOT/semrep.properties. The parameters relevant to Semrep processing in addition to those described in 1.b are as follows:

semrepLoadingProgram=resources/semreppingEntity\_40WithSH\_Novelty.pl

perlScript=/usr/bin/perl

schedulerAccount=yourUMLSAccount

schedulerPassword=yourUMLSPassword

schedulerEmail=yourEmailaddress

semrepLoadingProgram is the Perl script that is used in loading SemRep output into the database.

schedulerAccout, schedulerPassword and schedulerEmail are the account information that is used to login to UMLS.

1. Run gov.nih.nlm.skr.semrepProcess.Semrepping if you want to process semrepping from scratch. For example, if there is no database and you want to process semrepping for all the PubMed citations, run this step. This class takes (1) the MEDLINE\_BASELINE directory as input, (2) converting XML files into corresponding ASCII format, (3) generating input ASCII file for Semrep processing and (4) putting Semrep result. The shell script ‘*semrepping.sh*” does this step automatically. The arguments need to be:
   1. First argument – MEDLINE\_BASELINE directory
   2. Second argument – ASCII conversion directory from MEDLINE\_BASELINE
   3. Third argument – directory for Medline format
   4. Forth argument – directory for Semrep output

EXAMPLE:

Java gov.nih.nlm.skr.semrepProcess.Semrepping /net/lhcdevfiler/ /MEDLINE\_Baseline\_Repository/2019

./XML\_ASCII . /MEDLINE\_format ./SemrepOutput

In this example, the first argument is the MEDLINE\_BASELINE directory, the second argument, ./XML\_ASCII is the ASCII converted directory of the original XML directory, the third argument, . /MEDLINE\_format is the directory for Medline format and the fourth, ./SemrepOutput is the directory where SemRep output is stored.

* In this process, PMID, title and abstract are inserted into the FACT\_DATA table for the corresponding row. This is because if Semrep step fails for some citations, the MEDLINE format are recreated from these tables and resubmitted in step 6.
* The reason that this class needs ASCII converted XML directory is that it extracts metadata information from XML structure and inserts it into CITATIONS table in the SemMed database before any sentence and predication is inserted.
* From SemMedDB version 4.0, SENTENCE table is dependent on CITATIONS table and unless the metadata info for a PMID is created in CITATIONS table, the sentences are not inserted into SENTENCE table.

1. Run gov.nih.nlm.skr.semrepProcess.SemreppingAddition if you want to add new citations to the current database and update it. For example, if there is current SemMed database and you want to process citations that are added recently, run this step. This class takes (1) the MEDLINE\_BASELINE directory as input, (2) converting XML files into corresponding ASCII format, (3) generating input ASCII file for Semrep processing and (4) putting Semrep result. The shell script ‘*semreppingAddition.sh*” does this step automatically. The arguments need to be:
   1. First argument – MEDLINE\_BASELINE directory
   2. Second argument – ASCII conversion directory from MEDLINE\_BASELINE
   3. Third argument – directory for Medline format
   4. Forth argument – directory for Semrep output

EXAMPLE:

Java gov.nih.nlm.skr.semrepProcess.Semrepping /net/lhcdevfiler/ /MEDLINE\_Baseline\_Repository/2019

./XML\_ASCII . /MEDLINE\_format ./SemrepOutput

This process is the same as step 4 except that the only citations that are not in the SemMed database are processed. In fact, the entire citations in the MEDLINE\_BASELINE directory is translated into the directory specified in the second argument (./XML\_ASCII). However, only citations that are not in the SemMed database are converted into Medline format (whose directory is specified in the third argument). Those citations are being semrepped further and loaded into the SemMed database.

1. gov.nih.nlm.skr.semrepProcess.semrepingFromDB. This class is looking at the FACT\_DATA table of PREPROCESS database and finds the PMIDs whose EXIST\_SEMREP is 0 instead of 1. If EXIST\_SEMREP is 0, it means that the citation has not been semrepped yet (either because it was not semrepped at all or semrepping has failed) and needs to be semrepped in this step. SemreppingFromDB class is scanning from FACT\_DATA table, extracting *limit* PMIDs starting from *offset*, generating input MEDLINE format, and sending those to the Scheduler. The output SemRep result is stored in the temp directory specified in the third argument (see below). The shell script “*SemreppingFromDB.sh*” does this task automatically. This class takes three input parameters as follows:
   1. First argument is the integer specifying the *limit*, which is a number of each batch (number of MEDLINE citations that are to be taken to the Scheduler)
   2. Second argument is the starting offset from which the PMIDs whose EXIST\_SEMREP is 0 are extracted.
   3. Third argument is the temp directories where SemRep output is stored and loaded into the SemMedDB database.

* As SemRepping of some citations can fail in the Scheduler for various reasons (including timeout or Scheduler failure), gov.nih.nlm.skr.semrepProcess.SemrepingFromDB class needs to be executed repeatedly until the remaining citations cannot be semrepped any further.

EXAMPLE:

java gov.nih.nlm.skr.semrepProcess.SemreppingFromDB 40000 3000 ./Sample/DB

**Appendix A. Database schema for SemMed database**

-- table CITATIONS

CREATE TABLE METAINFO (

DBVERSION varchar(10) NOT NULL,

SEMREPVERSION varchar(10),

PUBMED\_TODATE varchar(10),

COMMENT varchar(500)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE CITATIONS (

PMID varchar(20) NOT NULL,

ISSN varchar(10),

DP varchar(50),

EDAT varchar(50),

PYEAR int(5),

PRIMARY KEY (`PMID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 ;

CREATE TABLE `GENERIC\_CONCEPT` (

`CONCEPT\_ID` int(10) unsigned NOT NULL auto\_increment,

`CUI` varchar(20) NOT NULL default '',

`PREFERRED\_NAME` varchar(200) character set utf8 NOT NULL default '',

PRIMARY KEY (`CONCEPT\_ID`),

KEY `PREFERRED\_NAME` (`PREFERRED\_NAME`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Stores basic UMLS concept information';

-- table SENTENCE

CREATE TABLE `SENTENCE` (

`SENTENCE\_ID` int(10) unsigned NOT NULL auto\_increment,

`PMID` varchar(20) NOT NULL default '',

`TYPE` varchar(2) NOT NULL default '',

`NUMBER` int(10) unsigned NOT NULL default '0',

`SENT\_START\_INDEX` int(10) unsigned NOT NULL default '0',

`SENT\_END\_INDEX` int(10) unsigned NOT NULL default '0',

`SECTION\_HEADER` varchar(100),

`NORMALIZED\_SECTION\_HEADER` varchar(50),

`SENTENCE` varchar(999) character set utf8 NOT NULL default '',

PRIMARY KEY (`SENTENCE\_ID`),

FOREIGN KEY (`PMID`) REFERENCES `CITATIONS` (`PMID`) ON DELETE CASCADE ON UPDATE CASCADE,

UNIQUE KEY `SENTENCE` (`PMID`,`TYPE`,`NUMBER`),

KEY `PMID\_INDEX` USING BTREE (`PMID`),

KEY `PMID\_HASH` USING HASH (`PMID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Stores sentences from Medline';

CREATE TABLE `PREDICATION` (

PREDICATION\_ID int(10) unsigned NOT NULL auto\_increment,

SENTENCE\_ID int(10) unsigned NOT NULL ,

PMID varchar(20),

PREDICATE varchar(50),

SUBJECT\_CUI varchar(255),

SUBJECT\_NAME varchar(999),

SUBJECT\_SEMTYPE varchar(50),

SUBJECT\_NOVELTY tinyint(1),

OBJECT\_CUI varchar(255),

OBJECT\_NAME varchar(999),

OBJECT\_SEMTYPE varchar(50),

OBJECT\_NOVELTY tinyint(1),

FACT\_VALUE char(20),

MOD\_SCALE char(20),

MOD\_VALUE float(3,2),

PRIMARY KEY (PREDICATION\_ID),

-- Commented on March 23 2010, Dongwook since InnoDB cannot be created because of Foreign key constraint

-- In "Creating InnoDB Tables" and the title "FOREIGN KEY Constraints"...

-- Foreign keys definitions are subject to the following conditions:

-- Both tables must be InnoDB type.

-- In the referencing table, there must be an index where the foreign key columns are listed as the first columns in the same order.

-- In the referenced table, there must be an index where the referenced columns are listed as the first columns in the same order.

-- Index prefixes on foreign key columns are not supported. One consequence of this is that BLOB and TEXT columns cannot be included in a foreign key, because indexes on those columns must always include a prefix length.

FOREIGN KEY (SENTENCE\_ID) REFERENCES SENTENCE(SENTENCE\_ID) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (PMID) REFERENCES SENTENCE(PMID) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Stores aggregate info of semantic predications';

create index pmid\_index\_btree using btree on PREDICATION (PMID);

-- table PREDICATION\_AUX

CREATE TABLE `PREDICATION\_AUX` (

`PREDICATION\_AUX\_ID` int(10) unsigned NOT NULL auto\_increment,

`PREDICATION\_ID` int(10) unsigned NOT NULL,

`SUBJECT\_TEXT` varchar(200) default '' COMMENT 'Should be NOT NULL eventually',

`SUBJECT\_DIST` int(10) unsigned default '0' COMMENT 'Should be NOT NULL eventually',

`SUBJECT\_MAXDIST` int(10) unsigned default '0' COMMENT 'Should be NOT NULL eventually',

`SUBJECT\_START\_INDEX` int(10) unsigned default '0' COMMENT 'Should be NOT NULL eventually',

`SUBJECT\_END\_INDEX` int(10) unsigned default '0' COMMENT 'Should be NOT NULL eventually',

`SUBJECT\_SCORE` int(10) unsigned default '0' COMMENT 'Should be NOT NULL eventually',

`INDICATOR\_TYPE` varchar(10) default '' COMMENT 'Should be NOT NULL eventually',

`PREDICATE\_START\_INDEX` int(10) unsigned default '0' COMMENT 'Should be NOT NULL eventually',

`PREDICATE\_END\_INDEX` int(10) unsigned default '0' COMMENT 'Should be NOT NULL eventually',

`OBJECT\_TEXT` varchar(200) default '' COMMENT 'Should be NOT NULL eventually',

`OBJECT\_DIST` int(10) unsigned default '0' COMMENT 'Should be NOT NULL eventually',

`OBJECT\_MAXDIST` int(10) unsigned default '0' COMMENT 'Should be NOT NULL eventually',

`OBJECT\_START\_INDEX` int(10) unsigned default '0' COMMENT 'Should be NOT NULL eventually',

`OBJECT\_END\_INDEX` int(10) unsigned default '0' COMMENT 'Should be NOT NULL eventually',

`OBJECT\_SCORE` int(10) unsigned default '0' COMMENT 'Should be NOT NULL eventually',

`CURR\_TIMESTAMP` timestamp NOT NULL default CURRENT\_TIMESTAMP on update CURRENT\_TIMESTAMP,

PRIMARY KEY (`PREDICATION\_AUX\_ID`),

FOREIGN KEY (`PREDICATION\_ID`) REFERENCES `PREDICATION` (`PREDICATION\_ID`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Stores semantic predications in sentences';

-- table COREFERENCE

CREATE TABLE `COREFERENCE` (

`COREFERENCE\_ID` int(10) unsigned NOT NULL auto\_increment,

`PMID` varchar(20) NOT NULL default '',

`ANA\_CUI` varchar(255),

`ANA\_NAME` varchar(999),

`ANA\_SEMTYPE` varchar(50),

`ANA\_TEXT` varchar(200) default '',

`ANA\_SENTENCE\_ID` int(10) unsigned NOT NULL,

`ANA\_START\_INDEX` int(10) unsigned default '0' ,

`ANA\_END\_INDEX` int(10) unsigned default '0' ,

`ANA\_SCORE` int(10) unsigned default '0' ,

`ANT\_CUI` varchar(255),

`ANT\_NAME` varchar(999),

`ANT\_SEMTYPE` varchar(50),

`ANT\_TEXT` varchar(200) default '',

`ANT\_SENTENCE\_ID` int(10) unsigned NOT NULL,

`ANT\_START\_INDEX` int(10) unsigned default '0' ,

`ANT\_END\_INDEX` int(10) unsigned default '0' ,

`ANT\_SCORE` int(10) unsigned default '0' ,

`CURR\_TIMESTAMP` timestamp NOT NULL default CURRENT\_TIMESTAMP on update CURRENT\_TIMESTAMP,

PRIMARY KEY (`COREFERENCE\_ID`),

UNIQUE KEY (`PMID`,`ANA\_START\_INDEX`,`ANA\_END\_INDEX`, `ANT\_START\_INDEX`,`ANT\_END\_INDEX`),

CONSTRAINT `PMID\_ibfk\_1` FOREIGN KEY (`PMID`) REFERENCES `SENTENCE` (`PMID`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `ANA\_SENTENCE\_ID\_ibfk\_4` FOREIGN KEY (`ANA\_SENTENCE\_ID`) REFERENCES `SENTENCE` (`SENTENCE\_ID`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `ANT\_SENTENCE\_ID\_ibfk\_5` FOREIGN KEY (`ANT\_SENTENCE\_ID`) REFERENCES `SENTENCE` (`SENTENCE\_ID`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Stores coreference';

-- table ENTITY

CREATE TABLE `ENTITY` (

`ENTITY\_ID` int(10) unsigned NOT NULL auto\_increment,

`SENTENCE\_ID` int(10) unsigned NOT NULL,

`PMID` varchar(20) NOT NULL default '',

`CUI` varchar(255),

`NAME` varchar(999),

`SEMTYPE` varchar(50),

`GENE\_ID` varchar(999) NOT NULL default '',

`GENE\_NAME` varchar(999) NOT NULL default '',

`TEXT` varchar(999) default '',

`SCORE` int(10) unsigned default '0' COMMENT 'Should be NOT NULL eventually',

`START\_INDEX` int(10) unsigned default '0' COMMENT 'Should be NOT NULL eventually',

`END\_INDEX` int(10) unsigned default '0' COMMENT 'Should be NOT NULL eventually',

PRIMARY KEY (`ENTITY\_ID`),

FOREIGN KEY (`SENTENCE\_ID`) REFERENCES `SENTENCE` (`SENTENCE\_ID`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Stores semantic predications in sentences';

create index pmid\_entity\_index\_btree using btree on ENTITY (PMID);