MetaMap Data File Builder

Willie Rogers, François-Michel Lang, Cliff Gay

December 31, 2015

Abstract

The MetaMap application is designed to automatically identify UMLS® Metathesaurus® concepts referred to in free text[1][2]. Although the UMLS focuses on biomedical information sources, MetaMap's algorithms are domain independent and can be used with any domain providing adequate knowledge sources. The MetaMap Data File Builder enables such cross-domain utilization of MetaMap by allowing users to create UMLS-like data models similar to the actual UMLS data models normally used by MetaMap.

1 Intended Use

We assume throughout this document that the full MetaMap distribution and the Data File Builder distribution have been downloaded from

`https://metamap.nlm.nih.gov/#Downloads`

and installed. Once the MetaMap distributions has been installed, users have the option of customizing the UMLS dataset or creating a dataset *de novo*. There are three principal ways to create a MetaMap knowledge base whose concepts the system is designed to identify:

1. **Full UMLS Metathesaurus**: To use MetaMap to map text to concepts in the current version of the UMLS Metathesaurus, there is nothing more to do beyond installing MetaMap, because the application is bundled with the latest version of the UMLS data. Full UMLS data sets other than the one included in the full MetaMap distribution (e.g., data sets from earlier releases of the UMLS) are also available from the same website given above.

2. **UMLS Metathesaurus Subset**: If copyright issues preclude the use of certain vocabularies in the MetaMap download, or if users want to focus on selected Metathesaurus vocabularies, one should use MetamorphoSys[1] to create an appropriate subset of the Metathesaurus and then use Data File Builder to create a customized data set that based based on the Metathesaurus subset.

3. **Non-UMLS Database**: Finally, in order to to use MetaMap to identify concepts in knowledge sources not based on the UMLS Metathesaurus, it is necessary to create from scratch a special-purpose data set; this approach also requires Data File Builder to create the database files used by MetaMap.

2 The Problem

MetaMap requires knowledge sources contained in a set of data files structured to facilitate rapid access to domain information. During the process of preparing

[1] `https://www.nlm.nih.gov/research/umls/implementation_resources/metamorphosys/index.html`

these data files that make up MetaMap's data model, Data File Builder filters out certain terms and strings that are unlikely to appear in normal English text or be identified as concepts by MetaMap; removing these data therefore enhances the performance of the application. Some of the data files are word-index files, which speed the mapping from strings to concepts; one file contains MeSH treecodes that can be included in the output to facilitate post-MetaMap processing; finally, other files contain precomputed lexical variants for strings in the knowledge base.

3 The Solution

We now explain how to use MetaMap's Data File Builder to create a special-purpose database. This process involves three principal phases:

1. **Creating a UMLS-like data set:** The first step creates a set of data files similar to those available from MetamorphoSys (specifically the MetamorphoSys files `MRCONSO`, `MRSAT`, and `MRSTY`)[3]. These data files must be either extracted from the UMLS Metathesaurus (e.g., using MetamorphoSys), or manually created from scratch; in either case, however, they must mirror the Metathesaurus files available through MetamorphoSys in structure, format, and semantics. Guidelines for creating these files are provided in Section 7.

2. **Generating MetaMap data files:** The second and most time consuming step actually runs Data File Builder to convert the files created in the previous step into a form supporting rapid access to these data by MetaMap. This processing is implemented by the *Build Data Files* module, which comprises seven shell scripts that call various GNU utilities and a specialized Java program.

3. **Loading the data files into a new database:** The third and final stage assembles the data files created in the second stage into a final data set indexed using Berkeley DB.[2] Once this database is created, it can be used for MetaMap processing.

4 Document Roadmap

Section 5 explains of how to tailor the instructions in this manual to specific operating systems (Solaris and Linux). Section 6 provides a historical perspective for Data File Builder processing. Next, sections 7, 8, and 9 explain in greater detail the three processing stages presented above. Section 10 presents some troubleshooting tips, and the manual concludes with section 11, which contains directions for using MetaMap with the new database.

5 Platform Differences

5.1 Mac OS/X

Most but not all GNU utilities are available preinstalled on Mac OS/X. Notably, GNU Sed and many of the GNU Coreutils are not provided in the default Mac OS/X installation. In many case non-compatible BSD and SYS/V versions are supplied. GNU Sed and Coreutils are described below:

GNU Sed This is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline). The data file builder scripts are written to use GNU sed rather than the original BSD/UNIX Sed that comes with Mac OS/X.

GNU Coreutils The GNU Core Utilities are the basic file, shell and text manipulation utilities of the GNU operating system.

The source code to these utilities are available at `https://www.gnu.org/software/`.

[2] `http://www.oracle.com/technology/products/berkeley-db/index.html`

Pre-compiled (binary) versions of these utilities can be obtained at the following:

Fink Project `http://www.finkproject.org/`

Mac Ports `http://www.macports.org/`

5.2 Linux

The Linux distribution includes the GNU text utilities, so there is nothing additional to download and install. You will be able to run the commands and programs in these instructions from most shells (e.g., sh, csh, tcsh, bash).

6 Background

This section discusses the processing required to generate the data files to be used by MetaMap. Several papers on the Reference Page of the MetaMap Portal[3] explain portions of the Data File Building process in greater depth:

- *Effective Mapping of Biomedical Text to the UMLS Metathesaurus: The MetaMap Program*: This recent paper provides not only a good overview of the processing available with MetaMap, but also discusses the data-maintenance process and variant generation. [4]
- *MetaMap Update Procedures*: This paper describes the maintenance of MetaMap data files. It was this process that was adapted for use with MetaMap. [5]
- *Filtering the UMLS Metathesaurus for MetaMap* A critical part of the MetaMap Data File generation is the filtering out of certain strings from the source files. A brief description of this process is provided below at the end of section 8.2, but this paper explains manual filtering, the seven kinds of lexical filtering, filtering by Metathesaurus term type (which is no longer done), and finally syntactic filtering. [6]
- *MetaMap: Mapping Text to the UMLS Metathesaurus*: Section 3 (Noun Phrase Variants, pp. 4-7) of this paper provides an explanation of the role of variant generation in the MetaMap matching algorithm. [7]

7 Creating a Custom Dataset

Recall that if MetaMap is to be used with the full UMLS Metathesaurus provided with MetaMap, there is nothing to do beyond installing MetaMap. We now explain the two methods of creating a custom Metathesaurus-like data set:

1. Using a subset of the actual UMLS Metathesaurus, and
2. Creating one's own customized knowledge base from scratch.

This section explains both these approaches.

7.1 Create the Workspace

Data File Builder expects all the knowledge sources to be in a common location, so one must first create a workspace in which all processing will be done. If the customized data source to be created will be based on the UMLS, we suggest giving the workspace a name that reflects the specific release year and version of the UMLS Metathesaurus that will serve as a basis for the custom data set. Specifically, we recommend using the last two digits of the year version of

[3] `https://metamap.nlm.nih.gov/`

[4] `https://skr.nlm.nih.gov/papers/references/metamap_01AMIA.pdf`

[5] `https://skr.nlm.nih.gov/papers/references/mm.update.pdf`

[6] `https://skr.nlm.nih.gov/papers/references/filtering08.pdf`

[7] `https://skr.nlm.nih.gov/papers/references/metamap06.pdf`

the UMLS joined by an underscore with a mnemonic name (e.g. `02_meshonly`). The following discussion assumes that the full MetaMap distribution has been downloaded and installed locally, as explained in Section 1.

To create a workspace on Unix:

1. Decide what to call the version of data to be used. For this example we will use `09_custom` as the workspace name.
2. After the full MetaMap distribution has been downloaded and installed, there will be a directory called `public_mm` just below the directory in which MetaMap was installed. We will refer to this `public_mm` directory as `$BASEDIR`. First, change directories to the `$BASEDIR`. directory:

   ```
   cd $BASEDIR
   ```

3. Next, in the `$BASEDIR` directory, create a directory for the workspace:

   ```
   mkdir sourceData/09_custom
   ```

4. Finally, Create a directory to store the knowledge sources:

   ```
   mkdir sourceData/09_custom/umls
   ```

7.2 Using a UMLS Metathesaurus Subset

As noted in Section 1, there are two basic approaches to creating customized knowledge sources. We next describe the first approach, which involves creating a subset of the UMLS Metathesaurus knowledge sources. The UMLS download provides a tool called MetamorphoSys, which is the UMLS' installation wizard and customization tool. MetamorphoSys is included in every UMLS release, and allows users to exclude and/or prioritize vocabularies in order to comply with licensing restrictions and exercise some control over naming of the Metathesaurus data. The MetamorphoSys Fact Sheet can be found at

`https://www.nlm.nih.gov/pubs/factsheets/umlsmetamorph.html`

The UMLS Knowledge Sources manual (the green book) discusses the operation of MetamorphoSys in section 6. This information is also available online at

`https://www.ncbi.nlm.nih.gov/books/NBK9682/`

The remaining documentation makes the following assumptions:

1. The latest UMLS release, including MetamorphoSys, has been downloaded to a directory denoted as `$UMLS`.
2. The `$UMLS` directory contains the MetamorphoSys download `mmsys.zip` file.
3. The `mmsys.zip` file has been unzipped by running `unzip mmsys.zip`.
4. Unzipping `mmsys.zip` deposited in the `$UMLS` directory files named
   - `linux_mmsys.sh`,
   - `macintosh_mmsys.sh`, and
   - `windows_mmsys.bat`

When preparing a subset with MetamorphoSys for use with MetaMap and the MetaMap Data File Builder, designate the workspace as the location for the target files.

To specify a MetamorphoSys configuration:

1. Move to to the `$UMLS` directory:

   ```
   cd $UMLS
   ```

2. Execute the script tailored appropriate to your operating system. For example, if running Linux, type:

   ```
   ./linux_mmsys.sh
   ```

3. Select Install UMLS.
4. For the Destination click on Browse....

5. Navigate the file browser until you can select the `$UMLS` directory created earlier.

6. Click the Open button. The selected directory should appear in the Destination box. Note: You may not need to install the other Knowledge Sources, so click on Semantic Network and SPECIALIST Lexicon to deselect them.

7. Select OK and then choose the desired UMLS subset.

8. When the UMLS MetamorphoSys Configuration tool appears, select the Output Options tab.

9. In the "Select Output Format" box, click on the "Browse" button and change the format to Rich Release Format.

10. In the "Eliminate Extended Unicode Characters" box, select option: "Remove records containing extended UTF-8 characters".

Complete the configuration of MetamorphoSys and run it according its documentation. After MetamorphoSys finishes, confirm that the `$UMLS` directory contains the files listed in the Required Files table of the next section.

7.3 Using a New Knowledge Source

The previous section described how to extract a subset of the UMLS Metathesaurus. This section now describes the process of building data sources not based on the UMLS Metathesaurus.

The goal is to create the files required to run the Data File Builder scripts in order to create a valid MetaMap database. Table 1 lists the files required, which must be placed in the `umls` directory of the workspace.

Preliminaries

As noted earlier, knowledge sources not based on the UMLS Metathesaurus must still mirror the Metathesaurus files available through MetamorphoSys in structure, format, and semantics.

File	When Required
MRCONSO.RRF	always
MRSAT.RRF	when treecode equivalent data is to be presented.
MRSTY.RRF	always
MRSAB.RRF	always
MRRANK.RRF	always
SM.DB	always

Table 1: Required Files

1. **Synonymy**: One of the fundamental bases of a thesaurus is the clustering of synonymous terms (i.e., text strings) into a single concept; for example, in the 2013AA version of the UMLS Metathesaurus, the concepts listed in each bullet below (among many others) are considered synonyms:

   - *Myocardial Infarction*; *Attack Coronary*; *Heart attack*; *Infarction of heart*; *Infarction, Myocardial*; *Infarct, Myocardial*; *Myocardial Infarct*
   - *Malignant neoplasm of lung*; *Cancers, Lung*; *Cancer of the Lung*; *Lung Cancer*; *malignant lung neoplasm*; *Malignant Lung Tumor*; *Pulmonary Cancer*
   - *Oxygen Therapy Care*; *Inhalation Therapies, Oxygen*; *Oxygen Inhalation Therapy*; *oxygen administration*; *oxygenation therapy*; *Therapy, Oxygen Inhalation*; *Warburg Therapy*
   - *Acetaminophen*; *4-Hydroxyacetanilide*; *Acetamidophenol*; *Hydroxyacetanilide*; *Paracetamol*; *Paracetamol product*; *p-Hydroxyacetanilide*

   Because all the terms in each bullet above are synonyms, identifying any one of them in input text should produce semantically identical results; moreover, including synonymous concepts

in the database should improve recall. It is therefore essential to determine groupings among synonymous *terms*. How exactly to represent synonymy in the database files will be explained below.

2. **Preferred Names**: For each group of synonymous concepts, one term should be designated as the *preferred name* for each concept. In each grouping above, the first concept is the preferred name. We therefore say, for example, that *Heart attack* is a synonym for the preferred name *Myocardial infarction*. Synonyms are also referred to as *non-preferred*. The preferred name for each concept is considered the one that best represents the concept in the vocabulary in question. How to identify preferred names in the database files will be explained below.

3. **Unique Term Identifier**: Select a unique identifier for each term in the knowledge source. This unique term identifier is analogous to the Metathesaurus' SUI, and is typically a sequential numbering of all terms (e.g., `T00001`, `T00002`, `T00003`, etc.) We will call this unique identifier the *term id*.

4. **Unique Concept Identifier**: Select a *concept id* for each concept (i.e., each group of synonymous terms) in the knowledge source. This unique concept identifier is analogous to the Metathesaurus' CUI, and again is typically a sequential numbering of all concepts (e.g., `D00001`, `D00002`, `D00003`, etc.).

We *strongly* recommend that you

1. use a consistent and mnemonic identification scheme for the unique term and concept identifiers following the example of the Metathesaurus,

2. use disjoint sets of identifiers for the unique term identifiers (SUI-like IDs) and unique concept identifiers (CUI-like IDs),

3. ensure that any SUI- or CUI-like IDs that you create *not* be identifiers used by the Metathesaurus; for example, as noted above, you could use strings beginning with `D` for your CUI-like IDs, and strings beginning with `T` for your SUI-like IDs.

Table Creation

We now present the outline of a process that can be used for creating the required database tables from a knowledge source. The following discussion will assume that the data-organization recommendations outlined above (synonymy, preferred names, SUI- and CUI-like identifiers) have been followed.

The next steps are the following:

- transforming the thesaurus into a computationally friendly form;
- creating the necessary identifiers for each term; and
- creating the Metathesaurus-like files using the identified synonymy, preferred names, and identifiers.

It is now necessary to format the data in the knowledge source as line-oriented, pipe-separated ("`|`") ASCII text files. A simple sequential numbering of the database records can serve as the basis for the term and concept identifiers. We next present the format and semantics of the required tables. For more information about the Metathesaurus tables, refer to Section 4 of UMLS Reference Manual:

`https://www.ncbi.nlm.nih.gov/books/NBK9682/`

Please note that the records in each table must be grouped by the first identifier in each record.

Table Descriptions

MRCONSO.RRF

The MRCONSO table contains a row for each term (not each concept) in the knowledge base; i.e., there

should be exactly one row for each (unique) CUI-SUI combination. Consider for example the groups of synonymous terms listed in the Synonymy discussion above. Each of the terms (*Myocardial infarction*; *Heart attack*; *Infarction of heart*; *Infarction, Myocardial*; *Myocardial Infarct*, etc.) would therefore have its own row in the MRCONSO table. See Table 2 for a description of the MRCONSO table's fields, sample values, and the fields' semantics. Note that all rows in the MRCONSO table that contain synonymous strings must have the same CUI; indeed identity of CUIs is precisely how synonymy is represented in the UMLS Metathesaurus.

The UMLS Metathesaurus contains terms drawn from a vast collection of different source vocabularies. Although assembling terms from many different vocabularies may not be necessary in all cases, the MRCONSO table identifies the vocabulary source(s) for a concept, term, and string.

The MRCONSO table contains a new element not found in the two tables it replaces, MRCON and MRSO which were used as input in the previous version of the Data File Builder; this element is the AUI field. The AUI or Atom Unique Identifer refers (or identifies) a unique combination of a string assigned to a concept and its vocabulary source. For datasets containing one vocabulary the space of Atom identifier would be a one-to-one mapping to String Unique Identifers (SUIs) (See Table 2).

As stated before, the records in MRCONSO table must be grouped by the first identifier in each record in this case the concept identifier (CUI). All MRCONSO records referring to the same concept must grouped together.

For example, the following are a subset the rows in the 2013AA MRCONSO file for the synonyms of the concept whose preferred name is *Myocardial Infarction* presented above:[8]

```
C0027051|ENG|P|L0027051|PF|S0064638|N|
  A0089429||M0014340|D009203|MSH|MH|D009203|
  Myocardial Infarction|0|N|1792|
C0027051|ENG|P|L0027051|PF|S0064638|N|
  A18061446||N0000002085||NDFRT|PT|
  N0000002085|
  Myocardial Infarction|0|N|1792|
C0027051|ENG|P|L0027051|PF|S0064638|N|
  A21143843||||MEDLINEPLUS|ET|5|
  Myocardial Infarction|0|N|1536|
C0027051|ENG|P|L0027051|PF|S0064638|N|
  A7572328||C27996||NCI|PT|C27996|
  Myocardial Infarction|0|N|1792|
C0027051|ENG|P|L0027051|PF|S0064638|Y|
  A0089430||||MTH|PN|NOCODE|
  Myocardial Infarction|0|N|1792|
C0027051|ENG|P|L0027051|VCW|S1916645|Y|
  A1861262||||ICPC2P|PT|K75013|
  Infarction;myocardial|3|N||
```

[8]These are not all the MRCONSO rows for CUI C0027051; also, the rows are indented for clarity; in the actual ASCII file, all eight pipe-separated fields appear on a single line.

MRSTY.RRF

Every concept in the knowledge base should be assigned at least one Semantic Type; the MRSTY table accordingly encodes the Semantic Type(s) associated with each concept. Table 3 presents the MRSTY table's fields, sample values, and the fields' semantics.

The actual MRSTY line represented in Table 3 is the following:

```
C0027051|T047|B2.2.1.2.1|
  Disease or Syndrome|AT32679180|3840|
```

MRSAT.RRF

The MRSAT table is used only to generate the treecode data that support the MMI output option of MetaMap. If you do not plan to generating MMI output with MetaMap, it is not necessary to supply a complete MRSAT file; an empty file will be sufficient. Please see Section 8.3 below for information about treecodes, which are used as a manageable representation of the parent/child relations of MeSH. Although the MetaMap algorithms make no direct use of treecodes, this information has been found useful

for post-MetaMap processing. If you choose to generate treecodes for your custom dataset, use the table below as a guide for the MRSAT file; the indication of "not used" in the table means that an arbitrary value in this column will work as well as the value from the MRCONSO file. You should also set SAB field to a mnemonic value such as "MED2009" containing the year of the current UMLS release, or the year in which your data are created.

```
C0027051|L0027051|S0052322|A0073387|AUI|
  D009203|AT43773908||TERMUI|MSH|T027447|N||
C0027051|L0027051|S0064638|A0089429|AUI|
  D009203|AT05237569||MED1951|NLM-MED|12|N||
C0027051|L0027051|S0064638|A0089429|AUI|
  D009203|AT05260220||MED1951|NLM-MED|
  *12|N||
C0027051|L0027051|S0064638|A0089429|AUI|
  D009203|AT05282871||MED1952|NLM-MED|
  181|N||
C0027051|L0027051|S0064638|A0089429|AUI|
  D009203|AT05305522||MED1952|NLM-MED|
  *181|N||
```

MRSAB.RRF

The MRSAB is used to generate a list of source abbreviations that are valid for a UMLS Knowledge Source Release.

Sample MRSAB.RRF record:

```
C3529121|C1135584|MSH2013_2013_01_21|MSH|
  Medical Subject Headings, 2013_2013_01_21|
  MSH|2013_2013_01_21|||2013AA||
  John Kilbourne, M.D.;Acting Head,
  MeSH Section;
  National Library of Medicine;
  6701 Democracy Boulevard,;Suite 202Q;
  Bethesda;MD;United States;20894;
  (301)-496-1495;(301)-402-2002;
  kilbourj@mail.nih.gov;
  https://www.nlm.nih.gov/mesh/meshhome.html|
  John Kilbourne, M.D.;Acting Head,
  MeSH Section;
  National Library of Medicine;
  6701 Democracy Boulevard,;Suite 202Q;
  Bethesda;MD;United States;20894;
  (301)-496-1495;(301)-402-2002;
  kilbourj@mail.nih.gov;
  https://www.nlm.nih.gov/mesh/meshhome.html|
  0|787946|335642|
  FULL-MULTIPLE|
  CE,DEV,DSV,EN,EP,HS,HT,MH,N1,NM,PCE,PEN,
  PEP,PM,PXQ,QAB,QEV,QSV,TQ,XQ|
  AN,AQL,CX,DC,DQ,DX,EC,FR,FX,HM,HN,II,LT,
  MDA,MMR,MN,OL,PA,PI,PM,RN,RR,SC,SOS,SRC,
  TERMUI,TH|ENG|UTF-8|Y|Y|MeSH|
  Medical Subject Headings (MeSH);
  National Library of Medicine;2013;
  Bethesda, MD|
```

Note: the record has been broken into multiple lines for readability.

See table 6 for explaination of MRSAB fields [9]

SM.DB

The SM.DB file is a list of semantically related terms. For additional information see:

https://www.ncbi.nlm.nih.gov/books/NBK9680/

Assuming the MetamorphoSys release is in the directory `$UMLS`, the SM.DB file can be found at

```
Install/<version>/LEX/LEX_DB/SM.DB
```

Each row of the database is of the form

```
Index String|TERM1|SCA1|TERM2|SCA2
```

[9] Excerpted from Table 2. Source Information (File = MRSAB) of 4.3.13 Source Information (File = MRSAB) of Metathesaurus - Rich Release Format (ORF): https://www.ncbi.nlm.nih.gov/books/NBK9682/table/ch04.T.source_information_file_mrsab/?report=objectonly

meaning that the term *TERM1* in syntactic category *SCA1* is semantically related to the term *TERM2* in syntactic category *SCA2*. Both terms are given in base form. Some sample rows follow:

```
alar|alar|adj|wing|noun
ocular|ocular|adj|eye|noun
auditory area|auditory area|noun|
  auditory center|noun
auditory area|auditory area|noun|
  auditory cortex|noun
vomitive|vomitive|adj|emetic|adj
vomitive|vomitive|noun|emetic|noun
iridescent virus|iridescent virus|noun|
  iridovirus|noun
typhloteritis|typhloteritis|noun|cecitis|noun
```

MRRANK.RRF

The MRRANK file ranks the precedence of records by source vocabulary. The information provided by this file is used during the UMLS filtering process to determine which morphologically equivalent UMLS records should be retained and which should be discarded.

Some sample rows follow:

```
0839|MTH|PN|N|
0838|RXNORM|MIN|N|
0837|MSH|MH|N|
0836|MSH|TQ|N|
0835|MSH|PEP|N|
0834|MSH|PEN|N|
0833|MSH|EP|N|
0832|MSH|EN|N|
0831|MSH|XQ|N|
0830|MSH|PXQ|N|
```

8 Generating the Data Files

We assume at this point that the required tables listed in Table 1 above on page 5 have been created. MetaMap's datafiles can be divided into five categories:

1. word-index files,
2. a treecode file,
3. variants files,
4. synonyms files, and
5. abbreviations and acronyms (AA) files.

In order to allow users to more easily review intermediate results, the ASCII files containing the data for each data category are created in separate directories using a series of scripts. Users must set up their workspace, navigate to each of the directories, and run one or more scripts to generate the data files for each category of data.

8.1 Setting up a Workspace

We begin this stage by completing the setup of the workspace undertaken in Section 7.1 above. The workspace will contain directories holding the intermediate files and final output files, as well as copies of the scripts that will be run.

To set up a new workspace:

1. First, run the `BuildDataFiles` program as follows:

   ```
   ${BASEDIR}/bin/BuildDataFiles
   ```

2. Next, when prompted, enter the name of the workspace:

   ```
   09_custom
   ```

3. If the path shown is the location of the source data directory (`umls`), answer `Yes`; otherwise, enter the correct workspace name, or move the data to the proper location. See Section 7.1 above.

4. The process described in this document requires the use of the lexicon distributed as part of the MetaMap download. Lexical data from the UMLS are used in the generation of variants and become part of the data files. Here enter the year of the lexicon to use (or simply accept the default).

   ```
   2009
   ```

If this script generates errors, it is likely that certain required files listed in Table 1 above are missing. Any missing files should be created before running the `BuildDataFiles` script.

8.2 Generating MetaWordIndex Files

The MetaWordIndex files contain the strings of the Metathesaurus (or the Metathesaurus-like knowledge source) indexed by the individual words they contain. These files constitute the majority of the final data files, and are generated by executing five scripts that filter and reorganize the raw data (MRCONSO, MRSAT, etc.).

Workfiles:

The primary work of this first step is to strip the MRCONSO input file down to only the English strings—those lines in the MRCONSO table whose second field is `ENG`. Note that this processing must still be done even if all the rows in a custom-crafted MRCONSO table have `ENG` as the second field.

To create the workfiles:

1. First, move to the workspace directory that contains the `umls` directory:

   ```
   cd $BASEDIR/sourceData/09_custom
   ```

2. Next, move to the `01metawordindex` directory:

   ```
   cd 01metawordindex
   ```

3. Then run the `01CreateWorkFiles` script:

   ```
   ./01CreateWorkFiles
   ```

The script takes about 2 minutes to complete; as it runs, it will generate some timing statistics and notes about its progress, and remind you what to do next;

Suppression:

This second step marks the terms that have been identified as problematic to MetaMap processing as suppressible synonyms, so that they will be skipped during the filtering step described below. The small number of such problematic Metathesaurus strings include numbers, single alphabetic characters, special cases, and ambiguities. There are two parts to this step, which together take about four minutes.

To run suppression:

1. First, as in the workfiles step immediately above, move to the same `01metawordindex` directory.

2. Then run the `02Suppress` script:

   ```
   ./02Suppress
   ```

FilterPrep:

This third step extracts tables mapping cuis to vocabularysource and semantic types from the `mrconso.suppressed` file modified by `02Suppress`; the raw data file is described in Table 2 to create the file `mrconso.eng`, which will contain data that will be used in the filtering process described below. This step takes about 9 minutes.

To run filtering:

1. First, as in the two preceding steps outlined above, move to the same `01metawordindex` directory.

2. Then run the `03FilterPrep` script:

   ```
   ./03FilterPrep
   ```

Filtering:

The filtering step uses studies previously performed on the full Metathesaurus dealing with how to

- process NEC/NOS,
- handle parentheticals, and
- handle possessives.

There are three forms of filtering:

1. Manual Filtering, which excludes those strings marked for suppression in the Suppression step above.
2. Lexical Filtering, the most benign type of filtering, which consists of removing strings mapped to a given concept that are effectively the same as another string for that concept.
3. Syntactic Filtering, the final filtering step, which excludes many long or complex phrases. Syntactic filtering uses MetaMap's syntactic analyzer to parse Metathesaurus(-like) strings and determine their syntactic complexity. Because normal MetaMap processing analyzes text one phrase at a time, it is highly unlikely that a concept represented by a complex Metathesaurus string consisting of multiple phrases will ever be identified. Most strings consisting of more than one simple phrase are therefore filtered out. Phrases containing certain simple prepositional phrases, e.g., *Cancer of the lung*, *Wax in ear*, *Homes for the aged* are exempted from this syntactic filtering.

The extent of filtering that is performed is determined by the model to be created (strict or relaxed). Broadly speaking, MetaMap can be used in two ways: constrained semantic processing, and less constrained browsing, which casts a wider net and places more emphasis on recall than precision. To support these two modes of processing, separate data models differing in the degree of filtering can be created:

- The **Strict Model** applies all forms of filtering described above; this view is most appropriate for semantic processing where the greater accuracy is desired.
- The **Relaxed Model** applies only manual and lexical filtering (and not syntactic filtering), thereby providing access to a far broader range of Metathesaurus(-like) strings, and improving recall, albeit at the of cost of slower processing.

At least one the models must be created; the strict model is probably the more useful of the two, but both can be created in succession.

To create a filtered model:

1. As described in the previous steps, first move to the `01metawordindex` directory.
2. Then, determine the more suitable model for the intended processing, and run either `04FilterStrict` or `04FilterRelaxed`, as appropriate.

   ```
   ./04FilterStrict
   ```

   or

   ```
   ./04FilterRelaxed
   ```

   Note: If using `04FilterStrict`, be sure that the SKR/MedPOST tagger is running; to run the tagger, move to the `public_mm` directory and invoke

   ```
   ./bin/skrmedpostctl start
   ```

The two `04FilterStrict` and `04FilterRelaxed` scripts run a program to perform the desired filtering of the `mrconso.eng` file created by the `03FilterPrep` script, as described on page 10. The filtering scripts then generate the file

```
Filter/mrconso.eng.strict
```

or

```
Filter/mrconso.eng.relaxed
```

from the intermediate `mrconso.eng.strict` or `mrconso.eng.relaxed` file.

Finally, the scripts create a directory for the target model (`model.strict` or `model.relaxed`) that contains the file `mrcon.filtered`, which is simply a symbolic link to either `Filter/mrconso.eng.strict` or `Filter/mrconso.eng.relaxed`.

It is possible—even desirable—to generate both the strict and relaxed models by running the two `04FilterStrict` and `04FilterRelaxed` scripts one after the other.

Modifying Semantic Type Translations:

If the `MRSTY` file does not contain any non-UMLS Semantic Types, there is nothing to do in this step; otherwise, the file `st.raw`, located in

```
cd $BASEDIR/data/dfbuilder/<YEAR>
```

will need to be modified. The `st.raw` file contains lines of the form

```
SemType Full Name|SemType Abbreviation
```

e.g.,

```
Body Part, Organ, or Organ Component|bpoc
Disease or Syndrome|dsyn
Injury or Poisoning|inpo
```

Simply add a line to the `st.raw` file for each new Semantic Type.

Generating MWI Files:

This final step for creating the MetaWordIndex files produces the `.txt` files, which are ultimately loaded into the database used by MetaMap. A Java program extracts word information from the filtered MRCONSO file and builds the necessary database index files. This step takes approximately 15 minutes for each of the filter models created.

To generate MWI files:

1. As always, navigate to the `01metawordindex` directory.
2. Then run the `05GenerateMWIFiles` script:

   ```
   ./05GenerateMWIFiles
   ```

If you subsequently decide to generate a second model, you may run the other `04Filter` script and then rerun this step. `05GenerateMWIFiles` will generate the MetaWordIndex files only once for each model.

8.3 Generating the Treecode Files

Treecodes represent the attribute of MeSH terms that encode their hierarchical relationships to other terms. The treecode file is not directly used in MetaMap processing, but ensures that treecode information is included in MetaMap's output. This is the only step that uses the UMLS MRSAT file.

To generate the treecodes file:

1. Move to the 02treecodes directory of your workspace. If you just finished generating the meta word index files, you can type:

   ```
   cd ../02treecodes
   ```

2. Run the `01GenerateTreecodes` script:

   ```
   ./01GenerateTreecodes
   ```

This script takes only about 3 minutes, and generates the file `treecodes.txt`.

8.4 Generating the Variants Files

MetaMap variants are linguistic variants of base words and terms (variant *generators*) that have been computed using a variety of generation methods, including

- Inflectional variants:
  - *heart* → *hearts*
  - *treat* → *treating*
  - *cold* → *colder*
- Derivational variants:
  - *treat* → *treatment*
  - *cure* → *curable*
  - *fatal* → *fatality*
- Synonyms:
  - *pyrosis* ↔ *heartburn*
  - *tylenol* ↔ *acetaminophen*
  - *precancerous* ↔ *premalignant*
- Acronyms and abbreviations and their expansions:
  - *vt* ↔ *ventricular tachycardia*
  - *ama* ↔ *american medical association*
  - *vldl* ↔ *very low density lipoprotein*
- Spelling variants:
  - *apnea* ↔ *apnoea*
  - *cardioactive* ↔ *cardio-active*
  - *leukemia* ↔ *leukaemia*

and useful combinations of all the above. Every derivation of a variant from its generator carries an associated score measuring the variant's distance from its generator. Once the variants have been computed, results are filtered so that all table entries contain strings actually occurring in the Metathesaurus.

Variant Generation is a computationally intensive process because variants are generated from several hundred thousand generators. This process typically takes requires several hours. Be patient!

To generate variants:

1. First, move to the `03Variants` directory in the workspace. If you just finished generating the treecode file, you can type simply:

   ```
   cd ../03Variants
   ```

2. Then run the `01GenerateVariants` script:

   ```
   ./01GenerateVariants
   ```

Because of the computational intensity of the `01GenerateVariants` script, external events may result in abnormal termination of the processing before completion, which can be recognized by the messages identifying the next step.

8.5 Generating Synonyms

MetaMap's synonyms database is closely modeled after the SM.DB UMLS datafile included in the UMLS download.

To generate synonyms:

1. Move to the `04synonyms` directory of your workspace. If you just finished generating the variants files, you can type simply:

   ```
   cd ../04synonyms
   ```

2. Run the `01GenerateSynonyms` script:

   ```
   ./01GenerateSynonyms
   ```

8.6 Generating Acronyms and Abbreviations

Acronyms and abbreviations (AAs)[12] are very useful for improving recall in acronym-laden domains such as biomedical literature and clinical texts. For example, consider Medline's PMID 15387473, whose title is *Non-exercise activity thermogenesis (NEAT)*. In that title (and in the first sentence of the abstract), MetaMap recognizes the string *NEAT* as an acronym for *non-exercise activity thermogenesis*. In

[12] For example, *AAs* is an acronym for *Acronyms and Abbreviations*!

the remainder of the abstract, the string *NEAT* appears no fewer than ten additional times; because of MetaMap's ability to handle AAs, each of those occurrences of *NEAT* is correctly interpreted to mean *non-exercise activity thermogenesis*.

To generate Acronyms and Abbreviations:

1. First move to the `05abbrAcronyms` directory of your workspace. As before, if you just finished generating the synonyms file, you can type simply:

   ```
   cd ../05abbrAcronyms
   ```

2. Then run the `01GenerateAbbrAcronyms` script:

   ```
   ./01GenerateAbbrAcronyms
   ```

9 Data Set Creation and Load

This final stage (whew!) creates the final version of the datafiles, and loads them into a Berkeley DB database used by MetaMap just as it would use one of the standard Berkeley DB databases available in the normal MetaMap download.

To create a data set and load it into Berkeley DB:

1. First, run the `LoadDataFiles` program as follows:

   ```
   ${BASEDIR}/bin/LoadDataFiles
   ```

2. Next, when prompted, enter the name of the workspace:

   ```
   09_custom
   ```

3. The script then displays the full path for the workspace. If it is correct, simply hit return; if not, enter "no" and try again.

4. Next, the script checks the workspace to see which model(s) have been built, and asks which one(s) to load. This step should take about a half hour.

The new dataset will be deposited in the directory `$BASEDIR/DB/BDB4/DB.09_custom.base` and `$BASEDIR/DB/BDB4/DB.09_custom.strict` or `$BASEDIR/DB/BDB4/DB.09_custom.relaxed`. A complete data set contains at least two subdirectories: The `base` subdirectory holds files shared by the strict and relaxed models, and the `strict` and `relaxed` directories hold all model-specific files. Figure 1 below shows the shared files and the model-specific files for the strict model.

If an error is generated by the `LoadDataFiles` script (especially "File not found"), likely causes include an unnoticed failure in one of the previous steps or the omission of one of the steps altogether. Consult the table in Section 10 below for more information about error recovery.

If no errors were generated, the Data File Builder suite has completed all processing. Congratulations! Users are now referred to Section 11 below for an explanation of how to point MetaMap to the newly created database.

10 Troubleshooting

When a data file is missing: If a missing file is reported, you should go to the directory where the missing file should be, and (re-)run the script that creates it. If intermediate files exist in the directory, check the output from the script carefully for error messages.

When an error is reported: Because errors can have such varied etiology, we can offer only general suggestions; the specific error message may offer some clues as to what might have gone wrong. If not, we recommend cleaning up the working directory for the step that failed, and simply running it again.

To clean up a working directory: Using the information from Table 7, identify the directory used

for the script that failed. The directory is often the one containing the script in question, but sometimes it is the one containing helper scripts or programs. In that working directory, remove all files that are not input files. For `03variants`, however, remove the subdirectories.

To regenerate MWI files: If the Data File Builder scripts are re-run, the `05GenerateMWIFiles` script will skip over those models for which data files had already been generated. To force `05GenerateMWIFiles` to regenerate MWI files, follow these steps:

To regenerate files for the strict model:

1. Go to the model directory.

   ```
   01metawordindex/model.strict
   ```

2. 2 Remove the `sui_cui.txt` file:

   ```
   rm sui_cui.txt
   ```

3. Make the other files writeable.

   ```
   chmod u+w *.*.*
   ```

4. Return to MetaWordIndex directory.

   ```
   cd ../
   ```

5. Rerun the `./05GenerateMWIFiles` script.

   ```
   ./05GenerateMWIFiles
   ```

To regenerate files for the relaxed model, simply follow the above instructions, substituting `relaxed` for `strict`.

If you need to completely start over: If a problem with the input data has surfaced, and you need to rerun the entire family of Data File Builder scripts *ab initio*, it is possible to (re-)run everything with a single command.

To run all Data File Builder steps at once:

1. Clean up the model directories, as described immediately above.

2. Run the top-level script, which calls all the other previously described scripts:

   ```
   $BASEDIR/bin/rundatafiles.sh
   ```

3. When prompted to specify the dataset, simply hit Enter to specify the default.

4. You will also be asked which model to generate. (The strict model is the default.) Enter 'yes' for the desired model(s).

11 Using MetaMap

This section gives a brief overview of using MetaMap with a custom data set. More complete information on using MetaMap and its options is available at

```
https://metamap.nlm.nih.gov
```

The `LoadDataFiles` program created a new MetaMap configuration file that sets options to point MetaMap to the new database and the new variants table.

To run MetaMap:

1. Move to the top level directory in your MetaMap installation:

   ```
   cd <parent directory>/public_mm
   ```

2. Run MetaMap using a input file of text to be processed (e.g., the file used to test the MetaMap installation):

   ```
   ./bin/metamap -V 09_custom
         resources/test.txt custom.out
   ```

   Note: The -A option (the default) specifies the strict model; use -C to specify the relaxed model instead.

3. Examine the custom.out file to see the MetaMap output generated using with your data set.

   ```
   diff resources/test.out custom.out
   ```

   This command will probably return some differences because your newly generated custom data files are different from the data in the MetaMap distribution. We recommend ensuring that these differences are reasonable and explainable.

Fieldname	**Sample Value**	**Description**
CUI (Concept Unique ID)	C0027051	Letter + 7 or more digits
Language of Term	ENG	Fixed
Term Status	P or S	P for preferred name, S for non-preferred names
LUI (Lexical Unique ID)	L0027051	Letter + 7 or more digits (not used)
String Type	PF	Fixed
SUI (String Unique ID)	S0064638	Letter + 7 or more digits
ISPREF (Is Preferred Name)	N	Y or N
AUI (Atom Unique ID)	A0089429	Letter + 7 or more digits, there is one AUI for each unique combination of String and Vocabulary Source
SAUI (Source Atom Unique ID)		determined by source
SCUI (Source Concept ID)	M0014340	determined by source
SDUI (Source Descriptor ID)	D009203	determined by source
SAB (Abbreviated Source Name)	MSH	Alpha String
TTY (Term Type)	MH	determined by source
CODE (Source Asserted Id)	D009203	determined by source
STR (String)	Myocardial Infarction	The term string
SRL (Source Restriction Level)	0	digit
SUPPRESS	N	Y if suppressed or N
CVF (Content View Flag)	1792	digits

Table 2: MRCONSO.RRF Fields

Fieldname	Value	Description
CUI	C0027051	Same as MRCONSO.RRF
TUI	T047	Unique ID for Semantic Type
STN	B2.2.1.2.1	Semantic Type tree number
STY	Disease or Syndrome	Name of semantic type
ATUI	AT32679180	Unique Identifier for attribute
CVF	3840	Content View Flag

Table 3: MRSTY.RRF Fields

Fieldname	Value	Source
CUI	C0027051	Same as MRCONSO
LUI	L0027051	Same as MRCONSO (not used)
SUI	S0052322	Same as MRCONSO (not used)
METAUI	A0073387	Metathesaurus atom identifier (will have a leading A) or Metathesaurus relationship identifier (will have a leading R) or blank if it is a concept attribute.
STYPE	AUI	identifier to which the attribute, i.e. AUI, CODE, CUI, RUI, SCUI, SDUI.
CODE	D009203	Most useful source asserted identifier
ATUI	AT43773908	Unique identifier for attribute
SATUI		Source asserted attribute identifier (optional - present if it exists)
ATN	TERMUI	Attribute name. Possible values appear in MRDOC.RRF and are described on the Attribute Names page.
SAB	MSH	Abbreviated source name (SAB).
ATV	T027447	Attribute Value
SUPPRESS	N	Suppressible flag. Values = O, E, Y, or N.
CVF		Content View Flag.

Table 4: MRSAT Fields

Fieldname	Value	Description
RANK	0624	Numeric order of precedence, higher value wins
SAB	AIR	Abbreviated source name (SAB). [10]
TTY	SY	Abbreviation for term type in source vocabulary, [11]
SUPPRESS	N	Flag indicating that this SAB and TTY will create a TS=s MRCONSO entry; see TS (Term Status) in Table **??**

Table 5: MRRANK Fields

```
DB/DB.normal.2008_level0.base:
config                    meta_mesh_tc_opt.txt
config.01                 nlsaa
config.02                 nls_aa.txt
config.common             nlsaau
config.vars               nls_aau.txt
cuisourceinfo             sab_rv
cui_sourceinfo.txt        sab_rv.txt
cuisrc                    sab_vr
cui_src.txt               sab_vr.txt
meshmh                    syns
mesh_mh_opt.txt           syns.txt
meshtcrelaxed             vars
mesh_tc_relaxed.txt       varsan
meshtcstrict              varsan.txt
mesh_tc_strict.txt        varsanu
metamesh                  varsanu.txt
meta_mesh_opt.txt         vars.txt
metameshtc                varsu
meta_mesh_tc_opt.txt      varsu.txt

DB/DB.normal.2008_level0.strict:
all_words                 first_words_of_one
all_words_counts          first_words_of_one.txt
all_words_counts.txt      first_words_of_two
all_words.txt             first_words_of_two.txt
conceptcui                first_words.txt
concept_cui.txt           meshmh
conceptst                 meshtcrelaxed
concept_st.txt            meshtcstrict
config                    metamesh
cuiconcept                metameshtc
cui_concept.txt           nlsaa
cuisourceinfo             nlsaau
cuisrc                    sab_rv
cuist                     sab_vr
cui_st.txt                sui_cui
cui_sui.txt               sui_cui.txt
first_words               sui_nmstr_str.txt
first_wordsb              suistrings
first_wordsb_counts       syns
first_wordsb_counts.txt   vars
first_wordsb.txt          varsan
first_words_counts        varsanu
first_words_counts.txt    varsu
cui_sts_source.txt
```

Figure 1: Contents of a typical MetaMap dataset

Bibliography

[1] Alan R Aronson, Franois-Michel Lang An overview of MetaMap: historical perspective and recent advances, Journal of the American Medical Informatics Association, 2010;17:229-236, doi:10.1136/jamia.2009.002733, http://jamia.bmj.com/content/17/3/229.long

[2] Unified Medical Language System (UMLS), National Library of Medicine, https://www.nlm.nih.gov/research/umls/knowledge_sources/metathesaurus/index.html

[3] UMLS Reference Manual [Internet], Metathesaurus - Rich Release Format (RRF), National Library of Medicine, https://www.ncbi.nlm.nih.gov/books/NBK9685/

Fieldname	Full Name	Description
VCUI	CUI	CUI of the versioned SRC concept for a source
RCUI	Root CUI	CUI of the root SRC concept for a source
VSAB	Versioned Source Abbreviation	The versioned source abbreviation for a source e.g., MSH2003_2002_10_24
RSAB	Root Source Abbreviation	The root source abbreviation for a source e.g., MSH
SON	Official Name	The official name for a source
SF	Source Family	The source family for a source
SVER	Version	The source version e.g., 2001
VSTART	Valid Start Date For A Source	The date a source became active, e.g., 2004_04_03
VEND	Valid End Date For A Source	The date a source ceased to be active, e.g., 2003_05_10
IMETA	Meta Insert Version	The version of the Metathesaurus in which a source first appeared, e.g., 2001AB
RMETA	Meta Remove Version	The version of the Metathesaurus in which a source last appeared, e.g., 2001AC
SLC	Source License Contact	The source license contact information
SCC	Source Content Contact	The source content contact information
SRL	Source Restriction Level	0, 1, 2, 3, 4, 9 - explained in the License Agreement
TFR	Term Frequency	The number of terms for this source in MRCONSO, e.g., 12343
CFR	CUI Frequency	The number of CUIs associated with this source, e.g., 10234
CXTY	Context Type	The type of contexts for this source. Values are FULL, FULL-MULTIPLE, FULL-NOSIB, FULL-NOSIB-MULTIPLE, FULL-MULTIPLE-NOSIB-RELA, null.
TTYL	Term Type List	Term type list from source, e.g., MH, EN, PM, TQ
ATNL	Attribute Name List	The attribute name list (from MRSAT), e.g., MUI, RN, TH
LAT	Language	The language of the source
CENC	Character Encoding	All UMLS content is provided in Unicode, encoded in UTF-8. MetamorphoSys will allow exclusion of extended characters with some loss of information. Transliteration to other character encodings is possible but not supported buy NLM; for further information, see `http://www.unicode.org`.
CURVER	Current Version	A Y or N flag indicating whether or not this row corresponds to the current version of the named source
SABIN	Source in Subset	A Y or N flag indicating whether or not this row is represented in the current MetamorphoSys subset. Initially always Y where CURVER is Y, but later is recomputed by MetamorphoSys.

Table 6: MRSAB Fields

Script	Helpers	Input Files	Output Files
BuildDataFiles	builddatafiles.sh	public_mm/dfbuilder/scripts/* public_mm/data/dfbuilder/*	(The workspace)
01metawordindex			
01CreateWorkFiles	(none)	umls/MRCONSO	mrconso.eng.0
02Suppress	Suppress/ 0Suppress1 0Suppress2	mrconso.eng.0	mrcon.eng
03FilterPrep 04Filter*	(none) Filter/ExpandIndex	mrconso.eng mrconso.eng	mrconso.eng mrconso.eng.<model>
05GenerateMWIFiles	Generate/ 0Generate1 0Generate2 glean_mrconso	Filter/mrconso.eng.<model>	model.*/*.txt (See list in Figure 1 for individual file names)
02treecodes			
01GenerateTreecodes	(none)	MRCONOS.mesh umls/MRSAT ../01metawordindex/mrconso.eng Filter/mrconso.eng.*	treecodes.txt
03variants			
01GenerateVariants	(many)	../01metawordindex/model.*/ all_words.txt	vars*.txt
04synonyms			
01GenerateSynonyms	write_syns.perl write_syns	../umls/SM.DB	syns.txt syn_nls_db.txt
05abbrAcronyms			
01GenerateAbbrAcronyms	extract_addr_perl.09 write_aas.perl	public_mm/lexicon/data/ lexiconStatic<year> lexiconStatic<year>IndByEui.dbx lexiconStatic<year>IndByInfl.dbx	nls_aa.txt
LoadDataFiles	loaddatafiles.sh create_bulk	../01metawordindex/model.*/ *.txt	public_mm/DB/BDB4/ public_mm/DB/BDB4/ DB.normal.<name>.base public_mm/DB/BDB4/ DB.normal.<name>.<model>

Table 7: Data File Builder Overview