MetaMap Update Procedures

Alan R. Aronson

December 14, 2000

1. Overview

These notes describe the process for updating MetaMap's data files following the release of a new version of the UMLS Knowledge Sources. The update procedures are described in the next section, and a summary of the process is given in section 3. Finally, procedures for updating the files following a MeSH release are described in section 4, the Appendix.

2. The Update Process

2.1 MetaMap Data

The data used by MetaMap fall into several groups:

- **STT** (Semantic Type Table): a table showing semantic types, their abbreviations, unique identifier and treecode (e.g., the semantic type "Disease or Syndrome" has abbreviation "dsyn", unique identifier T047, and treecode B2.2.1.2.1).
- **MWI** (Meta Word Index): many files providing access to UMLS strings from the words they contain.
- **MeSH** (MeSH Treecodes): files defining treecodes for MeSH headings.
- **AA** (NLS Abbreviations/Acronyms): files extracted from the SPECIALIST Lexicon showing words and their abbreviations/acronyms.
- **Syn** (Synonyms): a file of synonym relationships obtained from Dorland's Illustrated Medical Dictionary and a supplementary NLS file of synonyms.
- **Var** (Variants): precomputed files of words and their variants according to the MetaMap variant generation algorithm (see *MetaMap Technical Notes*).

2.2 Data Dependencies

Sources for the MetaMap data are

- <Net>, the UMLS Semantic Network;
- <Meta>, the UMLS Metathesaurus;
- <Lex>, the SPECIALIST Lexicon; and

- <NLS>, NLS Supplementary Synonyms

The dependencies of the data on these data sources are shown in Figure 1. Arrows indicate direct

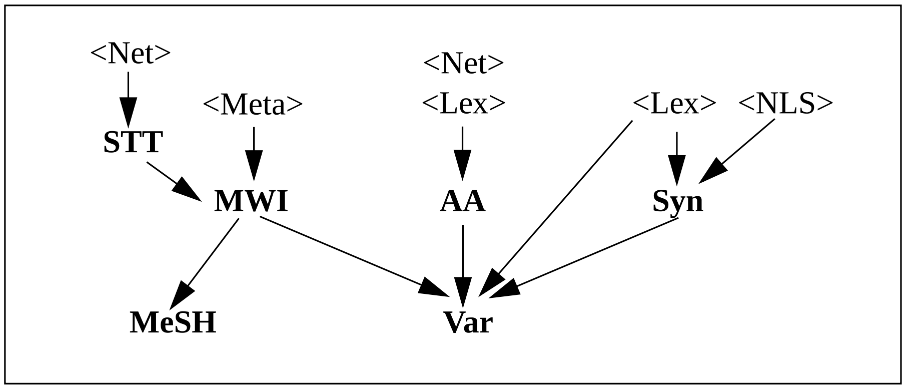

Figure 1. Data Dependencies

dependencies. Note, for example, that STT depends on <Net>, MWI depends on <Net>, STT and <Meta>, and Var depends on everything except MeSH.

2.3 Update Procedures

Procedures for updating each kind of MetaMap data are described here.

2.3.1 Semantic Type Table (STT)

The semantic type table is actually the Prolog predicate semtype_translation/4 in the module semtype_translation00 (located at /nfsvol/nls/specialist/module/semtype_translation/). The arguments of the predicate are a semantic type, its four-letter abbreviation, its unique identifier and its tree number. (See the example in section 2.1 above.) For use in MetaMap, the intermediate file `st.raw` is all that is needed. `st.raw` consists of lines containing a semantic type and its four-letter abbreviation delimited by the pipe symbol. The semantic types are obtained from the UMLS file `SRDEF`, and the abbreviations are defined by us.

Summary information
- Input files
  `SRDEF`
- Output files
  `st.raw`
- Time to create
  0.25 hours

2.3.2 Meta Word Index (MWI) Files

The Meta word index files are the most extensive files used by MetaMap. Three versions of them are created, one for each of the relaxed, moderate and strict MetaMap data models. (See *Filtering the UMLS Metathesaurus for MetaMap: 2000 Edition*.) Each version gives MetaMap the capability of finding Metathesaurus strings containing given words.

The data sources for the MWI files are the Metathesaurus files `MRCON`, `MRSO` and `MRSTY` and the semantic type table file `st.raw` defined in the previous section. Before the actual word index files are created, several studies and preliminary steps are performed. These preliminary processes are located at /nfsvol/nls/specialist/module/metawordindex/data.00/. Each is contained in a separate directory which contains descriptive information as well as scripts which automatically perform almost all of the work required to accomplish the tasks.

- Ambiguity study to discover ambiguities which are caused by overly aggressive synonymy (e.g., 54 strings 'other <n>') or by *contextual* terms such as 'Bladder' with meaning 'Benign neoplasm of bladder' or 'Carcinoma in situ of bladder' (See *Ambiguity in the UMLS Metathesaurus: 2000 Edition*.)
- Suppression of unwanted Metathesaurus strings: numbers, single alphabetic strings, unnecessarily ambiguous terms (from the Ambiguity study) and special cases such as the synonym 'Periods' for 'Menstruation'.
- Filtering the Metathesaurus based on *Filtering the UMLS Metathesaurus for MetaMap: 2000 Edition*. The filtering methods described in this document are manual, lexical, type-based, and syntactic. They produce three models of the Metathesaurus (relaxed, moderate and strict) based on the degree of filtering performed. The filtering process actually implies three subsidiary studies: a concept type study to determine which Metathesaurus term types (TTY) can be filtered out; an *of* study to determine how many consecutive *of* prepositional phrases occur in Metathesaurus strings (the current number is eight); and a study to determine which prepositions, conjunctions and determiners will be used to determine, for example, when to uninvert Metathesaurus strings.
- NEC/NOS study to determine how to ignore the many ways of expressing one or both of NEC and NOS in Metathesaurus strings.
- Parentheticals study to determine which parentheticals can be safely removed from Metathesaurus strings without affecting their meaning. The problem of parentheticals has almost disappeared due to the inclusion of strings both with and without non-essential parentheticals and to the fact that many strings with parentheticals have been marked as suppressible synonyms.
- Possessives study to confirm how possessives are expressed in the Metathesaurus. Like parentheticals, possessives are no longer of great concern.

Note that the NEC/NOS, parentheticals and possessives studies can produce results which require filtering to be recomputed. Because of the stability of this process, however, such iteration is not likely.

The main MWI processing, i.e., the creation of the output files for the three data models, is anticlimactic at this point and is accomplished completely by scripts. In addition there are some further studies which have been performed in the past but which are not necessary for updating MetaMap's data files.

Summary information

- Input files
  `MRCON`
  `MRSO`
  `MRSTY`
  `st.raw`
- Output files
  `all_words.txt`

```
all_words_counts.txt
concept.cui.txt
concept.st.txt
cui.concept.txt
cui.st.txt
first_words.txt
first_words_counts.txt
first_words_of_one.txt
first_words_of_two.txt
sui.cui.txt
sui.nmstr.str.txt
```

- Time to create
  3 days 4.5 hours (or 20.1 hours with parallelization of the most timeconsuming task)

2.3.3 MeSH Treecode (MeSH) Files

The MeSH treecode files include assignments of treecodes to MeSH headings (and also pseudo treecodes to chemicals and subheadings); but they also include information such as the MeSH heading associated with a given MeSH term and the MeSH terminology associated with Metathesaurus concepts. (The MeSH terminology associated with a given heading is often split among more than one Metathesaurus concepts.) Because these files depend on the MWI files which occur in three models, there should be three versions of these files. However, since MeSH is not strongly affected by the filtering process, only one version derived from the most inclusive, relaxed MWI model is computed. The files are located at /nfsvol/nls/specialist/module/db_access/data.00/MeSHTreeCodes/ and are created in the following four steps.

- Compute `mrcon.mesh`, a version of `MRCON` consisting only of MeSH, using the metamorphosys program.
- Do a preliminary computation of the files without pseudo treecodes for chemicals, subheadings and check tags. This work is based on Susanne Humphrey's hierarchy of subheadings.
- Study MeSH in order to assign pseudo treecodes to subheadings (using knowledge from Susanne's MEDINDEX) and chemicals.
- Compute the final files using the previous results.

Most of the above processing is done automatically using scripts. The manual effort occurs in the assignment of pseudo treecodes (mainly to subheadings, since chemicals all get the same pseudo treecode) within the MeSH study. Since subheadings do not change dramatically over time, this effort should remain stable.

Summary information

- Input files
  `mrcon.filtered` (from the relaxed MWI model)
  `MRCON`
  `MRSO`
  `MRSAT`
- Output files

```
mesh.mh.opt.txt
mesh.tc.relaxed.txt
mesh.tc.strict.txt
```

`meta.mesh.opt.txt`
`meta.mesh.tc.opt.txt`
- Time to create
  3.4 hours

2.3.4 NLS Abbreviation/Acronym (AA) Files

There are two versions of the AA files, one containing all abbreviations/acronyms (AAs), their expansions, and their syntactic category and a more restrictive version containing only those AAs having unique expansions. The basic AA information is obtained from the lexicon using the `lcat00` program; the information on AAs with unique expansions is obtained from the file `prevariants`, a non-release lexicon file. Except for the manual removal of a small number of anomalous AAs such as *Ms* with erroneous expansion *title for a woman* (a meta-description, not an expansion), the production of the AA files is automatically performed using scripts at /nfsvol/nls/specialist/module/db_access/data.00/NLSAbbrAcros/.

Summary information
- Input files
  `<lcat00 results>` (i.e., basic <Lex> information)
  `prevariants` (a non-release <Lex> file)
- Output files
  `nls_aa.txt`
  `nls_aau.txt`
- Time to create
  0.8 hours

2.3.5 Synonym (Syn) Files

A single file of synonyms together with their syntactic categories is produced by combining a file of Dorland synonyms (produced many years ago) and a small supplementary synonym file `SM.DB`, a non-release lexicon file. Categories not originally part of the Dorland synonym file were recently added in order that category information be available throughout MetaMap's variant generation process. Except for correcting errors in the `SM.DB` file, the synonym file is automatically created by scripts at /nfsvol/nls/specialist/module/db_access/data.00/Synonyms/.

Summary information
- Input files
  `<Dorland synonyms>`
  `SM.DB` (a non-release <Lex> file)
- Output files
  `syns.txt`
- Time to create
  0.3 hours

2.3.6 Variant (Var) Files

In order to reduce the time required to compute variants during MetaMap processing, four tables of variants are precomputed using both MetaMap's dynamic variant generation algorithm and the `lvg` program for generating lexical variants. The tables differ in whether they contain all AAs or only unique ones (i.e., AAs with a unique expansion in the data) and whether they contain all derivational variants or only adjective-noun ones. The universe of words whose variants are computed consists of all words from the SPECIALIST Lexicon, the previously computed AA and Syn files, and even the Metathesaurus, itself. (Because the latter are obtained from MWI files, there should be three versions of the variant files; but as with the MeSH files, only the most inclusive version is created.) Once the variants have been computed, the results are filtered so that only table entries with variants actually occurring in the Metathesaurus (the target of variant generation) are kept. This reduces table size and thereby lookup time dramatically. The complexity of the processing required to produce the Var files is second only to that for the MWI files. And the files are created by the following four steps.

- Compute all words in the knowledge sources, those from the Metathesaurus first (in order to keep track of how many words each knowledge source adds to the universe of words).
- Compute variants for all inflections in the lexicon. This is the step that uses MetaMap's dynamic variant generation algorithm. It is also where bad, rule-generated derivational variants are detected and saved for future filtering.
- Compute variants for words from knowledge sources other than the lexicon using `lvg`.
- Combine the previously computed variants and filter out variants not actually occurring in the Metathesaurus.

Most of the above processing is accomplished using scripts at /nfsvol/nls/specialist/metamap/tools/mm_variants/data.00/. Manual effort is only required for detecting bad derivational variants, and this process is now fairly easy given the stability of the lexicon.

Summary information

- Input files
  `all_words.txt` (an MWI file)
  `nls_aa.txt` (an AA file)
  `nls_aau.txt` (an AA file)
  `syns.txt` (a Syn file)
  `<lcat00 results>` (i.e., basic <Lex> information)
  `<meta words>` (from the relaxed data model)
- Output files
  `vars.txt`
  `varsan.txt`
  `varsanu.txt`
  `varsu.txt`
- Time to create
  12.6 hours

2.3.7 Creating the Actual Data Files

In order to make the files described in the previous sections available for MetaMap processing, they are loaded into relational tables using Berkeley DB. The DB files are created in two stages. The first stage involves all files except the Var files; indeed, some of the stage one files are used to create the Var files. The second stage includes the Var files and some additional files of chemicals for future use by MetaMap. Three versions of the DB files corresponding to the three MWI data models are created at /nfsvol/nls3aux/projects/DB/. Once the DB files have been created, they are accessed using the Prolog module `db_access` which, in turn, uses a C interface developed by Jim Mork.

Summary information

- Input files
  `all_words.txt`
  `all_words_counts.txt`
  `cui.concept.txt`
  `concept.cui.txt`
  `concept.st.txt`
  `cui.st.txt`
  `first_words.txt`
  `first_words_counts.txt`
  `first_words_of_one.txt`
  `first_words_of_two.txt`
  `sui.cui.txt`
  `sui.nmstr.str.txt`
  `mesh.mh.opt.txt`
  `mesh.tc.strict.txt`
  `mesh.tc.relaxed.txt`
  `meta.mesh.opt.txt`
  `meta.mesh.tc.opt.txt`
  `nls_aa.txt`
  `nls_aau.txt`
  `syns.txt`
  `vars.txt`
  `varsan.txt`
  `varsanu.txt`
  `varsu.txt`
  `stopwords.txt`
  `new_chem.txt`
- Output files (DB filenames or, equivalently, table names
  `all_words`
  `all_words_counts`
  `cuiconcept`
  `conceptcui`
  `conceptst`
  `cuist`
  `first_words`
  `first_words_counts`
  `first_words_of_one`
  `first_words_of_two`
  `sui_cui`
  `suistrings`
  `meshmh`
  `meshtcstrict`
  `meshtcrelaxed`
  `metamesh`
  `metameshtc`

```
nlsaa
nlsaau
syns
vars
varsan
varsanu
varsu
stpwrds
newchem
```

- Time to create
  3.7 hours (1.8 hours for the first stage and 1.9 hours for the second)

3. Summary

Several types of information are used by MetaMap and require updating whenever the underlying sources of that information change. The most common trigger for the update process is a new release of the UMLS Knowledge Sources (the SPECIALIST Lexicon, the UMLS Semantic Network, and the UMLS Metathesaurus). The update process can be described according to the different types of MetaMap data. That is what is done below where the data are updated in the order listed. (The one exception to this is that the DB files are computed in two stages, once before and once after computing the Var files.) Time estimates for each data type are given in brackets following the description of the data.

- Semantic Type Table (STT): A table giving the correspondence between semantic types and their abbreviations. [0.25 hours]
- Meta Word Index (MWI) Files: Index files providing access to UMLS strings from the words they contain. [3 days 4.5 hours, or 20.1 hours with parallelization of the most timeconsuming task]
- MeSH Treecode (MeSH) Files: Files defining the treecodes of MeSH headings (including pseudo-treecodes for chemicals, subheadings, and check tags). [3.4 hours]
- NLS Abbreviation/Acronym (AA) Files: A set of files containing abbreviations/acronyms and their expansions extracted from the SPECIALIST Lexicon. [0.8 hours]
- Synonym (Syn) Files: A supplementary knowledge base of synonyms. [0.3 hours]
- Variant (Var) Files: Precomputed files of words and those of their variants which occur in the Metathesaurus. [12.6 hours]
- DB Files: The Berkeley DB files corresponding to the above files. [3.7 hours]

The total amount of time to finish the entire update process is approximately 4 days 1.6 hours (or 1 day 17.2 hours with parallelization).

Because the annual release of MeSH occurs asynchronously with UMLS releases and because MetaMap needs to be updated in a timely manner following a MeSH release, we have begun experimenting with the process of updating a current UMLS release with a new MeSH release in order to update MetaMap's data using the same update process described above. The modified UMLS release will, of course, only be an approximation of the next official UMLS release; but it will meet the demand for keeping recommended indexing terms current with MeSH.

4. Appendix: Update following a MeSH release

The purpose of this appendix is to describe how to handle a MeSH release before the information becomes part of a UMLS release. This is necessary, for example, because the Index Section uses a new year's MeSH before it is available through the UMLS Knowledge Sources.

The MeSH Section provides several files which define the changes to MeSH from one year to the next. Florence Chang has written scripts to convert the change files into more manageable form, and I have added one more file to handle descriptor deletions more accurately. These files are located in /nfsvol/nls3aux/projects/ind/MeSHUpdate/ and its subdirectories.

Two programs, `mesh_update_prep` and `mesh_update` are used to update a file, `mrconso.eng`, which is the join of the English entries in MRCON with MRSO and forms the basis for computing MetaMap's word index files. `mesh_update_prep` converts the change files into instructions for updating `mrconso.eng`, and `mesh_update` actually performs the update.

The updated `mrconso.eng` file is then used to recompute the MWI and then the Var files. Some MeSH files should also be recomputed but are not since the change files do not contain the information (treecodes) needed to update them.

Summary information

- MeSH change files (also referred to as LTIMeSH files)
  `meshrdef.12502` (where 12502 denotes the version of the file)
  `meshrjoin.12502`
  `meshrrel.12502`
  `meshrsty.12501`
- Modified change files (produced by Florence's scripts)
  `mmi.chem.add`
  `mmi.chem.del`
  `mmi.desc.add`
  `mmi.desc.del` (my additional file)
  `mmi.desc.repl`
  `mmi.q.del`
- Instruction files for modifying `mrconso.eng`
  `instr.mmi.chem.add`
  `instr.mmi.chem.del`
  `instr.mmi.desc.add`
  `instr.mmi.desc.del`
  `instr.mmi.desc.repl`
  `instr.mmi.q.del`