MetaMap: Mapping Text to the UMLS Metathesaurus

Alan R. Aronson

July 14, 2006

The task of automatically determining the concepts referred to in text is a common one. It is a prerequisite to many applications including information retrieval, text mining, categorization and classification, text summarization, question answering, knowledge discovery and other natural language processing tasks. MetaMap was originally developed for use in information retrieval, i.e., for improving the retrieval of relevant MEDLINE® citations based on queries formulated in English. In this case the text can consist of not only user queries but also the titles and abstracts of MEDLINE citations; and the concepts to be found in the text are those of the UMLS® Metathesaurus®. The task becomes one of mapping text to the Metathesaurus. This report describes the development of MetaMap and provides guidance on its usage. Section 1 contains the results of manually examining the mapping problem for a small collection of utterances. It provides the basis for defining a strategy for automatically mapping to the Metathesaurus as outlined in Section 2. The automatic approach is characterized by determining how to map from the phrases discovered by the SPECIALIST parser to appropriate concepts in the Metathesaurus. The results of such a mapping can be used to normalize text so that each referenced concept is represented uniquely. Details of the MetaMap implementation are given in Section 3 through Section 6. Options defining MetaMap s behavior are described in Section 7; and additional options for SKR, a generalization of MetaMap, are defined in Section 8. Section 9 described MetaMap usage by specifying useful combinations of MetaMap options. Finally, Section 10 consists of several Appendices. It should be noted that the MetaMap algorithms are not specific to the biomedical domain and can be applied to any domain with adequate knowledge sources.

1. A Preliminary Examination of the Mapping Problem

In order to determine the scope of the problem of mapping text to the Metathesaurus, a set of 99 utterances (16 queries and 83 citation titles) was taken from the NLM Test Collection. The SPECIALIST parser was applied to the utterances producing 301 noun phrases. Each of the phrases was manually mapped to the 1992 version of the Metathesaurus and classified into one of four categories based on how well it maps to the Metathesaurus.[1] Membership in a category is determined by lexical properties of the mapping as defined below; and in each case inflectional and spelling variation are ignored:

Simple match the noun phrase maps exactly to a Metathesaurus string. For example, *intensive care unit* maps to Intensive Care Units.

1. All examples in Section 1 through Section 6 of this document are taken from the 1992 edition of the UMLS knowledge sources. Examples in Section 7 forward use the 2006AA edition.

Complex match there is a partitioning of the words in the noun phrase so that each element of the partition has a Simple match to the Metathesaurus. For example, *intensive care medicine* maps to the two Metathesaurus terms Intensive Care and Medicine

Partial match the noun phrase maps to a Metathesaurus string in such a way that at least one word of either the noun phrase or the Metathesaurus string (or both) does not participate in the mapping. Partial matches have the following variations:

Normal partial match The simplest type of partial match occurs when a Metathesaurus string maps part of the noun phrase without gaps in what it *does* map. For example, *liquid crystal thermography* maps to Thermography where the mapping does not involve *liquid crystal*. Similarly, *cochlear implant subjects* maps to Cochlear Implant where *subjects* is not involved. Normal partial matches provide good results for the part of the noun phrase involved.

Gapped partial match Gapped partial matches involve a gap either in the noun phrase or Metathesaurus string or both. For the mapping of *ambulatory monitoring* to AMBULATORY CARDIAC MONITORING, the gap CARDIAC occurs in the Metathesaurus string. For the mapping of *obstructive sleep apnea* to Obstructive Apnea, the gap *sleep* occurs in the noun phrase. And for the mapping of *continuous pump-driven hemofiltration* to Continuous Arteriovenous Hemofiltration, gaps occur in both. Gapped partial matches often provide better results than normal partial matches because of their greater matching involvement. However, when the gap occurs in the Metathesaurus string, the string tends to be too specific.

Overmatch An overmatch occurs when a match does not involve words at either or both ends of the Metathesaurus string. An overmatch is similar to a normal partial match except that part of the Metathesaurus string is uninvolved in the mapping. For example, the Metathesaurus string Postoperative Complications is an overmatch for *ocular complications*. The phrase *application* has many overmatches including Job Application, Heat/Cold Application and Medical Informatics Application. Overmatches almost always give poor results unless browsing is the object of the mapping.

No match no part of the noun phrase maps to any string in the Metathesaurus.[1]

The categories above are listed in order of the strength of the mapping, a simple match being the strongest. It should be emphasized, however, that the semantic or conceptual quality of the mapping varies widely. Even simple matches can map text to unrelated Metathesaurus terms. For example, the noun phrase *the numeric values* maps to the Metathesaurus concept Values with semantic type Qualitative Concept. In the Metathesaurus, Values is a term from psychology referring to social values; it is not a quantitative concept. Furthermore, even when the Metathesaurus contains the correct concept, that concept may be ambiguous. For example, the Metathesaurus contains two *ventilation* concepts, one related to air flow in buildings and the other related to respiration.

Some examples from the manual study illustrating the types of match just discussed are given below. In each case a phrase and one or more Metathesaurus terms are listed together with the type of match. Note that each phrase generally maps to more Metathesaurus terms than shown.

1. It is becoming very difficult to find such matches. For example, the phrase *improvement* had no mappings to the 1992 Metathesaurus, but maps to Improved and also to the eleven-word 1994 Metathesaurus concept Coreoplasty by photocoagulation, one or more sessions for improvement of vision.

computerized system > Computer Systems (simple) and Computerized Medical Record Systems (gapped partial);

indications > Indicators (simple);

ischaemic limbs > Ischemic and Limbs (complex);

diabetic foot > Diabetes and Foot (complex), DIABETIC FOOT CARE (overmatch), and Diabetic Foot (simple);

obstructive sleep apnea > Obstructive Apnea (gapped partial) and Sleep apnea (normal partial);

membrane plasma filtration > Membranes, Plasma, and Filtration (complex);

inferior vena caval stent filter > Inferior Vena Cava Filter and Stent (complex);

laboratory tests > laboratory Tests (simple) but also TEST[1] (normal partial);

cardiokymography > Kymography (simple); and

phrenic motoneurones > Motor Neurons (normal partial).

The results of manually mapping the 301 noun phrases to the Metathesaurus are summarized in Table 1.[2] [Note that 70 percent of the 113 partial matches involved the head of the noun phrase.]

Lexical Mapping Category	Count	Percent
Simple match	91	30%
Complex match	24	8%
Partial match	113	38%
No match	73	24%
Total	301	100%

Table 1. **Summary of Noun Phrase Mappings to the Metathesaurus**

2. The Basic Mapping Strategy

The experience gained from the manual mapping exercise described above led naturally to the following strategy for accomplishing the mapping automatically. Perform the following steps for each textual utterance:

1. Parse the text into noun phrases and perform the remaining steps for each phrase;[3]

1. In the Metathesaurus, TEST is the chemical Ethanesulfonic acid, 2-((2-hydroxy-1,1-bis(hydroxymethyl)ethyl)amino)-, mixt. with 2-amino-2-(hydroxymethyl)-1,3-propanediol.

2. The results shown were obtained using the 1992 version of the Metathesaurus; later versions would give better results due to increased domain coverage.

3. Parsing is accomplished using the SPECIALIST parser which produces a high-level syntactic analysis rather than a full syntactic analysis. The parser optionally uses the MedPost/SKR part of speech tagger which assigns syntactic labels to all textual items. The parser is very good at determining the simple noun phrases in text; and the errors it does make are normally inconsequential to MetaMap. The tagger also improves parsing results.

2. Generate the variants for the noun phrase where a variant essentially consists of one or more noun phrase words together with all of its spelling variants, abbreviations, acronyms, synonyms, inflectional and derivational variants, and meaningful combinations of these;
3. Form the *candidate set* of all Metathesaurus strings containing one of the variants;
4. For each candidate, compute the mapping from the noun phrase and calculate the strength of the mapping using an evaluation function. Order the candidates by mapping strength; and
5. Combine candidates involved with disjoint parts of the noun phrase, recompute the match strength based on the combined candidates, and select those having the highest score to form a set of best Metathesaurus mappings for the original noun phrase.

Descriptions of steps 2-5 of the mapping strategy are given in the next four sections.

3. Noun Phrase Variants

The Metathesaurus mapping algorithm begins by computing a set of variant generators for each noun phrase discovered by the parser. A variant generator is any *meaningful* subsequence of words in the phrase where a subsequence is meaningful if it is either a single word or occurs in the SPECIALIST lexicon. For example, the variant generators for the noun phrase *of liquid crystal thermography*[1] are *liquid crystal thermography*, *liquid crystal*, *liquid*, *crystal* and *thermography* (prepositions, determiners, conjunctions, auxiliaries, modals, pronouns and punctuation are ignored). Note the multi-word generators. A simpler example which will be used throughout the sequel is based on the noun phrase *ocular complications.*[2] Its generators are simply *ocular* and *complications*.

The approach taken in computing variants is a canonicalization approach. This simply means that a variant represents not only itself but all of its inflectional and spelling variants.[3] Collapsing inflectional and spelling variants results in significant computational savings. Variants are computed for each of the variant generators according to the scheme pictured in Figure 1. The computation for each generator proceeds as follows:

1. Compute all acronyms, abbreviations and synonyms of the generator. This results in the three sets Generator, Acronyms/Abbreviations, and Synonyms which are highlighted with boxes in Figure 1;
2. Augment the elements of the three sets by computing their derivational variants and the synonyms of the derivational variants;
3. For each member of the Acronyms/Abbreviations set, compute synonyms; and
4. For each member of the Synonyms set, compute acronyms/abbreviations.

The issue of whether to recursively generate variants of a given type is handled as follows:

1. A simplified syntactic analysis for *of liquid crystal thermography* is `[prep(of), head(liquid crystal thermography)]`.

2. A simplified syntactic analysis for *ocular complications* is `[mod(ocular), head(complications)]`.

3. A spelling variant of a word is just a variant having the same principal part as the word. For example, *haemorrhaged* is a spelling variant of *hemorrhaged*.

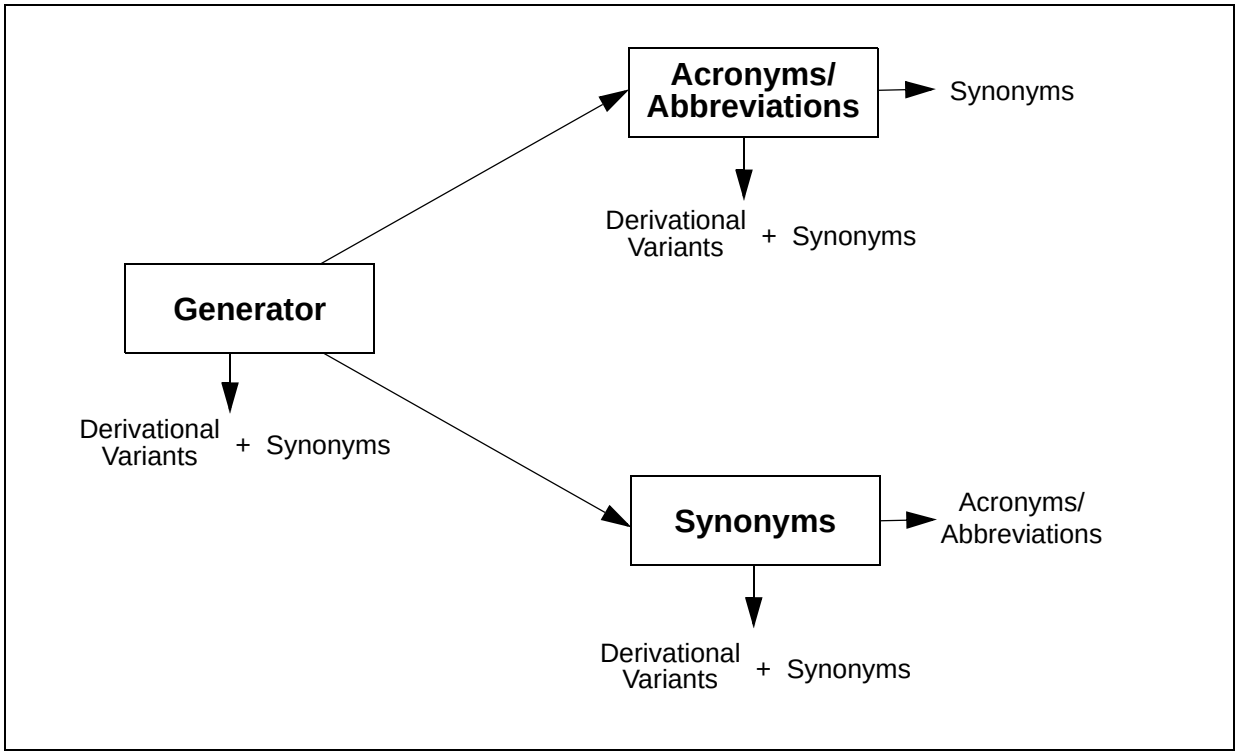

Figure 1. **Variant Generation**

Acronyms and abbreviations are not recursively generated since doing so almost always produces incorrect results. For example, the abbreviation *na* of *sodium* has expansions *nurse s aide* and *nuclear antigen* which are unrelated to *sodium*; and

Derivational variants and synonyms are recursively generated since this often produces meaningful variants.

The variants computed for the generator *ocular* are shown in Figure 2. Following each variant is its variant distance score, a rough measure of how much it varies from its generator (see Section 5) and the history of how it was computed. For example,

oculus (with variant distance 3 and history d) is simply a derivational variant of the generator *ocular*;

optical (with variant distance 7 and history ssd) is a derivational variant of a synonym (*optic*) of a synonym (*eye*) of *ocular*; and

vision (with variant distance 9 and history ssds) is a synonym of the derivational variant *optical* described above.

The variant generation algorithm described here is knowledge intensive. It uses the following knowledge sources:

the SPECIALIST lexicon and a table of canonical forms derived from it;

a SPECIALIST knowledge base of acronyms and abbreviations;

a SPECIALIST knowledge base containing rules of derivational morphology; and

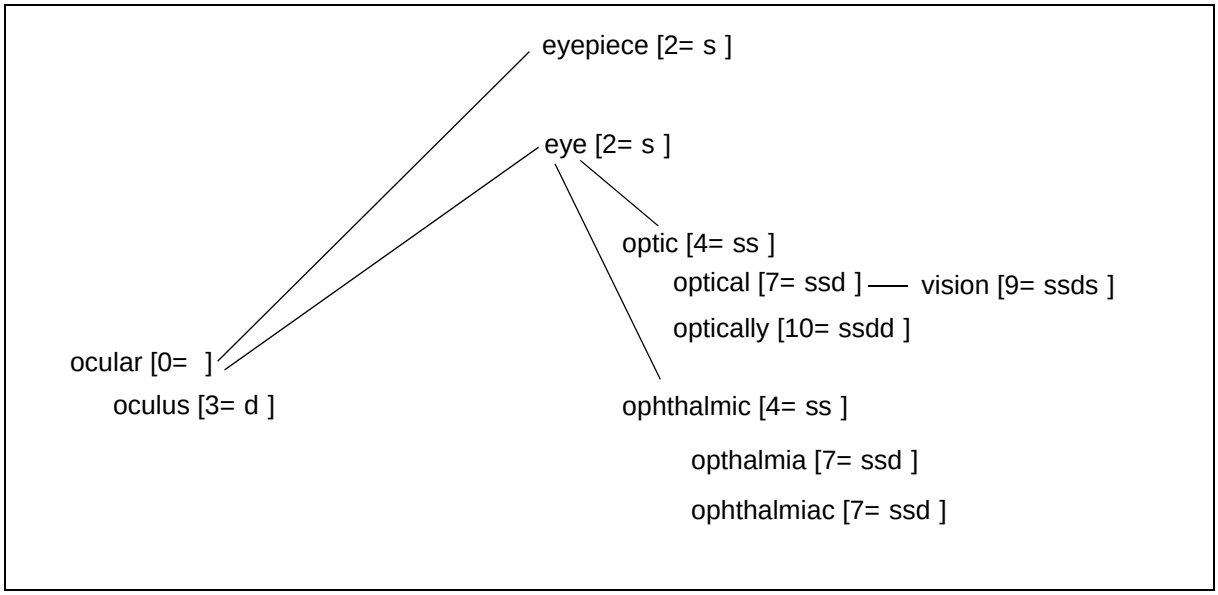

Figure 2. **Variants for the generator *ocular***

two knowledge bases of synonyms: one obtained by extracting synonyms from Dorland s Illustrated Medical Dictionary, and a supplemental synonym knowledge base developed for use with SPECIALIST.

4. Metathesaurus Candidates

The Metathesaurus candidates for a noun phrase consist of the set of all Metathesaurus strings containing at least one of the variants computed for the phrase.[1] The candidates are easily found by using a version of the Metathesaurus word index, an index from words to all Metathesaurus strings containing them. The Metathesaurus candidates for the noun phrase *ocular complications* are shown in Figure 3. When a string is not, itself, the preferred name for a Metathesaurus con-

861 Complications (Complication)

861 complications <1>

638 Eye

611 Optic (Optics)

588 Ophthalmia (Endophthalmitis)

Figure 3. **Metathesaurus Candidates for *ocular complications***

1. Here we are assuming normal MetaMap processing where correctness rather than breadth is most important. Since overmatches are rarely good matches, they are ignored for the rest of this discussion. An example of such an overmatch is Postoperative Complications for *ocular complications*. Similarly, Metathesaurus strings with gaps (e.g., Computerized Medical Record System for *computerized system*) are also ignored.

cept, the preferred name appears in parentheses following the string. The candidates are ordered according to the evaluation function described in the next section. The best candidates are Complications; and complications <1> both of which are simple matches involving the head of the phrase.[1] The remaining candidates are variants of *ocular* and are listed in order of similarity to *ocular.*

5. The Evaluation function

The evaluation function, or metric, computes a measure of the quality of the match between a phrase and a Metathesaurus candidate. For normal MetaMap operation the evaluation function is based on four components: *centrality, variation*, *coverage*, and *cohesiveness*. A normalized value between 0 (the weakest match) and 1 (the strongest match) is computed for each of these components. A weighted average is computed in which the coverage and cohesiveness components receive twice the weight as the centrality and variation components.[2] The result is then normalized to a value between 0 and 1,000, 0 indicating no match at all and 1,000 indicating an identical match (except for spelling variation, capitalization, NOS suffixes and inversions such as Cancer, Lung vs. Lung Cancer;). When MetaMap is set to ignore word order, the coverage component is replaced by an *involvement* component. Each of the evaluation function components is discussed below.

Centrality: The centrality value is simply 1 if the string involves the head of the phrase and 0 otherwise. For the noun phrase *ocular complications*, Complications has centrality value 1; and Eye has value 0.

Variation: The variation value estimates how much the variants in the Metathesaurus string differ from the corresponding words in the phrase. It is computed by first determining the *variation distance* for each variant in the Metathesaurus string. This distance is the sum of the distance values for each step taken during variant generation. The values for each step are listed in

Variant Type	Distance Value
spelling	0
inflectional	1
synonym or acronym/abbreviation	2
derivational	3

Table 2. **Variant Distances**

Table 2. The variation distance determines the variation value for the given variant according to the formula V=4/(D+4). As the total distance value, D, increases from its minimum value of 0, V decreases from a maximum value of 1 and is bounded below by 0. The final variation value

1. The symbol <1> denotes the first sense of an ambiguous Metathesaurus string. It is ignored during matching. Note that ambiguity designators such as <1> are now obsolete.

2. The actual weights used were determined empirically. Also, relative evaluation values were not particularly sensitive to small differences in the weights.

for the candidate is the average of the values for each of the variants. For *ocular complications*, Eye has a variant distance value of 2 and hence a variation value of 2/3 (4/(2+4)). Complications has a variant distance value of 0 and hence a variation value of 1.

Coverage: The coverage value indicates how much of the Metathesaurus string and the phrase are involved in the match. In order to compute the value, the number of words participating in the match is computed for both the Metathesaurus string and the phrase. These numbers are called the *Metathesaurus span* and *phrase span*, respectively. Note, however, that gaps are ignored.[1] The coverage value for the Metathesaurus string is the Metathesaurus span divided by the length of the string. Similarly, the coverage value for the phrase is the phrase span divided by the length of the phrase. The final coverage value is the weighted average of the values for the Metathesaurus string and the phrase where the Metathesaurus string is given twice the weight as the phrase. For *ocular complications* and either Eye or Complications, the Metathesaurus span and phrase span are both 1, and the coverage value is 5/6 (2/3*(1/1) + 1/3*(1/2)).

Cohesiveness: The cohesiveness value is similar to the coverage value but emphasizes the importance of connected components. A connected component is a maximal sequence of contiguous words participating in the match. The connected components for both the Metathesaurus string and the phrase are computed. This information is abstracted by noting the size of each component. This produces a set of connected component sizes for both the Metathesaurus string and the phrase. The cohesiveness value for the Metathesaurus string is the sum of the squares of the connected Metathesaurus string component sizes divided by the square of the length of the string. A similar cohesiveness value is computed for the phrase. The final cohesiveness value is the weighted average of the Metathesaurus string and phrase values where the Metathesaurus string is again given twice the weight as the phrase. For *ocular complications* and either Eye or Complications, the connected component sizes for both the Metathesaurus string and the phrase are {1} since one word from either the phrase or Metathesaurus string participates in the match. The cohesiveness value is 3/4 (2/3*(1/1) + 1/3(1/4)).

The final evaluation for Eye is the weighted average (0 + 2/3 + 2*(5/6) + 2*(3/4))/6 which normalizes to 638. Similarly, the final evaluation for Complications is (1 + 1 + 2*(5/6) + 2*(3/4))/6 which normalizes to 861.

Involvement: The involvement value is a replacement of the coverage value when word order is ignored. The strict word order implied by the matchmap is no longer followed. The involvement value for the phrase is the proportion of phrase words which *can* map to a Metathesaurus word whether or not they do according to the matchmap. For example, given the phrase *Advanced cancer of the lung* with words [advanced, cancer, lung] and the Metathesaurus string Lung Cancer with words [lung, cancer], the matchmap maps lung to lung, but does not map cancer because of word order. The phrase involvement value here is 2/3 as opposed to the coverage value of 1/3. Similarly, the involvement value for the Metathesaurus string is the proportion of words which *can* be mapped to from the phrase. For the current example, the Metathesaurus involvement value is 2/2 or 1 rather than 1/2 for coverage. Thus the final involvement value for this example is the weighted average (2/3 + 1)/2 or 0.83.[2]

1. This somewhat surprising scheme is illustrated by the following example. In computing the coverage for the phrase an inferior vena caval stent filter with the Metathesaurus string Inferior Vena Cava Filter, the phrase span is 5 even though stent does not participate in the match.

2. Note that the weighting of phrase involvement and Metathesaurus involvement is equal rather than the normal 1:2 ratio.

6. The Final Mapping

The final step in the mapping algorithm is straightforward. It consists of examining combinations of Metathesaurus candidates which participate in matches with disjoint parts of the noun phrase. The evaluation function is applied to the combined candidates, and the best ones form the final mapping result. The best mappings for *ocular complications* are shown in Figure 4. The central-

```
[mod([tokens([ocular]), metaconc([Eye])]),
 head([tokens([complications]), metaconc([Complica-
    tions])]),
 confid(861)]
  and
[mod([tokens([ocular]), metaconc([Eye]),
 head([tokens([complications]), metaconc([complica-
```

Figure 4. **The Best Metathesaurus Mappings for *ocular complications***

ity, variation, coverage and cohesiveness values for the mapping are 1, 2/3, 1 and 1, respectively. The final evaluation of the mapping is the weighted average (1 + 2/3 + 2*1 + 2*1)/6 which normalizes to 861 and is reported as a confidence value in the figure. (It is coincidence that this is the same value as the candidate score for Complications.)

7. MetaMap Options

MetaMap is highly configurable, and its behavior is controlled by option flags each of which has a short name (e.g., `-p`) and a long name (e.g., `--plain_syntax`). On the command line most of the options are toggle switches. Specifying a non-default option toggles it on; specifying a default option toggles it off. Options that take an argument are never defaults, so their presence always indicates that they are in effect. The options as of 2006 are described in the following sections. See Section 8 for a description of options specific to the related skr program. And see Section 10.1 for an example of MetaMap output.

7.1 The default options

MetaMap s default behavior consists of the following options (where default options are always shown in bold):

`-t` (**`--tag_text`**);

`-a` (**`--no_acros_abbrs`**) (new default as of 2006);

`-D` (**`--an_derivational_variants`**) (new default as of 2006);

`-l` (**`--stop_large_n`**);

`-p` (**`--plain_syntax`**);

-c (--**candidates**);

-s (**--semantic_types**); and

-m (**--mappings**).

-b (**--best_mappings_only**);

Each of these options is defined below.

7.2 Data options

Data options determine the underlying vocabularies and data model used by MetaMap.

`-V(--mm_data_version) <data version>` specifies which version of MetaMap s data files will be used for processing. The default data version is:

`normal`. It consists of all vocabularies in a given AA release of the Metathesaurus with the exception of the AMA vocabularies, CPT (Current Procedural Terminology) and CDT (Current Dental Terminology). Also excluded from the `normal` data version are CPT and CDT derivative vocabularies such as HCPCS (Healthcare Common Procedure Coding System) and MTHHH (Metathesaurus HCPCS Hierarchical Terms).

Two other data versions that are sometimes available are:

`level0` which consists of those UMLS vocabularies with the least restrictive source restriction level, namely level 0. Even level 0 vocabularies have some copyright restrictions, but they are less restrictive than those with restriction levels 1 through 3; and

`level0and4` which includes both level 0 and level 4 vocabularies. Currently SNOMEDCT and its derivatives are the only level 4 vocabularies in the Metathesaurus. (Note that, despite the numbering, level 4 is not as restrictive as levels 1 through 3, especially for USA users.)

`-A (--strict_model)`, `-B (--moderate_model)` and `-C (--relaxed_model)` determine which of the data models is used. If more than one model is specified, the strictest one is used; if none are specified then the strict model is used. See the report *Filtering the UMLS Metathesaurus for MetaMap* at the SKR website (http://skr.nlm.nih.gov/papers/index.shtml) for a description of the models.

7.3 Processing options

Processing options control MetaMap s internal behavior.

-t (**--tag_text**) specifies that the SPECIALIST parser will use the results of a tagger to assist in parsing. We previously used the Xerox Parc part of speech tagger but now use the MedPost/SKR tagger. The MedPost tagger was developed at NCBI specifically for tagging biomedical text; we modified it to use our part of speech tags.

`-L (--longest_lexicon_match)` causes lexical lookup to prefer matching as much text as possible to lexicon entries. This used to be the only form of lexical lookup, but it has been superseded by a shortest match algorithm. This is mainly due to the fact that the SPECIALIST lexicon is a syntactic lexicon; multi-word items contain no more information than their constituents which have their own lexicon entries. (This option, and the new shortest match lookup algorithm are new for 2006.)

`-P (--composite_phrases)` causes MetaMap to reconstruct longer, composite phrases from the simple phrases produced by the parser. A composite phrase is a simple phrase followed

by any prepositional phrase optionally followed by one or more *of* prepositional phrases. An example is pain on the left side of the chest which will map to Left sided chest pain rather than separate concepts as it would without the option. Note that `--composite_phrases` is experimental; it is currently both inefficient and not completely correct.

`-Q (--quick_composite_phrases)` is a version of `--composite_phrases` designed to overcome its inefficiency. It is both experimental and temporary.

`-a(--no_acros_abbrs)` prevents the use of any acronym/abbreviation variants which are the least reliable form of variation because it is normally the case that at most one of the expansions for an abbreviated form is correct. This option became a default in 2006.

`-u` (`--unique_acros_abbrs_only`) restricts the generation of acronym/abbreviation variants to those forms with unique expansions. This option produces better results than allowing all forms of acronym/abbreviation variants, but it is still better to prevent all such variants.

`-d (--no_derivational_variants)` prevents the use of any derivational variation in the computation of word variants. This option exists because derivational variants, as opposed to all other forms of variation, always involve a significant change in meaning.

`-D (--an_derivational_variants)` allows the use of derivational variation between adjectives and nouns, hence the name an_derivational variants. Adjective/noun derivational variants are generally the best of the derivational variants. This option became a default in 2006.

`-l` (**`--stop_large_n`**) prevents retrieval of Metathesaurus candidates for two-character words occurring in more than 2,000 Metathesaurus strings or one-character words occurring in more than 1,000 Metathesaurus strings. This option also prevents retrieval for words that can be a preposition, conjunction or determiner (according to the 2006 lexicon).

`-i (--ignore_word_order)` allows MetaMap to ignore the order of words in the phrases it processes. MetaMap was originally developed to process full text and consequently depended very strongly on normal English word order. When in effect, this option avoids the use of specialized word indexes used for efficient candidate retrieval, it ignores word order when matching phrase text to candidate words, and it replaces the normal coverage metric with an involvement metric for evaluating how well a candidate covers the words of a phrase. (See Section 5 for a description of the involvement metric.)

`-Y (--prefer_multiple_concepts)` causes MetaMap to score mappings with more concepts higher than those with fewer concepts. (It does so simply by inverting the normal cohesiveness value.) As a simplified example, with this option in effect, the input text lung cancer will score the mapping to the two concepts Lung and Cancer higher than the mapping to the single concept Lung Cancer . This option is useful for discovering higher-order relationships among concepts found in text (e.g., that Lung is the location of Cancer in the example).

`-y (--word_sense_disambiguation)` causes MetaMap to attempt to disambiguate among concepts scoring equally well in matching input text. The initial implementation of MetaMap word sense disambiguation uses a single method that chooses a concept (or concepts) having the most likely semantic type for the context in which the ambiguity arises. It is a new option for 2006.

`-z` (`--term_processing`) tells MetaMap to process terms rather than full text. When invoked, MetaMap treats each input as a single phrase (although the parser is still used to determine the head of that phrase). It also causes MetaMap to use the involvement metric rather than

coverage for evaluating Metathesaurus candidates (see Section 5). When used in conjunction with the `--allow_overmatches` and `--allow_concept_gaps` options, it constitutes MetaMap s *browse mode* for thorough searching of the Metathesaurus. In this case it is wise to also specify `-m (--mappings)` to toggle mapping construction off; otherwise, MetaMap spends too much time trying to combine the many candidates into final mappings.

`-o` (`--allow_overmatches`) causes MetaMap to retrieve Metathesaurus candidates which have words on one or both ends that do not match the text. For example, overmatches of medicine include Alternative Medicine , Medical Records and Nuclear medicine procedure, NOS . The use of `--allow_overmatches` greatly increases the number of candidates retrieved and is consequently much slower than MetaMap without overmatches. It is appropriate for browsing purposes.

`-g` (`--allow_concept_gaps`) causes MetaMap to retrieve Metathesaurus candidates with gaps (such as Unspecified childhood psychosis for *unspecified psychosis*). The use of this option does not appreciably affect MetaMap s performance. It is appropriate for browsing purposes.

`-8 (--dynamic_variant_generation)` forces MetaMap to generate variants dynamically rather than by looking up variants in a table. This option is normally used only for debugging purposes.

`-K (--ignore_stop_phrases)` simply prevents MetaMap from aborting its processing for commonly occurring phrases that are known to produce no mappings. This option is useful only for generating a new table of stop phrases after a change in UMLS data.

7.4 Output options

Output options control how MetaMap displays results.

`-q` (`--machine_output`) causes output to take the form of Prolog terms rather than human-readable form. The `--machine_output` option affects all other output options. An example of machine output is given in Section 10.2. For further information on machine output, including visually enhanced examples, see http://skr.nlm.nih.gov/Help/.

`-f (--fielded_output)` produces multi-line, tab-delimited output. Like machine output, it affects all other output options. It is described, with an example, in Section 10.3. For further information on fielded output, including visually enhanced examples, see http://skr.nlm.nih.gov/Help/.

`-T (--tagger_output)` displays the output of the MedPost/SKR tagger lining up input words on one line with their tags on a line below. See Section 10.1 for an example.

`-F (--formal_tagger_output)` displays the output of the MedPost/SKR tagger as a Prolog list of word/tag pairs.

`-p` (**`--plain_syntax`**) controls the output form of the results of the SPECIALIST parser. It simply outputs text without any syntactic information.

`-x (--syntax)` controls the output form of the results of the SPECIALIST parser. It outputs a Prolog term showing details of the syntactic processing. (See Section 10.1 for an example.)

`-H (--display_original_phrases)` displays the original (unexpanded) text of phrases rather than the expanded form that is produced when `--word_sense_disambigu-`

`ation` is in effect. Even if this option is used, it is the expanded form that determines MetaMap s output.

`-v` (`--variants`) displays the variants generated for each input word. (See Section 10.1 for an example.)

`-c` (**`--candidates`**) causes the list of Metathesaurus candidates to be displayed, best to worst, according to the MetaMap evaluation metric (see Section 5). Note that if a candidate is not the preferred name for a concept, the preferred name is displayed in parentheses immediately following the candidate. Displaying both the matching string and the preferred concept name when they differ is intended to avoid any confusion about why a concept appears on the candidate list.

`-O (--show_preferred_names_only)` prevents MetaMap from displaying both the matching string as well as the preferred name when it displays concepts.

`-r (--threshold) <integer>` restricts output to candidates with evaluation score of the threshold or better. Judicious use of this option can prevent MetaMap from making errors in situations where some input text has no close matches in the Metathesaurus. An appropriate threshold can usually be determined simply by examining MetaMap output for typical text in a given application.

`-X (--truncate_candidates_mappings)` first truncates the list of candidates to the 100 top-scoring ones before computing mappings and then truncates the list of mappings to the 8 top-scoring ones. This option can sometimes prevent a combinatorial explosion caused by computing a large number of mappings from a large number of candidates as is often encountered when using `--allow_overmatches`.

`-n` (`--number_the_candidates`) simply numbers the candidates in a displayed candidate list.

`-R (--restrict_to_sources) <list>` restricts output to those sources in the comma-separated `<list>` where spaces are not allowed. See Section 10.6 for the 2006AA sources.

`-e (--exclude_sources) <list>` excludes those sources in the comma-separated `<list>` where spaces are not allowed. See Section 10.6 for the 2006AA sources.

`-J (--restrict_to_sts) <list>` restricts output to those concepts with semantic types in the comma-separated `<list>` where spaces are not allowed. See Section 10.7 for the 2006AA semantic types.

`-k (--exclude_sts) <list>` excludes concepts not having a semantic type in the comma-separated `<list>` where spaces are not allowed. See Section 10.7 for the 2006AA semantic types.

`-I (--show_cuis)` shows the UMLS CUI for each concept displayed. See Section 10.1 for an example.

`-W (--preferred_name_sources)` lists all sources for the preferred names of displayed concepts. Note that this is just one of many possible choices for showing sources; showing all sources for any synonym in a concept, for example, would often produce very cluttered output.

`-s` (**`--semantic_types`**) causes the semantic types of Metathesaurus concepts to be displayed in square brackets for each concept in the candidate list or in a mapping.

-m (**--mappings**) causes mappings to be displayed. Most of the time it is useful to display both the candidate list and also the final mappings

-b (**--best_mappings_only**) restricts mappings displayed to only the top scoring ones. It is almost never useful to display all mappings because of their large number.

`-E (--indicate_citation_end)` This option causes an end-of transmission term to be written when processing of each unit of input is complete. It is only useful for processing using the Scheduler and only then with validated generic processing.

7.5 Miscellaneous options

`-h` (`--help`) displays MetaMap usage, i.e., the form of the command and a list of all options.

`-w` (`--warnings`) enables the display of conditions which are noteworthy if not erroneous. This option is normally used only for debugging purposes.

8. SKR Options

The SKR program is an extension of MetaMap that either summarizes MetaMap results for an entire document or uses MetaMap results as the basis for extracting higher-order concepts from text. All MetaMap options apply to SKR; the skr options that extend MetaMap s behavior are described here. Note, however, that SKR has no default options so that `metamap` (no options) is equivalent to `skr -taDlpcsmb`. One final thing to note is that sections of SKR output are bracketed for ease in parsing (e.g., lines >>>>> Phrase and <<<<< Phrase surround each phrase); see Section 10.1 for an example.

`-M (--mmi_output)` initiates the generation and display of MetaMap Indexing (MMI) output for a given processing unit (normally a citation). MMI output is a rudimentary form of indexing. It is far less accurate than that available from the NLM Indexing Initiative s Medical Text Indexer (MTI) which includes a second basic indexing methodology (based on PubMed s Related Articles algorithm) along with extensive postprocessing. See Section 10.4 for an example of both MMI output and fielded MMI output.

`-N (--fielded_mmi_output)` produces a detailed, pipe-delimited version of MetaMap Indexing (MMI) output. See Section 10.4 for an example of both `--mmi_output` and `--fielded_mmi_output`.

`-S (--semrep_output)` displays any higher-order concepts discovered by the SemRep (Semantic Representation) program. See Section 10.5 for examples.

9. Useful options and combinations thereof

MetaMap and SKR are used in various environments. The purpose of this section is to describe these environments and to specify the options that are appropriate for them. Additional information for some individual options that are useful in multiple environments is also provided. First, consider the following environments:

Semantic mode: xxx

Term processing mode: xxx

Browse mode: xxx

Second, the following options can be used with the above environments or in other situations:

additional options:

-y (--word_sense_disambiguation): xxx

-P (--composite_phrases) and -Q (--quick_composite_phrases): xxx

-i (include example of -i somewhere): xxx

-Y (--prefer_multiple_concepts): xxx

10. Appendices

All examples in the following appendices were produced using a preliminary version of the 2006 release of MetaMap based on the 2006AA UMLS knowledge sources.

10.1 Examples of MetaMap and SKR output

The following example was created by applying `metamap -TxvI` to the text Ocular complications of myasthenia gravis. The specified options generate tagger output, (full) syntax, variants and CUIs, respectively. The mappings generated in this example are shown in bold.

Processing 00000000.tx.1: Ocular complications of myasthenia gravis.

Ocular complications of myasthenia gravis .
adj noun prep noun noun pd

Phrase: "Ocular complications"

msu
mod([lexmatch([ocular]),inputmatch([Ocular]),tag(adj),tokens([ocular])])
head([lexmatch([complications]),inputmatch([complications]),tag(noun),tokens([complications])])

ocular [adj] variants (n=1):
ocular{[noun], 0=[]}

complications [noun] variants (n=2):
complication{[noun], 1="i"} complications{[noun], 0=[]}

Meta Candidates (3):
861 C0009566:Complications (Complication) [Pathologic Function]
861 C1171258:complications (Complication Aspects) [Pathologic Function]
694 C0015392:Ocular (Eye) [Body Part, Organ, or Organ Component]
Meta Mapping (888):
694 C0015392:Ocular (Eye) [Body Part, Organ, or Organ Component]
861 C0009566:Complications (Complication) [Pathologic Function]

Meta Mapping (888):
694 C0015392:Ocular (Eye) [Body Part, Organ, or Organ Component]
861 C1171258:complications (Complication Aspects) [Pathologic Function]

Phrase: "of myasthenia gravis."

msu
prep([lexmatch([of]),inputmatch([of]),tag(prep),tokens([of])])
head([lexmatch([myasthenia gravis]),inputmatch([myasthenia,gravis]),tag(noun),tokens([myasthenia,gravis])])
punc([inputmatch([.]),tokens([])])

myasthenia gravis [noun] variants (n=1):
myasthenia gravis{[noun], 0=[]}

myasthenia [noun] variants (n=3):
myasthenia{[noun], 0=[]} myasthenias{[noun], 1="i"} myasthenic{[adj], 3="d"}

gravis [adj] variants (n=1):
gravis{[adj], 0=[]}

Meta Candidates (3):
1000 C0026896:Myasthenia Gravis [Disease or Syndrome]
861 C0205082:Gravis (Severe) [Qualitative Concept]
861 C0947912:MYASTHENIA (Myasthenias) [Disease or Syndrome]
Meta Mapping (1000):
1000 C0026896:Myasthenia Gravis [Disease or Syndrome]

Note that the above results differ significantly from the examples presented in Section 3 and Section 4. This is due partly to changes in the UMLS knowledge sources from 1992 to 2006 and especially since MetaMap now has defaults **`-a (--no_acros_abbrs)`** and **`-D (--an_derivational_variants)`**. The results of processing the same text with `metamap -aDTxvI` (which turns off **`-aD`**) are given below. Note that all mappings are the same as before but that there are significant differences in intermediate results.

Processing 00000000.tx.1: Ocular complications of myasthenia gravis.

Ocular complications of myasthenia gravis .
adj noun prep noun noun pd

Phrase: "Ocular complications"

msu
mod([lexmatch([ocular]),inputmatch([Ocular]),tag(adj),tokens([ocular])])
head([lexmatch([complications]),inputmatch([complications]),tag(noun),tokens([complications])])

ocular [adj] variants (n=2):
ocular{[noun], 0=[]} ocularist{[noun], 3="d"}

complications [noun] variants (n=5):
complicate{[verb], 4="id"} complicated{[verb], 4="id"} complicating{[verb], 4="id"} complication{[noun], 1="i"} complications{[noun], 0=[]}

Meta Candidates (6):
  861 C0009566:Complications (Complication) [Pathologic Function]
  861 C1171258:complications (Complication Aspects) [Pathologic Function]
  777 C0231242:Complicated [Functional Concept]
  777 C1522701:Complicating [Functional Concept]
  694 C0015392:Ocular (Eye) [Body Part, Organ, or Organ Component]
  623 C1555959:Ocularist (Ocularist - NUCCProviderCodes) [Intellectual Product]
Meta Mapping (888):
  694 C0015392:Ocular (Eye) [Body Part, Organ, or Organ Component]
  861 C0009566:Complications (Complication) [Pathologic Function]
Meta Mapping (888):
  694 C0015392:Ocular (Eye) [Body Part, Organ, or Organ Component]
  861 C1171258:complications (Complication Aspects) [Pathologic Function]

Phrase: "of myasthenia gravis."

msu
 prep([lexmatch([of]),inputmatch([of]),tag(prep),tokens([of])])
 head([lexmatch([myasthenia gravis]),inputmatch([myasthenia,gravis]),tag(noun),tokens([myasthenia,gravis])])
 punc([inputmatch([.]),tokens([])])

myasthenia gravis [noun] variants (n=1):
myasthenia gravis{[noun], 0=[]}

myasthenia [noun] variants (n=3):
myasthenia{[noun], 0=[]} myasthenias{[noun], 1="i"} myasthenic{[adj], 3="d"}

gravis [adj] variants (n=1):
gravis{[adj], 0=[]}

Meta Candidates (3):
 1000 C0026896:Myasthenia Gravis [Disease or Syndrome]
  861 C0205082:Gravis (Severe) [Qualitative Concept]
  861 C0947912:MYASTHENIA (Myasthenias) [Disease or Syndrome]
Meta Mapping (1000):
 1000 C0026896:Myasthenia Gravis [Disease or Syndrome]

Similar SKR output, showing the bracketing of various sections of output and produced using the command `skr -taDlpcsmbN`, is:

Processing 00000000.tx.1: Ocular complications of myasthenia gravis.

Processing phrase: "Ocular complications"
>>>>> Phrase
ocular complications
<<<<< Phrase
>>>>> Candidates
Meta Candidates (3):
  861 Complications (Complication) [Pathologic Function]
  861 complications (Complication Aspects) [Pathologic Function]
  694 Ocular (Eye) [Body Part, Organ, or Organ Component]
<<<<< Candidates
>>>>> Mappings
Meta Mapping (888):
  694 Ocular (Eye) [Body Part, Organ, or Organ Component]
  861 Complications (Complication) [Pathologic Function]
Meta Mapping (888):
  694 Ocular (Eye) [Body Part, Organ, or Organ Component]
  861 complications (Complication Aspects) [Pathologic Function]
<<<<< Mappings

Processing phrase: "of myasthenia gravis."
>>>>> Phrase
myasthenia gravis
<<<<< Phrase
>>>>> Candidates
Meta Candidates (3):
 1000 Myasthenia Gravis [Disease or Syndrome]
  861 Gravis (Severe) [Qualitative Concept]
  861 MYASTHENIA (Myasthenias) [Disease or Syndrome]
<<<<< Candidates
>>>>> Mappings
Meta Mapping (1000):
 1000 Myasthenia Gravis [Disease or Syndrome]
<<<<< Mappings
>>>>> MMI
00000000|MM|15|Myasthenia Gravis|C0026896|[dsyn]|["Myasthenia Gravis"-tx-1-"myasthenia gravis"]|AB
00000000|MM|13|Eye|C0015392|[bpoc]|["Ocular"-tx-1-"Ocular"]|AB
00000000|MM|10|Complication Aspects|C1171258|[patf]|["complications"-tx-1-"complications"]|AB
00000000|MM|4|Complication|C0009566|[patf]|["Complications"-tx-1-"complications"]|AB
<<<<< MMI

10.2 An example of machine output

Continuing with the example in Section 10.1, the corresponding machine output (produced using `metamap -q`) is shown below. To improve the readability slightly, some lines have been broken in convenient places.

utterance(00000000.tx.1 ,"Ocular complications of myasthenia gravis.").

phrase(Ocular complications ,[mod([lexmatch([ocular]),inputmatch([Ocular]),tag(adj), tokens([ocular])]),head([lexmatch([complications]),inputmatch([complications]),tag(noun), tokens([complications])])]).

candidates([ev(-861, C0009566 , Complications , Complication ,[complications],[patf], [[[2,2],[1,1],0]],yes,no),ev(-861, C1171258 ,complications, Complication Aspects , [complications],[patf],[[[2,2],[1,1],0]],yes,no),ev(-694, C0015392 , Ocular , Eye , [ocular],[bpoc],[[[1,1],[1,1],0]],no,no)]).

mappings([map(-888,[ev(-694, C0015392 , Ocular , Eye ,[ocular],[bpoc],[[[1,1],[1,1],0]],no,no) ,ev(-861, C0009566 , Complications , Complication ,[complications],[patf], [[[2,2],[1,1],0]],yes,no)]),map(-888,[ev(-694, C0015392 , Ocular , Eye ,[ocular],[bpoc], [[[1,1],[1,1],0]],no,no),ev(-861, C1171258 ,complications, Complication Aspects , [complications],[patf],[[[2,2],[1,1],0]],yes,no)])]).

phrase(of myasthenia gravis. ,[prep([lexmatch([of]),inputmatch([of]),tag(prep),tokens([of])]), head([lexmatch([myasthenia gravis]),inputmatch([myasthenia,gravis]),tag(noun), tokens([myasthenia,gravis])]),punc([inputmatch([.]),tokens([])])]).

candidates([ev(-1000, C0026896 , Myasthenia Gravis , Myasthenia Gravis ,[myasthenia,gravis], [dsyn],[[[1,2],[1,2],0]],yes,no),ev(-861, C0205082 , Gravis , Severe ,[gravis],[qlco], [[[2,2],[1,1],0]],yes,no),ev(-861, C0947912 , MYASTHENIA , Myasthenias ,[myasthenia], [dsyn],[[[1,1],[1,1],0]],yes,no)]).

mappings([map(-1000,[ev(-1000, C0026896 , Myasthenia Gravis , Myasthenia Gravis , [myasthenia,gravis],[dsyn],[[[1,2],[1,2],0]],yes,no)])]).

EOU .

10.3 Fielded output

MetaMap s `-f (--fielded_output)` option produces multi-line, tab-delimited, fielded MetaMap output. The content for this option is the same as `-q (--machine_output)` and, like the machine output option, it ignores any other option intended to control human-readable output. The output is organized by utterances as determined by the SPECIALIST parser. Each utterance has a header and one or more instances of a phrase, its candidates and its mappings:

<label> <l> u <utterance>

<label> <l> p <phrase> <parsed-phrase>

<label> <l> c <j> <m> <score> <matching term> <matching concept>

<matching word list> <semantic type list> <matchmap> <involves head flag>

<is overmatch flag>

<label> <l> m <i> <n> <score>

<label> <l> mc <i> <n> <j> <m> <score> <matching term>

where

fields are separated by tab characters;

<l> is a line number for all lines for <label> beginning with 1;

u, p, c, m and mc are record types: utterance, phrase, candidate,

mapping, and mapping concept;

<parsed-phrase> and <matchmap> are currently shown as Prolog terms;

mappings are numbered <i> of <n>; and

candidates and mapping concepts are numbered <j> of <n>.

An example of fielded output (produced by `metamap -f`) using the same input as in Section 10.1 is given below. For readability, fields are shown pipe-separated with extra spaces rather than tab-delimited.

00000000.tx.1 | 1 | u | Ocular complications of myasthenia gravis.

00000000.tx.1 | 2 | p | Ocular complications | [mod([lexmatch([ocular]),inputmatch([Ocular]),tag(adj),tokens([ocular])]),head([lexmatch([complications]),inputmatch([complications]),tag(noun),tokens([complications])])]

00000000.tx.1 | 3 | c | 1 | 3 | 861 | Complications | Complication | complications | patf | [[[2,2],[1,1],0]] | yes | no

00000000.tx.1 | 4 | c | 2 | 3 | 861 | complications | Complication Aspects | complications | patf | [[[2,2],[1,1],0]] | yes | no

00000000.tx.1 | 5 | c | 3 | 3 | 694 | Ocular | Eye | ocular | bpoc | [[[1,1],[1,1],0]] | no | no

00000000.tx.1 | 6 | m | 1 | 2 | 888

00000000.tx.1 | 7 | mc | 1 | 2 | 1 | 2 | 694 | Ocular | Eye | ocular | bpoc | [[[1,1],[1,1],0]] | no | no

00000000.tx.1 | 8 | mc | 1 | 2 | 2 | 2 | 861 | Complications | Complication | complications | patf | [[[2,2],[1,1],0]] | yes | no

00000000.tx.1 | 9 | m | 2 | 2 | 888

00000000.tx.1 | 10 | mc | 2 | 2 | 1 | 2 | 694 | Ocular | Eye | ocular | bpoc | [[[1,1],[1,1],0]] | no | no

00000000.tx.1 | 11 | mc | 2 | 2 | 2 | 2 | 861 | complications | Complication Aspects | complications | patf | [[[2,2],[1,1],0]] | yes | no

00000000.tx.1 | 12 | p | of myasthenia gravis. | [prep([lexmatch([of]),inputmatch([of]),tag(prep),tokens([of])]),head([lexmatch([myasthenia gravis]),inputmatch([myasthenia,gravis]),tag(noun),tokens([myasthenia,gravis])]),punc([inputmatch([.]),tokens([])])]

00000000.tx.1 | 13 | c | 1 | 3 | 1000 | Myasthenia Gravis | Myasthenia Gravis | myasthenia,gravis | dsyn | [[[1,2],[1,2],0]] | yes | no

00000000.tx.1 | 14 | c | 2 | 3 | 861 | Gravis | Severe | gravis | qlco | [[[2,2],[1,1],0]] | yes | no

00000000.tx.1 | 15 | c | 3 | 3 | 861 | MYASTHENIA | Myasthenias | myasthenia | dsyn | [[[1,1],[1,1],0]] | yes | no

00000000.tx.1 | 16 | m | 1 | 1 | 1000

00000000.tx.1 | 17 | mc | 1 | 1 | 1 | 1 | 1000 | Myasthenia Gravis | Myasthenia Gravis | myasthenia,gravis | dsyn | [[[1,2],[1,2],0]] | yes | no

10.4 An example of MMI and fielded MMI output

The MMI output for the Section 10.1 example is given below followed by the fielded MMI output. They were created by the commands `skr -taDlpcsmbM` and `skr -taDlpcsmbN`, but only the actual MMI output is shown. For readability, pipe symbols in the fielded output are surrounded by spaces.

MMI output:
>>>>> MMI
MMI concepts for 00000000:
1.5 Myasthenia Gravis [Disease or Syndrome]
1.3 Eye [Body Part, Organ, or Organ Component]
1.0 Complication Aspects [Pathologic Function]
0.4 Complication [Pathologic Function]
<<<<< MMI

Fielded MMI output:
>>>>> MMI
00000000 | MM | 15 | Myasthenia Gravis | C0026896 | [dsyn] | ["Myasthenia Gravis"-tx-1-"myasthenia gravis"] | AB
00000000 | MM | 13 | Eye | C0015392 | [bpoc] | ["Ocular"-tx-1-"Ocular"] | AB
00000000 | MM | 10 | Complication Aspects | C1171258 | [patf] | ["complications"-tx-1-"complications"] | AB
00000000 | MM | 4 | Complication | C0009566 | [patf] | ["Complications"-tx-1-"complications"] | AB
<<<<< MMI

10.5 Examples of SemRep output

The SemRep output generated from aspirin for headache is
00000000.tx.1 | relation | Aspirin | orch,phsu | phsu | C0004057 | TREATS | Headache | sosy | sosy | C0018681

Similarly, the SemRep output for acidotic dogs is
00000000.tx.1 | relation | Acidosis | patf | patf | C0001122 | PROCESS_OF | Canis familiaris | mamm | mamm | C0012984

10.6 Codes for source vocabularies

The source vocabulary codes used by MetaMap are the versionless ones in the Metathesaurus. The following table lists them together with the number of strings in the 2006AA Metathesaurus (after some filtering):

AIR 630
ALT 4662
AOD 15901
BI 938
CCPSS 15266
CCS 1107
COSTAR 3074
CPM 3078
CSP 16532
CST 3856
CTCAE 5598
DDB 169
DSM3R 371
DSM4 452
DXP 7270
GO 18141
HHC 404
HL7V2.5 4909
HL7V3.0 7020
HUGO 20858
ICD10 11525
ICD10AE 1005
ICD10AM 24126
ICD10AMAE 2255
ICD9CM 19328
ICPC 748
ICPC2EENG 694
ICPC2ICD10ENG 38000
ICPC2P 6137
ICPCPAE 455
JABL 739
LCH 6587
LNC 61023
MCM 41
MDDB 9045
MDR 42493
MEDLINEPLUS 1270
MIM 240
MMSL 38035
MMX 8927
MSH 263631
MTH 82299
MTHFDA 15652
MTHHL7V2.5 1
MTHHL7V3.0 7
MTHICD9 15049
MTHICPC2EAE 27
MTHICPC2ICD107B 98
MTHICPC2ICD10AE 76
MTHMST 1636
MTHPDQ 48
MTHSCT 1963
NAN 235
NCBI 196206
NCI 39169
NCI-CTCAE 4504
NDDF 27519
NDFRT 31482
NEU 813
NIC 10006
NOC 2537
OMS 432
PCDS 2172
PDQ 8430
PNDS 268
PPAC 380
PSY 6737
QMR 940
RAM 210
RCD 186167
RCDAE 11149
RCDSA 821
RCDSY 9105
RXNORM 161620
SNM 35193
SNMI 112731
SNOMEDCT 297941
SPN 4807
SRC 261
ULT 84
UMD 9954
USPMG 1769
UWDA 61368
VANDF 19075
WHO 3169

10.7 Codes for semantic types

For the 2006AA UMLS knowledge sources, there are 136 semantic types. The semantic type codes used by MetaMap together with their full forms are listed in Table 3:

ST code	Semantic Type
acab	Acquired Abnormality
acty	Activity
aggp	Age Group
alga	Alga
amas	Amino Acid Sequence
aapp	Amino Acid, Peptide, or Protein
amph	Amphibian
anab	Anatomical Abnormality
anst	Anatomical Structure
anim	Animal
antb	Antibiotic
arch	Archaeon
bact	Bacterium
bhvr	Behavior
biof	Biologic Function
bacs	Biologically Active Substance
bmod	Biomedical Occupation or Discipline
bodm	Biomedical or Dental Material
bird	Bird
blor	Body Location or Region
bpoc	Body Part, Organ, or Organ Component
bsoj	Body Space or Junction
bdsu	Body Substance
bdsy	Body System
carb	Carbohydrate
crbs	Carbohydrate Sequence
cell	Cell
celc	Cell Component
celf	Cell Function
comd	Cell or Molecular Dysfunction
chem	Chemical
chvf	Chemical Viewed Functionally
chvs	Chemical Viewed Structurally
clas	Classification

Table 3. **Semantic Type Codes**

ST code	Semantic Type
clna	Clinical Attribute
clnd	Clinical Drug
cnce	Conceptual Entity
cgab	Congenital Abnormality
dora	Daily or Recreational Activity
diap	Diagnostic Procedure
dsyn	Disease or Syndrome
drdd	Drug Delivery Device
edac	Educational Activity
eico	Eicosanoid
elii	Element, Ion, or Isotope
emst	Embryonic Structure
enty	Entity
eehu	Environmental Effect of Humans
enzy	Enzyme
evnt	Event
emod	Experimental Model of Disease
famg	Family Group
fndg	Finding
fish	Fish
food	Food
ffas	Fully Formed Anatomical Structure
ftcn	Functional Concept
fngs	Fungus
gngp	Gene or Gene Product (pseudo ST for gene terminology)
gngm	Gene or Genome
genf	Genetic Function
geoa	Geographic Area
gora	Governmental or Regulatory Activity
grup	Group
grpa	Group Attribute
hops	Hazardous or Poisonous Substance
hlca	Health Care Activity
hcro	Health Care Related Organization
horm	Hormone
humn	Human
hcpp	Human-caused Phenomenon or Process
idcn	Idea or Concept
imft	Immunologic Factor

Table 3. **Semantic Type Codes**

ST code	Semantic Type
irda	Indicator, Reagent, or Diagnostic Aid
inbe	Individual Behavior
inpo	Injury or Poisoning
inch	Inorganic Chemical
inpr	Intellectual Product
invt	Invertebrate
lbpr	Laboratory Procedure
lbtr	Laboratory or Test Result
lang	Language
lipd	Lipid
mcha	Machine Activity
mamm	Mammal
mnob	Manufactured Object
medd	Medical Device
menp	Mental Process
mobd	Mental or Behavioral Dysfunction
mbrt	Molecular Biology Research Technique
moft	Molecular Function
mosq	Molecular Sequence
npop	Natural Phenomenon or Process
neop	Neoplastic Process
nsba	Neuroreactive Substance or Biogenic Amine
nnon	Nucleic Acid, Nucleoside, or Nucleotide
nusq	Nucleotide Sequence
ocdi	Occupation or Discipline
ocac	Occupational Activity
ortf	Organ or Tissue Function
orch	Organic Chemical
orgm	Organism
orga	Organism Attribute
orgf	Organism Function
orgt	Organization
opco	Organophosphorus Compound
patf	Pathologic Function
podg	Patient or Disabled Group
phsu	Pharmacologic Substance
phpr	Phenomenon or Process
phob	Physical Object
phsf	Physiologic Function

Table 3. **Semantic Type Codes**

ST code	Semantic Type
plnt	Plant
popg	Population Group
pros	Professional Society
prog	Professional or Occupational Group
qlco	Qualitative Concept
qnco	Quantitative Concept
rcpt	Receptor
rnlw	Regulation or Law
rept	Reptile
resa	Research Activity
resd	Research Device
rich	Rickettsia or Chlamydia
shro	Self-help or Relief Organization
sosy	Sign or Symptom
socb	Social Behavior
spco	Spatial Concept
strd	Steroid
sbst	Substance
tmco	Temporal Concept
topp	Therapeutic or Preventive Procedure
tisu	Tissue
vtbt	Vertebrate
virs	Virus
vita	Vitamin

Table 3. **Semantic Type Codes**