MetaMap Candidate Retrieval

Alan R. Aronson

July 13, 2001

1. Overview

This paper describes the methodology used by MetaMap to retrieve strings from the UMLS Metathesaurus as candidates for text mappings. Candidate retrieval occurs after parsing and variant generation. At that time the SPECIALIST parser has segmented incoming text into phrases each of which is analyzed independently. And variants have been computed for the words in a given phrase. Finding all Metathesaurus strings with at least one phrase variant in an efficient manner is the goal of candidate retrieval. Considerations such as the degree of focus of the search (browsing vs. semantic mode) and whether or not to respect word order affect the methods used and therefore the efficiency of the process.

2. Candidate Retrieval

The process for retrieving Metathesaurus candidates is described by defining the input to the process, describing the algorithm, and indicating how several MetaMap options affect the process. Throughout the descriptions, a text fragment *The effect of obstructive sleep apnea* is used for illustration.

2.1 Input

Input to candidate retrieval consists of a phrase and its variants. The text fragment mentioned above has two phrases, *The effect* and *of obstructive sleep apnea*. The second phrase has parse shown in Figure 1 consisting of two syntactic elements, a preposition and a head, each of which

```
[prep([lexmatch([of]), inputmatch([of]), tag(prep), tokens([of])]),
 head([lexmatch([obstructive sleep apnea]),
       inputmatch([obstructive,sleep,apnea]), tag(adj),
       tokens([obstructive,sleep,apnea])])
]
```

Figure 1. The parse for the phrase *of obstructive sleep apnea*

has information about a matching lexical entry, matching input text, the tag assigned by the tagger, and the individual text tokens for the element. Note that *obstructive sleep apnea* is a single, multi-word lexical entry and that the (erroneous) tag of adj for it is obtained from the tag for *obstructive*. (This will be corrected in the near future.)

Before computing variants for a phrase, it is filtered by removing the following syntactic elements: aux, compl, conj, det, modal, prep, pron, punc. These elements do not normally contribute to the mapping process. In the case of the phrase *of obstructive sleep apnea*, *of* is filtered out. Variants are now computed starting with the variant generators for the phrase. Variant generators are any multi-word sequence of words which occurs in the lexicon and any single word. The variant generators for our example phrase are the multi-word items *obstructive sleep apnea* and *sleep apnea* and the single words *obstructive*, *sleep* and *apnea*. The variants computed from these generators are shown in Figure 2. They are organized by generator and part of speech so that, for

```
obstructive sleep apnea [noun] variants (n=5):
obstructive sleep apnea{[noun], 0=[]} obstructive sleep apneae{[noun],
     1=“i”} obstructive sleep apneas{[noun], 1=“i”} osa{[noun], 2= “a”}
     osa’s{[noun], 3=“ai”}

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

sleep apnea [noun] variants (n=2):
sleep apnea{[noun], 0=[]} sleep apnoea{[noun], 0=“p”}

sleep [verb] variants (n=5):
sleep{[verb], 0=[]} sleeper{[noun], 3=“d”} sleepers{[noun], 4= “di”} sleep-
     ing{[verb], 1=“i”} sleeps{[verb], 1=“i”}

sleep [noun] variants (n=5):
hypnic{[adj], 2=“s”} sleep{[noun], 0=[]} sleeplessness{[noun], 3=“d”}
     sleepy{[adj], 3=“d”} somnus{[noun], 2=“s”}

apnea [noun] variants (n=3):
apnea{[noun], 0=[]} apneas{[noun], 1=“i”} apnoea{[noun], 0=“p”}
```

Figure 2. The variants for the phrase *of obstructive sleep apnea*

example, there are both verb and noun variants for *sleep*.[1] In the figure, each variant is listed with its part of speech and its derivational score and history, i.e. how it was derived from its generator. Each step in a variant’s history is either a spelling variant (history `“p”` and score 0), an inflection (history `“i”` and score 1), a synonym (history `“s”` and score 2), an acronym/abbreviation (history `“a”` and score 2), an expansion of an acronym/abbreviation (history `“e”` for score 2), or a derivational variant (history `“d”` and score 3). A variant’s total derivational score is the sum of the scores for each step in the history. Note that the null history, representing the variant itself, is denoted []. The actual representation of variants uses `v/6` terms with arguments the variant, itself, its part of

1. Note that if the category for *sleep* were known to be one of verb or noun, only those variants would be used. In this case, *sleep* inherits the adj category from *obstructive sleep apnea*. Since *sleep* has no adj category, all of its categories are used.

speech, its derivational score and history, its base forms, and its distance from the right end of the phrase. Knowing the distance from the right end of the phrase allows the use of smaller indexes for searching, at least when overmatches are not allowed. For instance, if the distance is 1, then the table `first_words_of_one`, which contains only single-word concepts, is used to search for matching Metathesaurus strings. Examples of the complete representation for variants are

`v(apneas,[noun],1,“i”,[apnea,apnoea],1)`,
`v(obstructive sleep apneas,[noun],1,“i”,[obstructive sleep apnea],3)`,and
`v(sleepers,[noun],4,“di”,[sleeper],2)`.

In order to efficiently access variant information during the evaluation process, the list is reorganized into an AVL tree according to the first word of the variants. This is shown in Figure 3. Each

```
apnea
 vinfo: apnea, [3,3], yes, apnea, [apnea]
apneas
 vinfo: apnea, [3,3], yes, apneas, [apneas]
apnoea
 vinfo: apnea, [3,3], yes, apnoea, [apnoea]
hypnic
 vinfo: sleep, [2,2], yes, hypnic, [hypnic]
obstructive
 vinfo: obstructive, [1,1], yes, obstructive, [obstructive]
 vinfo: obstructive sleep apnea, [1,3], yes, obstructive sleep apneas,
          [obstructive,sleep,apneas]
 vinfo: obstructive sleep apnea, [1,3], yes, obstructive sleep apneae,
          [obstructive,sleep,apneae]
 vinfo: obstructive sleep apnea, [1,3], yes, obstructive sleep apnea,
          [obstructive,sleep,apnea]
osa
 vinfo: obstructive sleep apnea, [1,3], yes, osa’s, [osa]
 vinfo: obstructive sleep apnea, [1,3], yes, osa, [osa]
sleep
 vinfo: sleep, [2,2], yes, sleep, [sleep]
 vinfo: sleep, [2,2], yes, sleep, [sleep]
 vinfo: sleep apnea, [2,3], yes, sleep apnoea, [sleep,apnoea]
 vinfo: sleep apnea, [2,3], yes, sleep apnea, [sleep,apnea]
sleeper
 vinfo: sleep, [2,2], yes, sleeper, [sleeper]
sleepers
 vinfo: sleep, [2,2], yes, sleepers, [sleepers]
sleeping
 vinfo: sleep, [2,2], yes, sleeping, [sleeping]
sleeplessness
 vinfo: sleep, [2,2], yes, sleeplessness, [sleeplessness]
sleeps
 vinfo: sleep, [2,2], yes, sleeps, [sleeps]
sleepy
 vinfo: sleep, [2,2], yes, sleepy, [sleepy]
somnus
 vinfo: sleep, [2,2], yes, somnus, [somnus]
```

Figure 3. Variants for the phrase *of obstructive sleep apnea* organized by first word

variant is represented by a `vinfo/5` predicate indicating the variant's generator, the position of the generator in the phrase, whether the generator involves the head of the phrase,[1] the variant, itself, and a list of its words. As in Figure 2, the basic information about a generator or variant is actually represented by a `v/6` term.

2.2 The Algorithm

Given the variants for some input text, retrieving Metathesaurus strings is fairly straightforward. For each variant of each generator, compute the candidate Metathesaurus strings by performing the steps:

- Extract the variant and its distance from the right of the text from its full representation (the first and last arguments of `v/6`). This is the only information needed to retrieve strings containing the variant;
- If the `stop_large_n` option is in effect, stop out the following variants:
  prepositions, conjunctions and determiners as specified by
  `nls_strings:prep_conj_det_atom/1`, and
  single-character variants with more than 2,000 occurrences and two-character variants with more than 1,000 occurrences (according to the `first_words_counts` table) where the occurrence values are subject to change as the Metathesaurus grows;
- Retrieve strings from the appropriate table:
  if the `allow_overmatches` option is in effect, use `all_words`,
  otherwise, if the `allow_concept_gaps` option is in effect, use `first_words`,
  otherwise, if the distance from the right is 1, use `first_words_of_one`,
  otherwise, if the distance from the right is 2, use `first_words_of_two`,
  otherwise, use `first_words`;
- Filter the results to those for which the entire variant occurs in the string (where words used for comparison of both multi-word variants and Metathesaurus strings are determined by the MetaMap tokenization regime encoded in `metamap_tokenization:tokenize_text_mm/2`):
  if the `all_words` table was used to retrieve the string, then the variant words must only be a subsequence of the string words,
  otherwise, the variant words must be a prefix of the string words;

Finally, candidates for all variants are sorted (to remove duplicates). Note that the actual form of each candidate is a usc/3 term with arguments the list of words (determined by MetaMap tokenization) for the string, the string, itself, and the string's concept. For example, the candidates for the variant *osa* (which has distance from the right of 3) are shown in Figure 4. Note that normally the string and concept are the same, but in the last case the string 'OSA - Obstructive sleep apnoea' has concept 'Sleep Apnea, Obstructive'.

Because a given variant can appear more than once in the list of variants for all generators (e.g., *sleep* in our example), some mechanism (e.g., caching) can be used to avoid duplicating effort

1. It is somewhat surprising that all variants involve the head of the phrase, but this is because the head of the phrase consists of all three words *obstructive*, *sleep* and *apnea*. And this is due to the fact that the lexicon has a single entry for *obstructive sleep apnea* in addition to entries for each individual word. It will soon be the case that such multi-word lexicon entries will be ignored for MetaMap processing. When this happens the head will be *apnea*, and variants of *obstructive* and *sleep* will no longer be said to involve the head.

```
usc([osa,antigen],OSA antigen,OSA antigen)

usc([osa,gene,product],osa gene product,osa gene product)

usc([osa,protein],osa protein,osa protein)

usc([osa,obstructive,sleep,apnea],OSA - Obstructive sleep apnea,Sleep
    Apnea, Obstructive)

usc([osa,obstructive,sleep,apnoea],OSA - Obstructive sleep apnoea,Sleep
    Apnea, Obstructive)
```

Figure 4. Candidate Metathesaurus strings for the variant *osa*

2.3 Options affecting the algorithm

The following options have an effect on the candidate retrieval process:

- `-o --allow_overmatches` and `-g --allow_concept_gaps`: these options affect which table is used to do actual retrieval of Metathesaurus strings;
- `-l --stop_large_n`: this option affects whether short variants or some content free words occurring in many Metathesaurus strings are ignored;
- `-z --term_processing`, `-i --ignore_word_order`, `-Y --prefer_multiple concepts`, and `-P --composite_phrases`: these options have an indirect effect in that they affect either the parsing that precedes candidate retrieval, the evaluation that follows retrieval, or both.