MetaMap Evaluation

Alan R. Aronson

May 8, 2001

1. Overview

The evaluation of Metathesaurus strings (or *candidates*), against the input text that caused them to be retrieved by MetaMap is described in this paper.[1] This evaluation occurs after the text has been parsed into manageable pieces called *phrases*, the variants for a phrase have been computed, and the Metathesaurus candidates containing at least one phrase variant have been retrieved. The purpose of evaluation is to determine how well each candidate matches the text.

2. Candidate Evaluation

The candidate evaluation process is described in the following steps:

- describing the input to the process and how it is filtered;
- defining the evaluation function;
- extending that function to a complete mapping of the text;
- summarizing the algorithm; and
- indicating how several MetaMap options affect the process.

Throughout the descriptions, input phrases such as *of obstructive sleep apnea* are used for illustration. For normal MetaMap processing, the phrase is filtered by removing non-content words such as prepositions, determiners, etc. In the above case, the filtered phrase is *obstructive sleep apnea*.

1. Much of the material here is derived from the paper *MetaMap: Mapping Text to the UMLS Metathesaurus*, 1996.

2.1 Input

Input to the evaluation process consists of a phrase, its variants, and Metathesaurus candidates. The variants for the phrase *of obstructive sleep apnea* are shown in Figure 1 organized according

```
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 1. Variants for the phrase *of obstructive sleep apnea* organized by first word

to the first word of the variants. Each 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, the variant, itself, and a list of its words. The basic information about a generator or variant is actually represented by a `v/6` term which has arguments the variant, itself, its part of speech, its variant distance score and history, its base forms, and its distance from the right end of the phrase. 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)`.

Note that *sleepers* is an inflection of a derivational variant of *sleep*, hence its history `“di”`; and its distance from *sleep* is determined by the values shown in Table 1. Here, the single-letter codes for

Variant Type	Distance Value
spelling (`p`)	0
inflectional (`i`)	1
synonym (`s`) or acronym/abbreviation (`a`, `e`)	2
derivational (`d`)	3

Table 1. Variant Distances

a single variation step are obvious except to note that the transformation from a complete form to an acronym or abbreviation is denoted `“a”`, and the transformation from an acronym/abbreviation to its expanded form is denoted `“e”`. Since *sleepers* is one derivational step and one inflectional step from *sleep*, its total distance is 4, the sum of 3 and 1.

130 Metathesaurus strings are retrieved for the phrase *of obstructive sleep apnea*. Each candidate is represented by a `usc/3` term with arguments the list of words (determined by MetaMap tokenization) for the string, the string, itself, and the string’s Metathesaurus concept. Some of the candidates are listed here according to the variant which retrieved them:

- **obstructive sleep apnea** (2 candidates)

```
usc([obstructive,sleep,apnea],Sleep Apnea, Obstructive,Sleep Apnea, Obstruc-
     tive)
usc([obstructive,sleep,apnea,syndrome],Syndrome, Sleep Apnea, Obstruc-
     tive,Sleep Apnea, Obstructive)
```

- **obstructive sleep apneas** (1 candidate)

```
usc([obstructive,sleep,apneas],Apneas, Obstructive Sleep,Sleep Apnea,
     Obstructive)
```

- **osa** (5 candidates)

```
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)
```

- **obstructive** (49 candidates)

```
usc([obstructive,hypertrophic,cardiomyopathy],Obstructive hypertrophic car-
     diomyopathy,Cardiomyopathy, Hypertrophic Obstructive)
```

```
usc([obstructive,hyperbilirubinemia],Obstructive hyperbilirubinemia,Jaun-
     dice, Obstructive)
usc([obstructive,liver,cirrhoses],Cirrhoses, Obstructive Liver,Liver Cirrho-
     sis, Obstructive)
usc([obstructive,lung,disease],Disease, Obstructive Lung,Chronic Obstructive
     Airway Disease)
usc([obstructive,jaundice],Jaundice, Obstructive,Jaundice, Obstructive)
...
usc([obstructive],Obstructive,Obstructed)
...
usc([obstructive,symptom],OBSTRUCTIVE SYMPTOM,OBSTRUCTIVE SYMPTOM)
usc([obstructive,sleep,apneas],Apneas, Obstructive Sleep,Sleep Apnea,
     Obstructive)
usc([obstructive,sleep,apnea],Sleep Apnea, Obstructive,Sleep Apnea, Obstruc-
     tive)
usc([obstructive,sleep,apnea,syndrome],Syndrome, Sleep Apnea, Obstruc-
     tive,Sleep Apnea, Obstructive)
usc([obstructive,hydrocephalus],Obstructive Hydrocephalus,Obstructive Hydro-
     cephalus)
usc([obstructive,genitourinary,defect],Obstructive genitourinary
     defect,Obstructive genitourinary defect)
```

- **sleep apnea** (1 candidate)
  ```
  usc([sleep,apnea],Apnea, Sleep,Sleep Apnea Syndromes)
  ```
- **sleep apnoea** (1 candidate)
  ```
  usc([sleep,apnoea],Sleep apnoea <1>,Sleep Apnea Syndromes)
  ```
- **sleep** (58 candidates)
  ```
  usc([sleep,walking],Sleep Walking,Somnambulism)
  usc([sleep,apnea],Apnea, Sleep,Sleep Apnea Syndromes)
  usc([sleep,apneas],Apneas, Sleep,Sleep Apnea Syndromes)
  usc([sleep],Sleep,Sleep)
  usc([sleep,deprivation],Sleep Deprivation,Sleep Deprivation)
  ...
  usc([sleep,analyzers],Sleep Analyzers,Polysomnography Analyzers, Computer-
       ized)
  usc([sleep,recorders],Sleep Recorders,Polysomnographs)
  usc([sleep],Sleep <3>,Sleep brand of diphenhydramine hydrochloride)
  ```
- **sleeper** (2 candidates)
  ```
  usc([sleeper,peptide],sleeper peptide,sleeper peptide)
  usc([sleeper,short],Short, sleeper,“Short-sleeper”)
  ```
- **sleeping** (6 candidates)
  ```
  usc([sleeping],Sleeping,Asleep)
  usc([sleeping,sickness],Sleeping sickness, NOS,Trypanosomiasis, African)
  usc([sleeping,excessive],Sleeping Excessive,Disorders of Excessive Somno-
       lence)
  usc([sleeping,out],Sleeping out,Sleeping out)
  usc([sleeping,rough],Sleeping rough,Sleeping out)
  usc([sleeping,pill],sleeping pill,sleeping pill)
  ```

- **sleeplessness** (1 candidate)
  `usc([sleeplessness],Sleeplessness,Sleep Initiation and Maintenance Disorders)`
- **sleepy** (1 candidate)
  `usc([sleepy],Sleepy,Sleepy)`
- **apnea** (1 candidate)
  `usc([apnea],Apnea,Apnea)`
- **apneas** (1 candidate)
  `usc([apneas],Apneas,Apnea)`
- **apnoea** (1 candidate)
  `usc([apnoea],Apnoea,Apnea)`

During normal MetaMap processing (called *semantic* mode processing), Metathesaurus candidates with extraneous words are filtered out because they are plentiful but produce mappings which are inferior to those associated with candidates with fewer words. Specifically *overmatches* and *concept gaps* are not allowed. An overmatch is a candidate with non-matching words on one end of the candidate. Examples of overmatches for the phrase *of obstructive sleep apnea*, where the extraneous words are italicized, include '*Central* sleep apnea', 'OSA - *Obstructive sleep apnea*',[1] 'Obstructive *appendicitis*', 'Sleep *Walking*', 'Sleeping *sickness*, NOS',[2] and '*Primary* sleep apnea *of newborn*'. Concept gaps are like overmatches except that the extraneous words occur in the middle of the candidate. An example of a concept gap is 'Computerized *Medical Record* System' for the text *computer system*. The extraneous words are again italicized.

2.2 Evaluating Candidates

The evaluation function 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 of the components is computed in which coverage and cohesiveness receive twice the weight as centrality and variation. The actual weights used were determined empirically; relative evaluation values were not particularly sensitive to differences in the weights. The result is normalized to a value between 0 and 1,000, 0 indicating no match at all and 1,000 indicating a perfect match (modulo capitalization and uninversion). Throughout the description of evaluation, the candidates 'Obstructive sleep apnoea', 'Sleep Apneas'[3] and 'Sleepy' will be used to illustrate.

In preparation for computing the evaluation function for a given phrase and candidate, a *matchmap* containing information about the mapping from phrase text to Metathesaurus candidate is computed. It consists of a list of matchmap elements each of which has a phrase component, a

1. Alternatively, this example could be interpreted as a left overmatch: *OSA* - Obstructive sleep apnea; in either case the hyphen is ignored.

2. NOS is ignored by MetaMap

3. This analysis also applies to the Metathesaurus string 'Apneas, Sleep' because MetaMap only sees strings uninverted.

concept component and a variation level. The phrase and concept components indicate a word span of matching words in the phrase and candidate, respectively; and the variation level is just the variant score for the words in the candidate. Example matchmaps for the phrase *of obstructive sleep apnea* are shown in Table 2.

Candidate	Matchmap
Obstructive sleep apnoea	[[[1,1],[1,1],0],[[2,3],[2,3],0]]
Sleep Apneas	[[[2,2],[1,1],0],[[3,3],[2,2],1]]
Sleepy	[[[2,2],[1,1],3]]

Table 2. Matchmaps for selected Metathesaurus candidates

In addition to computing which words of a phrase match which words of a candidate, it is important to know how many pieces were used to accomplish the match. The matchmap is used to derive the set of connected component sizes for both phrase and candidate. (See the definition of cohesiveness below for more information.) Examples of connected component sizes are given in Table 3. Each of these examples indicates that the match consists of one maximal connected component of the same size for both the phrase and the candidate. A more interesting example would be if we had the text *of sleep obstructive apnea* and the candidate 'Sleep Apneas', the connected component sizes would be [[1,1],[2]] indicating the phrase has two components of size 1 (*sleep* and *apnea*) and the candidate has a single component of size two (*sleep apneas*).

Candidate	Connected Component Sizes
Obstructive sleep apnoea	[[3],[3]]
Sleep Apneas	[[2],[2]]
Sleepy	[[1],[1]]

Table 3. Connected component sizes for selected Metathesaurus candidates

The remainder of this section consists of definitions for the four components of the evaluation function plus an additional component which is used when word order does not matter.

- Centrality: The centrality value is simply 1 if the string involves the head of the phrase and 0 otherwise. Centrality values for our examples are displayed in Table 4. Note that 'Sleepy'

Candidate	Centrality
Obstructive sleep apnoea	1
Sleep Apneas	1
Sleepy	1

Table 4. Centrality values for selected Metathesaurus candidates

would have gotten a centrality value of 0 if the head of the example phrase were *apnea* rather than *obstructive sleep apnea*.

- 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 were listed above in Table 1. 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 for the candidate is the average of the values for each of the variants. Variation values for the example candidates is shown in Table 5. Note that the values for Ds in the table

Candidate	Variation
Obstructive sleep apnoea	Ds = [0,0,0]; V = (1+1+1)/3 = 1.0
Sleep Apneas	Ds = [0,1]; V = (1+0.8)/2 = 0.9
Sleepy	Ds = [3]; V = (4/7)/1 = 0.571

Table 5. Variation values for selected Metathesaurus candidates

  are simply lists of the distance values for the candidate variants.

- Coverage: The coverage value indicates how much of the phrase string and the Metathesaurus string are involved in the match. In order to compute the value, the number of words participating in the match is computed for both the phrase and the Metathesaurus string. These numbers are called the *phrase span* and *Metathesaurus span*, respectively. Note, however, that gaps are ignored.[1] The coverage value for the phrase is the phrase span divided by the length of the phrase. Similarly, the coverage value for the Metathesaurus string is the Metathesaurus span divided by the length of the string. The final coverage value is the weighted average of the values for the phrase and the Metathesaurus string where the Metathesaurus string is given twice the weight as the phrase. Example coverage values are given in Table 6.

Candidate	Coverage
Obstructive sleep apnoea	(3/3 + 2*3/3)/3 = 1.0
Sleep Apneas	(2/3 + 2*2/2)/3 = 0.889
Sleepy	(1/3 + 2*1/1)/3 = 0.778

Table 6. Coverage values for selected Metathesaurus candidates

- 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 phrase and

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.

the Metathesaurus string 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 phrase is the sum of the squares of the connected phrase component sizes divided by the square of the length of the string. A similar cohesiveness value is computed for the Metathesaurus string. The final cohesiveness value is the weighted average of the phrase and Metathesaurus string values where the Metathesaurus string is again given twice the weight as the phrase. Examples of cohesiveness values are given in Table 7. (A slightly modified cohesiveness computation is used for final mappings; see section 2.3 below.)

Candidate	Cohesiveness
Obstructive sleep apnoea	$(3^2/3^2 + 2*3^2/3^2)/3 = 1.0$
Sleep Apneas	$(2^2/3^2 + 2*2^2/2^2)/3 = 0.815$
Sleepy	$(1^2/3^2 + 2*1^2/1^2)/3 = 0.704$

Table 7. Cohesiveness values for selected Metathesaurus candidates

Recall that the final evaluation value is the weighted average of the four evaluation components, coherence and cohesiveness getting twice the weight of centrality and variation, normalized to a value between 0 and 1,000. The final evaluation for our examples is shown in Table 8.

Candidate	Final Evaluation
Obstructive sleep apnoea	1,000*(1.0 + 1.0 + 2*1.0 +2*1.0)/6 = 1,000
Sleep Apneas	1,000*(1.0 + 0.9 + 2*0.889 + 2*0.815)/6 = 884
Sleepy	1,000*(1.0 + 0.571 + 2*0.778 + 2*0.704)/6 = 755

Table 8. Final evaluation values for selected Metathesaurus candidates

The final list of candidates for the phrase *of obstructive sleep apnea* is shown in Figure 2. Note

```
 1000 Obstructive sleep apnoea (Sleep Apnea, Obstructive) [Disease or Syn-
      drome]
  901 Apnea, Sleep (Sleep Apnea Syndromes) [Disease or Syndrome]
  827 Apnea [Finding]
  827 Obstructive (Obstructed) [Functional Concept]
  827 Sleep [Functional Concept]
  827 Sleep <3> (Sleep brand of diphenhydramine hydrochloride) [Organic
      Chemical,Pharmacologic Substance]
  793 Sleeping (Asleep) [Finding]
  755 Sleeplessness (Sleep Initiation and Maintenance Disorders) [Mental or
      Behavioral Dysfunction,Sign or Symptom]
  755 Sleepy [Finding]
```

Figure 2. The Evaluated Metathesaurus Candidates for *of obstructive sleep apnea*

that ‘Sleep Apneas’ (or ‘Apneas, Sleep’) does not appear here because it is a string for the concept

‘Sleep Apnea Syndromes’ which has a better scoring string ‘Apnea, Sleep’. The lesser scoring string can never contribute to a better mapping, so it is eliminated from the candidate list.

When word order does not matter, the coverage component of the evaluation function is replaced by the *involvement* component:

- Involvement: The involvement value is a rough approximation of the coverage except that 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 average (2/3 + 1)/2 or 0.833.[1]

2.3 Evaluating the Final Mapping

Once final mappings have been computed (see *The MetaMap Mapping Algorithm*), the evaluation function is applied to the combined candidates constituting the mappings, and the highest scoring mappings determine the final MetaMap result. The best mapping for *of obstructive sleep apnea* consists of the single candidate ‘Obstructive sleep apnoea’ and is not very interesting. Consider the non-optimal mapping consisting of ‘Obstructive’ and ‘Sleep Apneas’. The matchmap for this mapping is formed by simply combining the matchmaps of ‘Obstructive’ and ‘Sleep Apneas’: [[[1,1],[1,1],0], [[2,2],[1,1],0], [[3,3],[2,2],1]]. The connected component sizes for this matchmap are [[3], [1,2]]. For a final mapping, the connected component sizes for the candidates are ignored. Instead the lengths of the candidates are used. This has the effect of basing cohesiveness on concept chunks rather than word chunks, and it also ignores gaps and overmatches in the process. It also has the property that the candidate part of the cohesiveness value is always a perfect 1 when the mapping involves a single candidate. For our example, the lengths are [1,2], exactly the same as the connected component sizes. The final evaluation value for the mapping is 890; details are given in Table 9.

Evaluation Component	Value
Centrality	1
Variation	Ds = [0,0,1]; V = (1+1+0.8)/3 = 0.933
Coverage	(3/3 + 2*3/3)/3 = 1.0

Table 9. Evaluation for the mapping ‘Obstructive’ and ‘Sleep Apneas’

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

Evaluation Component	Value
Cohesiveness	$(3^2/3^2 + 2*(1^2+2^2)/3^2)/3 = 0.704$
Total	1,000*(1.0 + 0.933 + 2*1.0 + 2*0.704)/6 = 890

Table 9. Evaluation for the mapping ‘Obstructive’ and ‘Sleep Apneas’

As another example of computing cohesiveness for a final mapping, consider the text *chronic headache with sleep disorders* and the two candidates ‘Chronic Cluster Headache’ and ‘Sleep Walking Disorders’ (using the options `-zg`, `--term_processing and --allow_concept_gaps`). The connected component sizes for the two candidates are [1,1,1,1], but the candidate lengths are [3,3]. Since the connected component sizes for the phrase are [2,2], the cohesiveness value for this mapping is $((2^2+2^2)/5^2 + 2*(3^2+3^2)/6^2)/3 = 0.440$

Output from the evaluation process takes the form of ev/8 terms with arguments: the negation of the normalized value, the Metathesaurus string (candidate), its concept, the words in the string, the concept’s semantic types, the matchmap, a flag indicating whether the match involves the head, and a flag indicating whether the match is an overmatch. The evaluation terms for our examples are:

- `ev(-1000, Obstructive sleep apnoea, Sleep Apnea, Obstructive, [obstructive,sleep,apnoea], [dsyn], [[[1,1],[1,1],0],[[2,3],[2,3],0]], yes, no)`
- `ev(-884, Apneas, Sleep, Sleep Apnea Syndromes, [sleep,apneas], [dsyn], [[[2,2],[1,1],0],[[3,3],[2,2],1]], yes, no)`
- `ev(-755, Sleepy, Sleepy, [sleepy], [fndg], [[[2,2],[1,1],3]], yes, no)`

Examples of complete mappings with their corresponding evaluations are:

- `map(-1000, [ev(-1000, ‘Obstructive sleep apnoea’, ‘Sleep Apnea, Obstructive’, [obstructive,sleep,apnoea], [dsyn], [[[1,1],[1,1],0],[[2,3],[2,3],0]], yes, no)])`
- `map(-901, [ev(-827, ‘Obstructive’, ‘Obstructed’, [obstructive], [ftcn], [[[1,1],[1,1],0]], yes, no), ev(-901, ‘Apnea, Sleep’, ‘Sleep Apnea Syndromes’, [sleep,apnea], [dsyn], [[[2,3],[1,2],0]], yes, no)])`

2.4 Summary of the Algorithm

This section summarizes the evaluation process elaborating only those steps that have not already been discussed. For a given phrase and Metathesaurus candidate (or possibly several candidates for a full mapping):

- First check to see if the lowercased phrase text is a stop phrase according to `metamap_stop_phrase:stop_phrase/1`. This predicate is defined by those phrases appearing in one of several MEDLINE test collections which occur at least 40 times and do not produce any mappings. (This is determined, of course, without the use of `stop_phrase/1`, itself). If the phrase is a stop phrase, do not continue; otherwise,

- Check to see if this candidate has already been evaluated. If so, return the previous result; otherwise,
- Compute its matchmap and connected component sizes;
- Compute each of the evaluation components and combine them into a final value;
- Filter out redundant evaluations, where an evaluation is *redundant* if its score is no better than another evaluation involving the same concept and having the same phrase involvement. An example of a redundant evaluation is that for 'Sleep Apneas' (with a score of 884) because 'Sleep Apnea' (with a score of 901) involves the same concept, 'Sleep Apnea Syndromes', and has the same phrase involvement, *sleep apnea*;
- Filter out subsumed evaluations, where an evaluation is *subsumed* by another if its score is strictly worse and it has the same phrase involvement. The evaluations for 'Sleeping', 'Sleeplessness' and 'Sleepy' are all subsumed by either 'Sleep' or 'Sleep <3>' since in each case their score is less than 827 and they all involve *sleep* from the phrase. Note that neither 'Sleep' nor 'Sleep <3>' subsume each other because they have the same score of 827.

The purpose of filtering out redundant evaluations is to remove strings of a given concept which cannot outscore a similar string for the concept. The purpose of filtering out subsumed evaluations is to prepare for computing a final mapping by removing all but the best scoring strings covering a specific part of the phrase.

2.5 Options affecting the algorithm

The following options have an effect on the evaluation process:

- `-z --term_processing` affects parsing and, therefore, has an indirect effect on evaluation. If term processing is in effect for the input text *of obstructive sleep apnea*, then evaluation includes the word *of* which would not be the case without the option. A more realistic example is that the input text *Cancer of the lung <1>* appears as *cancer of the lung* if term processing is in effect. (Term processing assumes that it may be asked to process Metathesaurus strings with ambiguity designators. These are removed before further processing.);
- `-o --allow_overmatches` or `-g --allow_concept_gaps`: if either of these options is in force, the initial check for stop phrases is not performed since they were computed without these options;
- `-i --ignore_word_order` affects both the computation of matchmaps and also causes the coverage component of the evaluation function to be replaced by an involvement component. Matchmaps are formed by scanning the list of words in the candidate, matching variants from the text along the way. Under normal processing the sequence of variants must occur left to right in the phrase. If word order is being ignored, the variants can match in any order. The other effect of ignoring word order is the use of involvement instead of coverage; it was described previously;
- `-Y --prefer_multiple_concepts` causes the cohesiveness evaluation component to be inverted, i.e., the cohesiveness value is 1.0 - <original cohesiveness value>;

- `-r <integer> --threshold <integer>` invokes filtering out of evaluations not meeting the specified threshold. This is performed before filtering out of redundant evaluations;
- `-s --semantic_types` or `-q --machine output`: either of these options causes the four-letter abbreviations for semantic types to be added to evaluations. This addition is performed after filtering out redundant evaluations;
- `-X --truncate_candidates_mappings` truncates the list of candidate evaluations to the 100 top scoring ones; this is done before filtering out subsumed evaluations. (This option also truncates the list of top scoring mappings to 8 after the mappings have been constructed.)
- `-P --composite_phrases` or `-Q --quick_composite_phrases`: these options have the indirect effect of setting other options. For -P, the options -zogiX (where -z is --term_processing); for -Q, just -zi are set. (Note that these options are still under development.)