MetaMap 2012 Machine Output Explained

June 30, 2014

François-Michel Lang

1 Introduction

MetaMap Machine Output (MMO) is one form of MetaMap's output, and consists of Prolog terms; MMO is generated by calling MetaMap with the `--machine_output` (or `-q`) command-line option. Because the 2011v2 release of MetaMap included a number of changes to MMO, we created this document to explain MetaMap's MMO format, which provides comprehensive information.

A very brief summary of MetaMap processing follows: Input text is first chunked into its component utterances; next, each utterance is parsed into phrases; candidate concepts are then identified from the list of variants of each lexical item; finally, MetaMap maps the phrase to sets of UMLS concepts to identify the best coverage of the input phrase. All of this processing information is then displayed in the machine output format explained below.

The MMO structures displayed in this document are pretty-printed to make them readable by humans consulting this report; in actual MMO, however, each structure appears entirely on one line, because MMO is not intended to be human-readable, but rather postprocessed by downstream programs.

2 Pros and Cons of MMO

MMO contains all the information generated by MetaMap, and is most useful by allowing persistent storage of MetaMap results for automated postprocessing. MMO is a relatively compact output format, at least when compared to MetaMap's XML output. However, as mentioned above, MMO consists of Prolog terms, which are easy to parse because of Prolog's equivalence of programs and data—but only if a Prolog system is available. Otherwise, e.g., using C, Java, or Perl, parsing Prolog terms is not for the faint of heart.

In order to allow users to minimize disk usage by generating MMO, but still postprocess MetaMap's XML output, which is far simpler than postprocessing MMO, we also provide the program `mm_print`, which can generate equivalent MetaMap XML output (both formatted and unformatted) from MMO:

http://metamap.nlm.nih.gov/download/public_mm_linux_mm_print_2012.tar.bz2 (Linux; 4MB)

http://metamap.nlm.nih.gov/download/public_mm_win32_mm_print_2012.tar.bz2 (Windows; 4MB)

http://metamap.nlm.nih.gov/download/public_mm_darwin_mm_print_2012.tar.bz2 (Mac; 4MB)

If disk-space constraints are an issue, we recommend saving MetaMap output in MMO and using `mm_print` to generate the equivalent XML when needed for subsequent postprocessing.

3 High-Level MMO Structure

MMO consists of five main kinds of objects, which always appear in the order below. Each element is displayed on one possibly very long line. The various MMO components are explained below in greater detail.

1. **args**: The command used to invoke MetaMap along with all associated command-line options.
2. **aas**: List of user-defined acronym/abbreviations identified in the input text.
3. **neg_list**: List of negations identified in the input text; MetaMap's negation processing is based on Wendy Chapman's NegEx.
4. **utterance**: A sequence of tokens into which the input text was chunked; each utterance is followed by one or more sequences of the following three sub-components:
   (a) **phrase**: A sub-sequence of the utterance's tokens, along with its syntax; a phrase always appears with corresponding **candidates** and **mappings** objects.
   (b) **candidates**: A possibly empty list of UMLS candidate concepts identified in the phrase.
   (c) **mappings**: A possibly empty list of MetaMap's final mappings; each mapping is a subset of the candidate set, so the mappings list is empty iff the candidates list is empty.
5. **'EOU'**: **E**nd **o**f **U**tterance marker.

In addition, if MetaMap is invoked with the `--indicate_citation_end` (or `-E`) option and reads its input from a text file (and not user input), the string `'EOT'` (**E**nd **O**f **T**ext) is printed when MetaMap terminates normally.

4 The Devil in the Details

We now present in more detail each of the above MMO objects. All examples below will be based on the input text

The patient denied chest pain (CP), and heart attack (HA) was ruled out.

assumed to be in an ASCII text file called `SAMPLE.txt`; the MMO is written to the file `SAMPLE.MMO`.

4.1 The args Term

The form of the `args` term is

```
args( CommandLine , MetaMapOptions )
```

In this structure,

- CommandLine is the command-line call, including the name of the binary, and
- MetaMapOptions is a comma-separated list of terms of the form `OptionName-OptionValue` listing all the command-line options used in this invocation of MetaMap; `OptionValue` is `[]` if the corresponding option is a binary switch that requires no value.

The args term is of the form

```
args('MetaMap12.BINARY.Linux -L 2012 -Z 2012AA -q SAMPLE.txt SAMPLE.MMO',
     [lexicon_year-'2012',
      mm_data_year-'2012AA',
      machine_output-[],
      infile-'SAMPLE.txt',
      outfile-'SAMPLE.MMO'])
```

4.2 The aas Term

The form of the `aas` term is

```
aas(AcronymsAndAbbreviations)
```

where `AcronymsAndAbbreviations` is a list of 4-tuples:

```
Acronym * Expansion * CountList * CUIList
```

The "*" character was chosen as a separator because of its extremely low chance of appearing in the acronym or the expansion. The aas term for our input text is

```
aas(["CP"*"chest pain"*[1,2,3,10]*['C0008031','C2926613'],
     "HA"*"heart attack"*[1,2,3,12]*['C0027051']])
```

The four components of an element of the `AcronymsAndAbbreviations` list are the following:

- Acronym is the acronym itself (`CP` and `HA`);

- Expansion is the acronym's expansion (`chest pain` and `heart attack`);
- CountList is a list containing four integers (`[1,2,3,10]` and `[1,2,3,12]`), which are
  1. the number of tokens in the acronym (`1`)
  2. the character length of the acronym (`2`)
  3. the number of tokens (including whitespace tokens) in the expansion (`3`)
  4. the character length of the expansion (`10` and `12`)
- CUIList is the possibly empty list of the Concept Unique Identifiers (CUIs) of the concept(s) to which the acronym expansion was mapped by MetaMap. In this example, MetaMap maps *chest pain* to two concepts whose CUIs are C0008031 and C2926613; The CUIList is therefore `['C0008031','C2926613']`. The CUIs are surrounded by single quotation marks because they would otherwise be Prolog variables. If the acronym expansion was not mapped to any concepts, CUIList is the empty list (`[]`).

4.3 The neg_list Term

The form of the `neg_list` term is

```
neg_list(ListOfNegations)
```

where `ListOfNegations` is a list of terms of the form

```
negation(TypeOfNegation,
         NegationTrigger, TriggerPositionalInfo,
         NegatedConcept,  ConceptPositionalInfo)
```

Our input generates the `neg_list` term

```
neg_list([negation(nega,
                   denied, [12/6],
                   ['C0008031':'Chest Pain'], [19/10]),
          negation(negb,
                   'was ruled out', [58/13],
                   ['C0027051':'Heart attack'], [40/12])])
```

The positional information is a list of `StartPos/Length` terms representing the starting character position of the text string in question (relative to the beginning of the entire input) and its length in characters. There can be multiple `StartPos/Length` terms because triggers and concepts can be represented in the text by multiple non-contiguous strings.

The alert reader will have noticed that although two CUIs are associated with *chest pain* in the aas term, only one appears in the neg_list term. That apparent discrepancy is due to Semantic Types: The Semantic Type of C0008031 is Sign or Symptom (sosy), which triggers NegEx, but the semantic type of C2926613 is Clinical Attribute (clna), which does not.

4.4 The Utterance Term and Its Compoments

The form of the utterance term is

```
utterance( UtteranceID , UtteranceText , PosInfo , CRPos )
```

In this structure,

- UtteranceID identifies the position of the utterance within the input text. The UtteranceID is of the form CitationNumber.Section.UtteranceNumber; CitationNumber is the PMID, if available, or other possibly synthetic ID such as the default `00000000`; Section is either `ti` (title) or `ab` (abstract); UtteranceNumber is an integer representing the position of the utterance within the Section.
- UtteranceText is the actual text of the utterance.
- PosInfo is a `StartPos/Length` term identifying the starting character position (relative to the beginning of the entire input) and character length of the utterance.
- CRPos is a possibly empty list of integers representing the character positions of carriage returns in the utterance.

The utterance term for our input is

```
utterance( '00000000.tx.1' ,
           "The patient denied chest pain (CP),
            and heart attack (HA) was ruled out." ,
           0/72 , [] ).
```

Each utterance term is followed by one or more sequences of phrase, candidates, and mappings terms, which are now presented.

4.4.1 The Phrase Term

The form of the phrase term is

```
phrase( PhraseText , Syntax , PosInfo , CRPos )
```

In this structure,

- PhraseText is the text of the phrase,
- Syntax is a list representing MetaMap's syntactic analysis of the phrase,
- PosInfo is a term of the form `StartPos/Length`, identifying the starting character position character length of the phrase
- CRPos is a possibly empty list of integers representing the character positions of all carriage returns in the phrase.

Several phrase terms generated from our input text are

```
phrase( 'The patient' ,
        [det([lexmatch([the]),inputmatch(['The']),
              tag(det),tokens([the])]),
         head([lexmatch([patient]),inputmatch([patient]),
               tag(noun),tokens([patient])])] ,
        0/11 , [] )

phrase( 'denied chest pain (CP),' ,
        [pastpart([lexmatch([denied]),inputmatch([denied]),
                   tag(verb),ied])]),
         head([lexmatch(['chest pain']),inputmatch([chest,pain]),
               tag(noun),tokens([chest,pain])]),
         punc([inputmatch([',']),tokens([])])] ,
        12/23 , [] )
```

4.4.2 The Candidates Term

The form of the candidates term is

```
candidates( TotalCandidateCount ,  ExcludedCandidateCount ,
            PrunedCandidateCount ,  RemainingCandidateCount ,
            ListOfCandidates )
```

In this structure,

- TotalCandidateCount is the total number of candidates identified,
- ExcludedCandidateCount is the number of excluded candidates, i.e., those receiving a strictly lower score than another candidate with the same phrase coverage,
- PrunedCandidateCount is the number of pruned candidates, i.e., those removed from the candidate list for processing efficiency,
- RemainingCandidateCount is the number of remaining candidates, i.e., those neither excluded nor pruned, and
- ListOfCandidates is a list containing each remaining candidate term in an `ev/12` structure.

Excluded, pruned, and remaining candidate concepts are explained in detail in Section 2 of the MetaMap2011v2 Release notes, available at

http://metamap.nlm.nih.gov/MM_2011_v2_ReleaseNotes.pdf.

An example of a candidates term in which the candidates are abbreviated by `C1`, `C2`, etc., is

```
candidates( 1 , 0 , 0 , 1 , [C2,C2,...,CN] )
```

Each candidate term is an `ev/12` structure:

```
ev( NegScore , CUI , String , PreferredName , TextWords , SemanticTypes ,
    MatchMap , InvolvesHead , IsOvermatch , UniqueSources , PosInfo , Status )
```

in which (are you ready for this?)

- NegScore is an integer equal to -1 times the concept's MetaMap score.
- CUI is the UMLS Metathesaurus CUI of the UMLS string identified.
- String is the Metathesaurus string matched by MetaMap.

- PreferredName is the preferred name of the concept, (which may be the same as the Metathesaurus string).
- TextWords is a lowercased list of the words in the input text licensing the concept.
- SemanticTypes is a list of the short forms of the Semantic Types of the concept.
- MatchMap is a list representing (1) the correspondence of words in the candidate concept and words in the original phrase, and (2) any lexical variation in the matching.
- InvolvesHead is a yes/no field identifying if the concept is part of the head of the phrase.
- IsOvermatch is a yes/no field identifying if the concept match is an overmatch.
- UniqueSources is a list of the short forms (RSAB) of the UMLS source vocabularies in which the concept appears; in the examples below, the list is truncated for ease of presentation.
- PosInfo is a list of terms of the form `StartPos/Length`, identifying the starting character position within the input text and and character length of each word in the string.
- Status is a 0/1/2 field specifying whether the concept is remaining (0), excluded (1), or pruned (2).

Some candidate terms generated from our input text are

```
ev(-1000, 'C0030705', 'patient', 'Patients', [patient], [podg],
   [[[1,1],[1,1],0]], yes, no, ['MSH','SNOMEDCT'], [4/7], 0)

ev(-660, 'C0332319', 'Denied', 'Denied (qualifier)', [denied], [qlco],
   [[[1,1],[1,1],0]], no, no, ['MTH','SNMI','SNOMEDCT'], [12/6], 0)

ev(-901, 'C0008031', 'Chest Pain', 'Chest Pain', [chest,pain], [sosy],
   [[[2,3],[1,2],0]], yes, no, ['MSH','ICD10CM','CCS'], [19/10], 0)
```

4.4.3 The Mappings Term

The comparatively simple top-level form of the mappings term is

```
mappings(MappingsList)
```

in which `MappingsList` is a list of terms of the form

```
map( NegScore , CandidatesList )
```

`CandidatesList` is a list of `ev/12` terms as shown immediately above. A mappings term generated from our input text is

```
mappings([
  map( -901
       [ev(-660,'C0332319','Denied','Denied (qualifier)',[denied],[qlco],
           [[[1,1],[1,1],0]],no,no,['MTH','SNMI','SNOMEDCT'],[12/6],0),
       ev(-901,'C0008031','Chest Pain','Chest Pain',[chest,pain],[sosy],
           [[[2,3],[1,2],0]],yes,no,['MSH','ICD10CM','CCS'],[19/10],0)] ),
  map( -901 ,
       [ev(-660,'C0332319','Denied','Denied(qualifier)',[denied],[qlco],
           [[[1,1],[1,1],0]],no,no,['MTH','SNMI','SNOMEDCT'],[12/6],0),
        ev(-901,'C2926613','Chest pain','Chest pain',[chest,pain],[clna],
           [[[2,3],[1,2],0]],yes,no,['LNC','MTH'],[19/10],0)] )])
```

5 All MMO At Once

In order to show how the various MMO components fit together, we conclude this report with the MMO for the first two phrases our input text, i.e.,

The patient denied chest pain (CP):

```
args( 'MetaMap12.BINARY.Linux -L 2012 -Z 2012AA -q SAMPLE.txt SAMPLE.MMO' ,
      [lexicon_year-'2012',
       mm_data_year-'2012AA',
       machine_output-[],
       infile-'SAMPLE.txt',
       outfile-'SAMPLE.MMO'] ).

aas(["CP" * "chest pain" * [1,2,3,10] * ['C0008031','C2926613'] ,
     "HA" * "heart attack" * [1,2,3,12] * ['C0027051'] ]).

neg_list([negation( nega ,
                    denied , [12/6] ,
```

```
                ['C0008031':'Chest Pain'], [19/10]),
        negation(negb,
                'was ruled out', [58/13],
                ['C0027051':'Heart attack'], [40/12])])

utterance('00000000.tx.1',
          "The patient denied chest pain (CP),
           and heart attack (HA) was ruled out.",
          0/72, []).

phrase('The patient',
       [det([lexmatch([the]),inputmatch(['The']),
             tag(det),tokens([the])]),
        head([lexmatch([patient]),inputmatch([patient]),
              tag(noun),tokens([patient])])],
       0/11, []).

candidates(1, 0, 0, 1,
     [ev(-1000,'C0030705','patient','Patients',[patient],[podg],
         [[[1,1],[1,1],0]],yes,no,['MSH','SNOMEDCT'],[4/7],0)]).

mappings([
  map(-901
      [ev(-1000,'C0030705','patient','Patients',[patient],[podg],
          [[[1,1],[1,1],0]],yes,no,['MSH','SNOMEDCT'],[4/7],0)])]).

phrase('denied chest pain (CP),',
       [pastpart([lexmatch([denied]),inputmatch([denied]),
                  tag(verb),tokens([denied])]),
        head([lexmatch(['chest pain']),inputmatch([chest,pain]),
              tag(noun),tokens([chest,pain])]),
        punc([inputmatch([',']),tokens([])])],
       12/23, []).

candidates(17, 9, 0, 8,
     [ev(-901,'C0008031','Chest Pain','Chest Pain',[chest,pain],[sosy],
         [[[2,3],[1,2],0]],yes,no,['MSH','NDFRT','CST'],[19/10],0),
      ev(-901,'C2926613','Chest pain','Chest pain',[chest,pain],[clna],
         [[[2,3],[1,2],0]],yes,no,['LNC','MTH'],[19/10],0),
           . . .   other candidates omitted  . . .
      ev(-660,'C0332319','Denied','Denied (qualifier)',[denied],[qlco],
```

```
            [[[1,1],[1,1],0]],no,no,
            ['MTH','SNMI','SNOMEDCT','CHV'],[12/6],0),
        ev(-660,'C1549490','Denied','Day type - Denied',[denied],[icdn],
            [[[1,1],[1,1],0]],no,no,['MTH','HL7V2.5'],[12/6],0)]).

mappings([
  map(-901,
      [ev(-660,'C0332319','Denied','Denied (qualifier)',[denied],[qlco],
          [[[1,1],[1,1],0]],no,no,['MTH','SNMI','SNOMEDCT','CHV'],[12/6],0),
       ev(-901,'C0008031','Chest Pain','Chest Pain',[chest,pain],[sosy],
          [[[2,3],[1,2],0]],yes,no,['ICF','ICD10CM','CCS'],[19/10],0)]),
  map(-901,
      [ev(-660,'C0332319','Denied','Denied (qualifier)',[denied],[qlco],
          [[[1,1],[1,1],0]],no,no,['MTH','SNMI','SNOMEDCT','CHV'],[12/6],0),
       ev(-901,'C2926613','Chest pain','Chest pain',[chest,pain],[clna],
          [[[2,3],[1,2],0]],yes,no,['LNC','MTH'],[19/10],0)]),
            . . .   other mappings omitted  . . .
  map(-901,
      [ev(-660,'C1549490','Denied','Day type - Denied',[denied],[idcn],
          [[[1,1],[1,1],0]],no,no,['MTH','HL7V2.5'],[12/6],0),
       ev(-901,'C0008031','Chest Pain','Chest Pain',[chest,pain],[sosy],
          [[[2,3],[1,2],0]],yes,no,['ICF','ICD10CM','CCS'],[19/10],0)]),
  map(-901,
      [ev(-660,'C1549490','Denied','Day type - Denied',[denied],[idcn],
          [[[1,1],[1,1],0]],no,no,['MTH','HL7V2.5'],[12/6],0),
       ev(-901,'C2926613','Chest pain','Chest pain',[chest,pain],[clna],
          [[[2,3],[1,2],0]],yes,no,['LNC','MTH'],[19/10],0)])).

'EOU'.
```