Machine Output Explained

Each citation is first broken down into it's respective sentences or utterances. Each utterance is then broken down into it's respective noun phrases. Then, each noun phrase is tagged using a POS tagger (in our case the Xerox Parc tagger) or using the lexicon when there is no POS tagger available to identify nouns, verbs, prepositions, adjectives, punctuation, etc., and where the head or main idea of the noun phrase is located. Once we have all of this information, MMTx can then map UMLS concepts to the noun phrase identifying the best possible coverage.

```
The machine output consists of four (5) main objects:
   utterance     -- One per sentence

      phrase     -- As many as are needed to complete the sentence/utterance. Always matched up with both a "candidates"
                    object and a "mappings" object.

      candidates -- Always matched up with a phrase object and may be empty "candidates([])." depending on the phrase.
                    All of the possible elements available to matchup with the phrase in highest to lowest score order
                    within each set of items.

      mappings   -- Always matched up with a phrase object and may be empty "mappings([])." depending on the phrase.
                    Final combinations of candidates that matchup with the phrase in highest to lowest score order.

   'EOU'.        -- Marks the end of the utterance.
```

Format:

utterance:

```
   utterance('sentence/utterance identifier', "sentence/utterance").
      where sentence/utterance identifier consists of:
           MEDLINE ID.Location marker ti (title) or ab (abstract).one up number for each section.
```

phrase:

```
    phrase('identified noun phrase', [tagging information]).
```

candidates:

```
    candidates(
      [ev(negated candidate score, 'UMLS concept', 'preferred name for concept - may or may not be different',
         [matched word or words lowercased that this candidate matches in the phrase - comma separated list],
         [semantic type(s) - comma separated list],
         [match map list - see below], candidate involved with head of phrase - yes or no,
        is this an overmatch - yes or no
        ),
       ev(negated candidate score, 'UMLS concept', 'preferred name for concept - may or may not be different',
         [matched word or words lowercased that this candidate matches in the phrase - comma separated list],
         [semantic type(s) - comma separated list],
         [match map list - see below], candidate involved with head of phrase - yes or no,
        is this an overmatch - yes or no
        )
      ]
    ).
```

mappings:

```
    mappings(
      [map(negated overall score for this mapping,
            [ev(negated candidate score, 'UMLS concept', 'preferred name for concept - may or may not be different',
                 [matched word or words lowercased that this candidate matches in the phrase - comma separated list],
                 [semantic type(s) - comma separated list],
                 [match map list - see below], candidate involved with head of phrase - yes or no,
                 is this an overmatch - yes or no
               )
            ]
          )
      ]
    ).
```

Match Map List: The match map list consists of information on how the candidate concept matches up to words in the original phrase and if there is any lexical variation in the matching. NOTE: The span word counts don't include the following syntactic elements: aux, compl, conj, det, modal, prep, pron, and punc which are ignored by MetaMap. For example, in the phrase "of the drug therapy", the word "drug" would be counted as word #1 and the word "therapy" would be word #2.

```
    [[[phrase word span begin,phrase word span end],[concept word span begin,concept word span end],variation]]
```

Example: This mapping shows word 1 of the phrase maps to word 1 of the concept with 0 lexical variation

```
      [[[1,1],[1,1],0]]
         ^^^ Match up of words in TEXT
               ^^^ Match up of words in STRING
                    ^ Variation
```

Example: This shows word 2 of the phrase maps to word 1 of the concept with 0 lexical variation and word 3 of the text maps to word 2 of the concept with 0 lexical variation.

```
       [[2,2],[1,1],0],[[3,3],[2,2],0]
```

```
utterance('98157484.ab.1',"OBJECTIVE: To define the total allowable variability that is clinically tolerated for certain drug assays performed by the therapeutic drug monitoring (TDM) laboratory at our institution.").
phrase('OBJECTIVE',[head([lexmatch([objective]),inputmatch(['OBJECTIVE']),tag(adj),tokens([objective])])]).
candidates([ev(-1000,'Objective','Goals',[objective],[inpr],[[[1,1],[1,1],0]],yes,no),ev(-928,'Objectivity','Objectivity',[objectivity],[inbe],[[[1,1],[1,1],3]],yes,no)]).
mappings([map(-1000,[ev(-1000,'Objective','Goals',[objective],[inpr],[[[1,1],[1,1],0]],yes,no)])]).
phrase(:,[punc([inputmatch([:]),tokens([:])])]).
candidates([]).
mappings([]).
phrase('To',[adv([lexmatch([to]),inputmatch(['To']),tag(adv),tokens([to])])]).
candidates([]).
mappings([]).
phrase(define,[verb([lexmatch([define]),inputmatch([define]),tag(verb),tokens([define])])]).
candidates([]).
mappings([]).
```

```
phrase('the total allowable
variability',[det([lexmatch([the]),inputmatch([the]),tag(det),tokens([the])]),mod([lexmatch([total]),inputmatch([total]),tag(adj),tokens([total])]),m
od([lexmatch([allowable]),inputmatch([allowable]),tag(adj),tokens([allowable])]),head([lexmatch([variability]),inputmatch([variability]),tag(noun),to
kens([variability])])]).
candidates([ev(-755,'Variable','Variable',[variable],[qlco],[[[3,3],[1,1],3]],yes,no),ev(-660,'% total','percent of
total',[total],[qnco],[[[1,1],[1,1],0]],no,no),ev(-660,'Total','Total',[total],[qlco],[[[1,1],[1,1],0]],no,no),ev(-
577,allowing,allowing,[allowing],[socb],[[[2,2],[1,1],4]],no,no),ev(-560,'Allowance','Allowance',[allowance],[qnco],[[[2,2],[1,1],6]],no,no)]).
mappings([map(-800,[ev(-660,'Total','Total',[total],[qlco],[[[1,1],[1,1],0]],no,no),ev(-
577,allowing,allowing,[allowing],[socb],[[[2,2],[1,1],4]],no,no),ev(-755,'Variable','Variable',[variable],[qlco],[[[3,3],[1,1],3]],yes,no)]),map(-
800,[ev(-660,'% total','percent of total',[total],[qnco],[[[1,1],[1,1],0]],no,no),ev(-
577,allowing,allowing,[allowing],[socb],[[[2,2],[1,1],4]],no,no),ev(-755,'Variable','Variable',[variable],[qlco],[[[3,3],[1,1],3]],yes,no)])]).
phrase(that,[compl([lexmatch([that]),inputmatch([that]),tag(compl),tokens([that])])]).
candidates([]).
mappings([]).
phrase(is,[aux([lexmatch([is]),inputmatch([is]),tag(aux),tokens([is])])]).
candidates([]).
mappings([]).
phrase(clinically,[adv([lexmatch([clinically]),inputmatch([clinically]),tag(adv),tokens([clinically])])]).
candidates([]).
mappings([]).
phrase(tolerated,[verb([lexmatch([tolerated]),inputmatch([tolerated]),tag(verb),tokens([tolerated])])]).
candidates([]).
mappings([]).
phrase('for certain drug
assays',[prep([lexmatch([for]),inputmatch([for]),tag(prep),tokens([for])]),det([lexmatch([certain]),inputmatch([certain]),tag(det),tokens([certain])]
),mod([lexmatch([drug]),inputmatch([drug]),tag(noun),tokens([drug])]),head([lexmatch([assays]),inputmatch([assays]),tag(noun),tokens([assays])])]).
candidates([ev(-983,'Drug assay','Drug assay',[drug,assay],[lbpr],[[[1,1],[1,1],0],[[2,2],[2,2],1]],yes,no),ev(-
827,assay,assay,[assay],[lbpr],[[[2,2],[1,1],1]],yes,no),ev(-694,'Drug, NOS','Pharmaceutical
Preparations',[drug],[phsu],[[[1,1],[1,1],0]],no,no),ev(-601,'Medicament','Medicaments',[medicament],[phsu],[[[1,1],[1,1],5]],no,no)]).
mappings([map(-983,[ev(-983,'Drug assay','Drug assay',[drug,assay],[lbpr],[[[1,1],[1,1],0],[[2,2],[2,2],1]],yes,no)])]).
phrase(performed,[verb([lexmatch([performed]),inputmatch([performed]),tag(verb),tokens([performed])])]).
candidates([ev(-966,'Performing','Performing',[performing],[topp],[[[1,1],[1,1],1]],yes,no),ev(-
928,'Performance','Performance',[performance],[inbe],[[[1,1],[1,1],3]],yes,no)]).
mappings([map(-966,[ev(-966,'Performing','Performing',[performing],[topp],[[[1,1],[1,1],1]],yes,no)])]).
phrase('by the therapeutic drug
monitoring',[prep([lexmatch([by]),inputmatch([by]),tag(prep),tokens([by])]),det([lexmatch([the]),inputmatch([the]),tag(det),tokens([the])]),mod([lexm
atch([therapeutic]),inputmatch([therapeutic]),tag(adj),tokens([therapeutic])]),mod([lexmatch([drug]),inputmatch([drug]),tag(noun),tokens([drug])]),he
ad([lexmatch([monitoring]),inputmatch([monitoring]),tag(noun),tokens([monitoring])])]).
candidates([ev(-901,'Drug Monitoring','Drug Monitoring',[drug,monitoring],[diap],[[[2,2],[1,1],0],[[3,3],[2,2],0]],yes,no),ev(-
827,monitoring,'Preventive monitoring',[monitoring],[hlca],[[[3,3],[1,1],0]],yes,no),ev(-793,'Monitor, NOS','Monitor,
NOS',[monitor],[medd],[[[3,3],[1,1],1]],yes,no),ev(-793,monitor,monitor,[monitor],[hops,opco],[[[3,3],[1,1],1]],yes,no),ev(-660,'Drug,
NOS','Pharmaceutical Preparations',[drug],[phsu],[[[2,2],[1,1],0]],no,no),ev(-
660,'Therapeutic','Therapeutic',[therapeutic],[ftcn],[[[1,1],[1,1],0]],no,nc),ev(-660,'Therapeutic <2>','The science and art of
healing',[therapeutic],[topp],[[[1,1],[1,1],0]],no,no),ev(-567,'Medicament','Medicaments',[medicament],[phsu],[[[2,2],[1,1],5]],no,no)]).
mappings([map(-901,[ev(-660,'Therapeutic <2>','The science and art of healing',[therapeutic],[topp],[[[1,1],[1,1],0]],no,nc),ev(-901,'Drug
Monitoring','Drug Monitoring',[drug,monitoring],[diap],[[[2,2],[1,1],0],[[3,3],[2,2],0]],yes,no)]),map(-901,[ev(-
660,'Therapeutic','Therapeutic',[therapeutic],[ftcn],[[[1,1],[1,1],0]],no,nc),ev(-901,'Drug Monitoring','Drug
Monitoring',[drug,monitoring],[diap],[[[2,2],[1,1],0],[[3,3],[2,2],0]],yes,no)])]).
phrase('(TDM',[punc([inputmatch(['(']),tokens([])]),not_in_lex([inputmatch(['TDM']),tokens([tdm])])]).
candidates([]).
mappings([]).
phrase(')',[punc([inputmatch([')']),tokens([])])]).
candidates([]).
mappings([]).
phrase(laboratory,[head([lexmatch([laboratory]),inputmatch([laboratory]),tag(noun),tokens([laboratory])])]).
candidates([ev(-1000,'Laboratory','Laboratories',[laboratory],[mnob,orgt],[[[1,1],[1,1],0]],yes,no),ev(-
916,'Rennin','Chymosin',[rennin],[aapp,enzy,phsu],[[[1,1],[1,1],4]],yes,no),ev(-
907,'Renin','Renin',[renin],[aapp,enzy],[[[1,1],[1,1],5]],yes,no),ev(-907,'Renin <2>','Renin measurement',[renin],[lbpr],[[[1,1],[1,1],5]],yes,no)]).
mappings([map(-1000,[ev(-1000,'Laboratory','Laboratories',[laboratory],[mnob,orgt],[[[1,1],[1,1],0]],yes,no)])]).
```

```
phrase('at our
institution.',[prep([lexmatch([at]),inputmatch([at]),tag(prep),tokens([at])]),pron([lexmatch([our]),inputmatch([our]),tag(pron),tokens([our])]),head(
[lexmatch([institution]),inputmatch([institution]),tag(noun),tokens([institution])]),punc([inputmatch(['.']),tokens([])])]).
candidates([]).
mappings([]).
'EOU'.
```