Friday 04-05-18

Part(1):Our Polarity Marking Algorithm.

1. ***Polarity Table:*** Different type of words introduces different polarity effects on trees, I divided them into quantifiers and others.

   Quantifiers: I assumed they usually affect their restricted item; i.e. affect one argument?

```
polTable(*(universal),'-').
polTable(*(no),'-').
polTable(_,'+').
```

   Others:

```
polTable2('not',1,'-',_).
polTable2('lack',2,'-','+').
polTable2('doubt',2,'-','+').
polTable2(_,2,'+','+').
```

   a. It is assumed that for items affecting 2 arguments such as ‘lack’ the second argument most of the time appears first then the first one; dobj then subject –not always?
   b. It is assumed that 2 arguments are the maximum numbers of arguments a polarity word can affect?
   c. does polarity words taking 2 arguments but affecting only one means that the other argument will be taking the same mark that word is taking? (see example 3 & 4)
   d. Should time be marked with polarity, can a time be expanded or contracted when using natural logic matching algorithm?
   e. we handle scoping in later steps which is why ‘no’ is not marking everything it should be marking (see example 6).
   f.

2. ***Monotonicity composition operator(°),*** definition 2.3.Node A is given polarity Pol based on the polarity of the parent node ° the polarity that it should be assigned to it according to the parent node.

Definition 2.3: Having a set of polarity markers $P = \{+, -, \cdot\}$, the monotonicity composition operator $(\circ)$ decides on the final polarity mark as follows [Bernardi, 1999, Icard III and Moss, 2014]:

$$\begin{array}{lccc} \textit{No polarity } (\cdot): & (\cdot) \circ (+|-) = & (+|-) \circ (\cdot) = & (\cdot) \\ \textit{Positive polarity } (+): & (+) \circ (+) = & (-) \circ (-) = & (+) \\ \textit{Negative polarity } (-): & (+) \circ (-) = & (-) \circ (+) = & (-) \end{array}$$

Part(1):Examples

1.convSteps2('every man loves a woman.',X).

```
pm([.,
    arg(claim,
        *([time(tense(present),
                aspect(simple),
                aux(-),
                def(-),
                finite(tensed))]),
        [love:+,
         arg(dobj, *(indefinite), [woman>singular:+]),
         arg(subject, *(universal), [man>singular:-])])])
```

2.convSteps2('not every man loves a woman.',X).

```
pm([.,
    arg(claim,
        A,
        [not:+,
         arg(negComp,
             *([time(tense(present),
                     aspect(simple),
                     aux(-),
                     def(-),
```

```
                finite(tensed))]),
            [love:-,
             arg(dobj, *(indefinite), [woman>singular:-]),
             arg(subject, *(universal), [man>singular:+])])])])
```

```
3.convSteps2('a man does not lack a friend.',X).
pm([.,
    arg(claim,
        A,
        [not:+,
         arg(negComp,
             *([time(tense(present),
                     aspect(simple),
                     aux(+),
                     def(-),
                     finite(tensed)),
                time(tense(B),
                     aspect(simple),
                     aux(-),
                     def(C),
                     finite(infinitive))]),
             [lack:-,
              arg(dobj, *(indefinite), [friend>singular:+]),
              arg(subject, *(indefinite), [man>singular:-])])])])
```

```
4.convSteps2('he doubts she likes him.',X).
pm([.,
    arg(claim,
```

```
        *([time(tense(present),
                aspect(simple),
                aux(-),
                def(-),
                finite(tensed))]),
        [doubt:+,
         arg(xcomp,
             *([time(tense(present),
                     aspect(simple),
                     aux(-),
                     def(-),
                     finite(tensed))]),
             [like:-,
              arg(dobj, *(proRef), him:-),
              arg(subject, *(proRef), she:-)]),
         arg(subject, *(proRef), he:+)])])
```

```
5.convSteps2('John loves a pretty woman.',X).
pm([.,
    arg(claim,
        *([time(tense(present),
                aspect(simple),
                aux(-),
                def(-),
                finite(tensed))]),
        [love:+,
         arg(dobj,
             *(indefinite),
             [woman>singular:+, modifier(amod, pretty:+)]),
         arg(subject, *(name), [John:NP:+])])])
```

```
6.convSteps2('no man is an island.',X).
pm([.,
    arg(claim,
        *([time(tense(present),
                aspect(simple),
                aux(-),
                def(-),
                finite(tensed))]),
        [be:+,
         arg(predication(xbar(v(-), n(+))),
             *(indefinite),
             [island>singular:+]),
         arg(subject, *(no), [man>singular:-])])])


7.convSteps2('most men like women.',X).
pm([.,
    arg(claim,
        *([time(tense(A), aspect(B), aux(-), def(C), finite(tensed))]),
        [like:+,
         arg(dobj, *(existential=10), woman>plural:.),
         arg(subject, *(universal), [man>plural:-])])])

8.convSteps2('John left without Mary.',X).

pm([.,
    arg(claim,
        *([time(tense(past), aspect(simple), aux(-), def(A), finite(tensed))]),
        [left:+,
         arg(subject, *(name), [John:NP:+]),
         modifier(ppmod,
                  [without:+,
```

```
arg(comp, *(name), [Mary:NP:-])])])])
```