Implementation

December 29, 2013

1 Implementation

The crux of the dependency parser is Eisner's parsing algorithm. A slightly modified version of the same was used which is given below:

```
Initialization:
for s = 0 to n do
     Form Triangle of size 1
    C[s][s][$\rightarrow$][1] = C[s][s][$\leftarrow$][1] = 0.0 Form TriStop of size 1
    C[s][s][$\rightarrow$][2] = C[s][s][$\leftarrow$][2] = 0.0
end for
for  k = 1 to n + 1 do
    for s = 0 to n do
        t=s+k
        if t > n then break
        First: create Trapezium
        C[s][t][$\leftarrow$][0] = $max_{s \leq r < t}$ (C[s][r][$\rightarrow$][1] + C[r + 1][t][$\leftarrow$][1] + S(t, s))
        C[s][t][$\rightarrow$][0] = $max_{s \leq r < t}$ (C[s][r][$\rightarrow$][1] + C[r + 1][t][$\leftarrow$][1] + S(s, t))
        Second: create Triangle
        C[s][t][$\leftarrow$][1] = $max_{s \leq r < t}$ (C[s][r][$\leftarrow$][1] + C[r][t][$\leftarrow$][0])
        C[s][t][$\rightarrow$][1] = $max_{s < r \leq t}$ (C[s][r][$\rightarrow$][0] + C[r][t][$\rightarrow$][1])
        Second: create Tristop
        C[s][t][$\leftarrow$][1] = $max_{s \leq r < t}$ (C[s][r][$\leftarrow$][1] + C[r][t][$\leftarrow$][0])
        C[s][t][$\rightarrow$][1] = $max_{s < r \leq t}$ (C[s][r][$\rightarrow$][0] + C[r][t][$\rightarrow$][1])
    end for
end for
Return C[0][n][$\rightarrow$][1] as the highest score for any parse
```

An open source library called PyDecode is used to build a hypergraph, with all the triangles, trapezium and Tristop formed in the Eisner's parsing algorithm as its nodes. The edges of the hypergraph are assigned weights according to certain rules. An edge has head word, modifier word, direction, adjacency and state values stored in it. The direction indicates the direction in which the edge is being formed. The adjacency indicates if the modifier word is the first child of the head word, in which case it is “adj“, or not, in which case it is “non-adj“. The edge joining a Triangle with TriStop to form a Trapezium is responsible for dependency probability. The edge forming a TriStop from a Triangle is used to calculate the stop probability. If an edge does not have a modifier, the modifier word is “—“.

The insideOutside algorithm is run on the entire hypergraph. The inside probability of the root of the hypergraph gives the total probability of the sentence Z. The marginals of the hypergraph are computed using the PyDecode library. The marginals = marginals / Z gives the counts for the EM algorithm.

The em algorithm is run 20 times for all the sentences. The hypergraph is built for each sentence. The stop and dep probabilities are incremented according to the following algorithm:

```
for edge in hypergraph.edges do
    headWord, modWord, direct, adj, state = edge.label.split()
    if state == "1" and modWord != '—' then
        depCounts[headWord, modWord, direct] += marginals[edge.label]
    end if
    if state == "0" then
        stopCounts[headWord, direct, adj] += marginals[edge.label]
    end if
end for
```

2 EM

- sentence; $w_1 \dots w_n$
- vocabulary; $\mathcal{W}$
- modifier; $m \in \{1 \dots n\}$
- head; $h \in \{0 \dots n\}$
- direction; $\mathcal{D} = \{\mathrm{L}, \mathrm{R}\}$
- edge; $\mathcal{E} = \{\text{E(h, m!=–, dir, ADJ, cont=1)}, \text{E(h, m=–, dir, ADJ, cont=1)}, \text{E(h, dir, ADJ, cont=0)}\}$

  The first edge E(h, m!=–, ADJ, cont=1) is created when a Triangle whose head word is still taking children combines with a tringle whose headword has stopped taking children (TriStop) to form a trapezium.

  The edge E(h, m=–, ADJ, cont=1) is created when a Trapezium combines with a tringle whose headword has stopped taking children (TriStop) to form a Triangle.

  The edge E(h, dir, ADJ, cont=0) is created when a Triangle’s headword stops taking children to form TriStop.

- marginals $p(edge)$

The probabilities

- $p(\text{CONT}|w, dir, \text{ADJ})$; $\text{CONT} \in \{0,1\}$, $w \in \mathcal{W}$, $\text{ADJ} \in \{0,1\}$, $dir \in \{0,1\}$
- $c(\text{CONT}, w, \text{ADJ})$
- $p(u|v, dir, \text{ADJ})$; $\text{CONT} \in \{0,1\},\ u, v \in \mathcal{W},\ \text{ADJ} \in \{0,1\}$
- $c(u, v, dir)$
- $c(u, v, dir, \text{ADJ})$

Estimation Step
Fill in the c charts.

$$c(\text{CONT} = 0, h, dir, \text{ADJ} = 1) \leftarrow \sum p(\text{E(h, dir, ADJ = 1, CONT = 0)})$$

$$c(\text{CONT} = 0, h, dir, \text{ADJ} = 0) \leftarrow \sum p(\text{E(h, dir, ADJ = 0, CONT = 0)})$$

$$c(h, m, dir, \text{ADJ}) \leftarrow \sum p(\text{E(h, m! = --, dir, ADJ, CONT = 1)})$$

$$c(h, m, dir) \leftarrow \sum_{ADJ=\{0,1\}} p(\text{E(h, m! = --, dir, CONT = 1)})$$

Maximization Step

$$p(\text{CONT}|h, dir, \text{ADJ}) \leftarrow \frac{c(\text{CONT}, h, dir, \text{ADJ})}{\sum_{m \in \mathcal{W}} c(h, m, dir, ADJ) + c(\text{CONT}, h, dir, \text{ADJ})}$$

$$p(m|h, dir) \leftarrow \frac{c(h, m, dir)}{\sum_{m \in \mathcal{W}} c(h, m, dir)}$$