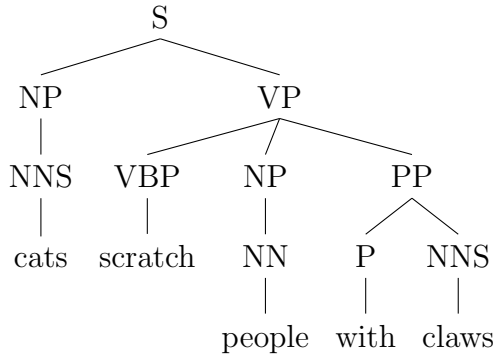
CS 4650/7650, Lecture 18
Integer Linear Programming for Semantic Role Labeling

Jacob Eisenstein

October 28, 2013

Here's our example:



Let's say that the correct annotation is

- $[cats]_{a0}$
- $[scratch]_{v}$
- $[people]_{a1}$
- $[with\ claws]_{am-mnr}$
- $[with]_{\varnothing}$
- $[claws]_{\varnothing}$
- $[scratch\ people\ with\ claws]_{\varnothing}$

(The tag AM-MNR means **argument modifier – manner**)
For each potential argument i, and for each tag t, define

$$y_{i,t} = \begin{cases} 1, & \text{argument } i \text{ takes tag } t \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

- Now, suppose $\theta_{i,t} = \log P(y_{i,t}|\boldsymbol{x}_i)$. Then

$$J = \sum_{i,t} \theta_{i,t} y_{i,t} = \log P(\boldsymbol{y}|\boldsymbol{x}) \tag{2}$$

- Or we could have $\theta_{i,t} = \boldsymbol{w}^\mathsf{T} \boldsymbol{f}(y_i = t, \boldsymbol{x}, i)$. Then J can be the sum of the inner products of a bunch of linear classifiers, and $\boldsymbol{w}^\mathsf{T} \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{x}) = \sum_i \boldsymbol{w}^\mathsf{T} \boldsymbol{f}(y_i, \boldsymbol{x}, i)$.
- In either case, J is a **linear** function of the integer (binary) variables $\boldsymbol{y}$. Such optimization problems are called **Integer Linear Programs** (ILP).

So far, we can optimize this easily: just optimize separately for each i. But this could lead to some bad outcomes.

- Multiple arguments of the same type:
  $[cats]_{a0}$ *scratch* $[people]_{a0}$ *with claws*
- Overlapping arguments:
  $[with\ [\text{claws}]_{a1}]_{a2}$
- Illegal arguments for the predicate:
  $[cats]_{a0}$ *scratch* $[people]_{a1}[$ *with claws*$]_{a2}$[1]
- Reference arguments without referents:
  - ok: $[The\ deregulation]_{A1}$ *of railroads* $[that]_{R-A1}$ *began* $[in\ 1980]_{AM-TMP}$
  - not ok: $[The\ deregulation]_{A1}$ *of railroads* $[that]_{R-A0}$ *began* $[in\ 1980]_{AM-TMP}$
- Etc

[1] I'm not sure if a2 is really illegal here, but the point is that scratch is not a ditransitive verb, so it can't take a second direct object.

Rather than incorporating these global factors into the objective, we'll treat them as **constraints**. So we now solve a constrained optimization problem,

$$\max_{\boldsymbol{y}} \sum_{i,t} \theta_{i,t} y_{i,t} \tag{3}$$

$$s.t. \boldsymbol{y} \in \mathcal{C}(\boldsymbol{x}) \tag{4}$$

Where the constraint set $\mathcal{C}(\boldsymbol{x})$ requires things like:

- All arguments get at most one label:

$$\forall i \sum_{t} y_{i,t} = 1 \tag{5}$$

  (equality, because you can always have the $\varnothing$ label)

- No duplicate argument classes

$$\forall t \neq \varnothing \sum_{i} y_{i,t} \leq 1 \tag{6}$$

- Overlapping arguments get at most one non-null label:

$$\forall \langle i, j \rangle : i \rightsquigarrow_{\tau} j, y_{i,\varnothing} + y_{j,\varnothing} \geq 1 \tag{7}$$

- Some arguments are forbidden

$$\sum_{i} y_{i,A2} = 0$$

- Reference arguments must have referents

$$\forall i y_{i,R-A0} \leq \sum_{j} y_{j,A0}$$

$$\forall i y_{i,R-A1} \leq \sum_{j} y_{j,A1}$$

$$\ldots$$

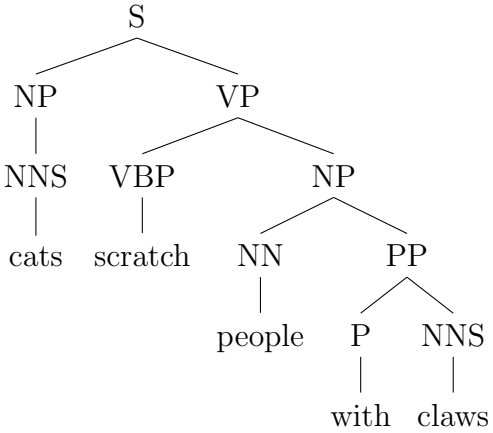
- Continuation argument must follow the referent

$$\forall i y_{i,C-A0} \leq \sum_{j<i} y_{j,A0}$$
$$\forall i y_{i,C-A1} \leq \sum_{j<i} y_{j,A1}$$
$$\ldots$$

Note that all of the constraints are also linear.

We precompute the scores $\theta_{i,t}$, and send the objective and the constraints to an ILP solver like GLPK (free) or CPLEX (expensive). It finds that best solution that meets all these linear constraints.

Adding more parses Now, suppose we ran another parser and got another parse:



Now we have another possible argument: [people with claws]. We can just include this argument with the ones that we got from the other parse, as long as we add the appropriate constraints to prevent overlapping.

This allows us to combine many possible parse trees, and even to select arguments that are constituents from different trees. Punyakanok, Roth, and Yi (2007) showed that this flexibility yields a big boost in accuracy.