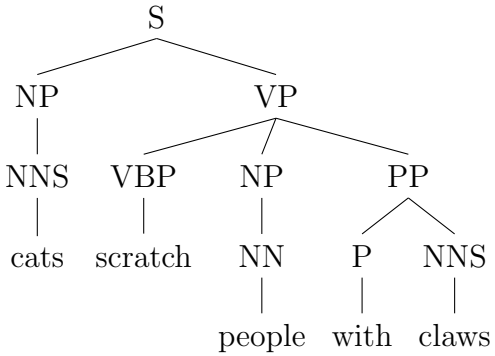
Integer Linear Programming for Semantic Role Labeling

Jacob Eisenstein

October 23, 2014

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 we define a feature function, where

$$\boldsymbol{f}(y, \boldsymbol{w}, i, t) = \delta(Y_{i,t} = y), \tag{2}$$

and set the weight

$$\theta(y, \boldsymbol{w}, i, t) = \log P(\boldsymbol{w}_i | y_i = t). \tag{3}$$

Then we have,

$$\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{w}) = \sum_{i,t} \boldsymbol{\theta}^\top \boldsymbol{f}(y_{i,t}, \boldsymbol{w}, i, t) \tag{4}$$

$$= \sum_{i} \log P(\boldsymbol{w}_i | y_i = t) \tag{5}$$

$$\approx \log P(\boldsymbol{w} | \boldsymbol{y}, \tau), \tag{6}$$

where τ is the parse tree.

More reasonably, we can define a rich local feature function $\boldsymbol{f}(y_{i,t}, \boldsymbol{w}, i, t)$, using features like the ones given by Gildea and Manning. In either case, J is a **linear** function of the integer (binary) variables $\boldsymbol{y}$. Such optimization problems are called **Integer Linear Programs** (ILP). They fit into our usual form for linear prediction:

$$\hat{\boldsymbol{y}} = \arg \max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{w}, \tau)} \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{w}, \tau), \tag{7}$$

where $\mathcal{Y}(\boldsymbol{w}, \tau)$ is the set of allowable labelings.

Because $\boldsymbol{y}$ is decoupled, we can optimize this easily: just solve separately for each y_i, obtaining the best label t for each constituent 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* $[in\ 1980]_{AM-TMP}$

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

$$\hat{\boldsymbol{y}} = \arg \max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{w}, \tau)} \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{w}, \tau) \tag{8}$$

$$s.t. \boldsymbol{y} \in \mathcal{C}(\boldsymbol{w}, \tau) \tag{9}$$

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

- All arguments get at most one label, $\forall i \sum_t y_{i,t} = 1$. Note we use equality, because you can always have the $\varnothing$ label.

[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.

- No duplicate argument classes, $\forall t \neq \varnothing, \sum_i y_{i,t} \leq 1$
- Overlapping arguments get at most one non-null label:

$$\forall \langle i, j \rangle : i \rightsquigarrow_\tau j, y_{i,\varnothing} + y_{j,\varnothing} \geq 1 \tag{10}$$

- Some arguments are forbidden, e.g. $\sum_i y_{i,A2} = 0$
- 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

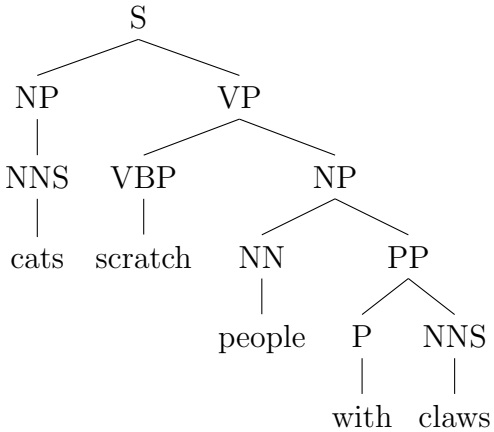
$$\forall i y_{i,C-A0} \leq \sum_{j<i} y_{j,A0}$$
$$\forall i y_{i,C-A1} \leq \sum_{j<i} y_{j,A1}, \ldots$$

All of the constraints are linear, meaning we can write them as $\mathbf{A}\boldsymbol{y} \leq \boldsymbol{b}$.

Unfortunately, constrained integer linear programming (ILP) is NP-Hard. It's easy to see this by reduction from TSP, boolean satisfiability, or minimum vertex cover. However, there are well-optimized algorithms for ILP, such as cutting plane and branch and bound, implemented in solvers like GLPK (free) or CPLEX (expensive). You just precompute the scores $\psi_{i,t} = \boldsymbol{\theta}^\mathsf{T} \boldsymbol{f}(y_{i,t}, \boldsymbol{w}, i, t)$ for all $\langle i, t \rangle$, write out the constraints, and send it to the solver.

Punyakanok et al (2004) "It only takes about 10 minutes to solve the inference problem for all 4305 sentences on a Pentium-III 800 Mhz machine in our experiments."

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.