Supplementary Note

February 28, 2015

1 Decoding for Individual Models

Under the independence hypothesis, the predicate-centric and argument-centric models can be seen as some kind of Markov models. We can use dynamic programming (DP) to decode the first two models. Here we illustrate the decoding method for the second order models, which can be easily extended for the kth order models. Given a sentence, our goal is to find a graph that maximizes

$$\begin{aligned} f(\boldsymbol{y}) &= \sum_{i=1}^{n} f_i(\boldsymbol{y}_i) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{m+1} \theta^\top \Phi(a_{j-1}, a_j, i, \mathbf{w}, \mathbf{p}). \end{aligned}$$

This equals to finding the optimal $\boldsymbol{y}_i$ for each i. Let L be the length of the sentence. To find the optimal $\boldsymbol{y}_i$, we can use a two dimensional array $s[1:L+1][1:L+1]$ as the DP table. $s[l][j]$ is the maximum score of the sequence $a_0, \cdots, a_l = j$. The initialization is

$$s[1][j] = \theta^\top \Phi(0, j, i, w),$$

and the functional equation is

$$s[l][j] = \max_{l-1 \le k \le j-1} s[l-1][k] + \theta^\top \Phi(k, j, i, w).$$

After the table is filled, the number of 1's of the optimal $\boldsymbol{y}_i$ is

$$\arg\max_{1 \le l \le L+1} s[l][L+1].$$

We can find the optimal $\boldsymbol{y}_i$ by maintaining a backtracing table during DP.

The decoding method for second order model cost time of $O(n^3)$ where n is the length of the sentence. For kth order model, it will cost $O(n^{k+1})$.

The decoding for the tree approximation model is simple. We train a tree parser with a converted tree corpus, and after the parser give the parse tree of a sentence, we convert the tree back to graph. The tree parser should and find a tree $\boldsymbol{t} = t(i, j)$ to maximize the score. Then we examine the labels on each arc to convert the tree back to graph.

2 Feature Templates

We implemented a perceptron with hash kernel (Bohnet, 2010), which allows us to extract a large amount of features. This technique saves us much memory and allow us to use rich contextual features for disambiguation. We also extract features from syntactic unlabeled dependency trees. The feature functions we use are shown in Table 1. Let $d(i, j) = i - j$ be the distance between i and j; $o(i, j, k) \in \{i < k < j, k < i < j, k < j < i\}$ be the relative order of i, j, k ($j > k$); $POS_{i \to j}$ be the POS-tag sequence along the path from i to j in the syntax tree; and $P_{i \to j}$ be the reduced POS-tag (here we mean the first letter of the POS-tag) sequence.

For predicate-centric model, all feature templates (i.e. $\Phi_p(prev, cur, i, \mathbf{w}, \mathbf{p})$) include:

- $\mathbf{g}_{\text{uni}}(i, \mathbf{w}, \mathbf{p}, d(i, cur))$;
- $\mathbf{g}_{\text{uni}}(i, \mathbf{w}, \mathbf{p}, o(i, cur, prev))$;
- $\mathbf{g}_{\text{uni}}(cur, \mathbf{w}, \mathbf{p}, d(i, cur))$;
- $\mathbf{g}_{\text{uni}}(cur, \mathbf{w}, \mathbf{p}, o(i, cur, prev))$;
- $\mathbf{g}_{\text{uni}}(prev, \mathbf{w}, \mathbf{p}, d(cur, prev))$;
- $\mathbf{g}_{\text{uni}}(prev, \mathbf{w}, \mathbf{p}, o(i, cur, prev))$;
- $\mathbf{f}_{\text{bi}}(w_i, p_i, w_{cur}, p_{cur}, d(i, cur))$;
- $\mathbf{f}_{\text{bi}}(w_i, p_i, w_{cur}, p_{cur}, o(i, cur, prev))$;
- $\mathbf{f}_{\text{bi}}(w_{prev}, p_{prev}, w_{cur}, p_{cur}, d(cur, prev))$;
- $\mathbf{f}_{\text{bi}}(w_{prev}, p_{prev}, w_{cur}, p_{cur}, o(i, cur, prev))$;
- $\mathbf{g}_{\text{bi}}(i, cur, \mathbf{w}, \mathbf{p}, d(i, cur))$;
- $\mathbf{g}_{\text{bi}}(i, cur, \mathbf{w}, \mathbf{p}, o(i, cur, prev))$;
- $\mathbf{f}_{\text{tri}}(w_{prev}, p_{prev}, p_{cur}, p_{cur}, w_i, p_i, o(i, cur, prev))$;

$\mathbf{f}_{\text{uni}}(w_i, p_i, d)$**:**	$w_i \circ d$; $p_i \circ d$; $w_i \circ p_i \circ d$
$\mathbf{f}_{\text{bi}}(w_i, p_i, w_j, p_j, d)$**:**	$w_i \circ p_i \circ w_j \circ p_j \circ d$; $w_i \circ p_i \circ w_j \circ d$; $w_i \circ p_i \circ p_j \circ d$; $w_i \circ w_j \circ p_j \circ d$; $p_i \circ w_j \circ p_j \circ d$; $w_i \circ w_j \circ d$; $p_i \circ p_j \circ d$
$\mathbf{f}'_{\text{bi}}(w_i, p_i, w_j, p_j, d)$**:**	$w_i \circ p_i \circ w_j \circ p_j \circ d$; $w_i \circ p_i \circ p_j \circ d$; $p_i \circ w_j \circ p_j \circ d$; $p_i \circ p_j \circ d$
$\mathbf{f}_{\text{tri}}(w_i, p_i, w_j, p_j, w_k, p_k, d)$**:**	$w_i \circ p_i \circ w_j \circ p_j \circ w_k \circ p_k \circ d$; $w_i \circ w_j \circ w_k \circ d$; $w_i \circ w_j \circ p_k \circ d$; $w_i \circ p_j \circ w_k \circ d$; $w_i \circ p_j \circ p_k \circ d$; $p_i \circ w_j \circ w_k \circ d$; $p_i \circ w_j \circ p_k \circ d$; $p_i \circ p_j \circ w_k \circ d$; $p_i \circ p_j \circ p_k \circ d$
$\mathbf{f}_{\text{path}}(w_i, p_i, w_j, p_j, P, d)$**:**	P; $P \circ d$; $w_i \circ p_i \circ w_j \circ p_j \circ P \circ d$; $w_i \circ p_i \circ w_j \circ P \circ d$; $w_i \circ p_i \circ p_j \circ P \circ d$; $w_i \circ w_j \circ p_j \circ P \circ d$; $p_i \circ w_j \circ p_j \circ P \circ d$; $w_i \circ w_j \circ P \circ d$; $p_i \circ p_j \circ P \circ d$
$\mathbf{g}_{\text{uni}}(i, \mathbf{w}, \mathbf{p}, d)$**:**	$\mathbf{f}_{\text{uni}}(w_i, p_i, d)$; $\mathbf{f}_{\text{uni}}(w_{i-1}, p_{i-1}, d)$; $\mathbf{f}_{\text{uni}}(w_{i+1}, p_{i+1}, d)$;
$\mathbf{g}_{\text{bi}}(i, j, \mathbf{w}, \mathbf{p}, d)$**:**	$\mathbf{f}'_{\text{bi}}(w_i, p_i, w_{i-1}, p_{i-1}, d)$; $\mathbf{f}'_{\text{bi}}(w_i, p_i, w_{i+1}, p_{i+1}, d)$; $\mathbf{f}'_{\text{bi}}(w_i, p_i, w_{j-1}, p_{j-1}, d)$; $\mathbf{f}'_{\text{bi}}(w_i, p_i, w_{j+1}, p_{j+1}, d)$; $\mathbf{f}'_{\text{bi}}(w_j, p_j, w_{j-1}, p_{j-1}, d)$; $\mathbf{f}'_{\text{bi}}(w_j, p_j, w_{j+1}, p_{j+1}, d)$; $\mathbf{f}'_{\text{bi}}(w_j, p_j, w_{i-1}, p_{i-1}, d)$; $\mathbf{f}'_{\text{bi}}(w_j, p_j, w_{i+1}, p_{i+1}, d)$;
$\mathbf{g}_{\text{tri}}(i, j, \mathbf{w}, \mathbf{p}, d)$**:**	$\mathbf{f}_{\text{tri}}(w_i, p_i, w_{i-1}, p_{i-1}, w_j, p_j, d)$; $\mathbf{f}_{\text{tri}}(w_i, p_i, w_{i+1}, p_{i+1}, w_j, p_j, d)$; $\mathbf{f}_{\text{tri}}(w_i, p_i, w_{j-1}, p_{j-1}, w_j, p_j, d)$; $\mathbf{f}_{\text{tri}}(w_i, p_i, w_{j+1}, p_{j+1}, w_j, p_j, d)$;

Table 1: Basic feature functions.

- $\mathbf{g}_{\text{tri}}(i, cur, \mathbf{w}, \mathbf{p}, o(i, cur, prev))$;
- $\mathbf{g}_{\text{tri}}(prev, cur, \mathbf{w}, \mathbf{p}, o(i, cur, prev))$;
- $\mathbf{f}_{\text{path}}(w_i, p_i, w_{cur}, p_{cur}, POS_{i \to cur}, d(i, cur))$;
- $\mathbf{f}_{\text{path}}(w_i, p_i, w_{cur}, p_{cur}, P_{i \to cur}, d(i, cur))$.

The Φ_a we use is the same as Φ_p described above. The Φ_t we use is the same as (Bohnet, 2010).

References

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational*

Linguistics (Coling 2010), pages 89–97. Coling 2010 Organizing Committee, Beijing, China.