Introduction

Transition-based Parsing with Neural Nets

Results and Analysis

Natural Language Understanding

Lecture 9: Dependency Parsing with Neural Networks

Frank Keller

School of Informatics
University of Edinburgh
keller@inf.ed.ac.uk

February 13, 2017

Dependency Parsing

Traditional dependency parsing (Nivre 2003):
- simple shift-reduce parser (see last lecture);
- classifier chooses which transition (parser action) to take for each word in the input sentence;
- features for classifier similar to MALT parser (last lecture):
  - word/PoS unigrams, bigrams, trigrams;
  - state of the parser;
  - dependency tree built so far.

Problems:
- feature templates need to be handcrafted;
- results in millions of features
- feature are sparse and slow to extract.

Chen and Manning (2014) propose:
- keep the simple shift-reduce parser;
- replace the classifier for transitions with a neural net;
- use dense features (embeddings) instead of sparse, handcrafted features.

Results:
- efficient parser (up to twice as fast as standard MALT parser);
- good performance (about 2% higher precision than MALT).

Reading: Chen and Manning (2014).
Goal of the network: predict correct transition $t \in T$, based on configuration $c$. Relevant information:

- words and PoS tags (e.g., has/VBZ);
- head of words with dependency label (e.g., nsubj, dobj);
- position of words on stack and buffer.

Correct transition: **SHIFT**

---

**Activation Function**

- **cube**
- **sigmoid**
- **tanh**
- **identity**

---

**Revision: Embeddings**

**CBOW** (Mikolov et al. 2013):

- $x_{ik}$: context words (one-hot)
- $h_i$: hidden units
- $y_j$: output units (one-hot)
- $W, W'$: weight matrices
- $V$: vocabulary size
- $N$: size of hidden layer
- $C$: number of context words

![Figure from Rong (2014).]
Revision: Embeddings

**CBOW** (Mikolov et al. 2013):

- $x_{ij}$: context words (one-hot)
- $h_i$: hidden units
- $y_j$: output units (one-hot)
- $W$, $W'$: weight matrices
- $V$: vocabulary size
- $N$: size of hidden layer
- $C$: number of context words

By **embedding** we mean the hidden layer $h_i$.

Chen and Manning (2014) use the following word embeddings $S^w$ (18 elements):

- top three words on stack and buffer: $s_1$, $s_2$, $s_3$, $b_1$, $b_2$, $b_3$;
- first and second leftmost/rightmost children of top two words on stack: $l_c(s_i)$, $r_c(s_i)$, $l_c(s_i)$, $r_c(s_i)$, $i = 1, 2$;
- leftmost of leftmost/rightmost of rightmost children of top two words on the stack: $l_c(l_c(s_i))$, $r_c(r_c(s_i))$, $i = 1, 2$.

Tag embeddings $S^t$ (18 elements): same as word embeddings.

Arc label embeddings $S^l$ (12 elements): same as word embeddings, excluding those the six words on the stack/buffer.

Generate examples $\{(c_i, t_i)\}_{i=1}^m$ from sentences with gold parse trees using **shortest stack** oracle (always prefers LEFT-ARC(l) over SHIFT), where $c_i$ is a configuration, $t_i \in T$ a transition.

Objective: minimize cross-entropy loss with $L_2$-regularization:

$$L(\theta) = -\sum_i \log \rho_{t_i} + \frac{\lambda}{2} ||\theta||^2$$

where $\rho_{t_i}$ is the probability of transition $t_i$ (from softmax layer), and $\theta$ is set of all parameters $\{W^w_t, W^l_t, W^f_t, b_1, W_2, E^w, E^t, E^l\}$.

Use pre-trained word embeddings to initialize $E^w$; use random initialization within $(-0.01, 0.01)$ for $E^t$ and $E^l$.

Word embeddings (Collobert et al. 2011) for English; 50-dimensional word2vec embeddings (Mikolov et al. 2013) for Chinese; compare with random initialization of $E_w$.

Mini-batched AdaGrad for optimization, dropout with 0.5 rate. Tune parameters on development set based UAS.

Hyper-parameters: embedding size $d = 50$, hidden layer size $h = 200$, regularization parameter $\lambda = 10^{-8}$, initial learning rate of AdaGrad $\alpha = 0.01$. 
Decoding

The parser performs greedy decoding:

- for each parsing step, extract all word, PoS, and label embeddings from current configuration \( c \);
- compute the hidden layer \( h(c) \);
- pick transition with the highest score:
  \[ t = \arg\max_t W^2(t, \cdot)h(c); \]
- execute transition \( c \rightarrow t(c) \).

Results: English with CoNLL Dependencies

<table>
<thead>
<tr>
<th>Parser</th>
<th>Dev UAS</th>
<th>Dev LAS</th>
<th>Test UAS</th>
<th>Test LAS</th>
<th>Speed (sent/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>standard</td>
<td>89.9</td>
<td>88.7</td>
<td>89.7</td>
<td>88.3</td>
<td>51</td>
</tr>
<tr>
<td>eager</td>
<td>90.3</td>
<td>89.2</td>
<td>89.9</td>
<td>88.6</td>
<td>63</td>
</tr>
<tr>
<td>Malt:sp</td>
<td>90.0</td>
<td>88.8</td>
<td>89.9</td>
<td>88.5</td>
<td>560</td>
</tr>
<tr>
<td>Malt:eager</td>
<td>90.1</td>
<td>88.9</td>
<td>90.1</td>
<td>88.7</td>
<td>535</td>
</tr>
<tr>
<td>MSTParser</td>
<td>92.1</td>
<td>90.8</td>
<td>92.0</td>
<td>90.5</td>
<td>12</td>
</tr>
<tr>
<td>Our parser</td>
<td>92.2</td>
<td>91.0</td>
<td>92.0</td>
<td>90.7</td>
<td>1013</td>
</tr>
</tbody>
</table>

Results: English with Stanford Dependencies

<table>
<thead>
<tr>
<th>Parser</th>
<th>Dev UAS</th>
<th>Dev LAS</th>
<th>Test UAS</th>
<th>Test LAS</th>
<th>Speed (sent/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>standard</td>
<td>90.2</td>
<td>87.8</td>
<td>89.4</td>
<td>87.3</td>
<td>26</td>
</tr>
<tr>
<td>eager</td>
<td>89.8</td>
<td>87.4</td>
<td>89.6</td>
<td>87.4</td>
<td>34</td>
</tr>
<tr>
<td>Malt:sp</td>
<td>89.8</td>
<td>87.2</td>
<td>89.3</td>
<td>86.9</td>
<td>469</td>
</tr>
<tr>
<td>Malt:eager</td>
<td>89.6</td>
<td>86.9</td>
<td>89.4</td>
<td>86.8</td>
<td>448</td>
</tr>
<tr>
<td>MSTParser</td>
<td>91.4</td>
<td>88.1</td>
<td>90.7</td>
<td>87.6</td>
<td>10</td>
</tr>
<tr>
<td>Our parser</td>
<td>92.0</td>
<td>89.7</td>
<td>91.8</td>
<td>89.6</td>
<td>654</td>
</tr>
</tbody>
</table>

Results: Chinese

<table>
<thead>
<tr>
<th>Parser</th>
<th>Dev UAS</th>
<th>Dev LAS</th>
<th>Test UAS</th>
<th>Test LAS</th>
<th>Speed (sent/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>standard</td>
<td>82.4</td>
<td>80.9</td>
<td>82.7</td>
<td>81.2</td>
<td>72</td>
</tr>
<tr>
<td>eager</td>
<td>81.1</td>
<td>79.7</td>
<td>80.3</td>
<td>78.7</td>
<td>80</td>
</tr>
<tr>
<td>Malt:sp</td>
<td>82.4</td>
<td>80.5</td>
<td>82.4</td>
<td>80.6</td>
<td>420</td>
</tr>
<tr>
<td>Malt:eager</td>
<td>81.2</td>
<td>79.3</td>
<td>80.2</td>
<td>78.4</td>
<td>393</td>
</tr>
<tr>
<td>MSTParser</td>
<td>84.0</td>
<td>82.1</td>
<td>83.0</td>
<td>81.2</td>
<td>6</td>
</tr>
<tr>
<td>Our parser</td>
<td>84.0</td>
<td>82.4</td>
<td>83.9</td>
<td>82.4</td>
<td>936</td>
</tr>
</tbody>
</table>
Effect of Activation Function

- Cube
- Tanh
- Sigmoid
- Identity

Pre-trained Embeddings vs. Random Initialization

- Pre-trained
- Random

Effect of PoS and Label Embeddings

- Word+POS+Label
- Word+POS
- Word+Label
- Word

Visualization of PoS Embeddings
Chen and Manning’s (2014) model builds on standard transition-based dependency parsing; uses neural net to select transitions; uses dense features (embeddings) instead of sparse, handcrafted features; embeddings over words, PoS, and arc labels; new cube activation function; good accuracy for English and Chinese dependency parsing; substantial improvement in speed.


