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
1 Introduction

2 Transition-based Parsing with Neural Nets
   - Network Architecture
   - Embeddings
   - Training and Decoding

3 Results and Analysis
   - Results
   - Analysis

Reading: Chen and Manning (2014).
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).
Network Architecture

Goal of the network: predict correct transition $t \in \mathcal{T}$, based on configuration $c$. Relevant information:

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

Correct transition: SHIFT

<table>
<thead>
<tr>
<th>Stack</th>
<th>Buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT has_VBZ</td>
<td>control_NN</td>
</tr>
<tr>
<td>good_JJ</td>
<td></td>
</tr>
<tr>
<td>nsubj He_PRP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Network Architecture

Softmax layer:
\[ p = \text{softmax}(W_2h) \]

Hidden layer:
\[ h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3 \]

Input layer: \([x^w, x^t, x^l]\)

Configuration

```
ROOT  has_VBZ  good_JJ
He_PRP
nsubj
```

Stack

Buffer

words
POS tags
arc labels
Introduction

Transition-based Parsing with Neural Nets

Results and Analysis

Network Architecture

Embeddings

Training and Decoding

Activation Function

\[
\begin{align*}
-1 & -0.8 & -0.6 & -0.4 & -0.2 & 0 & 0.2 & 0.4 & 0.6 & 0.8 & 1 \\
-1 & -0.5 & 0.5 & 1 \\
cube & sigmoid & tanh & identity
\end{align*}
\]
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):

- \( x_{ik} \): 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

By embedding we mean the hidden layer \( h \)!

[Figure from Rong (2014).]
Chen and Manning (2014) use the following word embeddings $S^w$ (18 elements):

1. top three words on stack and buffer: $s_1$, $s_2$, $s_3$, $b_1$, $b_2$, $b_3$;
2. first and second leftmost/rightmost children of top two words on stack: $lc_1(s_i)$, $rc_1(s_i)$, $lc_2(s_i)$, $rc_2(s_i)$, $i = 1, 2$;
3. leftmost of leftmost/rightmost of rightmost children of top two words on the stack: $lc_1(lc_1(s_i))$, $rc_1(rc_1(s_i))$, $i = 1, 2$.

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.
Training

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, $t_i \in \mathcal{T}$ a transition.

Objective: minimize cross-entropy loss with $l_2$-regularization:

$$L(\theta) = - \sum_{i} \log p_{t_i} + \frac{\lambda}{2} ||\theta||^2$$

where $p_{t_i}$ is the probability of transition $t_i$ (from softmax layer), and $\theta$ is set of all parameters $\{W^w_1, W^t_1, W^l_1, b_1, W_2, E^w, E^t, E^l\}$.
Training

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</th>
<th></th>
<th>Test</th>
<th></th>
<th>Speed (sent/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UAS</td>
<td>LAS</td>
<td>UAS</td>
<td>LAS</td>
<td></td>
</tr>
<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

- **PTB:CD**
  - Cube: 90
  - Tanh: 85
  - Sigmoid: 80
  - Identity: 75

- **PTB:SD**
  - Cube: 90
  - Tanh: 85
  - Sigmoid: 80
  - Identity: 75

- **CTB**
  - Cube: 80
  - Tanh: 75
  - Sigmoid: 70
  - Identity: 65
Pre-trained Embeddings vs. Random Initialization

![Graph showing comparison between pre-trained and random initialization for PTB:CD, PTB:SD, and CTB datasets. The x-axis represents the datasets (PTB:CD, PTB:SD, CTB), and the y-axis represents the UAS score ranging from 80 to 90. The bars for pre-trained embeddings are darker blue, and the bars for random initialization are lighter red. The UAS scores for pre-trained embeddings are consistently higher than those for random initialization.]
Effect of PoS and Label Embeddings

PTB:CD PTB:SD CTB
70 
75 
80 
85 
90 
95 UAS score
word+POS+label word+POS word+label word

- Blue: word+POS+label
- Red: word+POS
- Orange: word+label
- Black: 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.


