Projective Dependency Parsing with Perceptron

Xavier Carreras, Mihai Surdeanu, and Lluís Màrquez

Technical University of Catalonia
{carreras,surdeanu,lluism}@lsi.upc.edu

8th June 2006
Outline

Introduction

Parsing and Learning
  Parsing Model
  Parsing Algorithm
  Global Perceptron Learning Algorithm

Features

Experiments and Results
  Results
  Discussion
Outline

Introduction

Parsing and Learning
  Parsing Model
  Parsing Algorithm
  Global Perceptron Learning Algorithm

Features

Experiments and Results
  Results
  Discussion
Introduction

- **Motivation**
  - Blind treatment of multilingual data
  - Use well-known components

- **Our Dependency Parsing Learning Architecture:**
  - Eisner dep-parsing algorithm, for projective structures
  - Perceptron learning algorithm, run globally
  - Features: state-of-the-art, with some new ones

- **In CoNLL-X data, we achieve moderate performance:**
  - 74.72 of overall labeled attachment score
  - 10th position in the ranking
Outline

Introduction

Parsing and Learning
  Parsing Model
  Parsing Algorithm
  Global Perceptron Learning Algorithm

Features

Experiments and Results
  Results
  Discussion
A dependency tree is decomposed into labeled dependencies, each of the form $[h, m, l]$ where:

- $h$ is the position of the head word
- $m$ is the position of the modifier word
- $l$ is the label of the dependency

Given a sentence $x$ the parser computes:

$$d_{\text{parser}}(x, w) = \arg \max_{y \in \mathcal{Y}(x)} \sum_{[h, m, l] \in y} \text{score}([h, m, l], x, y, w)$$

$$= \arg \max_{y \in \mathcal{Y}(x)} \sum_{[h, m, l] \in y} w^l \cdot \phi([h, m], x, y)$$

- $w = (w^1, \ldots, w^l, \ldots, w^L)$ is the learned weight vector
- $\phi$ is the feature extraction function, given a priori
The Parsing Algorithm of Eisner (1996)

- Assumes that dependency structures are projective; in CoNLL data, this only holds for Chinese
- Bottom-up dynamic programming algorithm
- In a given span from word $s$ to word $e$:
  1. Look for the optimal point giving internal structures:
     ![Diagram showing internal structures]
  2. Look for the best label to connect the structures:
     ![Diagram showing label connections]
A third step assembles two dependency structures without using learning.
Perceptron Learning

- Global Perceptron (Collins 2002): trains the weight vector dependently of the parsing algorithm.
- A very simple online learning algorithm: it corrects the mistakes seen after a training sentence is parsed.

\[
\begin{align*}
\mathbf{w} &= \mathbf{0} \\
\text{for } t &= 1 \text{ to } T \\
\quad &\text{foreach training example } (x, y) \text{ do} \\
\quad &\quad \hat{y} = \text{dparser}(x, \mathbf{w}) \\
\quad &\quad \text{foreach } [h, m, l] \in y \setminus \hat{y} \text{ do} \quad /* \text{missed deps} */ \\
\quad &\quad \quad w_l' = w_l' + \phi(h, m, x, \hat{y}) \\
\quad &\quad \text{foreach } [h, m, l] \in \hat{y} \setminus y \text{ do} \quad /* \text{over-predicted deps} */ \\
\quad &\quad \quad w_l' = w_l' - \phi(h, m, x, \hat{y}) \\
\quad &\text{return } \mathbf{w}
\end{align*}
\]
Outline

Introduction

Parsing and Learning
  Parsing Model
  Parsing Algorithm
  Global Perceptron Learning Algorithm

Features

Experiments and Results
  Results
  Discussion
Feature Extraction Function

\[ \phi(h, m, x, y) : \text{represents in a feature vector a dependency from word positions } m \text{ to } h, \text{ in the context of a sentence } x \text{ and a dependency tree } y \]

\[
\phi(h, m, x, y) = \phi_{\text{token}}(x, h, "head") + \phi_{\text{tctx}}(x, h, "head") \\
+ \phi_{\text{token}}(x, m, "mod") + \phi_{\text{tctx}}(x, m, "mod") \\
+ \phi_{\text{dep}}(x, mM_{h,m}, d_{h,m}) + \phi_{\text{dctx}}(x, mM_{h,m}, d_{h,m}) \\
+ \phi_{\text{dist}}(x, mM_{h,m}, d_{h,m}) + \phi_{\text{runtime}}(x, y, h, m, d_{h,m})
\]

where
- \( mM_{h,m} \) is a shorthand for the tuple \( \langle \min(h, m), \max(h, m) \rangle \)
- \( d_{h,m} \) indicates the direction of the dependency
Context-Independent Token Features

- Represent a token $i$
- $type$ indicates the type of token being represented, i.e. “head” or “mod”
- Novel features are in red.

\[
\phi_{\text{token}}(x, i, \text{type})
\]

- $type \cdot \text{word}(x_i)$
- $type \cdot \text{lemma}(x_i)$
- $type \cdot \text{cpos}(x_i)$
- $type \cdot \text{fpos}(x_i)$

\[
\text{foreach } f \in \text{morphosynt}(x_i) : type \cdot f
\]

- $type \cdot \text{word}(x_i) \cdot \text{cpos}(x_i)$

\[
\text{foreach } f \in \text{morphosynt}(x_i) : type \cdot \text{word}(x_i) \cdot f
\]
Context-Dependent Token Features

- Represent the context of a token $x_i$
- The function extracts token features of surrounding tokens
- It also conjoins some selected features along the window

$$
\phi_{tctx}(x, i, \text{type})
$$

$$
\begin{align*}
&\phi_{token}(x, i - 1, \text{type} \cdot \text{string}(-1)) \\
&\phi_{token}(x, i - 2, \text{type} \cdot \text{string}(-2)) \\
&\phi_{token}(x, i + 1, \text{type} \cdot \text{string}(-1)) \\
&\phi_{token}(x, i + 2, \text{type} \cdot \text{string}(-2)) \\
&\quad \text{type} \cdot \text{cpos}(x_i) \cdot \text{cpos}(x_{i-1}) \\
&\quad \text{type} \cdot \text{cpos}(x_i) \cdot \text{cpos}(x_{i-1}) \cdot \text{cpos}(x_{i-2}) \\
&\quad \text{type} \cdot \text{cpos}(x_i) \cdot \text{cpos}(x_{i+1}) \\
&\quad \text{type} \cdot \text{cpos}(x_i) \cdot \text{cpos}(x_{i+1}) \cdot \text{cpos}(x_{i+2})
\end{align*}
$$
Features of the two tokens involved in a dependency relation

$\phi_{\text{dep}}(x, i, j, \text{dir})$

<table>
<thead>
<tr>
<th>$\text{dir}$</th>
<th>$\text{word}(x_i)$</th>
<th>$\text{cpos}(x_i)$</th>
<th>$\text{word}(x_j)$</th>
<th>$\text{cpos}(x_j)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{dir}$</td>
<td>$\text{word}(x_i)$</td>
<td>$\text{cpos}(x_i)$</td>
<td>$\text{word}(x_j)$</td>
<td>$\text{cpos}(x_j)$</td>
</tr>
<tr>
<td>$\text{dir}$</td>
<td>$\text{cpos}(x_i)$</td>
<td>$\text{word}(x_j)$</td>
<td>$\text{cpos}(x_j)$</td>
<td></td>
</tr>
<tr>
<td>$\text{dir}$</td>
<td>$\text{word}(x_i)$</td>
<td>$\text{word}(x_j)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{dir}$</td>
<td>$\text{cpos}(x_i)$</td>
<td>$\text{cpos}(x_j)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\text{dir}$ indicates whether the relation is left-to-right or right-to-left.
Context-Dependent Dependency Features

- Capture the context of the two tokens involved in a relation
- \textit{dir} indicates whether the relation is left-to-right or right-to-left

\[
\phi_{\text{dctx}}(x, i, j, \text{dir}) = \begin{align*}
\text{dir} \cdot \text{cpos}(x_i) \cdot \text{cpos}(x_{i+1}) \cdot \text{cpos}(x_{j-1}) \cdot \text{cpos}(x_j) \\
\text{dir} \cdot \text{cpos}(x_{i-1}) \cdot \text{cpos}(x_i) \cdot \text{cpos}(x_{j-1}) \cdot \text{cpos}(x_j) \\
\text{dir} \cdot \text{cpos}(x_i) \cdot \text{cpos}(x_{i+1}) \cdot \text{cpos}(x_j) \cdot \text{cpos}(x_{j+1}) \\
\text{dir} \cdot \text{cpos}(x_{i-1}) \cdot \text{cpos}(x_i) \cdot \text{cpos}(x_j) \cdot \text{cpos}(x_{j+1})
\end{align*}
\]
Surface Distance Features

- Features on the surface tokens found within a dependency relation
- Numeric features are discretized using “binning” to a small number of intervals

<table>
<thead>
<tr>
<th>( \phi_{\text{dist}}(x, i, j, \text{dir}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \text{foreach}(k \in (i, j)): \ \text{dir} \cdot \text{cpos}(x_i) \cdot \text{cpos}(x_k) \cdot \text{cpos}(x_j) ]</td>
</tr>
<tr>
<td>number of tokens between ( i ) and ( j )</td>
</tr>
<tr>
<td>number of verbs between ( i ) and ( j )</td>
</tr>
<tr>
<td>number of coordinations between ( i ) and ( j )</td>
</tr>
<tr>
<td>number of punctuations signs between ( i ) and ( j )</td>
</tr>
</tbody>
</table>
Runtime Features

- Capture the labels of the dependencies that attach to the head word
- This information is available in the dynamic programming matrix of the parsing algorithm

\[
\phi_{\text{runtime}}(x, y, h, m, \text{dir})
\]

<table>
<thead>
<tr>
<th>(\text{foreach} \ i, 1 \leq i \leq S)</th>
<th>(\text{dir} \cdot \text{cpos}(x_h) \cdot \text{cpos}(x_m) \cdot l_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{dir} \cdot \text{cpos}(x_h) \cdot \text{cpos}(x_m) \cdot l_1)</td>
<td></td>
</tr>
<tr>
<td>(\text{dir} \cdot \text{cpos}(x_h) \cdot \text{cpos}(x_m) \cdot l_1 \cdot l_2)</td>
<td></td>
</tr>
<tr>
<td>(\text{dir} \cdot \text{cpos}(x_h) \cdot \text{cpos}(x_m) \cdot l_1 \cdot l_2 \cdot l_3)</td>
<td></td>
</tr>
<tr>
<td>(\text{dir} \cdot \text{cpos}(x_h) \cdot \text{cpos}(x_m) \cdot l_1 \cdot l_2 \cdot l_3 \cdot l_4)</td>
<td></td>
</tr>
</tbody>
</table>
Outline

Introduction

Parsing and Learning
  Parsing Model
  Parsing Algorithm
  Global Perceptron Learning Algorithm

Features

Experiments and Results
  Results
  Discussion
## Results

<table>
<thead>
<tr>
<th>Language</th>
<th>GOLD</th>
<th>UAS</th>
<th>LAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japanese</td>
<td>99.16</td>
<td>90.79</td>
<td>88.13</td>
</tr>
<tr>
<td>Chinese</td>
<td>100.0</td>
<td>88.65</td>
<td>83.68</td>
</tr>
<tr>
<td>Portuguese</td>
<td>98.54</td>
<td>87.76</td>
<td>83.37</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>99.56</td>
<td>88.81</td>
<td>83.30</td>
</tr>
<tr>
<td>German</td>
<td>98.84</td>
<td>85.90</td>
<td>82.41</td>
</tr>
<tr>
<td>Danish</td>
<td>99.18</td>
<td>85.67</td>
<td>79.74</td>
</tr>
<tr>
<td>Swedish</td>
<td>99.64</td>
<td>85.54</td>
<td>78.65</td>
</tr>
<tr>
<td>Spanish</td>
<td>99.96</td>
<td>80.77</td>
<td>77.16</td>
</tr>
<tr>
<td>Czech</td>
<td>97.78</td>
<td>77.44</td>
<td>68.82</td>
</tr>
<tr>
<td>Slovene</td>
<td>98.38</td>
<td>77.72</td>
<td>68.43</td>
</tr>
<tr>
<td>Dutch</td>
<td>94.56</td>
<td>71.39</td>
<td>67.25</td>
</tr>
<tr>
<td>Arabic</td>
<td>99.76</td>
<td>72.65</td>
<td>60.94</td>
</tr>
<tr>
<td>Turkish</td>
<td>98.41</td>
<td>70.05</td>
<td>58.06</td>
</tr>
<tr>
<td>Overall</td>
<td>98.68</td>
<td>81.19</td>
<td>74.72</td>
</tr>
</tbody>
</table>
Feature Analysis

<table>
<thead>
<tr>
<th>Language</th>
<th>$\phi_{token}$</th>
<th>$+\phi_{dep}$</th>
<th>$+\phi_{tctx}$</th>
<th>$+\phi_{dist}$</th>
<th>$+\phi_{runtime}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japanese</td>
<td>38.78</td>
<td>78.13</td>
<td>86.87</td>
<td>88.27</td>
<td>88.13</td>
</tr>
<tr>
<td>Portuguese</td>
<td>47.10</td>
<td>64.74</td>
<td>80.89</td>
<td>82.89</td>
<td>83.37</td>
</tr>
<tr>
<td>Spanish</td>
<td>12.80</td>
<td>53.80</td>
<td>68.18</td>
<td>74.27</td>
<td>77.16</td>
</tr>
<tr>
<td>Turkish</td>
<td>33.02</td>
<td>48.00</td>
<td>55.33</td>
<td>57.16</td>
<td>58.06</td>
</tr>
</tbody>
</table>

- This table shows LAS at increasing feature configurations.
- All families of feature patterns help significantly.
Errors Caused by 4 Factors

1. Size of training sets: accuracy below 70% for languages with small training sets: Turkish, Arabic, and Slovene.

2. Modeling large distance dependencies: our distance features ($\phi_{dist}$) are insufficient to model well large-distance dependencies:

<table>
<thead>
<tr>
<th></th>
<th>to root</th>
<th>1</th>
<th>2</th>
<th>3 – 6</th>
<th>&gt;= 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spanish</td>
<td>83.04</td>
<td>93.44</td>
<td>86.46</td>
<td>69.97</td>
<td>61.48</td>
</tr>
<tr>
<td>Portuguese</td>
<td>90.81</td>
<td>96.49</td>
<td>90.79</td>
<td>74.76</td>
<td>69.01</td>
</tr>
</tbody>
</table>

3. Modeling context: our context features ($\phi_{dctx}$, $\phi_{tctx}$, and $\phi_{runtime}$) do not capture complex dependencies. Top 5 focus words with most errors:
   - Spanish: “y”, “de”, “a”, “en”, and “que”
   - Portuguese: “em”, “de”, “a”, “e”, and “para”

4. Projectivity assumption: Dutch is the language with most crossing dependencies in this evaluation, and the accuracy we obtain is below 70%.
Thanks!