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

Outline

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

Parsing Model

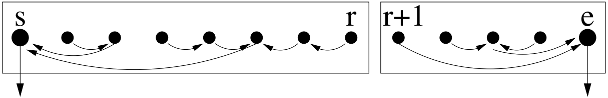
- A dependency tree is decomposed into labeled dependencies, each of the form $[h, m, l]$ where :
  - h is the position of the head word
  - m is the position of the modifier word
  - l is the label of the dependency
- Given a sentence x the parser computes:

$$\begin{aligned}
\text{dparser}(x, \mathbf{w}) &= \arg\max_{y \in \mathcal{Y}(x)} \text{score}(x, y, \mathbf{w}) \\
&= \arg\max_{y \in \mathcal{Y}(x)} \sum_{[h,m,l] \in y} \text{score}([h, m, l], x, y, \mathbf{w}) \\
&= \arg\max_{y \in \mathcal{Y}(x)} \sum_{[h,m,l] \in y} \mathbf{w}^l \cdot \phi([h, m], x, y)
\end{aligned}$$

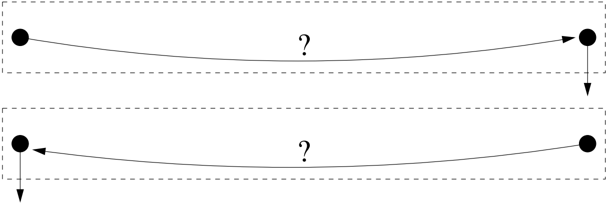
- $\mathbf{w} = (\mathbf{w}^1, \ldots, \mathbf{w}^l, \ldots, \mathbf{w}^L)$ is the learned weight vector
- ϕ is the feature extraction function, given a priori

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:

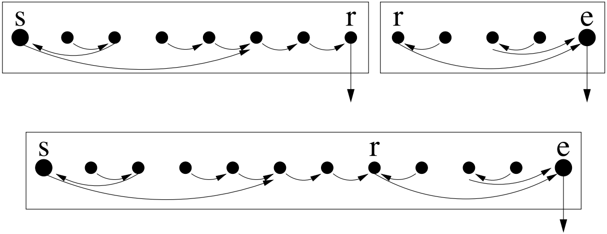
     

  2. Look for the best label to connect the structures:

The Parsing Algorithm of Eisner (1996) (II)

- 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.

$\mathbf{w} = \mathbf{0}$
for $t = 1$ *to* T
  foreach training example (x, y) *do*
    $\hat{y} = \mathrm{dparser}(x, \mathbf{w})$
    foreach $[h, m, l] \in y \backslash \hat{y}$ *do*   /* missed deps */
      $\mathbf{w}^l = \mathbf{w}^l + \phi(h, m, x, \hat{y})$
    foreach $[h, m, l] \in \hat{y} \backslash y$ *do*   /* over-predicted deps */
      $\mathbf{w}^l = \mathbf{w}^l - \phi(h, m, x, \hat{y})$
return $\mathbf{w}$

Outline

Feature Extraction Function

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

$$
\begin{aligned}
\phi(h, m, x, y) \;=\; & \phi_{token}(x, h, \text{“}head\text{”}) + \phi_{tctx}(x, h, \text{“}head\text{”}) \\
+\; & \phi_{token}(x, m, \text{“}mod\text{”}) + \phi_{tctx}(x, m, \text{“}mod\text{”}) \\
+\; & \phi_{dep}(x, mM_{h,m}, d_{h,m}) + \phi_{dctx}(x, mM_{h,m}, d_{h,m}) \\
+\; & \phi_{dist}(x, mM_{h,m}, d_{h,m}) + \phi_{runtime}(x, y, h, m, d_{h,m})
\end{aligned}
$$

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.

$\phi_{\textbf{token}}(\mathbf{x}, \mathbf{i}, \mathbf{type})$
$\mathit{type} \cdot \mathrm{word}(x_i)$
$\mathit{type} \cdot \mathrm{lemma}(x_i)$
$\mathit{type} \cdot \mathrm{cpos}(x_i)$
$\mathit{type} \cdot \mathrm{fpos}(x_i)$
foreach $f \in \mathrm{morphosynt}(x_i)$: $\mathit{type} \cdot f$
$\mathit{type} \cdot \mathrm{word}(x_i) \cdot \mathrm{cpos}(x_i)$
foreach $f \in \mathrm{morphosynt}(x_i)$: $\mathit{type} \cdot \mathrm{word}(x_i) \cdot f$

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

$\phi_{\mathbf{tctx}}(\mathbf{x}, \mathbf{i}, \mathbf{type})$
$\phi_{token}(x, i-1, type \cdot string(-1))$
$\phi_{token}(x, i-2, type \cdot string(-2))$
$\phi_{token}(x, i+1, type \cdot string(-1))$
$\phi_{token}(x, i+2, type \cdot string(-2))$
$type \cdot \mathrm{cpos}(x_i) \cdot \mathrm{cpos}(x_{i-1})$
$type \cdot \mathrm{cpos}(x_i) \cdot \mathrm{cpos}(x_{i-1}) \cdot \mathrm{cpos}(x_{i-2})$
$type \cdot \mathrm{cpos}(x_i) \cdot \mathrm{cpos}(x_{i+1})$
$type \cdot \mathrm{cpos}(x_i) \cdot \mathrm{cpos}(x_{i+1}) \cdot \mathrm{cpos}(x_{i+2})$

Context-Independent Dependency Features

- Features of the two tokens involved in a dependency relation
- *dir* indicates whether the relation is left-to-right or right-to-left

$\phi_{\mathbf{dep}}(\mathbf{x}, \mathbf{i}, \mathbf{j}, \mathbf{dir})$
$dir \cdot \mathrm{word}(x_i) \cdot \mathrm{cpos}(x_i) \cdot \mathrm{word}(x_j) \cdot \mathrm{cpos}(x_j)$
$dir \cdot \mathrm{cpos}(x_i) \cdot \mathrm{word}(x_j) \cdot \mathrm{cpos}(x_j)$
$dir \cdot \mathrm{word}(x_i) \cdot \mathrm{word}(x_j) \cdot \mathrm{cpos}(x_j)$
$dir \cdot \mathrm{word}(x_i) \cdot \mathrm{cpos}(x_i) \cdot \mathrm{cpos}(x_j)$
$dir \cdot \mathrm{word}(x_i) \cdot \mathrm{cpos}(x_i) \cdot \mathrm{word}(x_j)$
$dir \cdot \mathrm{word}(x_i) \cdot \mathrm{word}(x_j)$
$dir \cdot \mathrm{cpos}(x_i) \cdot \mathrm{cpos}(x_j)$

Context-Dependent Dependency Features

- Capture the context of the two tokens involved in a relation
- *dir* indicates whether the relation is left-to-right or right-to-left

$\phi_{\mathbf{dctx}}(\mathbf{x}, \mathbf{i}, \mathbf{j}, \mathbf{dir})$
$dir \cdot \mathrm{cpos}(x_i) \cdot \mathrm{cpos}(x_{i+1}) \cdot \mathrm{cpos}(x_{j-1}) \cdot \mathrm{cpos}(x_j)$
$dir \cdot \mathrm{cpos}(x_{i-1}) \cdot \mathrm{cpos}(x_i) \cdot \mathrm{cpos}(x_{j-1}) \cdot \mathrm{cpos}(x_j)$
$dir \cdot \mathrm{cpos}(x_i) \cdot \mathrm{cpos}(x_{i+1}) \cdot \mathrm{cpos}(x_j) \cdot \mathrm{cpos}(x_{j+1})$
$dir \cdot \mathrm{cpos}(x_{i-1}) \cdot \mathrm{cpos}(x_i) \cdot \mathrm{cpos}(x_j) \cdot \mathrm{cpos}(x_{j+1})$

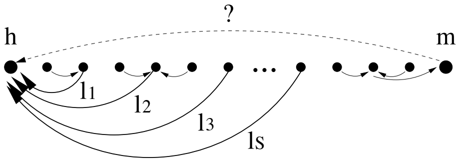
Surface Distance Features

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

$\phi_{\mathbf{dist}}(\mathbf{x}, \mathbf{i}, \mathbf{j}, \mathbf{dir})$
foreach(k $\in (i, j)$): *dir* $\cdot \mathrm{cpos}(x_i) \cdot \mathrm{cpos}(x_k) \cdot \mathrm{cpos}(x_j)$
number of tokens between *i* and *j*
number of verbs between *i* and *j*
number of coordinations between *i* and *j*
number of punctuations signs between *i* and *j*

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



$\phi_{\textbf{runtime}}(\mathbf{x}, \mathbf{y}, \mathbf{h}, \mathbf{m}, \mathbf{dir})$
foreach i, $1 \leq i \leq S$: $dir \cdot \mathrm{cpos}(x_h) \cdot \mathrm{cpos}(x_m) \cdot l_i$
$dir \cdot \mathrm{cpos}(x_h) \cdot \mathrm{cpos}(x_m) \cdot l_1$ $dir \cdot \mathrm{cpos}(x_h) \cdot \mathrm{cpos}(x_m) \cdot l_1 \cdot l_2$ $dir \cdot \mathrm{cpos}(x_h) \cdot \mathrm{cpos}(x_m) \cdot l_1 \cdot l_2 \cdot l_3$ $dir \cdot \mathrm{cpos}(x_h) \cdot \mathrm{cpos}(x_m) \cdot l_1 \cdot l_2 \cdot l_3 \cdot l_4$

Outline

Results

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

Feature Analysis

	$\phi_{\textbf{token}}$	$+\phi_{\textbf{dep}}$	$+\phi_{\textbf{tctx}}$ $+\phi_{\textbf{dctx}}$	$+\phi_{\textbf{dist}}$	$+\phi_{\textbf{runtime}}$
Japanese	38.78	78.13	86.87	88.27	88.13
Portuguese	47.10	64.74	80.89	82.89	83.37
Spanish	12.80	53.80	68.18	74.27	77.16
Turkish	33.02	48.00	55.33	57.16	58.06

- 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 (ϕ_{dist}) are insufficient to model well large-distance dependencies:

	to root	1	2	$3-6$	$>=7$
Spanish	83.04	93.44	86.46	69.97	61.48
Portuguese	90.81	96.49	90.79	74.76	69.01

3. Modeling context: our context features (ϕ_{dctx}, ϕ_{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!