Lecture 9: Dependency Parsing

They solved the problem with statistics.

CoNLL Format

```
1  Cathy  Cathy  N  N  eigen|ev|out  2  ev
2  zag  zie  V  V  trans|ov|lo2o3|ev  0  hoopt
3  hen  hen  Pron  Pron  per|3|ev|datafeer  2  obj
4  wild  wild  Adj  Adj  ette|tel|convrv  5  mod
5  zwaaie  zwaaie  N  N  assent|ev|neut  2  ve
6  .  .  Punct  Punct  punct  5  punct
```

(Non-)Projectivity

- Crossing Arcs needed to account for non-projective constructions
- Fairly rare in English but can be common in other languages (e.g. Czech):

```
root  John  saw  a  dog  yesterday  which  was  a  Yorkshire  Terrier
```

```
root  O  to  nové  většinou  nemá  ani  zájem  a  taky  na  to  většinou  nemá  peníze
```

http://ilk.uvt.nl/conll/
Formal Conditions

- For a dependency graph $G = (V, A)$
- With label set $L = \{l_1, \ldots, l_{|L|}\}$
- $G$ is (weakly) connected:
  - If $i, j \in V$, $i \rightarrow^* j$.
- $G$ is acyclic:
  - If $i \rightarrow j$, then not $j \rightarrow^* i$.
- $G$ obeys the single-head constraint:
  - If $i \rightarrow j$, then not $i' \rightarrow j$, for any $i' \neq i$.
- $G$ is projective:
  - If $i \rightarrow j$, then $i \rightarrow^* i'$, for any $i'$ such that $i < i' < j$ or $j < i' < i$.

Styles of Dependency Parsing

- **Transition-Based (tr)**
  - Fast, greedy, linear time inference algorithms
  - Trained for greedy search
  - Beam search

- **Graph-Based (gr)**
  - Slower, exhaustive, dynamic programming inference algorithms
  - Higher-order factorizations

![Accuracy vs. Time Graph](Image)

- **Arc-Factored Models**
  - Assumes that the score / probability / weight of a dependency graph factors by its arcs
  \[ w(G) = \prod_{(i,j,k) \in G} w_{ij} \]
  - $w_{ij}$ is the weight of creating a dependency from word $w_i$ to $w_j$ with label $l_k$
  - Thus there is an assumption that each dependency decision is independent
    - Strong assumption! Will address this later.

Graph-based Parsing

- Assumes that scores factor over the tree
- **Arc-factored models**
  - Score(tree) = $\sum$ edges
    \[ I \text{ washed dishes with detergent} = I \text{ washed} + \text{ washed dishes} + \text{ washed with} + \text{ with detergent} \]
As McGwire neared, fans went wild.
As McGwire neared, fans went wild.
**Graph-Based Parsing**

- **Searching**
- **Scoring**

\[ y^* = \arg \max_{y \in \mathcal{Y}} \sum_{p \in \mathcal{P}(y)} w \cdot \phi(p, x) \]

- Parsing algorithm
- Set of possible trees
- Factorization: local parts of the tree
- Features

**Arc-factored Projective Parsing**

- All projective graphs can be written as the combination of two smaller adjacent graphs

**Arc-factored Projective Parsing**

- Chart item filled in a bottom-up manner
  - First do all strings of length 1, then 2, etc. just like CKY

- Weight of new item: \( \max_{i, j, k} w(A) \times w(B) \times w_{h, h'} \)
- Algorithm runs in \( O(|L|n^2) \)
- Use back-pointers to extract best parse (like CKY)

**Eisner Algorithm**

- \( O(|L|n^2) \) is not that good
- [Eisner 1996] showed how this can be reduced to \( O(|L|n^3) \)
  - Key: split items so that sub-roots are always on periphery
Eisner First-Order Parsing

In practice also left arc version

As McGwire neared, fans went wild

As McGwire neared, fans went wild

As McGwire neared, fans went wild
As McGwire neared, fans went wild.
**Eisner First-Order Parsing**

As McGwire neared, fans went wild.

**Eisner Algorithm Pseudo Code**

Initialization: $C[s][a][d][c] = 0 \ \forall s, d, c$

for $k : 1..n$
for $s : 1..n$
    $t = s + k$
    if $t > n$ then break

% First: create incomplete items
$C[s][t][<-][0] = \max_{s \leq r < 1} \left( C[a][r][<-][1] + C[r + 1][t][<-][1] + s(t, s) \right)$
$C[s][t][<-][0] = \max_{s \leq r < 1} \left( C[a][r][<-][1] + C[r + 1][t][<-][1] + s(s, t) \right)$

% Second: create complete items
$C[s][t][<-][1] = \max_{s \leq r < 1} \left( C[a][r][<-][1] + C[r][t][<-][0] \right)$
$C[s][t][<-][1] = \max_{s \leq r < 1} \left( C[a][r][<-][0] + C[r][t][<-][1] \right)$

end for
end for

**Maximum Spanning Trees (MSTs)**

- A directed spanning tree of a (multi-)digraph $G = (V, A)$, is a subgraph $G' = (V', A')$ such that:
  - $V' = V$
  - $A' \subseteq A$, and $|A'| = |V'| - 1$
  - $G'$ is a tree (acyclic)

- A spanning tree of the following (multi-)digraphs

Can use MST algorithms for nonprojective parsing!
Chu-Liu-Edmonds

- \( x = \text{root} \), John saw Mary

Find Cycle and Contract

- If not a tree, identify cycle and contract
- Recalculate arc weights into and out-of cycle

Recalculate Edge Weights

- Incoming arc weights
  - Equal to the weight of best spanning tree that includes head of incoming arc, and all nodes in cycle
  - root → saw → John is 40 (**) 
  - root → John → saw is 29

Chu-Liu-Edmonds

- Find highest scoring incoming arc for each vertex

If this is a tree, then we have found MST!!
Theorem

The weight of the MST of this contracted graph is equal to the weight of the MST for the original graph.

Therefore, recursively call algorithm on new graph

Chu-Liu-Edmonds PseudoCode

\[
\text{Chu-Liu-Edmonds}(G, w) =
1. \text{Let } M = \{(i^*, j) : j \in V_x, i^* = \arg \max_{j} w_{ij}\}
2. \text{Let } G_M = (V_x, M)
3. \text{If } G_M \text{ has no cycles, then it is an MST: return } G_M
4. \text{Otherwise, find a cycle } C \text{ in } G
5. \text{Let } < G_C, c, ma >= \text{ contract}(G, C, w)
6. \text{Let } G = \text{Chu-Liu-Edmonds}(G_C, w)
7. \text{Find vertex } i \in C \text{ such that } (i', i) \in G \text{ and } ma(i', i) = i
8. \text{Find arc } (i'', i') \in C
9. \text{Find all arc } (c, i''') \in G
10. G = G \cup \{(ma(c, i'''), i''')_{(c, i''') \in G \cup C \cup \{(i', i)\}}\} - \{(i'', i')\}
11. \text{Remove all vertices and arcs in } G \text{ containing } c
12. \text{return } G
\]

Reminder: \(w_{ij} = \arg \max_k w_{ij}^k\)

Final MST

This is a tree and the MST for the contracted graph!!

Go back up recursive call and reconstruct final graph

Chu-Liu-Edmonds PseudoCode

\[
\text{contract}(G = (V, A), C, w) =
1. \text{Let } G_C \text{ be the subgraph of } G \text{ excluding nodes in } C
2. \text{Add a node } c \text{ to } G_C \text{ representing cycle } C
3. \text{For } i \in V - C : \exists i' \in C \text{ such that } (i', i) \in A
   \text{ Add arc } (c, i) \text{ to } G_C \text{ with }
   \text{ma}(c, i) = \arg \max_{j} w_{ij} \text{ score}(i', i)
   \text{ i' = ma}(c, i)
   \text{score}(c, i) = \text{score}(i', i)
4. \text{For } i \in V - C : \exists i' \in C \text{ such that } (i', i) \in A
   \text{ Add edge } (i, c) \text{ to } G_C \text{ with }
   \text{ma}(i, c) = \arg \max_{i''} \text{ score}(i', i'') \text{ score}(i', i)
   \text{ i' = ma}(i, c)
   \text{score}(i, c) = \text{score}(i', i') \text{ score}(i', i) + \text{score}(C)
   \text{ where } a(v) \text{ is the predecessor of } v \text{ in } C
   \text{ and } \text{score}(C) = \sum_{v \in C} \text{score}(a(v), v)
5. \text{return } < G_C, c, ma >
\]
Arc Weights

\[ w^k_{ij} = e^{w f(i,j,k)} \]

- Arc weights are a linear combination of features of the arc, \( f \), and a corresponding weight vector \( w \)
- Raised to an exponent (simplifies some math ...)
- What arc features?
- [McDonald et al. 2005] discuss a number of binary features

Arc Feature Ideas for \( f(i,j,k) \)

- Identities of the words \( w_i \) and \( w_j \) and the label \( l_k \)
- Part-of-speech tags of the words \( w_i \) and \( w_j \) and the label \( l_k \)
- Part-of-speech of words surrounding and between \( w_i \) and \( w_j \)
- Number of words between \( w_i \) and \( w_j \) and their orientation
- Combinations of the above

First-Order Feature Computation

(Structured) Perceptron

Training data: \( T = \{ (x_t, G_t) \}_{t=1}^T \)

1. \( w^{(0)} = 0 \); \( i = 0 \)
2. for \( n : 1..N \)
3. for \( t : 1..T \)
4. Let \( G' = \arg \max _{G'} w^{(i)} \cdot f(G') \)
5. if \( G' \neq G_t \)
6. \( w^{(i+1)} = w^{(i)} + f(G_t) - f(G') \)
7. \( i = i + 1 \)
8. return \( w' \)
Transition Based Dependency Parsing

- Process sentence left to right
- Different transition strategies available
- Delay decisions by pushing on stack

Arc-Standard Transition Strategy [Nivre ’03]

Initial configuration: ([], [0,...,n], [])
Terminal configuration: ([0], [], A)

shift: (σ, [i|β], A) \rightarrow ([σ|i], β, A)
left-arc (label): ([σ|i|j], B, A) \rightarrow ([σ|j], B, A∪{j,i})
right-arc (label): ([σ|i|j], B, A) \rightarrow ([σ|i], B, A∪{i,l,j})
I booked a flight to Lisbon
I booked a flight to Lisbon

I booked a flight to Lisbon

I booked a flight to Lisbon

I booked a flight to Lisbon
I booked a flight to Lisbon.

**Features ZPar Parser**

- # From single words
  - pair { stack.tag stack.word }
  - pair { input.tag input.word }
- # From word pairs
  - pair { input.tag input.word }
- # From word triples
  - triple { stack.tag stack.word input.word }
- # Distance
  - pair { stack.distance stack.word }
  - pair { stack.distance stack.tag }
  - pair { input.distance input.word }
  - pair { input.distance input.tag }

**SVM / Structured Perceptron Hyperparameters**

- • Regularization
- • Loss function
- • Hand-crafted features
Neural Network Transition Based Parser

[Chen & Manning '14] and [Weiss et al. '15]

Embedding Layer

Hidden Layer

Softmax

Atomic Inputs

words

pos

labels

[Weiss et al. '15]
Neural Network Transition Based Parser

[f_w \cdot w]

English Results (WSJ 23)

<table>
<thead>
<tr>
<th>Method</th>
<th>UAS</th>
<th>LAS</th>
<th>Beam</th>
</tr>
</thead>
<tbody>
<tr>
<td>3rd-order Graph-based (ZM2014)</td>
<td>93.22</td>
<td>91.02</td>
<td>-</td>
</tr>
<tr>
<td>Transition-based Linear (ZN2011)</td>
<td>93.00</td>
<td>90.95</td>
<td>32</td>
</tr>
<tr>
<td>NN Baseline (Chen &amp; Manning, 2014)</td>
<td>91.80</td>
<td>89.60</td>
<td>1</td>
</tr>
<tr>
<td>NN Better SGD (Weiss et al., 2015)</td>
<td>92.58</td>
<td>90.54</td>
<td>1</td>
</tr>
<tr>
<td>NN Deeper Network (Weiss et al., 2015)</td>
<td>93.19</td>
<td>91.18</td>
<td>1</td>
</tr>
</tbody>
</table>

NN Hyperparameters

- Regularization
- Loss function

NN Hyperparameters

- Regularization
- Loss function
- Dimensions
- Activation function
- Initialization
- Adagrad
- Dropout
NN Hyperparameters

- Regularization
- Loss function
- Dimensions
- Activation function
- Initialization
- Adagrad
- Dropout
- Mini-batch size
- Initial learning rate
- Learning rate schedule
- Momentum
- Stopping time
- Parameter averaging
- Adagrad
- Dropout
- Mini-batch size
- Initial learning rate
- Learning rate schedule
- Momentum
- Stopping time
- Parameter averaging

Random Restarts: How much Variance?

Effect of Embedding Dimensions

UAS (%) on WSJ Tune Set

Variance of Networks on Tuning/Dev Set

2nd hidden layer + pre training increases correlation

Effect of Embedding Dimensions

Word Tuning on WSJ (Tune Set, \( D_{\text{pos}}^{\text{D}_{\text{labels}}}=32 \))

Optimization matters!
Use random restarts, grid
Pick best using holdout data

Tune: WSJ S24
Dev: WSJ S22
Test: WSJ S23

Random Restarts: How much Variance?

Variance of Networks on Tuning/Dev Set

Word Tuning on WSJ (Tune Set, \( D_{\text{pos}}^{\text{D}_{\text{labels}}}=32 \))

Effect of Embedding Dimensions

Word Tuning on WSJ (Tune Set, \( D_{\text{pos}}^{\text{D}_{\text{labels}}}=32 \))

Pretrained 200x200
Pretrained 200
200x200
200

Pretrained 200x200
Pretrained 200
200x200
200

UAS (%) on WSJ Tune Set

91.2 91.4 91.6 91.8 92
92.1 92.2 92.3 92.4 92.5 92.6 92.7

90 90.5 91 91.5 92
89.5 90 90.5 91 91.5 92
Effect of Embedding Dimensions

How Important is Lookahead?

How Important is Lookahead?
How Important is Lookahead?

Alice saw Bob eat pizza with Charlie

How Important is Lookahead?

Bi-LSTM

How Important is Lookahead?

LSTM [Kiperwasser & Goldberg '16]
Beam Search with Local Model

Alice saw Bob eat pizza with Charlie

(Schematic)

Better

Beam Search with Local Model

Beam Start

UAS

Lookahead

0 1 2 3 4

80 85 90 95

Label Bias

- In Local Model every decision $0 \leq \phi \leq 1$ cannot penalize decision for pushing overall structure to far from gold
- Global Model learns to assign credit/blame

- Other views:
  - Need to learn to model states not along the gold path
  - Look into the future and overrule low-entropy (low-branching) states

Training with Early Updates

Globally normalized with respect to the beam:

$$\sum \phi_i^{(1)}$$
$$\sum \phi_i^{(2)}$$
$$\sum \phi_i^{(3)}$$
$$\sum \phi_i^{(4)}$$

Backpropagate through all steps, paths, and layers

[Collins and Roark ’04, Zhou et al.’15]
Globally Normalized Model

English Results (WSJ 23)

<table>
<thead>
<tr>
<th>Method</th>
<th>UAS</th>
<th>LAS</th>
<th>Beam</th>
</tr>
</thead>
<tbody>
<tr>
<td>3rd-order Graph-based (ZM2014)</td>
<td>93.22</td>
<td>91.02</td>
<td>-</td>
</tr>
<tr>
<td>Transition-based Linear (ZN2011)</td>
<td>93.00</td>
<td>90.95</td>
<td>32</td>
</tr>
<tr>
<td>NN Baseline (Chen &amp; Manning, 2014)</td>
<td>91.80</td>
<td>89.60</td>
<td>1</td>
</tr>
<tr>
<td>NN Better SGD (Weiss et al., 2015)</td>
<td>92.58</td>
<td>90.95</td>
<td>1</td>
</tr>
<tr>
<td>NN Deeper Network (Weiss et al., 2015)</td>
<td>93.19</td>
<td>90.60</td>
<td>1</td>
</tr>
<tr>
<td>NN Perceptron (Weiss et al., 2015)</td>
<td>93.99</td>
<td>92.05</td>
<td>8</td>
</tr>
<tr>
<td>NN CRF (Andor et al., 2016)</td>
<td>94.61</td>
<td>92.79</td>
<td>32</td>
</tr>
<tr>
<td>NN CRF Semi-Supervised (Andor et al.)</td>
<td>95.01</td>
<td>92.97</td>
<td>32</td>
</tr>
<tr>
<td>S-LSTM (Dyer et al., 2015)</td>
<td>93.20</td>
<td>90.90</td>
<td>1</td>
</tr>
<tr>
<td>Contrastive NN (Zhou et al., 2015)</td>
<td>92.83</td>
<td>—</td>
<td>100</td>
</tr>
</tbody>
</table>

Tri-Training

[Zhou et al. ’05, Li et al. ’14]

Berkeley Parser

ZPar Parser

UAS 89.96
LAS 87.26

UAS 96.35
LAS 95.02

UAS 89.84
LAS 95.02

~40% agreement

English Out-of-Domain Results

- Train on WSJ + Web Treebank + QuestionBank
- Evaluate on Web

UAS (%)

<table>
<thead>
<tr>
<th>Supervised</th>
<th>Semi-Supervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>87</td>
<td>90</td>
</tr>
<tr>
<td>87.25</td>
<td>89.25</td>
</tr>
<tr>
<td>87.75</td>
<td>89.75</td>
</tr>
<tr>
<td>88.5</td>
<td>89.5</td>
</tr>
<tr>
<td>88.75</td>
<td>89.75</td>
</tr>
</tbody>
</table>

3rd Order Graph (ZM2014)
Transition-based Linear (ZN 2011, B=32)
Transition-based NN (B=32)
Transition-based NN (B=8)
**Multilingual Results**

<table>
<thead>
<tr>
<th>Language</th>
<th>UAS (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalan</td>
<td>90.0</td>
</tr>
<tr>
<td>Chinese</td>
<td>95.0</td>
</tr>
<tr>
<td>Czech</td>
<td>85.0</td>
</tr>
<tr>
<td>English</td>
<td>80.0</td>
</tr>
<tr>
<td>German</td>
<td>90.0</td>
</tr>
<tr>
<td>Japanese</td>
<td>85.0</td>
</tr>
<tr>
<td>Spanish</td>
<td>90.0</td>
</tr>
</tbody>
</table>

- Tensor-Based Graph (Lei et al. ’14)
- 3rd-Order Graph (Zhang & McDonald ’14)
- Transition-based NN (Weiss et al. ’15)
- Transition-based CRF (Andor et al. ’16)

With morph features

**SyntaxNet and Parsey McParseface**

- Google Has Open Sourced Its AI for Understanding Language
- SyntaxNet, a natural-language understanding library for TensorFlow
- Googles Algorithms Decode Language like a Trained Linguist
- Alphabet subsidiary is making code freely available for anyone to distribute or modify

**LSTMs vs SyntaxNet**

<table>
<thead>
<tr>
<th>Feature</th>
<th>LSTMs</th>
<th>SyntaxNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Efficiency</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>End-to-End</td>
<td>++</td>
<td>- (yet)</td>
</tr>
<tr>
<td>Recurrence</td>
<td>+</td>
<td>- (yet)</td>
</tr>
</tbody>
</table>
Summary

- Constituency Parsing
  - CKY Algorithm
  - Lexicalized Grammars
  - Latent Variable Grammars
  - Conditional Random Field Parsing
  - Neural Network Representations

- Dependency Parsing
  - Eisner Algorithm
  - Maximum Spanning Tree Algorithm
  - Transition Based Parsing
  - Neural Network Representations