Advanced Graph-Based Parsing Techniques

Joakim Nivre

Uppsala University
Linguistics and Philology

Based on previous tutorials with Ryan McDonald
Overall Plan

1. Basic notions of dependency grammar and dependency parsing
2. Graph-based and transition-based dependency parsing
3. Advanced graph-based parsing techniques
4. Advanced transition-based parsing techniques
5. Neural network techniques in dependency parsing
6. Multilingual parsing from raw text to universal dependencies
Plan for this Lecture

- Projective parsing
  - Exact higher-order parsing
  - Approximations
- Non-projective parsing
  - NP-completeness
  - Exact higher-order parsing
  - Approximations
Graph-Based Parsing Trade-Off

[McDonald and Nivre 2007]

- Learning and inference are global
  - Decoding guaranteed to find highest scoring tree
  - Training algorithms use global structure learning
- But this is only possible with local feature factorizations
  - Must limit context statistical model can look at
  - Results in bad ‘easy’ decisions

The major question in graph-based parsing has been how to increase scope of features to larger subgraphs, without making inference intractable.
Higher-Order Parsing

- Two main dimensions of higher-order features
  - Vertical: e.g., “remain” is the grandparent of “emeritus”
  - Horizontal: e.g., “remain” is first child of “will”
Higher-Order Projective Parsing

- Easy – just modify the chart
- Usually asymptotic increase with each order modeled
- But we have a bag of tricks that help
2nd-Order Horizontal Projective Parsing

- Score factors by pairs of horizontally adjacent arcs
- Often called sibling dependencies
- \( s(i, j, j') = \) score of adjacent arcs \( x_i \rightarrow x_j \) and \( x_i \rightarrow x_{j'} \)

\[
\begin{align*}
s(T) &= \sum_{(i,j):(i,j') \in A} s(i, j, j') \\
&= \ldots + s(i_0, i_1, i_2) + s(i_0, i_2, i_3) + \ldots + s(i_0, i_{j-1}, i_j) + \\
&\quad s(i_0, i_{j+1}, i_{j+2}) + \ldots + s(i_0, i_{m-1}, i_m) + \ldots
\end{align*}
\]
2nd-Order Horizontal Projective Parsing

- Add a sibling chart item to get to $O(n^3)$
Higher-Order Projective Parsing

- People played this game since 2006
  - McDonald and Pereira [2006] (2nd-order sibling)
  - Carreras [2007] (2nd-order sibling and grandparent)
  - Koo and Collins [2010] (3rd-order grand-sibling and tri-sibling)
  - Ma and Zhao [2012] (4th-order grand-tri-sibling+)

\[
\begin{align*}
O(n^3) & \quad O(n^3) & \quad O(n^4) \\
\text{h} & \quad \text{m} & \quad \text{g} & \quad \text{h} & \quad \text{s} & \quad \text{m} & \quad \text{g} & \quad \text{h} & \quad \text{s'} & \quad \text{s} & \quad \text{m} \\
O(n^4) & \quad O(n^4) & \quad O(n^5) \\
\text{g} & \quad \text{h} & \quad \text{m} & \quad \text{h} & \quad \text{s'} & \quad \text{s} & \quad \text{m} & \quad \text{h} & \quad \text{s'} & \quad \text{s} & \quad \text{m} \\
\end{align*}
\]
Exact Higher-Order Projective Parsing

- Can be done via chart augmentation
- But there are drawbacks
  - $O(n^4), O(n^5), \ldots$ is just too slow
  - Every type of higher order feature requires specialized chart items and combination rules
Exact Higher-Order Projective Parsing

- Can be done via chart augmentation
- But there are drawbacks
  - $O(n^4), O(n^5), \ldots$ is just too slow
  - Every type of higher order feature requires specialized chart items and combination rules
- Led to research on approximations
  - Bohnet [2010]: feature hashing, parallelization
  - Koo and Collins [2010]: first-order marginal probabilities
  - Bergsma and Cherry [2010]: classifier arc filtering
  - Cascades
    - Rush and Petrov [2012]: structured prediction cascades
    - He et al. [2013]: dynamic feature selection
  - Zhang and McDonald [2012], Zhang et al. [2013]: cube-pruning
Structured Prediction Cascades

[Rush and Petrov 2012]

- Weiss et al. [2010]: train level $n$ w.r.t. to level $n + 1$
- Vine-parsing allows linear first stage [Dreyer et al. 2006]
- 100X+ faster than unpruned 3rd-order model with small accuracy loss (93.3→93.1) [Rush and Petrov 2012]
Cube Pruning
[Zhang and McDonald 2012, Zhang et al. 2013]

- Keep Eisner $O(n^3)$ as backbone
- Use chart item k-best lists to score higher order features

Example: Grandparent features

- Always $O(n^3)$ asymptotically
- No specialized chart parsing algorithms
Projective Parsing Summary

- Can augment chart (dynamic program) to increase scope of features but comes at complexity cost
- Solution: use pruning approximations

<table>
<thead>
<tr>
<th></th>
<th>En-UAS</th>
<th>Zh-UAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st order exact</td>
<td>91.8</td>
<td>84.4</td>
</tr>
<tr>
<td>2nd order exact</td>
<td>92.4</td>
<td>86.6</td>
</tr>
<tr>
<td>3rd order exact*</td>
<td>93.0</td>
<td>86.8</td>
</tr>
<tr>
<td>4th order exact†</td>
<td>93.4</td>
<td>87.4</td>
</tr>
<tr>
<td>struct. pred. casc.‡</td>
<td>93.1</td>
<td>–</td>
</tr>
<tr>
<td>cube-pruning*</td>
<td>93.5</td>
<td>87.9</td>
</tr>
</tbody>
</table>

* [Koo and Collins 2010], †[Ma and Zhao 2012], ‡[Rush and Petrov 2012], * [Zhang et al. 2013]

Cube-pruning is 2x slower than structured prediction cascades and 5x faster than third-order.
Higher-Order Non-Projective Parsing

- McDonald and Satta [2007]:
  - Parsing is \textit{NP-hard} for all higher-order features
  - Horizontal, vertical, valency, etc.
  - Even seemingly simple arc features like “Is this the only modifier” result in intractability
What to do?
What to do?

- Exact non-projective parsing
  - Integer Linear Programming
    - [Riedel and Clarke 2006, Martins et al. 2009]
  - Intractable in general, but efficient optimizers exact
  - Higher order parsing: asymptotic increase in constraint set size
Higher-Order Non-Projective Parsing

What to do?

- Exact non-projective parsing
  - Integer Linear Programming
    - [Riedel and Clarke 2006, Martins et al. 2009]
  - Intractable in general, but efficient optimizers exact
  - Higher order parsing: asymptotic increase in constraint set size

- Approximations (some return optimal in practice)
  - Approximate inference: \( T^* = \arg\max_{T \in G_x} s(T) \)
    - Post-processing [McDonald and Pereira 2006], [Hall and Novák 2005], [Hall 2007]
    - Dual Decomposition
    - Belief Propagation [Smith and Eisner 2008]
    - LP relaxations [Riedel et al. 2012]
    - Sampling [Nakagawa 2007]
  - Approximate search space: \( T^* = \arg\max_{T \in G_x} s(T) \)
    - Mildly non-projective structures
Dual Decomposition

- Assume a 2nd-order sibling model:

\[
s(T) = \sum_{(i,j):(i,j') \in A} s(i, j, j')
\]

\[
T^* = \arg\max_T s(T)
\]

- Computing \( T^* \) is hard, so let us try something simpler:

\[
s(D_i) = \sum_{(i,j):(i,j') \in A} s(i, j, j')
\]

\[
D_i^* = \arg\max_{D_i} s(D_i)
\]

- The highest scoring sequence of dependents for each word \( x_i \) can be computed in \( O(n) \) time using a semi-Markov model.
Dual Decomposition

- For each word $x_i$, find $D_i^*$:

  ROOT What did economic news have little effect on
  ROOT What did economic news have little effect on
  ROOT What did economic news have little effect on
  ROOT What did economic news have little effect on
  ROOT What did economic news have little effect on
Dual Decomposition

- For each word $x_i$, find $D_i^*$:

ROOT What did economic news have little effect on

ROOT What did economic **news** have little effect on

ROOT What did economic news **have** little effect on

ROOT What did economic news have little **effect** on

ROOT What did economic news have little effect **on**

**put it together**
Dual Decomposition

Why does this not work? **No tree constraint!**

- **ROOT** What did economic news have little effect on
- **ROOT** What did economic news have little effect on
- **ROOT** What did economic news have little effect on
- **ROOT** What did economic news have little effect on

**put it together**

- **ROOT** What did economic news have little effect on
Dual Decomposition

- First-order $O(n^2)$ model with tree constraint exists
  - MST algorithm [Chu and Liu 1965, Edmonds 1967]
- Second-order $O(n^3)$ model without tree constraint exists
  - The $O(n^3)$ sibling decoding algorithm
Dual Decomposition

- First-order $O(n^2)$ model with tree constraint exists
  - MST algorithm [Chu and Liu 1965, Edmonds 1967]
- Second-order $O(n^3)$ model without tree constraint exists
  - The $O(n^3)$ sibling decoding algorithm

- Dual Decomposition [Koo et al. 2010]
  - Add components for each feature
    - Independently calculate each efficiently
    - Tie together with agreement constraints (penalties)
Dual Decomposition

▶ For a sentence $x = x_1 \ldots x_n$, let:

- $s_1^o(T)$ be the first-order score of a tree $T$
- $T_1^o = \arg\max_{T \in G} s_1^o(T)$
- $s_2^o(G)$ be the second-order sibling score of a graph $G$
- $G_2^o = \arg\max_{G \in G} s_2^o(G)$
- Define structural variables $t_1^o(i, j) = 1$ if $(i, j) \in T_1^o$, 0 otherwise
- $g_2^o(i, j) = 1$ if $(i, j) \in G_2^o$, 0 otherwise

What we really want to find is $T = \arg\max_{T \in G} s_1^o(T) + s_2^o(T)$.
Dual Decomposition

For a sentence $x = x_1 \ldots x_n$, let:

- $s_{1o}(T)$ be the first-order score of a tree $T$
  - $T_{1o} = \arg\max_{T \in G_x} s_{1o}(T)$

- $s_{2o}(G)$ be the second-order sibling score of a graph $G$
  - $G_{2o} = \arg\max_{G \in G_x} s_{2o}(G)$
Dual Decomposition

- For a sentence $x = x_1 \ldots x_n$, let:
  - $s_{1o}(T)$ be the first-order score of a tree $T$
    - $T_{1o} = \arg\max_{T \in \mathcal{G}_x} s_{1o}(T)$
  - $s_{2o}(G)$ be the second-order sibling score of a graph $G$
    - $G_{2o} = \arg\max_{G \in \mathcal{G}_x} s_{2o}(G)$
**Dual Decomposition**

- For a sentence $x = x_1 \ldots x_n$, let:
  - $s_{1o}(T)$ be the first-order score of a tree $T$
  - $T_{1o} = \arg\max_{T \in G_x} s_{1o}(T)$
  - $s_{2o}(G)$ be the second-order sibling score of a graph $G$
  - $G_{2o} = \arg\max_{G \in G_x} s_{2o}(G)$

- Define structural variables
  - $t_{1o}(i,j) = 1$ if $(i,j) \in T_{1o}$, 0 otherwise
  - $g_{2o}(i,j) = 1$ if $(i,j) \in G_{2o}$, 0 otherwise
Dual Decomposition

- For a sentence $x = x_1 \ldots x_n$, let:
  - $s_{1o}(T)$ be the first-order score of a tree $T$
    - $T_{1o} = \arg\max_{T \in G_x} s_{1o}(T)$
  - $s_{2o}(G)$ be the second-order sibling score of a graph $G$
    - $G_{2o} = \arg\max_{G \in G_x} s_{2o}(G)$

- Define structural variables
  - $t_{1o}(i,j) = 1$ if $(i,j) \in T_{1o}$, 0 otherwise
  - $g_{2o}(i,j) = 1$ if $(i,j) \in G_{2o}$, 0 otherwise

- What we really want to find is

$$T = \arg\max_{T \in G_x} s_{1o}(T) + s_{so}(T)$$
Dual Decomposition

For a sentence $x = x_1 \ldots x_n$, let:

- $s_{1o}(T)$ be the first-order score of a tree $T$
  - $T_{1o} = \operatorname{argmax}_{T \in G_x} s_{1o}(T)$
- $s_{2o}(G)$ be the second-order sibling score of a graph $G$
  - $G_{2o} = \operatorname{argmax}_{G \in G_x} s_{2o}(G)$

Define structural variables

- $t_{1o}(i, j) = 1$ if $(i, j) \in T_{1o}$, 0 otherwise
- $g_{2o}(i, j) = 1$ if $(i, j) \in G_{2o}$, 0 otherwise

This is equivalent to:

$$ (T, G) = \operatorname{argmax}_{T \in G_x, G \in G_x} s_{1o}(T) + s_{so}(G) $$

s.t. $t_{1o}(i, j) = g_{2o}(i, j), \forall i, j \leq n$
Dual Decomposition

\[(T, G) = \operatorname{argmax}_{T \in G_x, G \in G_x} s_{1o}(T) + s_{so}(G), \text{ s.t. } t_{1o}(i, j) = g_{2o}(i, j)\]

Algorithm sketch
Dual Decomposition

\[(T, G) = \arg\max_{T \in G, G \in G_x} s_{1o}(T) + s_{so}(G), \text{ s.t. } t_{1o}(i, j) = g_{2o}(i, j)\]

**Algorithm sketch**

for \( k = 1 \) to \( K \)
Dual Decomposition

\[(T, G) = \arg\max_{T \in G_x, G \in G_x} s_{1o}(T) + s_{so}(G), \text{ s.t. } t_{1o}(i, j) = g_{2o}(i, j)\]

Algorithm sketch

for \(k = 1\) to \(K\)

1. \(T_{1o} = \arg\max_{T \in G_x} s_{1o}(T) - p\) // first-order decoding
Approximate Higher-Order Non-Projective Parsing

Dual Decomposition

\[(T, G) = \arg\max_{T \in G_x, G \in G_x} s_{1o}(T) + s_{so}(G), \; \text{s.t.} \; t_{1o}(i, j) = g_{2o}(i, j)\]

Algorithm sketch

for \( k = 1 \) to \( K \)

1. \( T_{1o} = \arg\max_{T \in G_x} s_{1o}(T) - p \) // first-order decoding
2. \( G_{2o} = \arg\max_{G \in G_x} s_{2o}(T) + p \) // second-order decoding
Dual Decomposition

\[ (T, G) = \arg\max_{T \in G_x, G \in G_x} s_{1o}(T) + s_{so}(G), \text{ s.t. } t_{1o}(i, j) = g_{2o}(i, j) \]

Algorithm sketch

for \( k = 1 \) to \( K \)

1. \( T_{1o} = \arg\max_{T \in G_x} s_{1o}(T) - p \) // first-order decoding
2. \( G_{2o} = \arg\max_{G \in G_x} s_{2o}(T) + p \) // second-order decoding
3. if \( t_{1o}(i, j) = g_{2o}(i, j), \forall i, j \), return \( T_{1o} \)
Dual Decomposition

\[(T, G) = \arg\max_{T \in G_x, G \in G_x} s_{1o}(T) + s_{so}(G), \text{ s.t. } t_{1o}(i, j) = g_{2o}(i, j)\]

Algorithm sketch

for \( k = 1 \) to \( K \)

1. \( T_{1o} = \arg\max_{T \in G_x} s_{1o}(T) - p \) // first-order decoding

2. \( G_{2o} = \arg\max_{G \in G_x} s_{2o}(T) + p \) // second-order decoding

3. if \( t_{1o}(i, j) = g_{2o}(i, j), \forall i, j \), return \( T_{1o} \)

4. else Update penalties \( p \) and go to 1
Dual Decomposition

\[ (T, G) = \arg\max_{T \in G_x, G \in G_x} s_{1o}(T) + s_{so}(G), \text{ s.t. } t_{1o}(i, j) = g_{2o}(i, j) \]

Algorithm sketch

for \( k = 1 \) to \( K \)

1. \( T_{1o} = \arg\max_{T \in G_x} s_{1o}(T) - p \) // first-order decoding
2. \( G_{2o} = \arg\max_{G \in G_x} s_{2o}(T) + p \) // second-order decoding
3. if \( t_{1o}(i, j) = g_{2o}(i, j), \forall i, j, \) return \( T_{1o} \)
4. else Update penalties \( p \) and go to 1

If \( K \) is reached, return \( T_{1o} \) from last iteration
Dual Decomposition

\[(T, G) = \arg\max_{T \in G_x, G \in G_x} s_{1o}(T) + s_{so}(G), \text{ s.t. } t_{1o}(i, j) = g_{2o}(i, j)\]

Algorithm sketch

for \( k = 1 \) to \( K \)

1. \( T_{1o} = \arg\max_{T \in G_x} s_{1o}(T) - p \) \( /\text{ first-order decoding} \)
2. \( G_{2o} = \arg\max_{G \in G_x} s_{2o}(T) + p \) \( /\text{ second-order decoding} \)
3. if \( t_{1o}(i, j) = g_{2o}(i, j), \forall i, j \), return \( T_{1o} \)
4. else Update penalties \( p \) and go to 1

What are the penalties \( p \)?
Dual Decomposition

Let $p(i, j) = t_{1o}(i, j) - g_{2o}(i, j)$

$p$ is the set of all penalties $p(i, j)$
Dual Decomposition

- Let $p(i, j) = t_{1o}(i, j) - g_{2o}(i, j)$
- $p$ is the set of all penalties $p(i, j)$
- We rewrite the decoding objectives as:

$$T_{1o} = \arg\max_{T \in G_x} \ s_{1o}(T) - \sum_{i,j} p(i, j) \times t_{1o}(i, j)$$

$$G_{2o} = \arg\max_{G \in G_x} \ s_{2o}(G) + \sum_{i,j} p(i, j) \times g_{2o}(i, j)$$

- Reward trees/graphs that agree with other model
Dual Decomposition

Let \( p(i,j) = t_{1o}(i,j) - g_{2o}(i,j) \)

\( p \) is the set of all penalties \( p(i,j) \)

We rewrite the decoding objectives as:

\[
T_{1o} = \arg\max_{T \in G_x} s_{1o}(T) - \sum_{i,j} p(i,j) \times t_{1o}(i,j)
\]

\[
G_{2o} = \arg\max_{G \in G_x} s_{2o}(G) + \sum_{i,j} p(i,j) \times g_{2o}(i,j)
\]

Reward trees/graphs that agree with other model

Since \( t_{1o} \) and \( g_{2o} \) are arc-factored indicator variables, we can easily include in decoding

\[
s(i,j) = s(i,j) - p(i,j)
\]

for first-order model
Dual Decomposition – 1-iter Example

First-order

Second-order sibling

penalties: \( p(5, 3) = 1, p(4, 3) = -1, p(7, 8) = -1 \)

first-order: \( s_{1o}(5, 3) = 1, s_{1o}(4, 3) = 1, s_{1o}(7, 8) = 1 \)

second-order: \( s_{2o}(5, *, 3) = 1, s_{2o}(4, *, 3) = 1, s_{2o}(7, *, 8) = 1 \)

*Indicates any sibling, even null if it is first left/right modifier.
Dual Decomposition

Goal: \((T, G) = \arg\max_{T \in G_x, G \in G_x} s_{1o}(T) + s_{so}(G), \text{ s.t. } t_{1o}(i, j) = g_{2o}(i, j)\)

for \(k = 1\) to \(K\)

1. \(T_{1o} = \arg\max_{T \in G_x} s_{1o}(T) - p\) /// first-order decoding
2. \(G_{2o} = \arg\max_{G \in G_x} s_{2o}(T) + p\) /// second-order decoding
3. if \(t_{1o}(i, j) = g_{2o}(i, j), \forall i, j\), return \(T_{1o}\)
4. else Update penalties \(p\) and go to 1

- Penalties push scores towards agreement
- **Theorem**: If for any \(k\), line 3 holds, then decoding is optimal
Dual Decomposition

- Koo et al. [2010]: grandparents, grand-sibling, tri-siblings
- Martins et al. [2011, 2013]: arbitrary siblings, head bigrams

<table>
<thead>
<tr>
<th>Order</th>
<th>UAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st order</td>
<td>90.52</td>
</tr>
<tr>
<td>2nd order</td>
<td>91.85</td>
</tr>
<tr>
<td>3rd order</td>
<td>92.41</td>
</tr>
</tbody>
</table>

[Martins et al. 2013]
Mildly Non-Projective Structures

- Dual decomposition approximates search over entire space
  \[ T = \arg\max_{T \in G} s(T) \]
Mildly Non-Projective Structures

- Dual decomposition approximates search over entire space
  \[ T = \arg\max_{T \in G_x} s(T) \]
- Another approach is to restrict search space
  \[ T = \arg\max_{T \in G_x} s(T) \]
  1. Allow efficient decoding
  2. Still cover all linguistically plausible structures
Mildly Non-Projective Structures

- Dual decomposition approximates search over entire space
  \[ T = \operatorname{argmax}_{T \in G_x} s(T) \]
- Another approach is to restrict search space
  \[ T = \operatorname{argmax}_{T \in G_x} s(T) \]
  1. Allow efficient decoding
  2. Still cover all linguistically plausible structures
- Do we really care about scoring such structures?
Mildly Non-Projective Structures

- Well-nested block-degree 2 [Bodirsky et al. 2005]
  - LTAG-like algorithms: $O(n^7)$ [Gómez-Rodríguez et al. 2011]
  - + 1-inherit: $O(n^6)$ [Pitler et al. 2012]
    - Empirical coverage identical to well-nested block-degree 2
  - + Head-split: $O(n^6)$ [Satta and Kuhlmann 2013]
    - Empirical coverage similar to well-nested block-degree 2
  - + Head-split + 1-inherit: $O(n^5)$ [Satta and Kuhlmann 2013]
- Gap Minding Trees: $O(n^5)$ [Pitler et al. 2012]
- 1-Endpoint-Crossing: $O(n^4)$ [Pitler et al. 2013]

*All run-times are for first-order parsing*
**1-Endpoint-Crossing** [Pitler et al. 2013]

- An arc $A$, is **1-endpoint-crossing** iff all arcs $A'$ that cross $A$ have a common endpoint $p$
- An endpoint $p$ is either a head or a modifier in an arc
- E.g., (arrived, What) is crossed by (ROOT,think) and (think,?), both have endpoint ‘think’
1-Endpoint-Crossing

- Can we design an algorithm that parses all and only 1-endpoint-crossing trees?
- Pitler et al. [2013] provides the solution
- Pitler’s algorithm works by defining 5 types of intervals

Location of exterior point, direction of arcs, etc, controlled via variables, similar to Eisner [1996] projective formulation.
1-Endpoint-Crossing

- On CoNLL-X data sets [Buchholz and Marsi 2006]

<table>
<thead>
<tr>
<th>Class</th>
<th>Tree coverage</th>
<th>Run-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projective</td>
<td>80.8</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>Well-nested block-degree 2</td>
<td>98.4</td>
<td>$O(n^7)$</td>
</tr>
<tr>
<td>Gap-Minding</td>
<td>95.1</td>
<td>$O(n^5)$</td>
</tr>
<tr>
<td>1-Endpoint-Crossing</td>
<td>98.5</td>
<td>$O(n^4)$</td>
</tr>
</tbody>
</table>

[Pitler et al. 2013]

Macro average over Arabic, Czech, Danish, Dutch, Portuguese
1-Endpoint-Crossing

- Good empirical coverage and low run-time
- Can be linguistically motivated [Pitler et al. 2013]

Phrae-impensetrability condition (PIC) [Chomsky 1998]
- Only head and edge words of phrase accessible to sentence
- Long-distance elements leave chain of traces at clause edges
- Pitler et al. [2013] conjecture: PIC implies 1-endpoint-crossing
1-Endpoint-Crossing

- Pitler [2014]: 1-endpoint-crossing + third-order
  - Merge of Pitler et al. [2013] and Koo and Collins [2010]
  - Searches 1-endpoint-crossing trees
  - Scores higher-order features when no crossing arc present
  - $O(n^4)$ – identical to third-order projective!
  - Significant improvements in accuracy
Coming Up Next

1. Basic notions of dependency grammar and dependency parsing
2. Graph-based and transition-based dependency parsing
3. Advanced graph-based parsing techniques
4. Advanced transition-based parsing techniques
5. Neural network techniques in dependency parsing
6. Multilingual parsing from raw text to universal dependencies
References and Further Reading

► Shane Bergsma and Colin Cherry. 2010. 

Well-nested drawings as models of syntactic structure. In Tenth Conference on Formal Grammar and Ninth Meeting on Mathematics of Language.

► Bernd Bohnet. 2010. 

► Sabine Buchholz and Erwin Marsi. 2006. 
CoNLL-X shared task on multilingual dependency parsing. In Proceedings of the Tenth Conference on Computational Natural Language Learning, pages 149–164.

► Xavier Carreras. 2007. 
References and Further Reading


References and Further Reading


Keith Hall. 2007.
K-best spanning tree parsing. In *Proceedings of the Association for Computational Linguistics (ACL)*.

He He, Hal Daumé III, and Jason Eisner. 2013.

Terry Koo and Michael Collins. 2010.


▶ Ryan McDonald and Joakim Nivre. 2007.
Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the Join Conference on Empirical Methods in Natural Language Processing and the Conference on Computational Natural Language Learning (EMNLP-CoNLL)*.


Emily Pitler. 2014. A crossing-sensitive third-order factorization for dependency parsing. *Transactions of the Association for Computational Linguistics (TACL).*


References and Further Reading


▷ Liang Zhang, Huang, Kai Zhao, and Ryan McDonald. 2013.