Graph-Based and Transition-Based Dependency Parsing

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

- Graph-based dependency parsing
  - First-order model
  - Learning and inference
- Transition-based dependency parsing
  - Arc-eager transition system
  - Learning and inference
- Contrastive error analysis [McDonald and Nivre 2007]
Graph-Based Parsing

Basic idea:
- Define a space of candidate dependency graphs for a sentence.
- **Learning**: Induce a model for scoring an entire dependency graph for a sentence.
- **Parsing**: Find the highest-scoring dependency graph, given the induced model.

Characteristics:
- Global training of a model for optimal dependency graphs
- Exhaustive search/inference
Graph-Based Parsing

- For input sentence $x$ define a graph $G_x = (V_x, A_x)$, where
  - $V_x = \{0, 1, \ldots, n\}$
  - $A_x = \{(i, j, k) \mid i, j \in V \land j \neq 0 \land i \neq j \land l_k \in L\}$
Graph-Based Parsing

▶ For input sentence $x$ define a graph $G_x = (V_x, A_x)$, where
  ▶ $V_x = \{0, 1, \ldots, n\}$
  ▶ $A_x = \{(i, j, k) | i, j \in V \text{ and } j \neq 0 \text{ and } i \neq j \text{ and } l_k \in L\}$
▶ Key observation:
  ▶ Valid dependency trees for $x = \text{directed spanning trees of } G_x$
Graph-Based Parsing

- For input sentence $x$ define a graph $G_x = (V_x, A_x)$, where
  - $V_x = \{0, 1, \ldots, n\}$
  - $A_x = \{(i, j, k) \mid i, j \in V \text{ and } j \neq 0 \text{ and } i \neq j \text{ and } l_k \in L\}$
- Key observation:
  - Valid dependency trees for $x =$ directed spanning trees of $G_x$
- Score of dependency tree $T$ factors by subgraphs $G_1, \ldots, G_m$:
  - $s(T) = \sum_{c=1}^{m} s(G_c)$
Graph-Based Parsing

- For input sentence $x$ define a graph $G_x = (V_x, A_x)$, where
  - $V_x = \{0, 1, \ldots, n\}$
  - $A_x = \{(i, j, k) \mid i, j \in V \text{ and } j \neq 0 \text{ and } i \neq j \text{ and } l_k \in L\}$
- Key observation:
  - Valid dependency trees for $x =$ directed spanning trees of $G_x$
- Score of dependency tree $T$ factors by subgraphs $G_1, \ldots, G_m$:
  - $s(T) = \sum_{c=1}^{m} s(G_c)$
- Learning: Scoring function $s(G_c)$ for subgraphs $G_c \in G$
Graph-Based Parsing

For input sentence $x$ define a graph $G_x = (V_x, A_x)$, where

- $V_x = \{0, 1, \ldots, n\}$
- $A_x = \{(i, j, k) \mid i, j \in V \text{ and } j \neq 0 \text{ and } i \neq j \text{ and } l_k \in L\}$

Key observation:
- Valid dependency trees for $x = \text{directed spanning trees of } G_x$
- Score of dependency tree $T$ factors by subgraphs $G_1, \ldots, G_m$:
  - $s(T) = \sum_{c=1}^m s(G_c)$
- Learning: Scoring function $s(G_c)$ for subgraphs $G_c \in G$
- Inference: Search for maximum spanning tree $T^*$ of $G_x$

$$T^* = \arg\max_{T \in G_x} s(T) = \arg\max_{T \in G_x} \sum_{c=1}^m s(G_c)$$
Learning

- Typical scoring function:
  - $s(G_i) = \mathbf{w} \cdot \mathbf{f}(G_i)$

  where
  - $\mathbf{f}(G_i) =$ high-dimensional feature vector over subgraphs
  - $\mathbf{w} =$ weight vector [$w_j =$ weight of feature $f_j(G_i)$]

- Structured learning [McDonald et al. 2005a]:
  - Learn weights that maximize the score of the correct dependency tree for every sentence in the training set
First-Order Model

- Scored subgraph $G_c$ is a single arc $(i, j, k)$
- $s(T) = \sum_{c=1}^{m} s(G_c) = \sum_{(i, j, k) \in T} s(i, j, k)$
- Often we drop $k$, since it is rarely structurally relevant
  - $s(T) = \sum_{(i, j) \in T} s(i, j)$
  - $s(i, j) = \max_k s(i, j, k)$
First-Order Model

- Scored subgraph \( G_c \) is a single arc \((i, j, k)\)
- \( s(T) = \sum_{c=1}^{m} s(G_c) = \sum_{(i,j,k) \in T} s(i,j,k) \)
- Often we drop \( k \), since it is rarely structurally relevant
  - \( s(T) = \sum_{(i,j) \in T} s(i,j) \)
  - \( s(i,j) = \max_k s(i,j,k) \)

- This search is **global**: consider all possible trees
First-Order Projective Parsing

Eisner algorithm

[1]

Chart items either:
1) Create a new dependency
2) Absorb left/right subtree

Each chart item store two indexes:
1) left boundary
2) right boundary

All operations require 3 indexes: $O(n^3)$
First-Order Non-Projective Parsing

- Equivalent to MST problem [McDonald et al. 2005b]
- For directed graphs, also called arborescence problem
- $O(n^2)$ parsing [Chu and Liu 1965, Edmonds 1967]
- Greedy algorithm, not dynamic programming
Feature Scope

- $f \in \mathbb{R}^n$ is a feature representation of the subgraph $G_c$
Feature Scope

- $f \in \mathbb{R}^n$ is a feature representation of the subgraph $G_c$
- For first-order models, $G_c$ is an arc
  - $G_c = (i, j)$ for a head $i$ and modifier $j$
Feature Scope

- $f \in \mathbb{R}^n$ is a feature representation of the subgraph $G_c$
- For first-order models, $G_c$ is an arc
  - $G_c = (i, j)$ for a head $i$ and modifier $j$
- This inherently limits features to a local scope
Feature Scope

- $f \in \mathbb{R}^n$ is a feature representation of the subgraph $G_c$
- For first-order models, $G_c$ is an arc
  - $G_c = (i, j)$ for a head $i$ and modifier $j$
- This inherently limits features to a local scope

- For arc (had, effect) below, can have features over properties of arc and context within sentence

```
Economic news had little effect on financial markets
```

```
adj   noun   verb   adj   noun   prep   adj   noun
amod  nsubj  dobj   amod  prep   pmod   amod
```

Graph-Based and Transition-Based Dependency Parsing 10(32)
Feature Scope

- $f \in \mathbb{R}^n$ is a feature representation of the subgraph $G_c$
- For first-order models, $G_c$ is an arc
  - $G_c = (i,j)$ for a head $i$ and modifier $j$
- This inherently limits features to a local scope

- For arc (had, effect) below, cannot have features over multiple arcs (siblings, grandparents), valency, etc.
Graph-Based Parsing Trade-Off

- Learning and inference are global
  - Decoding guaranteed to find highest scoring tree
  - Training algorithms use global structure learning
Graph-Based Parsing Trade-Off

- 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
    - For example, first-order models often predict two subjects
    - No parameter exists to discourage this

Diagram:
- John (noun) - Smith (noun) - was (verb) - tall (adj)
- nsubj - nsubj - acomp
Transition-Based Parsing

- Basic idea:
  - Define a transition system (state machine) for mapping a sentence to its dependency graph.
  - **Learning**: Induce a model for predicting the next state transition, given the transition history.
  - **Parsing**: Construct the optimal transition sequence, given the induced model.

- Characteristics:
  - Local training of a model for optimal transitions
  - Greedy search/inference
Transition-Based Parsing

- A transition system for dependency parsing defines
  - a set $C$ of parser configurations
  - a set $T$ of transitions, each a function $t : C \rightarrow C$
  - initial configuration and terminal configurations for sentence $x$
- Key idea:
  - Valid dependency trees for $S$ defined by terminating transition sequences $C_{0,m} = t_1(c_0), \ldots, t_m(c_{m-1})$
- Score of $C_{0,m}$ factors by config-transition pairs $(c_{i-1}, t_i)$:
  - $s(C_{0,m}) = \sum_{i=1}^{m} s(c_{i-1}, t_i)$
- Learning:
  - Scoring function $s(c_{i-1}, t_i)$ for $t_i(c_{i-1}) \in C_{0,m}$
- Inference:
  - Search for highest scoring sequence $C_{0,m}^*$ given $s(c_{i-1}, t_i)$
Arc-Eager Transition System [Nivre 2003]

**Configuration:** \((S, B, A)\) \([S = \text{Stack}, B = \text{Buffer}, A = \text{Arcs}]\)

**Initial:** \(([\; ], [0, 1, \ldots, n], \{ \; \})\)

**Terminal:** \((S, [\; ], A)\)

**Shift:** \((S, i|B, A)\) \(\Rightarrow (S|i, B, A)\)

**Reduce:** \((S|i, B, A)\) \(\Rightarrow (S, B, A)\)

**Right-Arc\((k)\):** \((S|i, j|B, A)\) \(\Rightarrow (S|i+j, B, A \cup \{(i, j, k)\})\)

**Left-Arc\((k)\):** \((S|i, j|B, A)\) \(\Rightarrow (S, j|B, A \cup \{(j, i, k)\})\) \(\neg h(i, A) \land i \neq 0\)

**Notation:**
- \(S|i = \text{stack with top } i \text{ and remainder } S\)
- \(j|B = \text{buffer with head } j \text{ and remainder } B\)
- \(h(i, A) = i \text{ has a head in } A\)
Example Transition Sequence

\[ \text{ROOT}_S \ [\text{Economic, news, had, little, effect, on, financial, markets, .}]_B \]

\begin{align*}
\text{ROOT} & \quad \text{Economic} & \quad \text{news} & \quad \text{had} & \quad \text{little} & \quad \text{effect} & \quad \text{on} & \quad \text{financial} & \quad \text{markets} & \quad . \\
\text{adj} & \quad \text{noun} & \quad \text{verb} & \quad \text{adj} & \quad \text{noun} & \quad \text{prep} & \quad \text{adj} & \quad \text{noun} & \quad .
\end{align*}
Example Transition Sequence

\[[\text{ROOT, Economic}]_S \ [\text{news, had, little, effect, on, financial, markets, .}]_B\]

ROOT Economic news had little effect on financial markets .

adj noun verb adj noun prep adj noun .
Example Transition Sequence

\[ \text{ROOT}_S \text{ [news, had, little, effect, on, financial, markets, .]} \text{B} \]

ROOT Economic news had little effect on financial markets .

amod

ROOT adj noun verb adj noun prep adj noun .
Example Transition Sequence

\[ [\text{ROOT, news}]_S \quad [\text{had, little, effect, on, financial, markets, .}]_B \]

- ROOT
- Economic news had little effect on financial markets .
  - adj
  - noun
  - verb
  - adj
  - noun
  - prep
  - adj
  - noun
Example Transition Sequence

\[ \text{ROOT} \_S \quad [\text{had, little, effect, on, financial, markets, .}]_B \]

![Diagram showing the transition sequence and dependencies between words in the sentence: Economic news had little effect on financial markets.](Diagram)
Example Transition Sequence

\[\text{ROOT, had}_S \ [\text{little, effect, on, financial, markets, .}]_B\]

\[
\begin{align*}
\text{root} & \\
\text{amod} & \quad \text{nsubj} \\
\text{ROOT} & \quad \text{Economic} \quad \text{news} \quad \text{had} \quad \text{little} \quad \text{effect} \quad \text{on} \quad \text{financial} \quad \text{markets} \quad .
\end{align*}
\]
Example Transition Sequence

\[ \text{ROOT, had, little}_S \quad \text{[effect, on, financial, markets, .]}_B \]

Graph-Based and Transition-Based Dependency Parsing 15(32)
Example Transition Sequence

\[
\begin{align*}
\text{ROOT, had}_S & \quad \text{effect, on, financial, markets, .}_B \\
\text{root} & \\
\text{ROOT} & \quad \text{Economic} \\
\text{adj} & \\
\text{nsubj} & \\
\text{amod} & \quad \text{had} \\
\text{nsubj} & \quad \text{little} \\
\text{amod} & \quad \text{effect} \\
\text{nsubj} & \quad \text{on} \\
\text{amod} & \quad \text{financial} \\
\text{amod} & \quad \text{markets} \\
\text{pmod} & \\
\text{p} & \\
\text{root} & \\
\text{dobj} & \\
\text{amod} & \quad 
\end{align*}
\]
Example Transition Sequence

[ROOT, had, effect]_S  [on, financial, markets, .]_B
Economic news had little effect on financial markets.

Example Transition Sequence

\[ \text{ROOT, had, effect, on}_S \quad \text{financial, markets, .}_B \]

Diagram:

- **ROOT**: root
- **amod**: amod
- **nsubj**: nsubj
- **dobj**: dobj
- **prep**: prep

Labels:

- **ROOT**: Economic
  - adj: adj
  - noun: noun
- **news**: had
  - verb: verb
- **little**: effect
  - adj: prep
  - noun: financial
  - prep: markets
  - adj: noun
Example Transition Sequence

\[
\text{[ROOT, had, effect, on, financial]}_S \quad \text{[markets, .]}_B
\]
Example Transition Sequence

\[
\text{[ROOT, had, effect, on]}_S \quad \text{[markets, .]}_B
\]
Example Transition Sequence

\[ [\text{ROOT, had, effect, on, markets}]_S \quad [.]_B \]
Example Transition Sequence

\[ [\text{ROOT, had, effect, on}]_S \quad [.]_B \]
Example Transition Sequence

\[
[\text{ROOT, had, effect}]_S \quad [\cdot]_B
\]
Example Transition Sequence

\[[\text{ROOT}, \text{had}]_S \ [\cdot]_B\]
Example Transition Sequence

\[
\begin{array}{c}
\text{ROOT, had, .}_{S} \\
\text{ROOT, } \text{Economic news had little effect on financial markets .}_{B}
\end{array}
\]
Greedy Inference

- Given an oracle $o$ that correctly predicts the next transition $o(c)$, parsing is deterministic:

  $$\text{Parse}(w_1, \ldots, w_n)$$

  1. $c \leftarrow ([ ]_S, [0, 1, \ldots, n]_B, \{ \})$
  2. while $B_c \neq []$
  3. $t \leftarrow o(c)$
  4. $c \leftarrow t(c)$
  5. return $G = (\{0, 1, \ldots, n\}, A_c)$

- Complexity given by upper bound on number of transitions
- Parsing in $O(n)$ time for the arc-eager transition system
An oracle can be approximated by a (linear) classifier:

\[ o(c) = \arg\max_t w \cdot f(c, t) \]

- History-based feature representation \( f(c, t) \)
- Weight vector \( w \) learned from treebank data
Feature Representation

- Features over input tokens relative to \( S \) and \( B \)

### Configuration

```
[S: [ROOT, had, effect] \( S \)]
```

```
[B: [on, financial, markets, .] \( B \)]
```

### Features

- \( \text{pos}(S_2) = \text{ROOT} \)
- \( \text{pos}(S_1) = \text{verb} \)
- \( \text{pos}(S_0) = \text{noun} \)
- \( \text{pos}(B_0) = \text{prep} \)
- \( \text{pos}(B_1) = \text{adj} \)
- \( \text{pos}(B_2) = \text{noun} \)
Feature Representation

- Features over input tokens relative to $S$ and $B$

**Configuration**

```latex
\begin{array}{c}
\text{ROOT, had, effect}_S \quad \text{on, financial, markets, .}_B \\
\text{ROOT} \quad \text{Economic} \quad \text{news} \quad \text{had} \quad \text{little} \quad \text{effect} \quad \text{on} \quad \text{financial} \quad \text{markets} \quad . \\
\text{ROOT} \quad \text{ROOT} \quad \text{adj} \quad \text{noun} \quad \text{verb} \quad \text{adj} \quad \text{noun} \quad \text{prep} \quad \text{adj} \quad \text{noun} \quad .
\end{array}
```

**Features**

- $\text{word}(S_2) = \text{ROOT}$
- $\text{word}(S_1) = \text{had}$
- $\text{word}(S_0) = \text{effect}$
- $\text{word}(B_0) = \text{on}$
- $\text{word}(B_1) = \text{financial}$
- $\text{word}(B_2) = \text{markets}$
Feature Representation

- Features over input tokens relative to $S$ and $B$
- Features over the (partial) dependency graph defined by $A$

Configuration

```
ROOT, had, effect]$_{S}$  [on, financial, markets, .]$_{B}$
```

Features

- $\text{dep}(S_1) = \text{root}$
- $\text{dep}(\text{lcs}(S_1)) = \text{nsubj}$
- $\text{dep}(\text{rcs}(S_1)) = \text{dobj}$
- $\text{dep}(S_0) = \text{dobj}$
- $\text{dep}(\text{lcs}(S_0)) = \text{amod}$
- $\text{dep}(\text{rcs}(S_0)) = \text{NIL}$
Feature Representation

- Features over input tokens relative to $S$ and $B$
- Features over the (partial) dependency graph defined by $A$
- Features over the (partial) transition sequence

Configuration

```
[ROOT, had, effect]_S [on, financial, markets, ]_B
```

```
ROOT Economic news had little effect on financial markets .
```

Features

$t_{i-1} = \text{Right-Arc(dobj)}$
$t_{i-2} = \text{Left-Arc(amod)}$
$t_{i-3} = \text{Shift}$
$t_{i-4} = \text{Right-Arc(root)}$
$t_{i-5} = \text{Left-Arc(nsubj)}$
$t_{i-6} = \text{Shift}$
Feature Representation

- Features over input tokens relative to $S$ and $B$
- Features over the (partial) dependency graph defined by $A$
- Features over the (partial) transition sequence

### Configuration

![Configuration Diagram]

$[\text{ROOT, had, effect}]_S$ $[\text{on, financial, markets, .}]_B$

### Features

$t_{i-1} = \text{Right-Arc(dobj)}$
$t_{i-2} = \text{Left-Arc(amod)}$
$t_{i-3} = \text{Shift}$
$t_{i-4} = \text{Right-Arc(root)}$
$t_{i-5} = \text{Left-Arc(nsubj)}$
$t_{i-6} = \text{Shift}$

- Feature representation unconstrained by parsing algorithm
Local Learning

- Given a treebank:
  - Reconstruct oracle transition sequence for each sentence
  - Construct training data set \( D = \{(c, t) \mid o(c) = t\} \)
  - Maximize accuracy of local predictions \( o(c) = t \)
- Any (unstructured) classifier will do (SVMs are popular)
- Training is local and restricted to oracle configurations
Transition-Based Parsing Trade-Off

- Advantages:
  - Highly efficient parsing – linear time complexity with constant time oracles and transitions
  - Rich history-based feature representations – no rigid constraints from inference algorithm

- Drawback:
  - Sensitive to search errors and error propagation due to greedy inference and local learning
CoNLL 2006

- CoNLL 2006: Shared Task on Dependency Parsing
  - Evaluation of 13 different languages
- Top 2 systems statistically identical: One graph-based (MSTParser) and the other transition-based (MaltParser)
- Question: do the systems learn the same things?
### MSTParser and MaltParser

<table>
<thead>
<tr>
<th>Language</th>
<th>MSTParser</th>
<th>MaltParser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>66.91</td>
<td>66.71</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>87.57</td>
<td>87.41</td>
</tr>
<tr>
<td>Chinese</td>
<td>85.90</td>
<td>86.92</td>
</tr>
<tr>
<td>Czech</td>
<td>80.18</td>
<td>78.42</td>
</tr>
<tr>
<td>Danish</td>
<td>84.79</td>
<td>84.77</td>
</tr>
<tr>
<td>Dutch</td>
<td>79.19</td>
<td>78.59</td>
</tr>
<tr>
<td>German</td>
<td>87.34</td>
<td>85.82</td>
</tr>
<tr>
<td>Japanese</td>
<td>90.71</td>
<td>91.65</td>
</tr>
<tr>
<td>Portuguese</td>
<td>86.82</td>
<td>87.60</td>
</tr>
<tr>
<td>Slovene</td>
<td>73.44</td>
<td>70.30</td>
</tr>
<tr>
<td>Spanish</td>
<td>82.25</td>
<td>81.29</td>
</tr>
<tr>
<td>Swedish</td>
<td>82.55</td>
<td>84.58</td>
</tr>
<tr>
<td>Turkish</td>
<td>63.19</td>
<td>65.68</td>
</tr>
<tr>
<td>Overall</td>
<td>80.83</td>
<td>80.75</td>
</tr>
</tbody>
</table>
Comparing the Models

- **Inference:**
  - Exhaustive (MSTParser)
  - Greedy (MaltParser)

- **Training:**
  - Global structure learning (MSTParser)
  - Local decision learning (MaltParser)

- **Features:**
  - Local features (MSTParser)
  - Rich decision history (MaltParser)

- **Fundamental trade-off:**
  - Global learning and inference vs. rich feature space
Error Analysis [McDonald and Nivre 2007]

- **Aim:**
  - Relate parsing errors to linguistic and structural properties of the input and predicted/gold standard dependency graphs
- **Three types of factors:**
  - **Length factors:** sentence length, dependency length
  - **Graph factors:** tree depth, branching factor, non-projectivity
  - **Linguistic factors:** part of speech, dependency type
- **Statistics:**
  - Labeled accuracy, precision and recall
  - Computed over the test sets for all 13 languages
MaltParser is more accurate than MSTParser for short sentences (1–10 words) but its performance degrades more with increasing sentence length.
MaltParser is more precise than MSTParser for short dependencies (1–3 words) but its performance degrades drastically with increasing dependency length (> 10 words).

MSTParser has more or less constant precision for dependencies longer than 3 words.

Recall is very similar across systems.
MSTParser is much more precise than MaltParser for dependents of the root and has roughly constant precision for depth $> 1$, while MaltParser’s precision improves with increasing depth (up to 7 arcs).

Recall is very similar across systems.
Part of Speech

- MSTParser is more accurate for verbs, adjectives, adverbs, adpositions, and conjunctions.
- MaltParser is more accurate for nouns and pronouns.
Contrastive Error Analysis

Dependency Type: Root, Subject, Object

- MSTParser has higher precision (and recall) for roots.
- MSTParser has higher recall (and precision) for subjects.
Discussion

- Many of the results are indicative of the fundamental trade-off: global learning/inference versus rich features.
- Global inference improves decisions for long sentences and those near the top of graphs.
- Rich features improve decisions for short sentences and those near the leaves of the graphs.
- Dependency parsing post-2007:
  - How do we use this to improve parser performance?
Voting and Stacking

- Early improvements were based on system combination
- Voting:
  - Let parsers vote for heads [Zeman and Žabokrtský 2005]
  - Use MST algorithm for tree constraint [Sagae and Lavie 2006]
- Stacking:
  - Use the output of one parser as features for the other
    [Nivre and McDonald 2008, Torres Martins et al. 2008]
- Focus in these lectures:
  - Work on evolving the approaches themselves
  - Richer feature representations in graph-based parsing
  - Improved learning and inference in transition-based parsing
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


▶ Ryan McDonald and Joakim Nivre. 2007.


References and Further Reading


