Transition-Based Dependency Parsing

Recent Advances in Parsing Technologies
Seminar Presentation

Anjana Vakil
Universität des Saarlandes
December 20, 2012
Presentation Outline

- Dependency Graphs (review)
- Transition-Based Parsers
  - Components, configurations & transitions
  - Deterministic parsing algorithm
  - Example parse
- The Oracle: Predicting the next transition
  - Parser histories as feature vectors
  - Discriminative machine learning
- Implementation: MaltParser
- Transition- vs. graph-based approaches
  - Comparison
  - Integration
Dependency Graphs

- Dependency Graphs
- Transition-based parsing
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches

```
Sentence (+ ROOT)       x = (w_0, w_1, \ldots, w_n)
Dep. types (arc labels) R = \{r_1, \ldots, r_m\}
Vertices/nodes (words)  V \subseteq \{w_0, w_1, \ldots, w_n\} = V_S
Labeled Arcs           A \subseteq V \times R \times V
Dependency Graph       G = (V, A)
```

Reproduced from (Kübler et al., 2009; p. 2)
Dependency Graphs

\[ G = (V, A) \]
\[ V = V_S = \{ \text{ROOT, Economic, news, had, little, effect, on, financial, markets, .} \} \]
\[ A = \{(\text{root, PRED, had}), (\text{had, SBJ, news}), (\text{had, OBJ, effect}), (\text{had, PU, .}), (\text{news, ATT, Economic}), (\text{effect, ATT, little}), (\text{effect, ATT, on}), (\text{on, PC, markets}), (\text{markets, ATT, financial})\} \]

Reproduced from (Kübler et al., 2009; p. 2)

- Dependency Graphs
- Transition-based parsing
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches
Dependency Graphs

Constraints:

- Root
- Connectedness
- Single-Headedness
- Acyclicity
- Projectivity (?)
Transition-Based Parsing

Components:
- Buffer ($\beta$): token nodes to be processed
- Stack ($\sigma$): token nodes being processed
- Partial graph ($A$): set of labeled arcs $(w_i, r, w_j)$

Configurations:
- Initial: $c_0 = ([w_0]_\sigma, [w_1, \ldots, w_n]_\beta, \emptyset)$
- Terminal: $c_{term} = (\sigma, [\ ]_\beta, A)$

- Dependency Graphs
- Transition-based parsing
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches
Transition-Based Parsing

**Transitions** map a non-terminal configuration to a new configuration.

“Arc-eager” Transition system:

<table>
<thead>
<tr>
<th>Transition</th>
<th>Precondition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left-Arc</td>
<td>((\sigma</td>
</tr>
<tr>
<td>Right-Arc</td>
<td>((\sigma</td>
</tr>
<tr>
<td>Reduce</td>
<td>((\sigma</td>
</tr>
<tr>
<td>Shift</td>
<td>((\sigma, w_i</td>
</tr>
</tbody>
</table>

Reproduced from (Kübler et al., 2009; p. 34)

Alternatives: “arc-standard”, non-projective

- Dependency Graphs
- **Transition-based parsing**
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches
A simple deterministic algorithm:

Given:
- Sentence \( S \)
- Transitions \( t \in T \)
- Oracle \( o \)

Time complexity: \( O(n) \)

Alternatives: multi-pass, incremental (non-deterministic)

Transition-Based Parsing

Reproduced from (Kübler et al., 2009; p. 26)

\begin{align*}
1 & \quad c \leftarrow c_0(S) \\
2 & \quad \textbf{while} \ c \text{ is not terminal} \\
3 & \quad t \leftarrow o(c) \\
4 & \quad c \leftarrow t(c) \\
5 & \quad \textbf{return} \ G_c
\end{align*}
Economic news had little \ldots \beta

- Dependency Graphs
- Transition-based parsing
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches

Adapted from (Kübler et al., 2009; p. 2)
Economic news had little ... β

- Dependency Graphs
- Transition-based parsing
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition-vs. Graph-based approaches

Adapted from (Kübler et al., 2009; p. 2)
Adapted from Kübler et al., 2009; p. 2
Transition-Based Parsing

news had little effect ...

• Dependency Graphs
• Transition-based parsing
  • Components, configurations, transitions
  • Deterministic algorithm
  • Example
• The Oracle
• MaltParser
• Transition- vs. Graph-based approaches

Adapted from (Kübler et al., 2009; p. 2)
Transition-Based Parsing

- Dependency Graphs
- Transition-based parsing
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches

Adapted from (Kübler et al., 2009; p. 2)

The diagram shows a sentence with the words "had little effect on ..." and transitions labeled as "LEFT-ARC_{SBJ}". The sentence structure is represented with dependency arcs and labels such as "ROOT", "SBJ", "ATT", and "Economic".
Transition-Based Parsing

Adapted from (Kübler et al., 2009; p. 2)
Transition-Based Parsing

- Dependency Graphs
- Transition-based parsing
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches

Adapted from (Kübler et al., 2009; p. 2)
Transition-Based Parsing

- Dependency Graphs
- Transition-based parsing
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches

Adapted from (Kübler et al., 2009; p. 2)
Transition-Based Parsing

had effect on financial markets.

Adapted from (Kübler et al., 2009; p. 2)

- Dependency Graphs
- Transition-based parsing
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches
Transition-Based Parsing

- Dependency Graphs
- Transition-based parsing
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition-vs. Graph-based approaches

Adapted from (Kübler et al., 2009; p. 2)
Transition-Based Parsing

- **Dependency Graphs**
- **Transition-based parsing**
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- **The Oracle**
- **MaltParser**
- **Transition- vs. Graph-based approaches**

Adapted from (Kübler et al., 2009; p. 2)
Transition-Based Parsing

- Dependency Graphs
- Transition-based parsing
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches

Adapted from (Kübler et al., 2009; p. 2)
Transition-Based Parsing

- Dependency Graphs
- Transition-based parsing
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches

Adapted from (Kübler et al., 2009; p. 2)
Transition-Based Parsing

适应自 (Kübler et al., 2009; p. 2)

- Dependency Graphs
- Transition-based parsing
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches
Transition-Based Parsing

- Dependency Graphs
- Transition-based parsing
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition-vs. Graph-based approaches

Adapted from (Kübler et al., 2009; p. 2)
Transition-Based Parsing

- Dependency Graphs
- Transition-based parsing
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches

Adapted from (Kübler et al., 2009; p. 2)
Transition-Based Parsing

Adapted from (Kübler et al., 2009; p. 2)

- Dependency Graphs
- Transition-based parsing
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches
Transition-Based Parsing

- Dependency Graphs
- Transition-based parsing
  - Components, configurations, transitions
  - Deterministic algorithm
  - Example
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches

TERMINAL CONFIGURATION ($c_{term}$)

Adapted from (Kübler et al., 2009; p. 2)
The Oracle: Predicting the next transition

Parse a sentence by moving from initial to terminal configuration through a series of transitions

Easy: Using some $t$ to find the next $c$

Hard: Choosing the right $t$

→ Visit the “oracle”
  - Mechanism for predicting the next transition
  - Approximated with:
    - History-based feature models
    - Discriminative machine learning
  - Parsing essentially reduced to classification

• Dependency Graphs
• Transition-based parsing
• The Oracle
  • Histories as feature vectors
  • Discriminative learning
• MaltParser
• Transition- vs. Graph-based approaches
The Oracle: Predicting the next transition

- Goal: predict a decision (transition) given a particular history (parser configuration)
- Problem: Infinite # of possible configurations
- Solution: represent configurations as multidimensional feature vectors
- Which features are helpful?
  - Attributes of input tokens
    - Static: word form, POS, etc.
    - Dynamic: dependency type (w.r.t. head)
  - Dependency Graphs
  - Transition-based parsing
  - The Oracle
    - Histories as feature vectors
    - Discriminative learning
  - MaltParser
  - Transition- vs. Graph-based approaches
The Oracle: Predicting the next transition

Formally:

\[ P(x, y) = P(d_1, \ldots, d_n) \]

\[ = \prod_{i=1}^{n} P(d_i \mid d_1, \ldots, d_{i-1}) \]

\[ = \prod_{i=1}^{n} P(d_i \mid \Phi(d_1, \ldots, d_{i-1})) \]

\[ \Phi_{1,p} = (\phi_1, \ldots, \phi_p) \]

Estimate:

\[ \text{arg max}_{d_i} P(d_i \mid \Phi(d_1, \ldots, d_{i-1})) \]

Adapted from (Nivre et al., 2006; p. 104)
The Oracle: Predicting the next transition

Goal: Learn a classifier that predicts the next transition based on the history feature vector of the current configuration

- Classification task:
  - Input: history = feature vector (Φ(c))
  - Output: class = transition (t)

- Training data:
  - Pairs (Φ(c), t)
  - Extracted from treebank by reconstructing the transition sequence for each sentence

- Dependency Graphs
- Transition-based parsing
- The Oracle
  - Histories as feature vectors
  - Discriminative learning
- MaltParser
- Transition- vs. Graph-based approaches
The Oracle: Predicting the next transition

Example feature model (MaltParser Standard):

- **POS features**: POS of...
  - Top two tokens in stack: \( p(\sigma_0), p(\sigma_1) \)
  - Next four in buffer: \( p(\beta_0), p(\beta_1), p(\beta_2), p(\beta_3) \)

- **Lexical features**: word form of...
  - Top token in stack: \( w(\sigma_0) \)
  - Its head: \( w(h(\sigma_0)) \)
  - Next two in buffer: \( w(\beta_0), w(\beta_1) \)

- **Dependency type features**: relation-to-head of...
  - Top token in stack: \( d(\sigma_0) \)
  - Its left- and rightmost dependents: \( d(l(\sigma_0)), d(r(\sigma_0)) \)
  - Leftmost child of next token in buffer: \( d(l(\beta_0)) \)

- **Dependency Graphs**
- **Transition-based parsing**
- **The Oracle**
  - Histories as feature vectors
  - Discriminative learning
- **MaltParser**
- **Transition- vs. Graph-based approaches**
Discriminative learning algorithms:

- **Memory-Based Learning (MBL)**
  - Essentially k-nearest neighbor classification
  - Lazy learning: “experiences” stored in memory
  - Classification: new experience compared to old
    - Distance metric/feature weighting used to compute similarity to stored experiences
    - New experience classified by similarity
  - Fast learning $\rightarrow$ efficient training
  - Slow classification $\rightarrow$ inefficient parsing

- **Dependency Graphs**
- **Transition-based parsing**
- **The Oracle**
  - Histories as feature vectors
  - **Discriminative learning**
  - MaltParser
  - Transition- vs. Graph-based approaches
The Oracle: Predicting the next transition

Discriminative learning algorithms:

- Memory-Based Learning (MBL)
- Support Vector Machines (SVM)
  - Max-margin linear classifier
    - Try to maximize “distance” between classes
    - Kernel functions map feature vector to more dimensions, to make classes more separable
  - Binary classifier, can be adapted for multi-class problem with various techniques
  - Only numerical features allowed
  - Training is slow, various techniques to speed it up

- Dependency Graphs
- Transition-based parsing
- The Oracle
  - Histories as feature vectors
  - Discriminative learning
- MaltParser
- Transition- vs. Graph-based approaches
“A language-independent system for data-driven dependency parsing” (Nivre et al., 2006)

- Induces a parser for a given language from (a limited amount of) training data

- **Learning:**
  - Derives training data \(((\Phi(c), t) \text{ pairs})\) from dependency-annotated corpus
  - Uses this to train a classifier

- **Parsing:**
  - Uses classifier as oracle for parsing test data
User can specify:

- Parsing algorithm/transition system
  - Arc-eager, projective (default)
  - Arc-standard, projective
- Feature model
  - Add/omit/replace features w.r.t. standard model
- Learning algorithm
  - MBL using TiMBL (default)
    - Adjustable settings (e.g. $k$, distance metric, weighting)
  - SVM using LIBSVM
- Non-projective/pseudo-projective parsing

- Dependency Graphs
- Transition-based parsing
- The Oracle
  - MaltParser
    - Design
    - Performance
- Transition- vs. Graph-based approaches
Overview of data sets for experimental evaluation:

Table 1. Data sets. AS = Annotation scheme (C = Constituency, D = Dependency, G = Grammatical functions); Pro = Projective; #D = Number of dependency types; #P = Number of PoS tags; TA = Tagging accuracy; #W = Number of words; #S = Number of sentences; SL = Mean sentence length; EM = Evaluation method (T = Held-out test set, CV_k = k-fold cross-validation)

<table>
<thead>
<tr>
<th>Language</th>
<th>AS</th>
<th>Pro</th>
<th>#D</th>
<th>#P</th>
<th>TA</th>
<th>#W</th>
<th>#S</th>
<th>SL</th>
<th>EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulgarian</td>
<td>C</td>
<td>no</td>
<td>14</td>
<td>51</td>
<td>93.5</td>
<td>72k</td>
<td>5.1k</td>
<td>14.1</td>
<td>CV_8</td>
</tr>
<tr>
<td>Chinese</td>
<td>CG</td>
<td>yes</td>
<td>12</td>
<td>35</td>
<td>100.0</td>
<td>509k</td>
<td>18.8k</td>
<td>27.1</td>
<td>T</td>
</tr>
<tr>
<td>Czech</td>
<td>D</td>
<td>no</td>
<td>26</td>
<td>28</td>
<td>94.1</td>
<td>1507k</td>
<td>87.9k</td>
<td>17.2</td>
<td>T</td>
</tr>
<tr>
<td>Danish</td>
<td>D</td>
<td>no</td>
<td>54</td>
<td>33</td>
<td>96.3</td>
<td>100k</td>
<td>5.5k</td>
<td>18.2</td>
<td>T</td>
</tr>
<tr>
<td>Dutch</td>
<td>CD</td>
<td>no</td>
<td>23</td>
<td>165</td>
<td>95.7</td>
<td>186k</td>
<td>13.7k</td>
<td>13.6</td>
<td>T</td>
</tr>
<tr>
<td>English</td>
<td>CG</td>
<td>yes</td>
<td>12</td>
<td>48</td>
<td>96.1</td>
<td>1174k</td>
<td>49.2k</td>
<td>23.8</td>
<td>T</td>
</tr>
<tr>
<td>German</td>
<td>CG</td>
<td>no</td>
<td>31</td>
<td>55</td>
<td>100.0</td>
<td>382k</td>
<td>22.1k</td>
<td>17.3</td>
<td>CV_10</td>
</tr>
<tr>
<td>Italian</td>
<td>D</td>
<td>no</td>
<td>17</td>
<td>89</td>
<td>93.1</td>
<td>42k</td>
<td>1.5k</td>
<td>27.7</td>
<td>CV_10</td>
</tr>
<tr>
<td>Swedish</td>
<td>CG</td>
<td>yes</td>
<td>17</td>
<td>46</td>
<td>95.6</td>
<td>98k</td>
<td>6.3k</td>
<td>15.5</td>
<td>T</td>
</tr>
<tr>
<td>Turkish</td>
<td>D</td>
<td>no</td>
<td>24</td>
<td>484</td>
<td>100.0</td>
<td>48k</td>
<td>5.6k</td>
<td>8.6</td>
<td>CV_10</td>
</tr>
</tbody>
</table>

Reproduced from (Nivre et al., 2006; p. 110)
Performance across languages:

<table>
<thead>
<tr>
<th>Language</th>
<th>Model</th>
<th>Settings</th>
<th>$AS_U$</th>
<th>$AS_L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulgarian</td>
<td>$\forall a[w(a) \rightarrow s_6(w(a))]$</td>
<td>Standard</td>
<td>81.3</td>
<td>73.6</td>
</tr>
<tr>
<td>Chinese</td>
<td>Standard</td>
<td>k = 6, l = 8</td>
<td>81.1</td>
<td>79.2</td>
</tr>
<tr>
<td>Czech</td>
<td>Standard</td>
<td>Standard</td>
<td>80.1</td>
<td>72.8</td>
</tr>
<tr>
<td>Danish</td>
<td>$[w(h(\sigma_0)) \rightarrow s_6(w(h(\sigma_0))); -w(\tau_1)]$</td>
<td>Standard</td>
<td>85.6</td>
<td>79.5</td>
</tr>
<tr>
<td>Dutch</td>
<td>Standard</td>
<td>k = 10</td>
<td>84.7</td>
<td>79.2</td>
</tr>
<tr>
<td>English</td>
<td>Standard</td>
<td>k = 7, l = 5</td>
<td>88.1</td>
<td>86.3</td>
</tr>
<tr>
<td>German</td>
<td>$[-w(h(\sigma_0)); -w(\tau_1); +p(\sigma_2)]$</td>
<td>k = 13, IL</td>
<td>88.1</td>
<td>83.4</td>
</tr>
<tr>
<td>Italian</td>
<td>Standard</td>
<td>Standard</td>
<td>82.9</td>
<td>75.7</td>
</tr>
<tr>
<td>Swedish</td>
<td>Standard</td>
<td>Standard</td>
<td>86.3</td>
<td>82.0</td>
</tr>
<tr>
<td>Turkish</td>
<td>$[-p(\sigma_1); -p(\tau_2); -p(\tau_3); -w(h(\sigma_0)); -w(\tau_1)]$</td>
<td>Standard</td>
<td>81.6</td>
<td>69.0</td>
</tr>
</tbody>
</table>

Reproduced from (Nivre et al., 2006; p. 113) (emphasis added)
# Transition- vs. Graph-based Dependency Parsing

<table>
<thead>
<tr>
<th>Approach</th>
<th>Transition-based (Malt) (Nivre et al. 2006)</th>
<th>Graph-based (MSTParser) (McDonald et al. 2006)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference algorithm</td>
<td>Local, greedy, based on current parser history</td>
<td>Global, exhaustive, searches all graphs to find max. score</td>
</tr>
<tr>
<td>Learning</td>
<td>Trains model to make a single classification</td>
<td>Trains model to maximize score of correct graphs</td>
</tr>
<tr>
<td>Feature representation</td>
<td>Rich, expressive</td>
<td>Restricted for efficiency</td>
</tr>
<tr>
<td>Most accurate dependencies</td>
<td>Short Far from root</td>
<td>Long Close to root</td>
</tr>
<tr>
<td>Most accurate dependents</td>
<td>nouns, pronouns</td>
<td>verbs, adjectives, adverbs, adpositions, conjunctions</td>
</tr>
</tbody>
</table>

Adapted from (Nivre and McDonald, 2008, p. 952)
Transition- vs. Graph-based Dependency Parsing

One model guides the other ("parser stacking")

Feature vectors for training:

- **Malt (base):** $f(c, t)$
- **Malt$_{MST}$ (guided):** $f(c, t, G^x_{MST})$
- **MST (base):** $f(i, j, l)$
- **MST$_{Malt}$ (guided):** $f(i, j, l, G^x_{Malt})$

- Dependency Graphs
- Transition-based parsing
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches
  - Comparison
  - Integration
Transition- vs. Graph-based Dependency Parsing

Results

<table>
<thead>
<tr>
<th>Language</th>
<th>MST</th>
<th>MST\textsubscript{Malt}</th>
<th>Malt</th>
<th>Malt\textsubscript{MST}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>66.91</td>
<td>68.64 (+1.73)</td>
<td>66.71</td>
<td>67.80 (+1.09)</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>87.57</td>
<td>89.05 (+1.48)</td>
<td>87.41</td>
<td>88.59 (+1.18)</td>
</tr>
<tr>
<td>Chinese</td>
<td>85.90</td>
<td>88.43 (+2.53)</td>
<td>86.92</td>
<td>87.44 (+0.52)</td>
</tr>
<tr>
<td>Czech</td>
<td>80.18</td>
<td>82.26 (+2.08)</td>
<td>78.42</td>
<td>81.18 (+2.76)</td>
</tr>
<tr>
<td>Danish</td>
<td>84.79</td>
<td>86.67 (+1.88)</td>
<td>84.77</td>
<td>85.43 (+0.66)</td>
</tr>
<tr>
<td>Dutch</td>
<td>79.19</td>
<td>81.63 (+2.44)</td>
<td>78.59</td>
<td>79.91 (+1.32)</td>
</tr>
<tr>
<td>German</td>
<td>87.34</td>
<td>88.46 (+1.12)</td>
<td>85.82</td>
<td>87.66 (+1.84)</td>
</tr>
<tr>
<td>Japanese</td>
<td>90.71</td>
<td>91.43 (+0.72)</td>
<td>91.65</td>
<td>92.20 (+0.55)</td>
</tr>
<tr>
<td>Portuguese</td>
<td>86.82</td>
<td>87.50 (+0.68)</td>
<td>87.60</td>
<td>88.64 (+1.04)</td>
</tr>
<tr>
<td>Slovene</td>
<td>73.44</td>
<td>75.94 (+2.50)</td>
<td>70.30</td>
<td>74.24 (+3.94)</td>
</tr>
<tr>
<td>Spanish</td>
<td>82.25</td>
<td>83.99 (+1.74)</td>
<td>81.29</td>
<td>82.41 (+1.12)</td>
</tr>
<tr>
<td>Swedish</td>
<td>82.55</td>
<td>84.66 (+2.11)</td>
<td>84.58</td>
<td>84.31 (-0.27)</td>
</tr>
<tr>
<td>Turkish</td>
<td>63.19</td>
<td>64.29 (+1.10)</td>
<td>65.58</td>
<td>66.28 (+0.70)</td>
</tr>
<tr>
<td>Average</td>
<td>80.83</td>
<td>82.53 (+1.70)</td>
<td>80.74</td>
<td>82.01 (+1.27)</td>
</tr>
</tbody>
</table>

Table 2: Labeled attachment scores for base parsers and guided parsers (improvement in percentage points).

Reproduced from (Nivre and McDonald, 2008, p. 954)
Conclusion

- Transition-based parsing (e.g. MaltParser)
  - Uses a greedy deterministic algorithm to build dependency graphs through a series of transitions
  - Uses history-based feature models and discriminative machine learning to train an “oracle” to guide the parser
  - Language-independent, needs little training data
  - Can be adapted for non-projectivity

- Integration of transition- and graph-based approaches
  - Combine rich features and global inference/learning
  - Performance improves over either approach alone

- Dependency Graphs
- Transition-based parsing
- The Oracle
- MaltParser
- Transition- vs. Graph-based approaches
