Transition-Based Dependency Parsing

Saarbrücken, December 23rd 2011

David Przybilla – davida@coli.uni-saarland.de
Outline

1. MaltParser

2. Transition Based Parsing
   a. Example
   b. Oracle

3. Integrating Graph and Transition Based

4. Non–Projective Dependency Parsing
MaltParser

- Different Languages (No tuning for a Specific Lang)
- Language independent: accurate parsing for a wide variety of languages
- Accuracy between 80% and 90%
- Deterministic
Transition Based Parsing

Transitions
- Shift
- Left-arc
- Right-arc
- Reduction

Stack
- $W_k$
- $W_x$
- $W_i$
- $W_{i+1}$
- $W_{i+2}$

Buffer

Actions
Sentence

$C_1 \xrightarrow{t_1} C_2 \xrightarrow{t_2} \ldots \xrightarrow{t_n} C_T$

$C_i =$ configuration $i$

$C_T =$ Final State

t = transition, a single change to a configuration

Oracle (Classifier)

"predict" the optimal (best) transition given the current configuration

Trained

Treebank
Example

John hit the ball

Stack

John hit the ball

Buffer
Example

Transition=Shift

John hit the ball
Example

Transition = left Arc

John hit the ball

Only if $h(john) = 0$
Example

Transition=Shift

John hit the ball

Stack  Buffer

hit  the ball

Subj
Example

Transition=Shift

John hit the ball

Stack    Buffer

the
hit
ball

Subj
Example

Transition = left Arc

John hit the ball

Only if \( h(\text{the}) = 0 \)
Example

Transition=Right Arc

John hit the ball

Subj

Det

Obj

Only if $h(ball) = 0$
Example

John hit the ball

Buffer is Empty = Terminal Configuration
Transition Based Parsing

Sentence: $W_x \ W_k \ldots W_i \ W_{i+1} \ldots$

Only if $h(W_k) \neq 0$
Oracle

- Greedy Algorithm, choose a local optimal hoping it will lead to the global optimal

- It makes Transition Based Algorithm Deterministic.
  - Originally there might be more than one possible transition from one configuration to another

- Construct the Optimal Transition sequence for the Input Sentence

- How to Build the Oracle? Build a Classifier
Classifier

The Classifier

Classes:
- Shift
- Left-arc
- Right-arc
- Reduction

Feature Vector (Features)
- POS of words in the Buffer and Stack
- Words themselves
- The First Word in the Stack
- The L World in the Buffer
- The current arcs in the Graph
Results of the MaltParser

Evaluation Metrics:

- **ASU (Unlabeled Attachment Score)**: Proportion of Tokens assigned the correct head

- **ASL (Labeled Attachment Score)**: Proportion of tokens assigned with the correct head and the correct dependency type
Results of the MaltParser

Goal ->
Evaluate if Maltparser can do reasonably accurate parsing for a wide variety of languages
### Results of the MaltParser

<table>
<thead>
<tr>
<th>Language</th>
<th>Asu</th>
<th>Asl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech</td>
<td>80.1</td>
<td>72.8</td>
</tr>
<tr>
<td>English</td>
<td>88.1</td>
<td>86.3</td>
</tr>
<tr>
<td>Italian</td>
<td>82.9</td>
<td>75.7</td>
</tr>
<tr>
<td>Chinese</td>
<td>81.1</td>
<td>79.2</td>
</tr>
<tr>
<td>Dutch</td>
<td>84.7</td>
<td>79.2</td>
</tr>
<tr>
<td>Swedish</td>
<td>86.3</td>
<td>82.0</td>
</tr>
<tr>
<td>German</td>
<td>88.1</td>
<td>83.4</td>
</tr>
<tr>
<td>Danish</td>
<td>85.6</td>
<td>79.5</td>
</tr>
<tr>
<td>Turkish</td>
<td>81.6</td>
<td>69.0</td>
</tr>
</tbody>
</table>
Results of the MaltParser

- Results:
  - Above 80% unlabeled dependency Accuracy (ASU) for all languages
  - morphological richness and word order are the cause of variation across languages

In General lower accuracy for languages like Czech and Turkish.
  - There are more non-projective structures in those languages

- It is difficult to do Cross-Language Comparison:
  - Big difference in the amount of annotated data
  - existence of accurate POS Taggers..

State of the art for Italian, Swedish, Danish, Turkish
Graph Based VS Transition Based

**Graph Based**
- Search for Optimal Graph (Highest Scoring Graph)
- Globally Trained (Global Optimal)
- Limited History of Parsing Decisions
- Less rich feature representation

**Transition Based**
- Search for Optimal Graph by finding the best transition between two states. (Local Optimal Decisions)
- Locally Trained (configurations)
- Rich History of Parsing Decisions
- More rich feature but Error Propagation (Greedy Alg.)
Graph Based vs Transition Based

Graph Based (MST)
- Better for Long Dependencies
- More accurate for dependents that are:
  - Verbs
  - Adjectives
  - Adverbs

Transition Based (Malt)
- Better for Short dependencies
- More accurate for dependents that are:
  - Nouns
  - Pronouns

Integrate Both Approaches
Integrating Graph and Transition Based

- Integrate both approaches at learning time.
- Base MSTParser guided by Malt

```
Treebank T  Malt Parser  Transition Based Parser  Parsed T  MST Parser
```

- Base MALTParser guided by MLT

```
Treebank T  MST Parser  Transition Based Parser  Parsed T  Malt Parser
```
Features used in the Integration

- **MSTParser guided by Malt**
  - Is arc \((i, j, *)\) in \(G_{malt}\)
  - Is arc \((i, j, l)\) in \(G_{malt}\)
  - Is arc \((i, j, *)\) not in \(G_{malt}\)
  - Identity of \(l'\) such that \((i, j, l')\) is in \(G_{malt}\)
  - ..

- **MaltParser guided by MST**
  - Is arc \((S^0, B^0, *)\) in \(G_{mst}\)
  - Is arc \((B^0, S^0, *)\) in \(G_{mst}\)
  - Head direction of \(B^0\) in \(G_{mst}\) (left, right, root..)
  - Identity of \(l'\) such that \((*, B^0, l')\) is in \(G_{mst}\)

\(S^0\) = fist element of the Stack, \(B^0\) = First element of the Buffer
Results of Integration

Asl(Correct head And Correct Label)
Results of Integration

Asl(Correct head And Correct Label)
## Results of Integration

<table>
<thead>
<tr>
<th>Language</th>
<th>MST</th>
<th>\text{MST}_{\text{Malt}}</th>
<th>Malt</th>
<th>\text{Malt}_{\text{MST}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>66.91</td>
<td>\textbf{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>\textbf{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>\textbf{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>\textbf{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>\textbf{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>\textbf{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>\textbf{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>\textbf{88.64} (+1.04)</td>
</tr>
<tr>
<td>Slovene</td>
<td>73.44</td>
<td>\textbf{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>\textbf{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>\textbf{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>\textbf{66.28} (−0.70)</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>80.83</td>
<td>\textbf{82.53} (+1.70)</td>
<td>80.74</td>
<td>82.01 (+1.27)</td>
</tr>
</tbody>
</table>

\text{Asl(Correct head And Correct Label)}
Results of Integration

- Graph-based models predict better long arcs
- Each model learns strengths from the others
- The integration actually improves accuracy
- Trying to do more chaining of systems do not gain better accuracy
Non-Projectivity

- Some Sentences have long distance dependencies which cannot be parsed with this algorithm
  - Cause it only consider relations between neighbors words
- 25% or more of the sentences in some languages are non-projective
- Useful for some languages with less constraints on word order
- Harder Problem, There could be relations over unbounded distances.
Non-Projectivity

A dependency Tree $T$ is Projective:

if for every Arc $(W_i, W_j, rel)$ there is a path from $W_i$ to $W_k$, if $W_k$ is between $W_i$ and $W_j$

From ‘Scheduled’ $W_2$ there is an arc to $W_5$ however there is no way to get to $W_4, W_3$ from $W_2$
Non-Projectivity

- Why the previous transition algorithm would not be able to generate this tree?

Stack

- is
- hearing

Buffer

- On
- ...
- ...

'is' can never be reduced. 'hearing' and 'on' will never get an arc.
Handling Non-Projectivity

- Add a new Transition – ""Swap"

Stack

$W_{i+1}$

$W_i$

Buffer

$W_k$

Stack

$W_{i+1}$

Buffer

$W_i$

$W_k$

- Re-Order the initial Input Sentence
Non-Projectivity

Stack

Buffer

is

hearing

On

…

…

swap

Stack

Buffer

is

Hearing

On

…

…

[Diagram of dependency tree]
Non-Projective Dependency Parsing

- Useful for some languages with less constraints on word order

Theoretically

- Best case $O(N)$, that is: no swaps
- Worst Case $O(N^2)$,
Results
Non-Projective Dependency Parsing

Running Time

- Test on 5 languages (Danish, Arabic, Czech, Slovene, Turkish)
- In practice the running time is $O(N)$.

Parsing Accuracy

- Criteria
  - **Attachment Score**: Percentage of tokens with correct head and dependency label
  - **Exact match**: completely correct labeled dependency tree
Results
Non-Projective Dependency Parsing

- Systems Compared
  - $S_u$ = allowing Non Projective
  - $S_p$ = Just Projective
  - $S_{pp}$ = Handling non-Projectivity as a pos-processing
  - **AS**: Percentage of tokens with correct head and dependency label
  - **EM**: completely correct labeled dependency tree

<table>
<thead>
<tr>
<th>System</th>
<th>Arabic</th>
<th></th>
<th>Czech</th>
<th></th>
<th>Danish</th>
<th></th>
<th>Slovene</th>
<th></th>
<th>Turkish</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AS</td>
<td>EM</td>
<td>AS</td>
<td>EM</td>
<td>AS</td>
<td>EM</td>
<td>AS</td>
<td>EM</td>
<td>AS</td>
</tr>
<tr>
<td>$S_u$</td>
<td>67.1</td>
<td>(9.1)</td>
<td>11.6</td>
<td>82.4</td>
<td>(73.8)</td>
<td>35.3</td>
<td>84.2</td>
<td>(22.5)</td>
<td>26.7</td>
</tr>
<tr>
<td>$S_p$</td>
<td>67.3</td>
<td>(18.2)</td>
<td>11.6</td>
<td>80.9</td>
<td>(3.7)</td>
<td>31.2</td>
<td>84.6</td>
<td>(0.0)</td>
<td>27.0</td>
</tr>
<tr>
<td>$S_{pp}$</td>
<td>67.2</td>
<td>(18.2)</td>
<td>11.6</td>
<td>82.1</td>
<td>(60.7)</td>
<td>34.0</td>
<td>84.7</td>
<td>(22.5)</td>
<td>28.9</td>
</tr>
<tr>
<td>Malt-06</td>
<td>66.7</td>
<td>(18.2)</td>
<td>11.0</td>
<td>78.4</td>
<td>(57.9)</td>
<td>27.4</td>
<td>84.8</td>
<td>(27.5)</td>
<td>26.7</td>
</tr>
<tr>
<td>MST-06</td>
<td>66.9</td>
<td>(0.0)</td>
<td>10.3</td>
<td>80.2</td>
<td>(61.7)</td>
<td>29.9</td>
<td>84.8</td>
<td>(62.5)</td>
<td>25.5</td>
</tr>
<tr>
<td>MST$_{Malt}$</td>
<td>68.6</td>
<td>(9.4)</td>
<td>11.0</td>
<td>82.3</td>
<td>(69.2)</td>
<td>31.2</td>
<td><strong>86.7</strong></td>
<td>(60.0)</td>
<td><strong>29.8</strong></td>
</tr>
</tbody>
</table>
Results
Non-Projective Dependency Parsing

- **AS**
  - Performance of $S_u$ is better for:
    - Czech and Slovene $\rightarrow$ more non-projective arcs in this language.
  - In AS $S_u$ is lower than $S_p$, however the drop is not really significant.
  - For Arabic the results are not meaningful since there are only 11 non-projective arcs in the whole set.

- **ME**
  - $S_u$ outperforms all other parsers.
  - The positive effect of $S_u$ is dependent on the non-projectivity arcs in the language.
Happy holidays
References


