Transition-based Dependency Parsing

Niranjan Balasubramanian

September 01, 2015

Slides Adapted from Nivre and Manning.
• **Dependency tree** -- A dependency tree is a tree structure composed of the input words and meets a few constraints:
  
  - Single-head
  - Connected
  - Acyclic

**Projective Parse:**

Arcs don’t cross each other.
Mostly true for English.
More from our awesome volunteers!
Rules of the game!
-- Keep move items from buffer to stack.
-- If the top item on stack is a dependent of the top buffer item output dependency relation and drop the item from stack.
-- If the top buffer item is a dependent of any item in stack, move buffer item to stack, but keep the head in stack.
Example Transition Sequence

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

ROOT Economic news had little effect on financial markets .
adj noun verb adj noun prep adj noun .

Assume we have some black-box that takes two words and magically gives you the dependency relation between them if one exists.
Root: Economic news had little effect on financial markets.
Example Transition Sequence

**Shift:**
Move Economic to stack.

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

\[\text{ROOT Economic news had little effect on financial markets .}\]
\[\text{adj noun verb adj noun prep adj noun .}\]
Example Transition Sequence

**Left Arc:**

Add left-arc `amod(news, Economic)` to A.
Remove Economic from stack since it now has head in A.
NOTE: Left-arc was possible only as Economic did not previously have a head in A.

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

Shift

Move news to stack.

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

ROOT Economic news had little effect on financial markets .

dependent mod

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

**Left Arc:**

Add left-arc *nsubj* (had, news) to A.
Remove news from stack since it now has head in A.

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

**Shift**

Move had to stack.

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

```
ROOT    Economic    news    had    little    effect    on    financial    markets .
   adj    noun     verb    adj    noun    prep    adj    noun .
```

\text{amod} \quad \text{nsubj}
Example Transition Sequence

**Right Arc:**

Add right-arc \textit{root}(\textsc{ROOT}, \textit{had}) to A.
Keep had in stack.

\textbf{NOTE:} We are keeping had because it can have other dependents on the left.

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

Black-box did not find any dependence relation for had and little.

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

**Left-arc:**

Add amod(effect, little) to A.
Remove little from stack.

\[
\begin{array}{c}
\text{ROOT, had}_S \\
\text{ROOT} \\
\text{amod} \\
\text{ROOT} \\
\text{Economic} \\
\text{adj} \\
\text{nsubj} \\
\text{news} \\
\text{noun} \\
\text{had} \\
\text{verb} \\
\text{little} \\
\text{adj} \\
\text{effect} \\
\text{noun} \\
\text{on} \\
\text{financial} \\
\text{adj} \\
\text{markets} \\
\text{.} \\
\end{array}
\]
And on it goes until …
As a supervised classification task.

- Given the current state (i.e., stack, buffer and A) predict the next action.

- Can be viewed as a supervised learning problem.
  - Four way classification (if un-typed dependencies)
  - m-way classification, where \( m = 2 \times \text{number of types} + 2 \)

- Features
  - Compute features of the current configuration of the stack, buffer and A.
  - Word in stack, POS of word, Word in buffer and POS of Word in buffer.
  - Other features: Length of dependency arc

- Greedy classifier (no search involved)
  - At each stage ask the classifier to predict the next transition.
  - Select the best legal transition and apply it.
  - Works quite well, close to PCFG.

- Quite fast!
  - \( O(N) \) in length of sentence.