Natural Language Processing I

lecture 8: constituent parsing (end), dependency parsing (start)

Ivan Titov

Institute for Logic, Language and Computation
Outline

- Unsupervised estimation of PCFGs
- Grammar refinement (producing a state-of-the-art parser)
- Dependency parsing

Why important?
PCFG estimation with EM

- Notation
- Calculating inside probabilities
- Calculating outside probabilities
- The inside-outside algorithm (EM)
Notation

- **Non-terminal symbols (latent variables):** \( \{N^1, \ldots, N^n\} \)
- **Sentence (observed data):** \( \{w_1, \ldots, w_m\} = w_{1m} \)
- \( N^j_{pq} \) denotes that \( N^j \) spans \( w_{pq} \) in the sentence
Inside probability

- Definition (compare with backward prob for HMMs):
  \[
  \beta_j(p, q) = P(w_p, \ldots, w_q | N^j_{pq}, G) = P(N^j_{pq} \rightarrow w_{pq} | G)
  \]

- Computed recursively
  - Base case: \[ \beta_j(k, k) = P(w_k | N^j_{kk}, G) = P(N_j \rightarrow w_k | G) \]
  - Induction:
    \[
    \beta_j(p, q) = \sum_{r,s} \sum_{d=p}^{q-1} P(N^j \rightarrow N^r N^s) \beta_r(p, d) \beta_s(d + 1, q)
    \]

The grammar is binarized

Does it remind you something?

Probabilistic CKY (Viterbi)! It is again an instance of Sum-product!
Inside probability: example

- Consider the following example:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP → DET N</td>
<td>0.8</td>
</tr>
<tr>
<td>DET → a</td>
<td>0.6</td>
</tr>
<tr>
<td>N → apple</td>
<td>0.8</td>
</tr>
<tr>
<td>NP → N</td>
<td>0.2</td>
</tr>
<tr>
<td>DET → the</td>
<td>0.4</td>
</tr>
<tr>
<td>N → orange</td>
<td>0.2</td>
</tr>
</tbody>
</table>

\[ \beta_{DET}(1,1) = \]
Consider the following example:

\[
\begin{array}{ccc}
\text{NP} & \text{DET} & \text{N} \\
0.8 & 0.6 & 0.8 \\
\text{DET} & \text{a} & \text{DET} \rightarrow \text{the} \\
0.2 & 0.4 & \\
\text{N} & \text{apple} & \text{N} \rightarrow \text{orange} \\
0.2 & 0.2 &
\end{array}
\]

\[
\beta_{DET}(1,1) = P(\text{the} \mid DET_{11}, G) = P(DET \rightarrow \text{the} \mid G) = 0.4
\]
Consider the following example:

\[
\begin{align*}
\text{NP} & \rightarrow \text{DET N} \quad 0.8 & \quad \text{NP} & \rightarrow \text{N} \quad 0.2 \\
\text{DET} & \rightarrow \text{a} \quad 0.6 & \quad \text{DET} & \rightarrow \text{the} \quad 0.4 \\
\text{N} & \rightarrow \text{apple} \quad 0.8 & \quad \text{N} & \rightarrow \text{orange} \quad 0.2
\end{align*}
\]

\[
\begin{align*}
\beta_{\text{DET}}^{(1,1)} &= P(\text{the} \mid \text{DET}_{11}, G) = P(\text{DET} \rightarrow \text{the} \mid G) = 0.4 \\
\beta_{\text{N}}^{(2,2)} &= P(\text{N} \rightarrow \text{orange} \mid G) = 0.2
\end{align*}
\]
Consider the following example:

\[
\begin{align*}
\beta_{DET}(1,1) &= P(\text{the} \mid DET_{11}, G) = P(DET \rightarrow \text{the} \mid G) = 0.4 \\
\beta_{N}(2,2) &= P(N \rightarrow \text{orange} \mid G) = 0.2 \\
\beta_{NP}(1,2) &= 
\end{align*}
\]
Consider the following example:

\[
\begin{align*}
\beta_{\text{DET}} (1,1) &= P(\text{the} \mid \text{DET}_{11}, G) = P(\text{DET} \rightarrow \text{the} \mid G) = 0.4 \\
\beta_{\text{N}} (2,2) &= P(\text{N} \rightarrow \text{orange} \mid G) = 0.2 \\
\beta_{\text{NP}} (1,2) &= P(\text{NP} \rightarrow \text{DET} \cdot \text{N}) \beta_{\text{DET}} (1,1) \beta_{\text{N}} (2,2)
\end{align*}
\]
Inside probability: example

Consider the following example:

\[
\begin{align*}
\text{NP} & \rightarrow \text{DET} \ \text{N} \quad 0.8 & \quad \text{NP} & \rightarrow \text{N} \quad 0.2 \\
\text{DET} & \rightarrow \text{a} \quad 0.6 & \quad \text{DET} & \rightarrow \text{the} \quad 0.4 \\
\text{N} & \rightarrow \text{apple} \quad 0.8 & \quad \text{N} & \rightarrow \text{orange} \quad 0.2
\end{align*}
\]

\[
\beta_{\text{DET}}(1,1) = P(\text{the} \mid \text{DET}_{11}, G) = P(\text{DET} \rightarrow \text{the} \mid G) = 0.4
\]

\[
\beta_{\text{N}}(2,2) = P(\text{N} \rightarrow \text{orange} \mid G) = 0.2
\]

\[
\beta_{\text{NP}}(1,2) = P(\text{NP} \rightarrow \text{DET} \cdot \text{N}) \beta_{\text{DET}}(1,1) \beta_{\text{N}}(2,2) = 0.8 \times 0.4 \times 0.2
\]

\[
\beta_{\text{NP}}(1,2) = 0.064
\]
Consider the following example:

\[
\begin{align*}
\text{NP} &\rightarrow \text{DET N} \quad 0.8 & \text{NP} &\rightarrow \text{N} \quad 0.2 \\
\text{DET} &\rightarrow \text{a} \quad 0.6 & \text{DET} &\rightarrow \text{the} \quad 0.4 \\
\text{N} &\rightarrow \text{apple} \quad 0.8 & \text{N} &\rightarrow \text{orange} \quad 0.2 \\
\end{align*}
\]

\[
\beta_{\text{DET}}(1,1) = P(\text{the} \mid \text{DET}_{11}, G) = P(\text{DET} \rightarrow \text{the} \mid G) = 0.4 \\
\beta_{\text{N}}(2,2) = P(\text{N} \rightarrow \text{orange} \mid G) = 0.2 \\
\beta_{\text{NP}}(1,2) = P(\text{NP} \rightarrow \text{DET} \cdot \text{N}) \beta_{\text{DET}}(1,1) \beta_{\text{N}}(2,2) \\
\quad = 0.8 \times 0.4 \times 0.2 \\
\beta_{\text{NP}}(1,2) = 0.064
\]

The probability of a sentence under a PCFG?

\[
\beta_S(1, m) = P(S \rightarrow w_1, \ldots, w_m \mid G)
\]
Outside probability

Definition (compare with forward prob for HMMs):

\[ \alpha_j(p, q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m} | G) \]

The joint probability of starting with \( S \), generating words \( w_1, \ldots, w_{p-1}, \) the non terminal \( N^j \) and words \( w_{q+1}, \ldots, w_m \).
Calculating outside probability

- Computed recursively, base case

\[ \alpha_1(1, m) = \alpha_S(1, m) = 1 \quad \alpha_{j \neq 1}(1, m) = 0 \]

- Induction?

- Intuition: \( N_{pq}^j \) must be either the L or R child of a parent node. We first consider the case when it is the L child.

\[ \alpha_j(p, q) = P(w_{1(p-1)}, N_{pq}^j, w_{(q+1)m}|G) \]
Calculating outside probability

- The yellow area is the probability we would like to calculate
  - How do we decompose it?
Step 1: We assume that $N^f_{pe}$ is the parent of $N^j_{pq}$. Its outside probability, $\alpha_f(p, e)$, (represented by the yellow shading) is available recursively. But how do we compute the green part?
Calculating outside probability

- Step 1: The red shaded area is the inside probability for $N_{(q+1)e}^g$, i.e. $\beta_g(q + 1, e)$
Calculating outside probability

- Step 3: The blue shaded area is just the production \( N^f \rightarrow N^j N^g \), the corresponding probability \( P(N^f \rightarrow N^j N^g | N^f, G) \).
Calculating outside probability

- If we multiply the terms together, we have the joint probability corresponding to the yellow, red and blue areas, assuming $N^j$ was the L child of $N^f$, and give fixed non-terminals $f$ and $g$, as well as a fixed partition $e$

$$\alpha_f(p, e) \beta_g(q + 1, e) P(N_f \rightarrow N^j N^g)$$

What if we do not want to assume this?
Calculating outside probability

- The joint probability corresponding to the yellow, red and blue areas, assuming $N_f$ was the L child of some non-terminal:

$$\sum_{f,g} \sum_{e=q+1}^{m} \alpha_f(p, e) \beta_g(q + 1, e) P(N_f \rightarrow N^j N^g)$$
Calculating outside probability

- The joint probability corresponding to the yellow, red and blue areas, assuming $N^f$ was the R child of some non-terminal:

$$\sum\sum_{f,g}^{p-1} \alpha_f(e, q) \beta_g(e, p-1) P(N_f \rightarrow N^g N^j)$$
Calculating outside probability

- The joint final joint probability (the sum over the L and R cases):

\[
\alpha_j(p, q) = \sum_{f,g} \sum_{e=q+1}^{m} \alpha_f(p, e) \beta_g(q + 1, e) P(N_f \rightarrow N^j N^g) + \sum_{f,g} \sum_{e=1}^{p-1} \alpha_f(e, q) \beta_g(e, p - 1) P(N_f \rightarrow N^g N^j)
\]
Recall EM

- **Maximum likelihood estimation**
  \[ P(y|z) = \frac{C(z, y)}{C(z)} \]

- **Supervised case**: counts are observable

- **EM**:
  \[ \theta^k = \arg \max_{\theta} Q(\theta, \theta^{k-1}) \]
  where \( Q \) is expected likelihood

\[
Q(\theta, \theta^{k-1}) = \sum_{l=1}^{L} \sum_{s} P(s|x^{(l)}, \theta^{k-1}) \log P(x^{(l)}, s|\theta)
\]

- Equivalently, estimated ("expected") counts based on the previous model:

\[
\theta^t = P(y|z) = \frac{E_{\theta^{t-1}}[C(z, y)|\text{observations}]}{E_{\theta^{t-1}}[C(z)|\text{observations}]}
= \frac{E_{\theta^{t-1}}[C(z, y)|x^1, \ldots, x^L]}{E_{\theta^{t-1}}[C(z)|x^1, \ldots, x^L]}
\]

- \( z \) – some latent variable (event), \( y \) – some other variable (event)
- \( s \) – are all latent variables in the model (latent state)
Inside-outside algorithm

- For PCFGs we need to compute:

\[ \theta^t = P(N^j \rightarrow N^r N^s | N^j) = \frac{E_{\theta^{t-1}}[C(N^j \rightarrow N^r N^s, N^j) | observations]}{E_{\theta^{t-1}}[C(N^j) | observations]} \]

Let start with the numerator.
Inside-outside algorithm

What we want here is, for given non-terminals \( r \) and \( s \), a probability that \( N^j \) is both used at some point in the derivation and accounts for span \( w_{pq} \). Since there are two symbols in the RHS, we need to pick a partition between the \( p \) and \( q \), (call it \( d \)):

\[
\alpha_j(p, q) P(N^j \rightarrow N^r N^s) \beta_r(p, d) \beta_s(d + 1, q) = P(N_{pq}^j \rightarrow N_{pd}^r, N_{d,q}^s, w_1, \ldots, w_m)
\]
What we want here is, for given non-terminals $r$ and $s$, a probability that $N^j$ is both used at some point in the derivation and accounts for span $w_{pq}$. Since there are two symbols in the RHS, we need to pick a partition between the $p$ and $q$, (call it $d$):

$$
\alpha_j(p, q) P(N^j \rightarrow N^r N^s) \sum_{d=p}^{q-1} \beta_r(p, d) \beta_s(d + 1, q)
$$

$$
= P(N^j_{pq} \rightarrow N^r, N^s, w_1, \ldots, w_m)
$$

Now sum over all partitions

The joint probability
Inside-outside algorithm

For PCFGs we need to compute the numerator:

\[ E_{\theta t-1}[C(N^j \rightarrow N^r N^s, N^j)|\text{observations}] = \sum_{l=1}^{L} E_{\theta t-1}[C(N^j \rightarrow N^r N^s, N^j)|w_1^{(l)}, \ldots, w_m^{(l)}] \]

\[ = \sum_{l=1}^{L} \sum_{p=1}^{m} \sum_{q=p+1}^{m} E_{\theta t-1} C(N^j_{pq} \rightarrow N^r N^s, N^j)|w_1^{(l)}, \ldots, w_m^{(l)}] \]

Varied the span

We need to get from joint probabilities

\[ P(N^j_{pq} \rightarrow N^r, N^s, w_1, \ldots, w_m) \]

to conditional ones

Easy: divide the joint probs by

\[ P(w_1, \ldots, w_m) = \beta_S(1, m) \]
The result is similar to posterior probabilities of transitions in the forward-backward algorithm

\[ E_{\theta t-1}[C(N^j \rightarrow N^r N^s, N^j)|\text{observations}] = \]

\[
\sum_{l} \sum_{p=1}^{m} \sum_{q=p+1}^{m} \frac{\alpha_j(p, q) P(N^j \rightarrow N^r N^s) \left[ \sum_{d=p}^{q-1} \beta_r(p, d) \beta_s(d + 1, q) \right]}{P(w_1^{(l)}, \ldots, w_m^{(l)})}
\]

The probability of coming from Start

The probability of transition

The probability of getting the rest

The number of opportunities for the transition to happen

The probability of the sentence \[ \beta_S(1, m) \]
For PCFGs we need to compute:

\[ \theta^t = P(N^j \rightarrow N^r N^s | N^j) = \frac{E_{\theta^{t-1}}[C(N^j \rightarrow N^r N^s, N^j)|observations]}{E_{\theta^{t-1}}[C(N^j)|observations]} \]

Know how to do this!

How do we get the denominator?

We can just sum over \( r \) and \( s \! \)!

We are done with the IO algorithm
Summary PCFG

- PCFG supervised estimation – done
- PCFG parsing – done
- PCFG unsupervised estimation – done
- Grammar refinement – learn a better grammar – in a moment

Actually, unsupervised estimation (IO algorithm) does not result in linguistically appropriate grammars

... but serves a basis for many important methods (incl. grammar refinement, machine translation, …)
Outline

- Unsupervised estimation of PCFGs
- Grammar refinement (producing a state-of-the-art parser)
- Dependency parsing
Weaknesses of (treebank) PCFGs

- They do not encode lexical preferences (i.e. information about words)
- They do not encode structural properties (beyond single rules)
Context-free constraint

- Subject and object NPs are (statistically) very different
  - NPs under S vs. NPs under VP

Independence assumptions in PCFGs are too strong for this grammar
Subject and object NPs are (statistically) very different
  - NPs under S vs. NPs under VP

<table>
<thead>
<tr>
<th>Types of NP</th>
<th>NP PP</th>
<th>D N</th>
<th>PN</th>
</tr>
</thead>
<tbody>
<tr>
<td>All NPs</td>
<td>11%</td>
<td>9%</td>
<td>6%</td>
</tr>
<tr>
<td>NPs under S (subjects)</td>
<td>9%</td>
<td>9%</td>
<td>21%</td>
</tr>
<tr>
<td>NPs under VP (objects)</td>
<td>23%</td>
<td>7%</td>
<td>4%</td>
</tr>
</tbody>
</table>

Many more pronouns as subjects; much less frequently prepositional phrases as subjects
The game of designing a grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Parent annotation [Johnson ’98]
The game of designing a grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Parent annotation [Johnson ’98]
  - Head lexicalization [Collins ’99, Charniak ’00]
The game of designing a grammar

Annotation refines base treebank symbols to improve statistical fit of the grammar

- Parent annotation [Johnson ‘98]
- Head lexicalization [Collins ‘99, Charniak ‘00]
- Automatic clustering?
The game of designing a grammar

- Manually split categories
  - NP: subject vs object
  - DT: determiners vs demonstratives
  - IN: sentential vs prepositional

- Advantages:
  - Fairly compact grammar
  - Linguistic motivations

- Disadvantages:
  - Performance leveled out
  - Manually annotated

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Treebank Grammar</td>
<td>72.6</td>
</tr>
<tr>
<td>Klein &amp; Manning ’03</td>
<td>86.3</td>
</tr>
</tbody>
</table>
Advantages:

- **Automatically learned:**
  - Label *all* nodes with latent variables.
  - Same number \( k \) of subcategories for all categories.

\[ \text{He was right} \]

\[ \text{He was right} \]
Automatic Annotation Induction

- **Advantages:**
  - **Automatically learned:**
    - Label *all* nodes with latent variables.
    - Same number $k$ of subcategories for all categories.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klein &amp; Manning ’03</td>
<td>86.3</td>
</tr>
<tr>
<td>Matsuzaki et al. ’05</td>
<td>86.7</td>
</tr>
</tbody>
</table>
Learning Latent Annotations

**EM algorithm:**
- Brackets are known
- Base non-terminals are known
- Only induce subcategories

Just like Inside-Outside but for a fixed tree with partially observable non-terminals
Overview

- Hierarchical Training
- Adaptive Splitting
- Parameter Smoothing

Limit of computational resources
Refinement of the DT tag

```
DT
  the (0.50)
  a (0.24)
  The (0.08)
```

```
DT-1
  a (0.61)
  the (0.19)
  an (0.11)

DT-2
  the (0.80)
  The (0.15)
  a (0.01)

DT-3
  this (0.39)
  that (0.28)
  That (0.11)

DT-4
  some (0.20)
  all (0.19)
  those (0.12)
```
Refinement of the DT tag

DT

the (0.50)
a (0.24)
The (0.08)

DT-1
DT-2
DT-3
DT-4
DT-5
DT-6
DT-7
DT-8
Hierarchical refinement of the DT tag
Hierarchical refinement

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>87.3</td>
</tr>
<tr>
<td>Hierarchical Training</td>
<td>88.4</td>
</tr>
</tbody>
</table>

[ Petrov and Klein, '06 ]
Refinement of the , tag

- Splitting all categories the same amount is wasteful:
Revisiting the DT tag

Oversplit?
Adaptive Splitting

- Want to split complex categories more
- Idea: split everything, roll back splits which were least useful

```
+------+
|   (0.54)   |
| a (0.25) |
| The (0.09) |
+------+
```

```
+------+
|   (0.61)   |
| a (0.19) |
| The (0.11) |
+------+
```

```
+------+
|   (0.80)   |
| a (0.01) |
+------+
```

```
+------+
|   (0.96)   |
| The (0.01) |
+------+
```

```
+------+
|   (0.93)   |
| A (0.02) |
| No (0.01) |
+------+
```
Adaptive Splitting

- Want to split complex categories more
- Idea: split everything, roll back splits which were least useful
Adaptive Splitting

- Want to split complex categories more
- Idea: split everything, roll back splits which were least useful
Adaptive Splitting

- Evaluate loss in likelihood from removing each split

  **Data likelihood with split reversed**

  **Data likelihood with split**

- No loss in accuracy when 50% of the splits are reversed.
Adaptive Splitting Results

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>88.4</td>
</tr>
<tr>
<td>With 50% Merging</td>
<td>89.5</td>
</tr>
</tbody>
</table>

[Petrov and Klein, '06]
Number of Phrasal Subcategories

NP
VP
PP
ADVP
S
ADJP
SBAR
QP
WHNP
PRN
NX
SINV
PRT
WHPP
SQ
CONJP
FRAG
NAC
UCP
WHADVP
INTJ
SBARQ
RRC
WHADJP
X
ROOT
LST
Number of Phrasal Subcategories
Number of PoS tags
Number of PoS tags

NNP
JJ
NNS
NN
A feel for what it learns

- **Proper Nouns (NNP):**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NNP-12</td>
<td>John</td>
<td>Robert</td>
<td>James</td>
</tr>
<tr>
<td>NNP-2</td>
<td>J.</td>
<td>E.</td>
<td>L.</td>
</tr>
<tr>
<td>NNP-1</td>
<td>Bush</td>
<td>Noriega</td>
<td>Peters</td>
</tr>
<tr>
<td>NNP-15</td>
<td>New</td>
<td>San</td>
<td>Wall</td>
</tr>
<tr>
<td>NNP-3</td>
<td>York</td>
<td>Francisco</td>
<td>Street</td>
</tr>
</tbody>
</table>

- **Personal pronouns (PRP):**

<table>
<thead>
<tr>
<th>PRP-0</th>
<th>it</th>
<th>He</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRP-1</td>
<td>it</td>
<td>he</td>
<td>they</td>
</tr>
<tr>
<td>PRP-2</td>
<td>it</td>
<td>them</td>
<td>him</td>
</tr>
</tbody>
</table>
### Relative adverbs (RBR):

<table>
<thead>
<tr>
<th>RBR-0</th>
<th>further</th>
<th>lower</th>
<th>higher</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBR-1</td>
<td>more</td>
<td>less</td>
<td>More</td>
</tr>
<tr>
<td>RBR-2</td>
<td>earlier</td>
<td>Earlier</td>
<td>later</td>
</tr>
</tbody>
</table>

### Cardinal Numbers (CD):

<table>
<thead>
<tr>
<th>CD-7</th>
<th>one</th>
<th>two</th>
<th>Three</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD-4</td>
<td>1989</td>
<td>1990</td>
<td>1988</td>
</tr>
<tr>
<td>CD-11</td>
<td>million</td>
<td>billion</td>
<td>trillion</td>
</tr>
<tr>
<td>CD-0</td>
<td>1</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>CD-3</td>
<td>1</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>CD-9</td>
<td>78</td>
<td>58</td>
<td>34</td>
</tr>
</tbody>
</table>
We have not really looked into but do we have all the components?

We will leave this for later.
Outline

- Unsupervised estimation of PCFGs
- Grammar refinement (producing a state-of-the-art parser)
- Dependency syntax
- Graph-based dependency parsing
**NLP Problems**
- Doc. classification
- Topic analysis
- Shallow synt. parsing /tagging
- Syntactic parsing
- Relation extraction
- Semantic parsing
- Models of inference
- Machine translation
- Question answering
- Opinion analysis
- Summarization
- Dialogue systems
- ...

**Types of structures**
- Bags
- Sequences / Chains
- Spanning trees
- Hierarchical trees
- DAGs
- Bipartite graphs
- ...

**Models/Views**
- Naive Bayes
- Topic models
- HMMs
- History- / transition-based models
- PCFGs
- DOP
- Global scoring (e.g., MST)
- "IBM" models
- ...

**Set-ups**
- Supervised estimation
- Unsupervised
- Partially/semi-supervised
- ...

**Modeling frameworks**
- Generative ML
- Generative Bayes
- Discriminative
- Discriminative Bayes
- Representation learning (factorizations / NNs)
Dependency representation

Terminology:
- *news* is the head of *economic*,
- *economic* is the dependent of *news*

[Some illustrations and slides here are from Ryan McDonald and Joakim Nivre]
Constituent (phrase-structure) representation

S
  VP
    NP
      PP
        NP
          NP
            JJ
            NN
            VBD
            JJ
          NN
          IN
          JJ
          NNS
        NP
          PU

Economic news had little effect on financial markets.
Dependency vs Constituency

- **Dependency structures** explicitly represent
  - head-dependent relations (**directed arcs**),
  - functional categories (**arc labels**)
  - possibly some structural categories (**parts of speech**)

- **Phrase (aka constituent) structures** explicitly represent
  - phrases (**nonterminal nodes**),
  - structural categories (**nonterminal labels**)

- **Hybrid representations** may combine all elements
Dependency Graphs

- A dependency structure can be defined as a directed graph $G$, consisting of
  - a set $V$ of nodes (vertices),
  - a set $A$ of arcs (directed edges),
  - a linear precedence order $<$ on $V$ (word order).

- Labeled graphs
  - nodes in $V$ are labeled with word forms (and annotation).
  - arcs in $A$ are labeled with dependency types
    - $L = \{l_1, \ldots, l_{|L|}\}$ is the set of permissible arc labels;
    - Every arc in $A$ is a triple $(i,j,k)$, representing a dependency from $w_i$ to $w_j$
      with label $l_k$. 
For dependency graph $G = (V, A)$ with label set $L = \{l_1, \ldots, l_{|L|}\}$:

- $i \rightarrow j \equiv \exists k : (i, j, k) \in A$
- $i \leftrightarrow j \equiv i \rightarrow j \land j \rightarrow i$
- $i \rightarrow^* j \equiv i = j \lor \exists i' : i \rightarrow i', i' \rightarrow^* j$
- $i \leftrightarrow^* j \equiv i = j \lor \exists i' : i \leftrightarrow i', i' \leftrightarrow^* j$
Formal Conditions on Dependency Graphs

- **G is (weakly) connected:**
  - If $i, j \in V$, $i \leftrightarrow^* j$.

- **G is acyclic:**
  - If $i \rightarrow j$, then not $j \rightarrow^* i$.

- **G obeys the single-head constraint:**
  - If $i \rightarrow j$, then not $i' \rightarrow j$, for any $i' \neq i$.

- **G is projective:**
  - If $i \rightarrow j$, then $i \rightarrow^* i'$, for any $i'$ such that $i < i' < j$ or $j < i' < i$. 
**Connectedness, Acyclicity and Single-Headness**

- **Intuitions**
  - Syntactic structure is complete (**connectedness**)
  - Syntactic structure is hierarchical (**acyclicity**)
  - Every word has at most one syntactic head (**single-head constraint**)
  - Connectedness can be enforced by adding a special root node

![Diagram of syntactic structure](image)
Connectedness, Acyclicity and Single-Headness

- **Intuitions**
  - Syntactic structure is complete (**connectedness**)
  - Syntactic structure is hierarchical (**acyclicity**)
  - Every word has at most one syntactic head (**single-head constraint**)
- Connectedness can be enforced by adding a special root node

![Syntactic Tree Diagram]

```
p
  pred
    nmod nmod
      root Economic news had
  obj
    nmod nmod
      little effect on
    nmod
      financial
  pc
    nmod
      markets
```
Most theoretical frameworks do not assume projectivity. Non-projective structures are needed to account for long-distance dependencies, free word order.
Dependency Parsing

The dependency parsing problem:

- **Input:** sentence \( x = w_0, w_1, \ldots, w_n \) (\( w_0 = \text{root} \))
- **Output:** dependency graph \( G(V,A) \)
  - \( V = \{0, 1, \ldots, n\} \) is the vertex set,
  - \( A \) is the arc set, i.e., \( (i, j, k) \in A \) represents a dependency from \( w_i \) to \( w_j \) with label \( l_k \in L \)

Two main approaches

- **Grammar-based parsing**
  - Context-free dependency grammars
  - Constraint dependency grammars
- **Data-driven parsing**
  - Graph-based models
  - Transition-based models
Conclusions

- We now know how to build an accurate constituent parser
  - The algorithm generalizes across languages and can be used to predict non-syntactic representations (e.g., meaning representations)

- We learnt about dependency syntax
- More about dependency parsing next time