IN2110: Methods in Language Technology

Dependency Parsing

Stephan Oepen

Language Technology Group (LTG)

April 23, 2019

Topics for Today

- Short recap:
  - (Local) Ambiguity in natural languages
  - Chart parsing: The Cocke–Kasami–Younger algorithm

Topics for Today

- Short recap:
  - (Local) Ambiguity in natural languages
  - Chart parsing: The Cocke–Kasami–Younger algorithm
- Granular evaluation of statistical parsers

Topics for Today

- Short recap:
  - (Local) Ambiguity in natural languages
  - Chart parsing: The Cocke–Kasami–Younger algorithm
- Granular evaluation of statistical parsers
- Move on to dependency syntax
  - Typological motivation
  - Contrast to constituent syntax
  - Properties of dependency graphs

Topics for Today

- Short recap:
  - (Local) Ambiguity in natural languages
  - Chart parsing: The Cocke–Kasami–Younger algorithm
- Granular evaluation of statistical parsers
- Move on to dependency syntax
  - Typological motivation
  - Contrast to constituent syntax
  - Properties of dependency graphs
- Data-driven dependency parsing
  - Variations on shift–reduce parsing
  - The arc-eager transition system
  - Thorough walk-through example

Topics for Today

- Short recap:
  - (Local) Ambiguity in natural languages
  - Chart parsing: The Cocke–Kasami–Younger algorithm
- Granular evaluation of statistical parsers
- Move on to dependency syntax
  - Typological motivation
  - Contrast to constituent syntax
  - Properties of dependency graphs
- Data-driven dependency parsing
  - Variations on shift–reduce parsing
  - The arc-eager transition system
  - Thorough walk-through example
- Sample exam questions

Exercise (1): Natural Language Ambiguity

Assume the following 'toy' grammar of English:

$$\mathrm{S} \rightarrow \mathrm{NP}$$
$$\mathrm{NP} \rightarrow \mathrm{Det}\ \mathrm{N}$$
$$\mathrm{N} \rightarrow \mathrm{N}\ \mathrm{N}$$

$$\mathrm{Det} \rightarrow \textit{the}$$
$$\mathrm{N} \rightarrow \textit{kitchen} \mid \textit{table} \mid \textit{towel} \mid \textit{rack}$$

Exercise (1): Natural Language Ambiguity

Assume the following 'toy' grammar of English:

$$S \rightarrow NP$$
$$NP \rightarrow Det\ N$$
$$N \rightarrow N\ N$$

$$Det \rightarrow \textit{the}$$
$$N \rightarrow \textit{kitchen} \mid \textit{table} \mid \textit{towel} \mid \textit{rack}$$

(1) How many different syntactic analyses, if any, does the grammar assign to the following strings?

(a) *the kitchen towel rack*
(b) *the kitchen table towel rack*

Exercise (2): CKY Parsing

Assume the following grammar and CKY parse table:

S → NP VP
VP → V NP
VP → VP PP
NP → NP PP
PP → P NP

	1	2	3	4	5
0	NP		S		S
1		V	VP		VP
2			NP		NP
3				P	PP
4					NP

Exercise (2): CKY Parsing

Assume the following grammar and CKY parse table:

S $\rightarrow$ NP VP
VP $\rightarrow$ V NP
VP $\rightarrow$ VP PP
NP $\rightarrow$ NP PP
PP $\rightarrow$ P NP

	1	2	3	4	5
0	NP		S		S
1		V	VP		VP
2			NP		NP
3				P	PP
4					NP

(2) Which pair(s) of 'input' cells and which production(s) give rise to the derivation of category S in 'target' cell $\langle 0, 5 \rangle$?

Evaluation: How to Quantify Parser Quality?

- A statistical parser (e.g. using a PCFG) will inevitably make 'mistakes'.
- The parser output (most probable derivation) differs from gold standard.
- How to measure parser quality? Suggestions for an evaluation metric?

Evaluation: How to Quantify Parser Quality?

- A statistical parser (e.g. using a PCFG) will inevitably make ‘mistakes’.
- The parser output (most probable derivation) differs from gold standard.
- How to measure parser quality? Suggestions for an evaluation metric?
- Sentence accuracy (‘exact match’) easy to interpret. Any deficiencies?

Evaluation: How to Quantify Parser Quality?

- A statistical parser (e.g. using a PCFG) will inevitably make 'mistakes'.
- The parser output (most probable derivation) differs from gold standard.
- How to measure parser quality? Suggestions for an evaluation metric?
- Sentence accuracy ('exact match') easy to interpret. Any deficiencies?
- The ParsEval metric (Black et al., 1991) measures constituent overlap.
- Precision, recall, and F_1 for labeled brackets $\rightarrow$ allow partial credit.

Evaluation: How to Quantify Parser Quality?

- A statistical parser (e.g. using a PCFG) will inevitably make 'mistakes'.
- The parser output (most probable derivation) differs from gold standard.
- How to measure parser quality? Suggestions for an evaluation metric?
- Sentence accuracy ('exact match') easy to interpret. Any deficiencies?
- The ParsEval metric (Black et al., 1991) measures constituent overlap.
- Precision, recall, and F_1 for labeled brackets $\rightarrow$ allow partial credit.
- De-facto standard for 25 years (combined with crossing brackets count).

ParsEval

Gold Standard

```
(NP (DT a)
    (ADVP (RB pretty)
          (JJ big))
    (NOM (NN dog)
         (POS 's)
         (NN house)) )
```

System Output

```
(NP (DT a)
    (JJ pretty)
    (NOM (JJ big)
         (NOM (NN dog)
              (POS 's)
              (NN house))))
```

ParsEval

Gold Standard

```
(NP (DT a)
    (ADVP (RB pretty)
          (JJ big))
    (NOM (NN dog)
         (POS 's)
         (NN house)) )
```

System Output

```
(NP (DT a)
    (JJ pretty)
    (NOM (JJ big)
         (NOM (NN dog)
              (POS 's)
              (NN house))))
```

0,6 NP

ParsEval

Gold Standard

```
(NP (DT a)
    (ADVP (RB pretty)
          (JJ big))
    (NOM (NN dog)
         (POS 's)
         (NN house)) )
```

System Output

```
(NP (DT a)
    (JJ pretty)
    (NOM (JJ big)
         (NOM (NN dog)
              (POS 's)
              (NN house))))
```

0,6 NP
0,1 DT

ParsEval

Gold Standard

```
(NP (DT a)
    (ADVP (RB pretty)
          (JJ big))
    (NOM (NN dog)
         (POS 's)
         (NN house)) )
```

System Output

```
(NP (DT a)
    (JJ pretty)
    (NOM (JJ big)
         (NOM (NN dog)
              (POS 's)
              (NN house))))
```

0,6 NP
0,1 DT
1,3 ADVP

ParsEval

Gold Standard

(NP (DT *a*)
(ADVP (RB *pretty*)
(JJ *big*))
(NOM (NN *dog*)
(POS *'s*)
(NN *house*)))

System Output

(NP (DT *a*)
(JJ *pretty*)
(NOM (JJ *big*)
(NOM (NN *dog*)
(POS *'s*)
(NN *house*))))

0,6 NP
0,1 DT
1,3 ADVP
1,2 RB

ParsEval

Gold Standard

```
(NP (DT a)
    (ADVP (RB pretty)
          (JJ big))
    (NOM (NN dog)
         (POS 's)
         (NN house)) )
```

System Output

```
(NP (DT a)
    (JJ pretty)
    (NOM (JJ big)
         (NOM (NN dog)
              (POS 's)
              (NN house))))
```

0,6	NP	1,2	RB
0,1	DT	2,3	JJ
1,3	ADVP		

ParsEval

Gold Standard

```
(NP (DT a)
    (ADVP (RB pretty)
          (JJ big))
    (NOM (NN dog)
         (POS 's)
         (NN house)) )
```

System Output

```
(NP (DT a)
    (JJ pretty)
    (NOM (JJ big)
         (NOM (NN dog)
              (POS 's)
              (NN house))))
```

0,6	NP	1,2	RB
0,1	DT	2,3	JJ
1,3	ADVP	3,6	NOM

ParsEval

Gold Standard

```
(NP (DT a)
    (ADVP (RB pretty)
          (JJ big))
    (NOM (NN dog)
         (POS 's)
         (NN house)) )
```

System Output

```
(NP (DT a)
    (JJ pretty)
    (NOM (JJ big)
         (NOM (NN dog)
              (POS 's)
              (NN house))))
```

0,6	NP	1,2	RB	3,4	NN
0,1	DT	2,3	JJ		
1,3	ADVP	3,6	NOM		

ParsEval

Gold Standard

```
(NP (DT a)
    (ADVP (RB pretty)
          (JJ big))
    (NOM (NN dog)
         (POS 's)
         (NN house)) )
```

System Output

```
(NP (DT a)
    (JJ pretty)
    (NOM (JJ big)
         (NOM (NN dog)
              (POS 's)
              (NN house))))
```

0,6	NP	1,2	RB	3,4	NN
0,1	DT	2,3	JJ	4,5	POS
1,3	ADVP	3,6	NOM		

ParsEval

Gold Standard

(NP (DT *a*)
    (ADVP (RB *pretty*)
              (JJ *big*))
    (NOM (NN *dog*)
              (POS *'s*)
              (NN *house*)))

System Output

(NP (DT *a*)
    (JJ *pretty*)
    (NOM (JJ *big*)
         (NOM (NN *dog*)
              (POS *'s*)
              (NN *house*))))

0,6	NP	1,2	RB	3,4	NN
0,1	DT	2,3	JJ	4,5	POS
1,3	ADVP	3,6	NOM	5,6	NN

ParsEval

Gold Standard

```
(NP (DT a)
    (ADVP (RB pretty)
          (JJ big))
    (NOM (NN dog)
         (POS 's)
         (NN house)) )
```

0,6	NP	1,2	RB	3,4	NN
0,1	DT	2,3	JJ	4,5	POS
1,3	ADVP	3,6	NOM	5,6	NN

System Output

```
(NP (DT a)
    (JJ pretty)
    (NOM (JJ big)
         (NOM (NN dog)
              (POS 's)
              (NN house))))
```

0,6	NP	2,6	NOM	3,4	NN
0,1	DT	2,3	JJ	4,5	POS
1,2	JJ	3,6	NOM	5,6	NN

ParsEval

Gold Standard

```
(NP (DT a)
    (ADVP (RB pretty)
          (JJ big))
    (NOM (NN dog)
         (POS 's)
         (NN house)) )
```

0,6	NP	1,2	RB	3,4	NN
0,1	DT	2,3	JJ	4,5	POS
1,3	ADVP	3,6	NOM	5,6	NN

System Output

```
(NP (DT a)
    (JJ pretty)
    (NOM (JJ big)
         (NOM (NN dog)
              (POS 's)
              (NN house))))
```

0,6	NP	2,6	NOM	3,4	NN
0,1	DT	2,3	JJ	4,5	POS
1,2	JJ	3,6	NOM	5,6	NN

Correct: 7

ParsEval

Gold Standard

(NP (DT *a*)
    (ADVP (RB *pretty*)
          (JJ *big*))
    (NOM (NN *dog*)
         (POS *'s*)
         (NN *house*)))

0,6	NP	1,2	RB	3,4	NN
0,1	DT	2,3	JJ	4,5	POS
1,3	ADVP	3,6	NOM	5,6	NN

System Output

(NP (DT *a*)
    (JJ *pretty*)
    (NOM (JJ *big*)
         (NOM (NN *dog*)
              (POS *'s*)
              (NN *house*))))

0,6	NP	2,6	NOM	3,4	NN
0,1	DT	2,3	JJ	4,5	POS
1,2	JJ	3,6	NOM	5,6	NN

Recall: $\frac{Correct}{Gold} = \frac{7}{9}$ Precision: $\frac{Correct}{System} = \frac{7}{9}$

ParsEval

Gold Standard

```
(NP (DT a)
    (ADVP (RB pretty)
          (JJ big))
    (NOM (NN dog)
         (POS 's)
         (NN house)) )
```

0,6	NP	1,2	RB	3,4	NN
0,1	DT	2,3	JJ	4,5	POS
1,3	ADVP	3,6	NOM	5,6	NN

System Output

```
(NP (DT a)
    (JJ pretty)
    (NOM (JJ big)
         (NOM (NN dog)
              (POS 's)
              (NN house))))
```

0,6	NP	2,6	NOM	3,4	NN
0,1	DT	2,3	JJ	4,5	POS
1,2	JJ	3,6	NOM	5,6	NN

Recall: $\frac{Correct}{Gold} = \frac{7}{9}$ Precision: $\frac{Correct}{System} = \frac{7}{9}$ F_1 score: $\frac{7}{9}$

ParsEval

Gold Standard

(NP (DT *a*)
    (ADVP (RB *pretty*)
          (JJ *big*))
    (NOM (NN *dog*)
         (POS *'s*)
         (NN *house*)))

0,6 NP	1,2 RB	3,4 NN
0,1 DT	2,3 JJ	4,5 POS
1,3 ADVP	3,6 NOM	5,6 NN

System Output

(NP (DT *a*)
    (JJ *pretty*)
    (NOM (JJ *big*)
         (NOM (NN *dog*)
              (POS *'s*)
              (NN *house*))))

0,6 NP	2,6 NOM	3,4 NN
0,1 DT	2,3 JJ	4,5 POS
1,2 JJ	3,6 NOM	5,6 NN

Recall: $\frac{Correct}{Gold} = \frac{2}{3}$ Precision: $\frac{Correct}{System} = \frac{2}{3}$ F_1 score: $\frac{2}{3}$

ParsEval

Gold Standard

(NP (DT *a*)
(ADVP (RB *pretty*)
(JJ *big*))
(NOM (NN *dog*)
(POS *'s*)
(NN *house*)))

0,6	NP	1,2	RB	3,4	NN
0,1	DT	2,3	JJ	4,5	POS
1,3	ADVP	3,6	NOM	5,6	NN

System Output

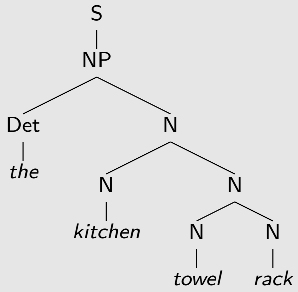
(NP (DT *a*)
(JJ *pretty*)
(NOM (JJ *big*)
(NOM (NN *dog*)
(POS *'s*)
(NN *house*))))

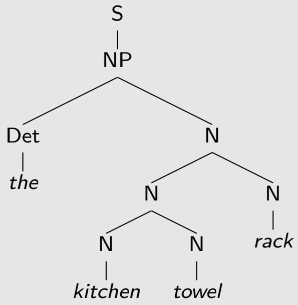
0,6	NP	2,6	NOM	3,4	NN
0,1	DT	2,3	JJ	4,5	POS
1,2	JJ	3,6	NOM	5,6	NN

Recall: $\frac{Correct}{Gold} = \frac{2}{3}$ Precision: $\frac{Correct}{System} = \frac{2}{3}$ F_1 score: $\frac{2}{3}$

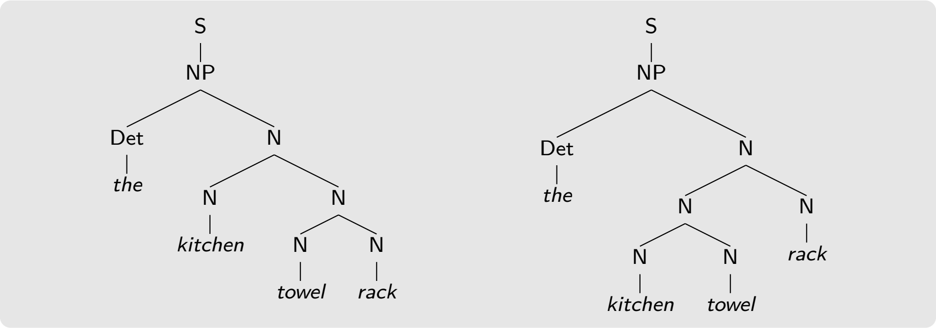
Crossing Brackets: 1

Exercise (3): Parser Evaluation





Exercise (3): Parser Evaluation



(3) What are the ParsEval precision and recall scores for this pair of trees (gold on the left; system on the right)?

From Constituent Structure to Dependency Grammar

- Phrase structure grammar works well for configurational languages.
- For example English: rigid word order; subject typically before verb.
- Grammatical functions (implicitly) defined as structural relations.

From Constituent Structure to Dependency Grammar

- Phrase structure grammar works well for configurational languages.
- For example English: rigid word order; subject typically before verb.
- Grammatical functions (implicitly) defined as structural relations.
- Alternatively, use these relations as primary, explict building blocks.
- Dependency syntax in terms of directed relations between words.

From Constituent Structure to Dependency Grammar

- Phrase structure grammar works well for configurational languages.
- For example English: rigid word order; subject typically before verb.
- Grammatical functions (implicitly) defined as structural relations.
- Alternatively, use these relations as primary, explict building blocks.
- Dependency syntax in terms of directed relations between words.
- More forgiving to word order variation, e.g. Slavic or German:

(weil) [die Frau]$_{\text{NOM}}$ *[dem Kind]*$_{\text{DAT}}$ *[ein Buch]*$_{\text{ACC}}$ *gab.*
(weil) [die Frau]$_{\text{NOM}}$ *[ein Buch]*$_{\text{ACC}}$ *[dem Kind]*$_{\text{DAT}}$ *gab.*
(weil) [dem Kind]$_{\text{DAT}}$ *[die Frau]*$_{\text{NOM}}$ *[ein Buch]*$_{\text{ACC}}$ *gab.*
(weil) [ein Buch]$_{\text{ACC}}$ *[dem Kind]*$_{\text{DAT}}$ *[die Frau]*$_{\text{NOM}}$ *gab.*

...

From Constituent Structure to Dependency Grammar

- Phrase structure grammar works well for configurational languages.
- For example English: rigid word order; subject typically before verb.
- Grammatical functions (implicitly) defined as structural relations.
- Alternatively, use these relations as primary, explict building blocks.
- Dependency syntax in terms of directed relations between words.
- More forgiving to word order variation, e.g. Slavic or German:

(weil) [die Frau]$_{\mathrm{NOM}}$ *[dem Kind]*$_{\mathrm{DAT}}$ *[ein Buch]*$_{\mathrm{ACC}}$ *gab.*
(weil) [die Frau]$_{\mathrm{NOM}}$ *[ein Buch]*$_{\mathrm{ACC}}$ *[dem Kind]*$_{\mathrm{DAT}}$ *gab.*
(weil) [dem Kind]$_{\mathrm{DAT}}$ *[die Frau]*$_{\mathrm{NOM}}$ *[ein Buch]*$_{\mathrm{ACC}}$ *gab.*
(weil) [ein Buch]$_{\mathrm{ACC}}$ *[dem Kind]*$_{\mathrm{DAT}}$ *[die Frau]*$_{\mathrm{NOM}}$ *gab.*
...

- Arguably dominant approach to syntactic structure in NLP today.

Recent Advances in Dependency Parsing

Tutorial, EACL, April 27th, 2014

Ryan McDonald[1] Joakim Nivre[2]

[1]Google Inc., USA/UK
E-mail: ryanmcd@google.com

[2]Uppsala University, Sweden
E-mail: joakim.nivre@lingfil.uu.se

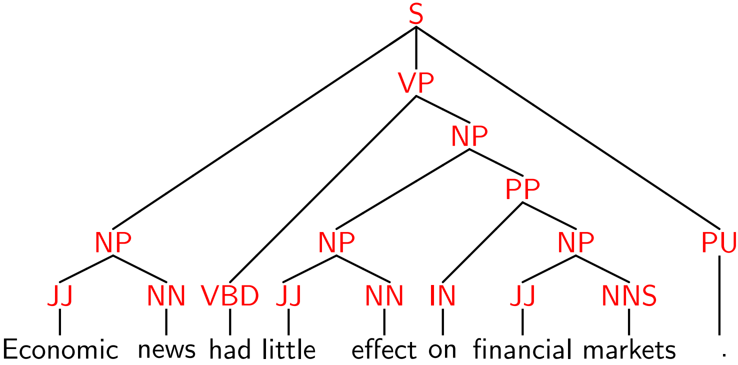
Dependency Syntax

- The basic idea:
  - Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies.
- In the words of Lucien Tesnière [Tesnière 1959]:
  - La phrase est un *ensemble organisé* dont les éléments constituants sont les *mots*. [1.2] Tout mot qui fait partie d'une phrase cesse par lui-même d'être isolé comme dans le dictionnaire. Entre lui et ses voisins, l'esprit aperçoit des *connexions*, dont l'ensemble forme la charpente de la phrase. [1.3] Les connexions structurales établissent entre les mots des rapports de *dépendance*. Chaque connexion unit en principe un terme *supérieur* à un terme *inférieur*. [2.1] Le terme supérieur reçoit le nom de *régissant*. Le terme inférieur reçoit le nom de *subordonné*. Ainsi dans la phrase *Alfred parle* [...], *parle* est le régissant et *Alfred* le subordonné. [2.2]

Dependency Syntax

- The basic idea:
  - Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies.
- In the words of Lucien Tesnière [Tesnière 1959]:
  - The sentence is an *organized whole*, the constituent elements of which are *words*. [1.2] Every word that belongs to a sentence ceases by itself to be isolated as in the dictionary. Between the word and its neighbors, the mind perceives *connections*, the totality of which forms the structure of the sentence. [1.3] The structural connections establish *dependency* relations between the words. Each connection in principle unites a *superior* term and an *inferior* term. [2.1] The superior term receives the name *governor*. The inferior term receives the name *subordinate*. Thus, in the sentence *Alfred parle* [...], *parle* is the governor and *Alfred* the subordinate. [2.2]

Phrase Structure



Dependency Structure

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

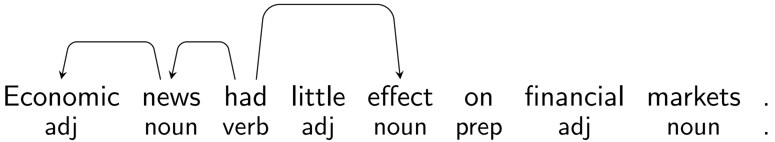
Dependency Structure

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

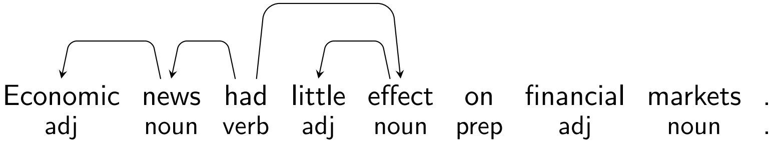
Dependency Structure

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

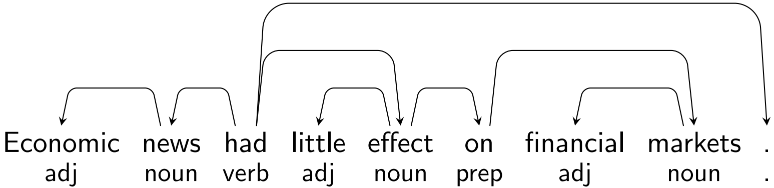
Dependency Structure



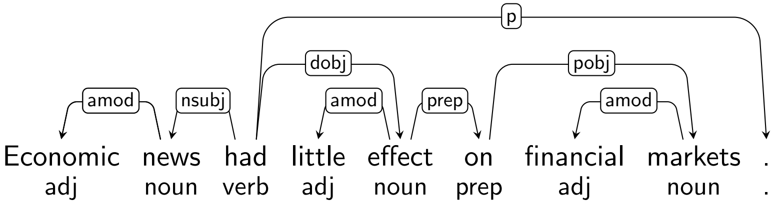
Dependency Structure



Dependency Structure



Dependency Structure



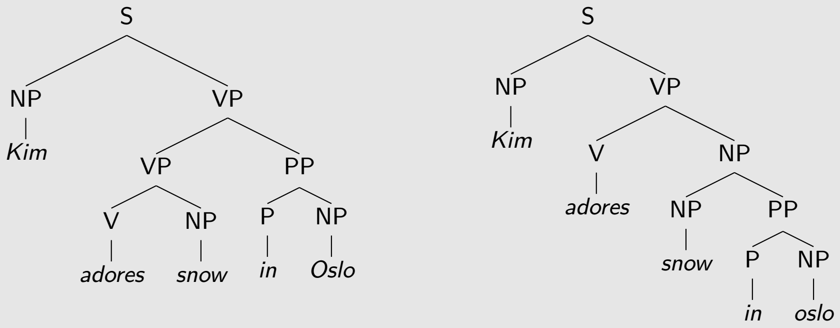
Terminology

Superior	**Inferior**
Head	Dependent
Governor	Modifier
Regent	Subordinate
⋮	⋮

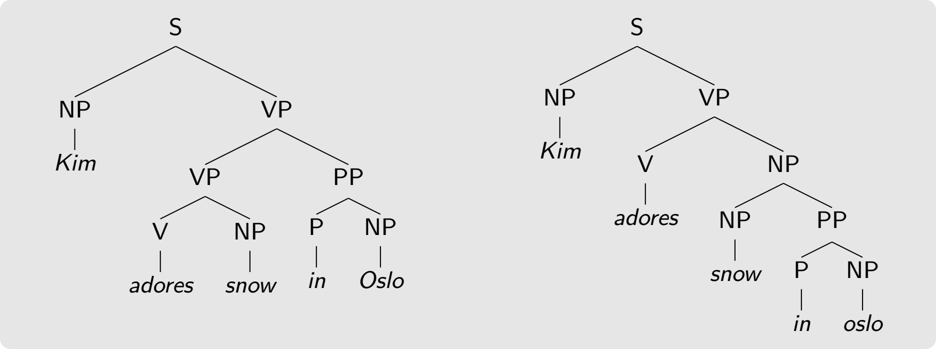
Comparison

- Dependency structures explicitly represent
  - head-dependent relations (directed arcs),
  - functional categories (arc labels),
  - possibly some structural categories (parts-of-speech).
- Phrase structures explicitly represent
  - phrases (nonterminal nodes),
  - structural categories (nonterminal labels),
  - possibly some functional categories (grammatical functions).
- Hybrid representations may combine all elements.

Exercise (4): Dependency Syntaxx



Exercise (4): Dependency Syntaxx



(4) Draw the dependency trees for the two readings.
Where does the attachment ambiguity manifest itself?

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.

Dependency Graph Notation

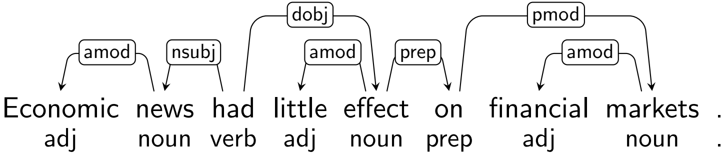
- For a dependency graph $G = (V, A)$
- With label set $L = \{l_1, \ldots, l_{|L|}\}$
  - $i \rightarrow j \ \equiv \ \exists k : (i, j, k) \in A$
  - $i \leftrightarrow j \ \equiv \ i \rightarrow j \lor j \rightarrow i$
  - $i \rightarrow^* j \ \equiv \ i = j \lor \exists i' : i \rightarrow i', \, i' \rightarrow^* j$
  - $i \leftrightarrow^* j \ \equiv \ i = j \lor \exists i' : i \leftrightarrow i', \, i' \leftrightarrow^* j$

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 $j < i' < i$.

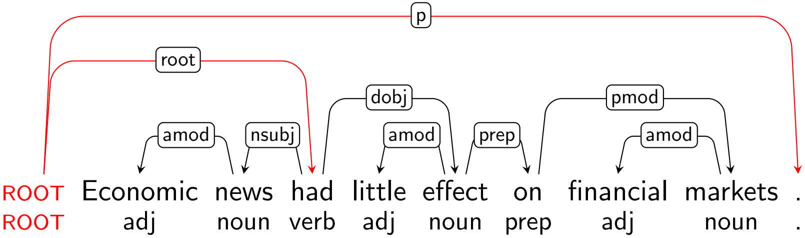
Connectedness, Acyclicity and Single-Head

- Intuitions:
  - Syntactic structure is complete (Connectedness).
  - Syntactic structure is hierarchical (Acyclicity).
  - Every word has at most one syntactic head (Single-Head).
- Connectedness can be enforced by adding a special root node.



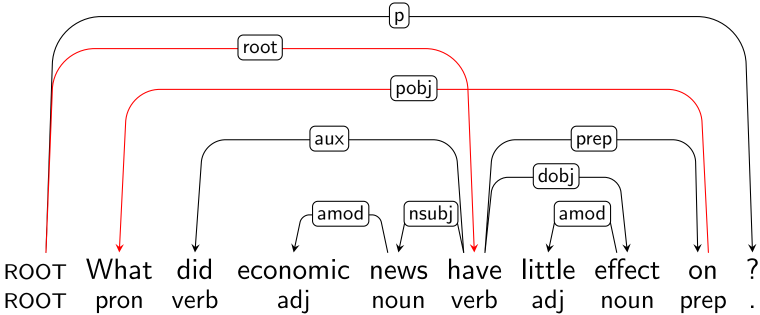
Connectedness, Acyclicity and Single-Head

- Intuitions:
  - Syntactic structure is complete (Connectedness).
  - Syntactic structure is hierarchical (Acyclicity).
  - Every word has at most one syntactic head (Single-Head).
- Connectedness can be enforced by adding a special root node.



Projectivity

- Most theoretical frameworks do not assume projectivity.
- Non-projective structures are needed to account for
  - long-distance dependencies,
  - free word order.



Dependency Treebanks

- Constituent tree with head information can be automatically converted.
- Dependency types based on structural relations (and parts of speech).

Dependency Treebanks

- Constituent tree with head information can be automatically converted.
- Dependency types based on structural relations (and parts of speech).
- Much dependency parsing using conversions of constituency treebanks.
- Including, of course, the venerable Penn Treebank (PTB) for English.
- Notable exceptions, e.g. PDT (Czech), NeGra and TiGer (German).

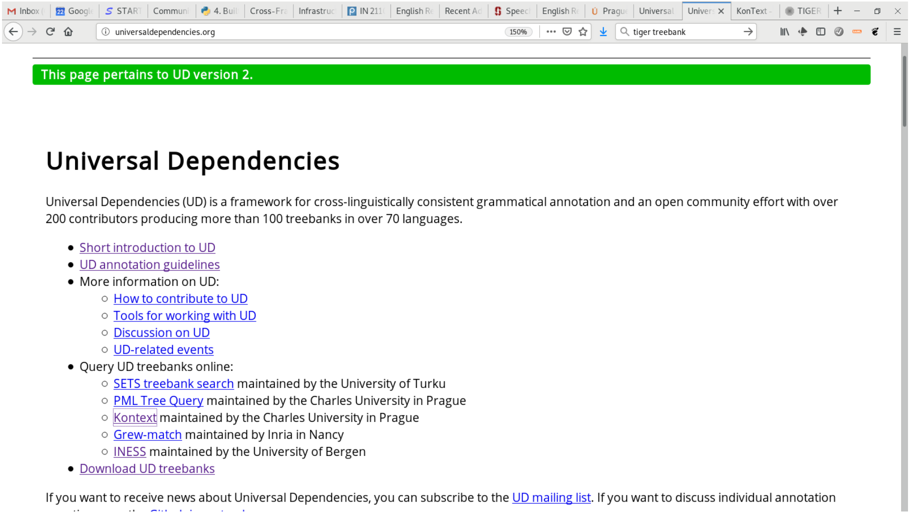
Dependency Treebanks

- Constituent tree with head information can be automatically converted.
- Dependency types based on structural relations (and parts of speech).
- Much dependency parsing using conversions of constituency treebanks.
- Including, of course, the venerable Penn Treebank (PTB) for English.
- Notable exceptions, e.g. PDT (Czech), NeGra and TiGer (German).
- Recently, greatly increased interest in dependency syntax ‘world-wide’.
- New annotation initiatives now more often ‘natively’ in dependencies.

Dependency Treebanks

- Constituent tree with head information can be automatically converted.
- Dependency types based on structural relations (and parts of speech).
- Much dependency parsing using conversions of constituency treebanks.
- Including, of course, the venerable Penn Treebank (PTB) for English.
- Notable exceptions, e.g. PDT (Czech), NeGra and TiGer (German).
- Recently, greatly increased interest in dependency syntax 'world-wide'.
- New annotation initiatives now more often 'natively' in dependencies.
- Many languages have their own linguistic traditions and terminology.
- Ongoing cross-linguistic harmonization: Universal Dependencies (UD).
- 'Mainstream': treebanks for 70^{+} languages (including NOB & NNO).

Universal Dependencies



This page pertains to UD version 2.

Universal Dependencies

Universal Dependencies (UD) is a framework for cross-linguistically consistent grammatical annotation and an open community effort with over 200 contributors producing more than 100 treebanks in over 70 languages.

- Short introduction to UD
- UD annotation guidelines
- More information on UD:
  - How to contribute to UD
  - Tools for working with UD
  - Discussion on UD
  - UD-related events
- Query UD treebanks online:
  - SETS treebank search maintained by the University of Turku
  - PML Tree Query maintained by the Charles University in Prague
  - Kontext maintained by the Charles University in Prague
  - Grew-match maintained by Inria in Nancy
  - INESS maintained by the University of Bergen
- Download UD treebanks

If you want to receive news about Universal Dependencies, you can subscribe to the UD mailing list. If you want to discuss individual annotation

Outlook

Next Week

- Statistical dependency parsing: The arc-eager transition system
- Supervised machine learning: The transition 'oracle'
- Feature functions for transition-based parsing
- Common evaluation metrics for dependency parsing
- Variations on syntactico-semantic dependency parsing