729G86/TDP030 Language Technology (VT2025)

Syntactic Analysis

Marcel Bollmann

Department of Computer and Information Science (IDA)

LINKÖPING UNIVERSITY

Partly adapted from slides by Marco Kuhlmann.

Today's lecture

1. Introduction
   - Dependency Structure
   - Information Extraction
2. Dependency Parsing
   - Annotations
   - Universal Dependencies
   - Projectivity
   - Evaluation
3. Transition-Based Dependency Parsing
   - Arc-Standard Model
   - Shift Transition
   - Left-Arc Transition
   - Right-Arc Transition
   - Terminal Condition
4. Outlook

What is Syntactic Analysis?

What is syntactic analysis?

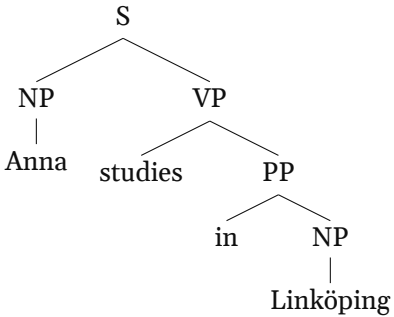
Definition

Syntactic analysis, or **syntactic parsing**, is the task of mapping a sentence to a formal representation of its syntactic structure.

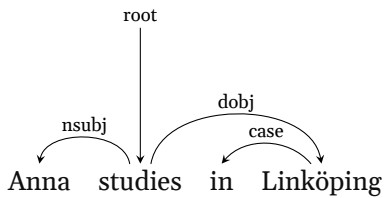
- **Syntactic structure** provides important clues about the meaning of a sentence.
  - can help with information extraction
  - can help resolve (structural) ambiguity

Different syntactic representations

Phrase structure tree

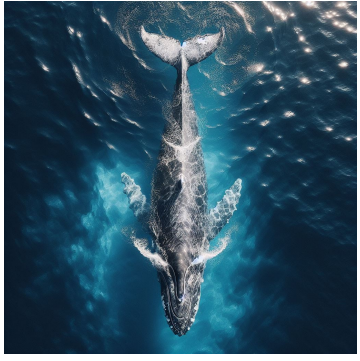


Dependency tree



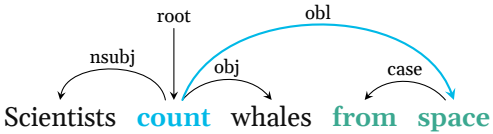
Syntactic ambiguity

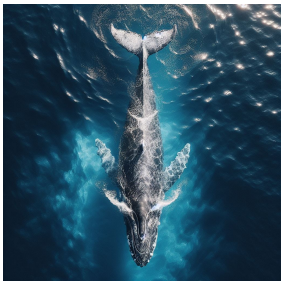
"Scientists count whales from space"

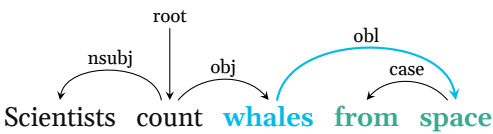


Images generated by Microsoft Copilot

Syntactic ambiguity

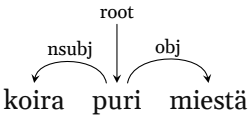


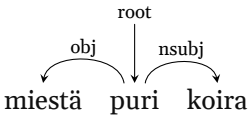


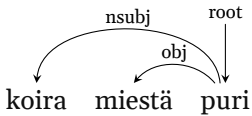


An example from Finnish

- In **languages with freer word order**, dependencies can reveal common structure:







"the dog bit the man"

- All three sentences contain the **same dependency relations**:

$$\emptyset \xrightarrow{\text{root}} \text{puri}, \quad \text{puri} \xrightarrow{\text{nsubj}} \text{koira}, \quad \text{puri} \xrightarrow{\text{obj}} \text{miestä}$$

Example taken from Wikipedia

Information extraction

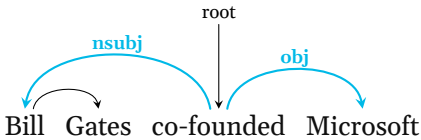
Definition

Information extraction (IE) is the task of extracting structured information from text.

- "Structured information" refers to **semantic relations** between **named entities**.
  - *e.g.* X is-leader-of Y, X bought Y, X was-born-in Y
- We already know how to find the named entities; syntactic structure can help us find the relations between them.

From syntactic structure to semantic relations

- **Syntactic structure** in form of a dependency tree:

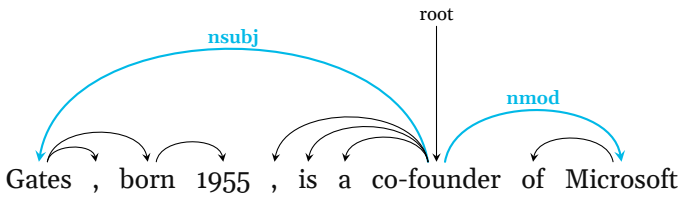


- **Semantic relation** in the knowledge graph DBpedia:

```
dbr:Microsoft  dbo:foundedBy  dbr:Bill_Gates
```

From syntactic structure to semantic relations

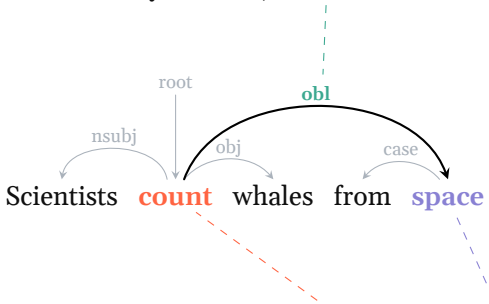
- Parsing the syntactic structure allows us to extract these relations even from **more complex sentences**:



Dependency Parsing

Dependency structure

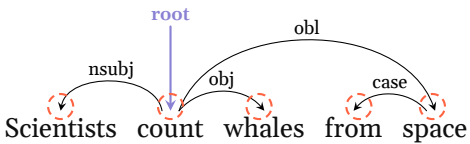
- **Syntactic dependencies** are asymmetric, **labelled** relations.



- Dependency relations are defined between a **head** and a **dependent**.

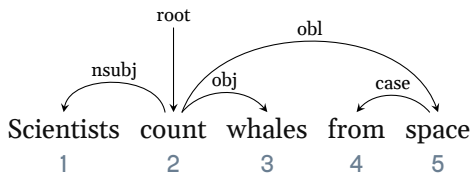
Dependency trees

- A **dependency tree** is a directed graph with the following properties:
  1. Every node has exactly **one incoming edge**.
  2. Every node is **reachable from the root** node.



Representing dependency relations

- We can view dependency trees as **annotating each word** with its **head** and **relation**.



Dependent	1	2	3	4	5
Head	2	0	2	5	2
Relation	nsubj	root	obj	case	obl

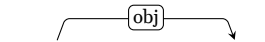
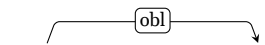
Dependency parsing

Definition

Dependency parsing is the task of annotating a sentence with its dependency structure.

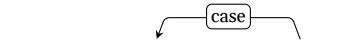
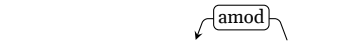
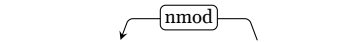
- Like with POS tagging, there are **different formalisms** for dependency parsing.
- We will look at the **Universal Dependencies (UD)** formalism.
  - same project that gave us the "universal part-of-speech tagset"
  - There is data in **over 100 languages** annotated within UD!

Universal dependencies

Label	Category	Example
nsubj	*nominal subject*	
obj	*object*	
iobj	*indirect object*	
obl	*oblique nominal*	

Source and more details: Universal dependencies

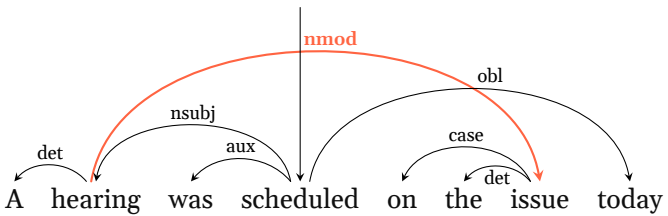
Universal dependencies

Label	Category	Example
det	*determiner*	
case	*case marking*	
amod	*adjectival modifier*	
nmod	*nominal modifier*	

Source and more details: Universal dependencies

Projectivity

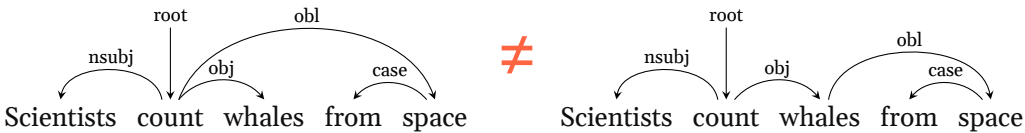
- A tree is **projective** *if and only if* any subtree is a **contiguous span**.
  - equivalent: there are no crossing arcs
- This tree is **non-projective** because it has **crossing arcs**:



How to **evaluate** a dependency parser?

Exact match

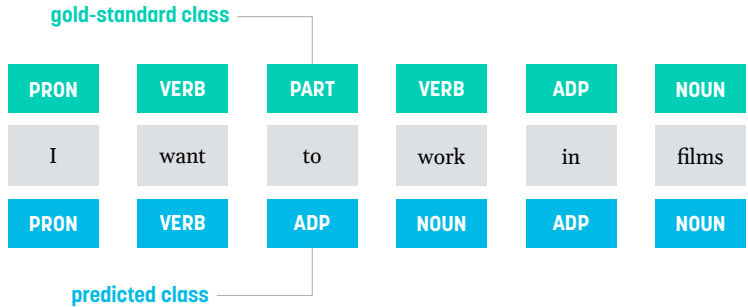
- The **exact match (EM)** metric counts how many sentences were parsed correctly.



- We can distinguish between **labelled** and **unlabelled** exact match.
  - "unlabelled": ignore the semantic relation labels

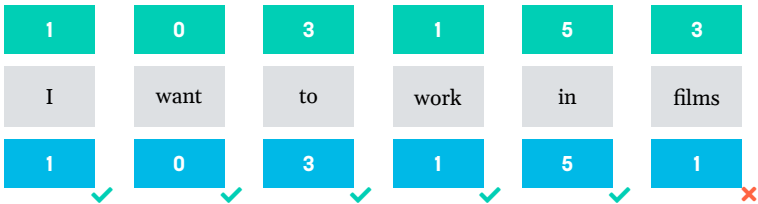
Reminder: Evaluation of part-of-speech taggers

- With part-of-speech tagging, we commonly evaluate with **accuracy**.



Unlabelled attachment score

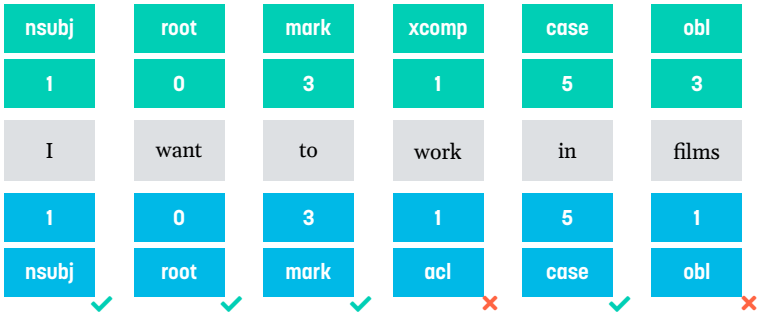
- The **unlabelled attachment score (UAS)** is the accuracy on the **heads**.
  - we're ignoring the semantic relation labels



$$\text{UAS} = \frac{5}{6} \approx 83.33\%$$

Labelled attachment score

- The **labelled attachment score (LAS)** is the accuracy on **heads + relations**.



$$\text{LAS} = \frac{4}{6} \approx 66.66\%$$

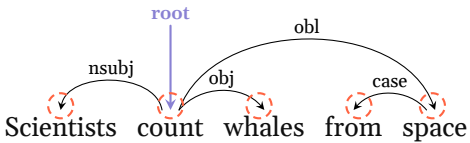
Important concepts

- dependency parsing, dependency trees
- head & dependent
- projectivity
- labelled & unlabelled attachment score (LAS & UAS)

Transition-Based Dependency Parsing

Reminder: Dependency trees

- A dependency tree is a directed graph with the following properties:
  1. Every node has exactly **one incoming edge**.
  2. Every node is **reachable from the root** node.

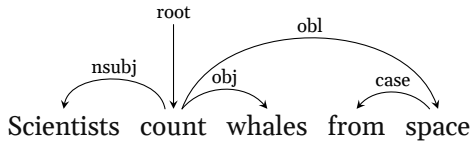


- If we simply predicted the head for each token independently, we might end up with an **invalid dependency tree**.

  → we need a specialized parsing algorithm!

Transition-based parsing with arc-standard

- **Transition-based parsers** predict a **sequence of transitions** to build up a tree.



- **Arc standard** is one model that defines a set of transitions.
  - In arc standard, the tree above could be built with the following transitions:

    SH, SH, LA, SH, RA, SH, SH, LA, RA, RA

- Important limitation: can only build **projective** trees!

Parser configuration

Stack	Buffer
ROOT	Scientists count whales from space

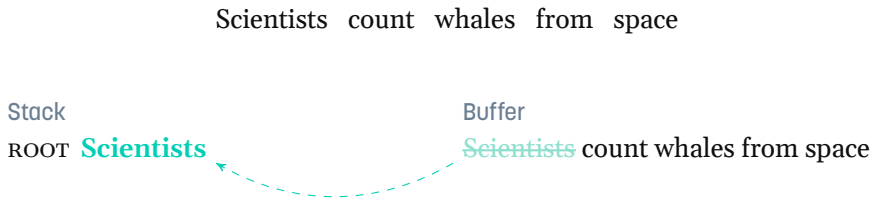
1. The **stack** contains words that are currently being processed.
   - Initially, it only contains the special ROOT symbol.
2. The **buffer** contains words that still need to be processed.
   - Initially, it contains the entire input sentence.
3. The **list of arcs** contains the arcs of the dependency tree.
   - Initially, it is empty.

Arc standard: transitions

Stack	Buffer
ROOT	Scientists count whales from space

- An arc-standard parser now predicts **one of three actions**:
  1. **Shift (SH)** moves the first item from the buffer to the stack.
  2. **Left-arc (LA)** creates a new arc from the topmost to the second-topmost item on the stack, and removes the target from the stack.
  3. **Right-arc (RA)** creates a new arc from the second-topmost to the topmost item on the stack, and removes the target from the stack.

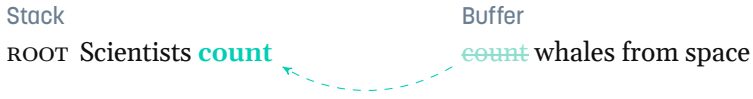
Arc standard: shift transition



- The **shift (SH)** transition moves the first item from the buffer to the top of the stack.

Arc standard: shift transition

Scientists count whales from space



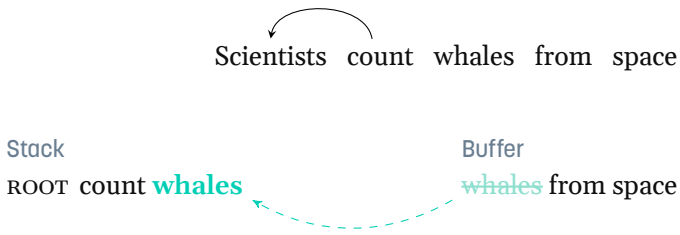
- If the parser predicts another **shift (SH)** transition, we do the same thing again.

Arc standard: left-arc transition

Stack	Buffer
ROOT ~~Scientists~~ **count**	whales from space

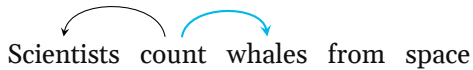
- **Left-arc (LA)** creates a **new arc** from the **topmost** to the **second-topmost** item on the stack.
- Afterwards, we remove the target *(= the second-topmost item)* from the stack.

Arc standard: left-arc transition



- For this example sentence, we'll do another **shift (SH)** transition now.

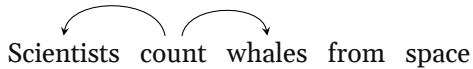
Arc standard: right-arc transition



Stack	Buffer
ROOT **count** ~~whales~~	from space

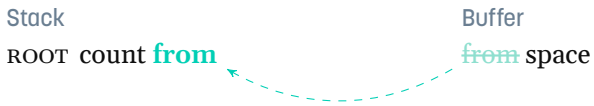
- **Right-arc (RA)** creates a **new arc** from the **second-topmost** to the **topmost** item on the stack.
- Afterwards, we remove the target *(= the topmost item)* from the stack.

Arc standard: right-arc transition



Stack	Buffer
ROOT count	from space

Arc standard: continuing the example

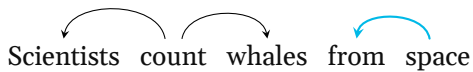


- In the next step, we predict **shift (SH)**.

Arc standard: continuing the example

- In the next step, we predict **shift (SH)** again.

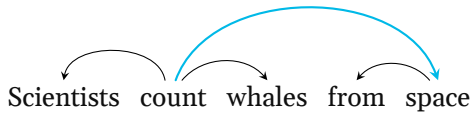
Arc standard: continuing the example



Stack	Buffer
ROOT count ~~from~~ space	

- In the next step, we predict **left-arc (LA)**.

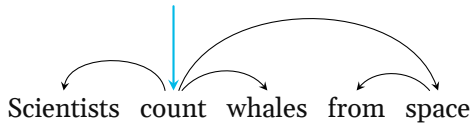
Arc standard: continuing the example



Stack	Buffer
ROOT **count** ~~space~~	

- In the next step, we predict **right-arc (RA)**.

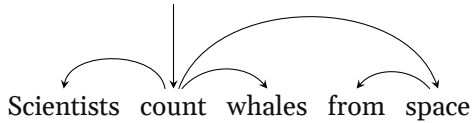
Arc standard: continuing the example



Stack	Buffer
ROOT ~~count~~	

- With only one word left on the stack, we must predict **right-arc (RA)**.

Arc standard: terminating the algorithm



Stack	Buffer
ROOT	

- We cannot perform any more transitions, so **we're done**!
  - terminal condition: no words left on the stack or buffer, except ROOT

Important concepts

- transition-based parsing, stack vs. buffer
- arc-standard model
- shift, left-arc, right-arc transitions

Outlook

Training a transition-based dependency parser

- Our **training data** looks like this:

Dependent	1	2	3	4	5
Head	2	0	2	5	2

- But a transition-based parser should now **predict** this:

SH, SH, LA, SH, RA, SH, SH, LA, RA, RA

How can we **train our parser** then?

Oracles

- In general, many **different transition sequences** can lead to the **same tree**!

SH, SH, LA, SH, RA, SH, SH, LA, RA, RA

SH, SH, SH, RA, SH, SH, LA, RA, LA, RA

SH, SH, SH, RA, LA, SH, SH, LA, RA, RA

Definition

An **oracle** is a function that compares a transition sequence against a gold-standard tree, in order to provide feedback during model training.


LINKÖPING
UNIVERSITY