DD2417 Language Engineering
2a: Syntax

Johan Boye, KTH

Formalizing syntax

We want an (automatic) procedure separating sequences of words that belong to the language from those who don't.

We'd like to identify *who* does *what when/how/to whom* .

- *who*: the subject
- *does*: the predicate
- *what/when/how/to whom* : the object, or adverbial, or complement

Information extraction

Among Republicans, 78% say that Biden did not win and 54% believe there is solid evidence of that, despite the fact that no such evidence exists.

CNN Politics, 2021-09-15.

Who?	78% of Republicans
Did?	say
What?	Biden did not win
When?	-
How?	-

Information extraction

Among Republicans, 78% say that Biden did not win and 54% believe there is solid evidence of that, despite the fact that no such evidence exists.

CNN Politics, 2021-09-15.

Who?	54% of Republicans
Did?	believe
What?	there is solid evidence that Biden did not win the election
When?	-
How?	despite the fact that no such evidence exists

Formalizing syntax

We'll have a look at two different formalisms:

- Context-free grammars (phrase-structure grammars), based on the notion of constituent or phrase
  - noun phrase, verb phrase, ...
- Dependency structures, based on the notion of grammatical functions
  - subject, predicate, complement, ...

Word classes and interchangeability

Words of the same class can often be substituted for each other (but there are also many exceptions).

- *There was a fly on the wall.*
- *There was a fear on the wall.*
- Syntactically correct although the meaning is unclear.

But also a group of words can have the same role.

Constituents

A constituent is a group of words acting as a unit.

E.g. a noun phrase:

- Ideas entered his brain.
- An idea entered his brain.
- A brilliant idea entered his brain.
- A brilliant idea about prime numbers entered his brain.
- A brilliant idea about prime numbers that had struck him before several years earlier when he was working as a visiting professor in Paris entered his brain.
- etc.

Head words and modifiers

A phrase has a head and modifiers, e.g.:

A brilliant $\boxed{\textit{idea}}$ *about prime numbers*

- idea is the head
- Modifiers:
  - a
  - brilliant
  - about prime numbers (a prepositional phrase)

Context-free grammars (CFG)

Noam Chomsky introduced context-free grammars in 1956.

Recall that a CFG consists of

- A set of terminal symbols (words, in our case)
- A set of non-terminal symbols (constituents, in our case)
- A set of rewrite rules on the form

$$A \to \alpha$$

where α is a string of terminals and non-terminals.

Noun phrase

A noun phrase is a constituent where the head word is a noun, a pronoun, or a proper name.

$NP \rightarrow N$	Ideas
$NP \rightarrow P$	He
$NP \rightarrow PM$	John
$NP \rightarrow DET\ N$	An idea
$NP \rightarrow DET\ Adj\ N$	A great idea
$NP \rightarrow NP\ CConj\ NP$	A great idea and a bad idea
$NP \rightarrow DET\ Adj\ N\ PP$	A great idea about prime numbers
etc.	

where *PP* is a prepositional phrase.

Nominals

English (but not Swedish!) allows sequences of nouns. These are called nominals.

numbers
prime numbers
prime number theorem
prime number limit theorem
etc.

$Nominal \rightarrow N$
$Nominal \rightarrow Nominal\ N$
$NP \rightarrow Nominal$

Prepositional phrases

A prepositional phrase has a preposition as a head word.

$PP \rightarrow P\ NP$ about prime numbers

Verb phrases

$VP \rightarrow V$	run
$VP \rightarrow V\ NP$	run a business
$VP \rightarrow V\ NP\ PP$	run a business in Sweden
$VP \rightarrow V\ PP$	run for president

etc.

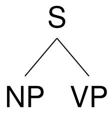
DD2417 Language Engineering
2b: Syntax trees

Johan Boye, KTH

Syntax trees

The phrase structure can be depicted in a syntax tree (or parse tree).

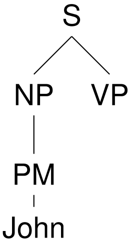
$S \rightarrow NP\ VP$
$NP \rightarrow PM \mid Det\ N \mid Det\ N\ PP$
$VP \rightarrow V\ NP \mid V\ NP\ PP$
$PP \rightarrow P\ NP$
$N \rightarrow pie \mid fridge$
$V \rightarrow made$
$P \rightarrow in$
$Det \rightarrow the$
$PM \rightarrow John$



Syntax trees

The phrase structure can be depicted in a syntax tree (or parse tree).

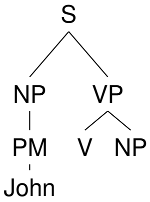
$$
\begin{aligned}
&S \rightarrow NP\ VP \\
&NP \rightarrow PM \mid Det\ N \mid Det\ N\ PP \\
&VP \rightarrow V\ NP \mid V\ NP\ PP \\
&PP \rightarrow P\ NP \\
&N \rightarrow pie \mid fridge \\
&V \rightarrow made \\
&P \rightarrow in \\
&Det \rightarrow the \\
&PM \rightarrow John
\end{aligned}
$$



Syntax trees

The phrase structure can be depicted in a syntax tree (or parse tree).

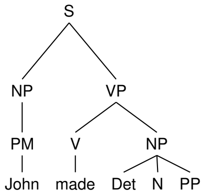
$S \rightarrow NP\ VP$
$NP \rightarrow PM \mid Det\ N \mid Det\ N\ PP$
$VP \rightarrow V\ NP \mid V\ NP\ PP$
$PP \rightarrow P\ NP$
$N \rightarrow pie \mid fridge$
$V \rightarrow made$
$P \rightarrow in$
$Det \rightarrow the$
$PM \rightarrow John$



Syntax trees

The phrase structure can be depicted in a syntax tree (or parse tree).

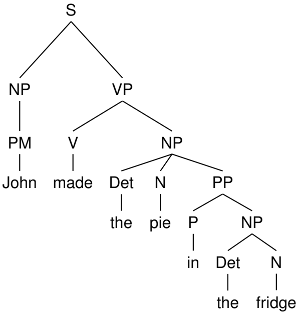
$S \rightarrow NP\ VP$
$NP \rightarrow PM \mid Det\ N \mid Det\ N\ PP$
$VP \rightarrow V\ NP \mid V\ NP\ PP$
$PP \rightarrow P\ NP$
$N \rightarrow pie \mid fridge$
$V \rightarrow made$
$P \rightarrow in$
$Det \rightarrow the$
$PM \rightarrow John$



Syntax trees

The phrase structure can be depicted in a syntax tree (or parse tree).

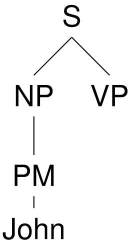
$$
\begin{aligned}
&S \rightarrow NP\ VP\\
&NP \rightarrow PM \mid Det\ N \mid Det\ N\ PP\\
&VP \rightarrow V\ NP \mid V\ NP\ PP\\
&PP \rightarrow P\ NP\\
&N \rightarrow pie \mid fridge\\
&V \rightarrow made\\
&P \rightarrow in\\
&Det \rightarrow the\\
&PM \rightarrow John
\end{aligned}
$$



Syntax trees

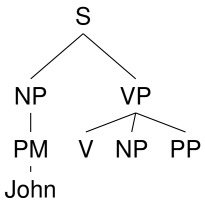
Suppose, at this point, we choose another rule.

$$
\begin{aligned}
&S \rightarrow NP\ VP \\
&NP \rightarrow PM \mid Det\ N \mid Det\ N\ PP \\
&VP \rightarrow V\ NP \mid V\ NP\ PP \\
&PP \rightarrow P\ NP \\
&N \rightarrow pie \mid fridge \\
&V \rightarrow made \\
&P \rightarrow in \\
&Det \rightarrow the \\
&PM \rightarrow John
\end{aligned}
$$



Syntax trees

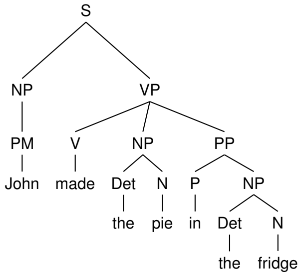
$S \rightarrow NP\ VP$
$NP \rightarrow PM \mid Det\ N \mid Det\ N\ PP$
$VP \rightarrow V\ NP \mid V\ NP\ PP$
$PP \rightarrow P\ NP$
$N \rightarrow pie \mid fridge$
$V \rightarrow made$
$P \rightarrow in$
$Det \rightarrow the$
$PM \rightarrow John$



Syntax trees

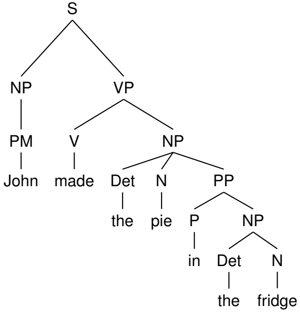
We could derive the same sentence, but in another way.

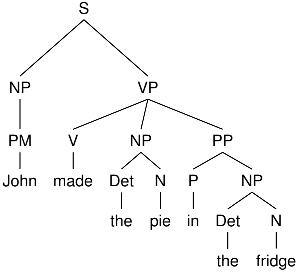
$S \rightarrow NP\ VP$
$NP \rightarrow PM \mid Det\ N \mid Det\ N\ PP$
$VP \rightarrow V\ NP \mid V\ NP\ PP$
$PP \rightarrow P\ NP$
$N \rightarrow pie \mid fridge$
$V \rightarrow made$
$P \rightarrow in$
$Det \rightarrow the$
$PM \rightarrow John$



Ambiguity

This sentence has two different syntax trees, so it is ambiguous.





Exercise

Write syntax trees for the sentences

John bought a watch.
John bought a watch in cash.
John bought a watch in pure gold.
John bought a watch in pure gold in cash.

Invent the necessary grammar rules as you go along.

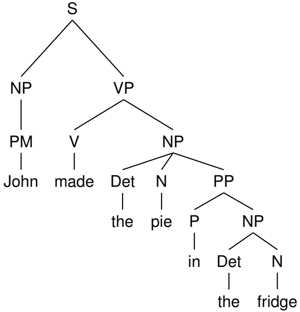
Use grammatically motivated constituents like NP, VP, etc.

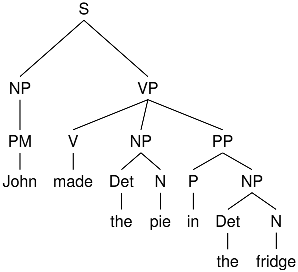
DD2417 Language Engineering
2c: Ambiguity

Johan Boye, KTH

Ambiguity

This sentence has two different syntax trees, so it is ambiguous.





Prepositional attachment ambiguity

3 prepositional phrases give 5 analyses

- Put the block [[in the box on the table] in the kitchen]
- Put the block [[in the box] on the table in the kitchen]
- Put [[the block in the box] on the table] in the kitchen
- Put [the block [in the box on the table]] in the kitchen
- Put [the block in the box] [on the table in the kitchen]

Number of analyses grows as the Catalan numbers: 1, 2, 5, 14, 42, 132,...

Coordination ambiguity

THE NEWS-GAZETTE

NATION / WO

PRESIDENT'S FIRST PHYSICAL

Doctor: No heart, cognitive issues

But Trump needs to reduce his cholesterol, lose weight

By JILL COLVIN

6-foot-3 president weighed in at 239 pounds — three pounds heavier than he was in September 2016, the last time Trump revealed his weight to the public.

Trump's blood pressure was 122 over 74, and his

with no medical issues." Trump has no heart disease and no family history of it.

The 71-year-old president performed "exceedingly well" on cognitive screening, which is not

The News Gazette, 2018

Modifier ambiguity

numbers, including some that featured a bucket and bells brigade
buckets and trash cans with drums sticks and hammer mallets. PHOTO BY JENNIFER STULTZ

MENTORING DAY

Students get first hand job experience

By Gale Rose
grose@pratttribune.com

Eager students invaded businesses all over Pratt Tuesday, October 24 as they looked for future job opportunities on Disability Mentoring Day.

The 97 students from 12 schools fanned out across Pratt and got first hand experience what it would be like to work at those 40 businesses. They asked questions and got some hands on experience with various operations.

Paola Luna of Pratt High School, Gina Patton of Kingman High School and America Fernandez of St. John chose the Main Street Small Animal Veterinarian Clinic for their business. Students got a tour of the facility, learned what happens in an examination, got to handle various animals and watched a snake eat a mouse.

Luna said she was interested in animal health and wanted to know more about caring for hurt animals. Patton likes all kinds of animals and said she learned a lot from the experience. Watching the snake eat the mouse impressed her the most.

Fernandez wants to become a veterinarian and enjoyed learning everything that veterinarians

SEE MENTORING, 6

Pratt Tribune, Oct 28, 2017

Ambiguity

Time flies like an arrow.

Ambiguity

Time	flies	like	an	arrow
Noun	Verb	Prep	Det	Noun

Ambiguity

Time	flies	like	an	arrow
Noun	Verb	Prep	Det	Noun
Noun	Noun	Verb	Det	Noun

Ambiguity

Time	flies	like	an	arrow
Noun	Verb	Prep	Det	Noun
Noun	Noun	Verb	Det	Noun
Verb	Noun	Prep	Det	Noun

Ambiguity

Time	flies	like	an	arrow
Noun	Verb	Prep	Det	Noun
Noun	Noun	Verb	Det	Noun
Verb	Noun	Prep	Det	Noun
Verb	Noun	Conj	Det	Noun

DD2417 Language Engineering
2d: Parsing

Johan Boye, KTH

Parsing strategies

Top-down

- Start at the root node, expand tree by matching left-hand side of rules.
- Derive a tree whose leaves match the input.
- We might use rules that could never match the input.
- Watch out for loops $VP \rightarrow VP\ PP$

Bottom-up

- Start at the leaves, build tree by matching right-hand side of rules.
- Derive a tree whose root is S.
- Builds structures that will never be used in the tree.

Parsing and dynamic programming

The number of possible trees grows exponentially with sentence length.

A naive backtracking approach is too inefficient.

But alternative trees share subtrees, so we can use dynamic programming.

- No work is repeated.
- Gives polynomial algorithm.
- Example: CKY (Cocke-Kasami-Younger)

Chomsky Normal Form

The CKY algorithm requires the grammar to be in Chomsky Normal Form (CNF).

All rules have the form $A \rightarrow B\,C$ or $A \rightarrow \textit{word}$.

If the grammar is not in CNF, it has to be rewritten.

Chomsky Normal Form

Translation into CNF:

- $A \rightarrow B\,C\,D$ is replaced by $\begin{array}{lcl} A & \rightarrow & B\,X \\ X & \rightarrow & C\,D \end{array}$

  where X is a new symbol.

- $A \rightarrow B$ is replaced by $\begin{array}{lcl} A & \rightarrow & word_1 \\ \ldots & & \\ A & \rightarrow & word_n \end{array}$

  where $word_1 \ldots word_n$ are all the words derivable from B.

Chomsky Normal Form

Quiz: Translate into CNF:

$$\begin{aligned}
&S \rightarrow NP\ VP\\
&NP \rightarrow NN \mid Det\ NN \mid Det\ NN\ PP\\
&VP \rightarrow V \mid V\ PP\\
&PP \rightarrow P\ NP\\
&NN \rightarrow girl \mid rain \mid coat\\
&V \rightarrow runs\\
&P \rightarrow in\\
&Det \rightarrow the
\end{aligned}$$

$A \rightarrow B\,C\,D \Rightarrow$

$$\begin{aligned}
A &\rightarrow B\,X\\
X &\rightarrow C\,D
\end{aligned}$$

where X is a new symbol.

$A \rightarrow B \Rightarrow$

$$\begin{aligned}
A &\rightarrow word_1\\
&\ldots\\
A &\rightarrow word_n
\end{aligned}$$

where $word_1 \ldots word_n$ are all the words derivable from B.

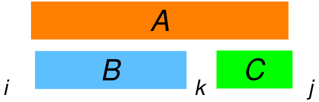
Chomsky Normal Form

$S \rightarrow NP\ VP$
$NP \rightarrow Det\ N \mid Det\ X$
$X \rightarrow N\ PP$
$VP \rightarrow V\ PP$
$PP \rightarrow P\ NP$
$N \rightarrow girl \mid rain \mid coat$
$NP \rightarrow girl \mid rain \mid coat$
$V \rightarrow runs$
$VP \rightarrow runs$
$P \rightarrow in$
$Det \rightarrow the$

CKY algorithm

$A \rightarrow B\ C$

- If there is an A somewhere in the input, then there has to be a B followed by a C



- If A extends from i to j, there must be a $k, i < k < j$, such that B extends from i to k, and C extends from k to j
- $_0$ *the* $_1$ *girl* $_2$ *in* $_3$ *the* $_4$ *coat* $_5$ *runs* $_6$
- *PP* covers [2,5], $PP \rightarrow P\ NP$
- *P* covers [2,3], *NP* covers [3,5]

CKY algorithm

Partial parses are represented in (half a) $N \times N$ table

- (N = length of input string)
- cell $[i, j]$ contains A if A covers i to j in the input string

i \ *j*	1	2	3	4	5
0	0-1	0-2	0-3	0-4	0-5
1		1-2	1-3	1-4	1-5
2			2-3	2-4	2-5
3				3-4	3-5
4					4-5

CKY algorithm

For A to cover $[i, j]$:

- $A \rightarrow B\,C$ is a rule in the grammar
- there is a B in $[i, k]$, and a C in $[k, j]$, for some $i < k < j$
- to apply $A \rightarrow B\,C$, look for a B in $[i, k]$, and a C in $[k, j]$

i \ j	1	2	3	4	5
0	0-1	0-2	0-3	0-4	0-5
1		1-2	1-3	1-4	1-5
2			2-3	2-4	2-5
3				3-4	3-5
4					4-5

CKY example

Example: Parse the sentence

$$_0\ \textit{giant}\ _1\ \textit{cuts}\ _2\ \textit{in}\ _3\ \textit{welfare}$$

given the grammar

S	$\rightarrow$	$NP\ VP$	N	$\rightarrow$	*giant*
NP	$\rightarrow$	$JJ\ NP$	N	$\rightarrow$	*cuts*
NP	$\rightarrow$	$NP\ PP$	N	$\rightarrow$	*welfare*
NP	$\rightarrow$	N	V	$\rightarrow$	*cuts*
PP	$\rightarrow$	$P\ NP$	JJ	$\rightarrow$	*giant*
VP	$\rightarrow$	$V\ PP$	P	$\rightarrow$	*in*

CKY example

First translate into Chomsky Normal Form:

- remove $NP \rightarrow N$
- add $NP \rightarrow giant \mid cuts \mid welfare$

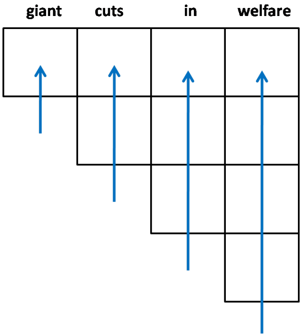
$S \rightarrow NP\ VP$	$N \rightarrow giant$
$NP \rightarrow JJ\ NP$	$N \rightarrow cuts$
$NP \rightarrow NP\ PP$	$N \rightarrow welfare$
$NP \rightarrow giant$	$V \rightarrow cuts$
$NP \rightarrow cuts$	$JJ \rightarrow giant$
$NP \rightarrow welfare$	$P \rightarrow in$
$PP \rightarrow P\ NP$	$VP \rightarrow V\ PP$

CKY example

Now fill the table left to right, bottom up.

When a cell is considered, everything needed to fill that cell is already in the table.

Nice property: Analyses input from left to right, in one sweep.

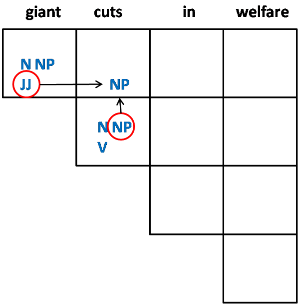


CKY example

giant	cuts	in	welfare
N NP JJ			
	N NP V		

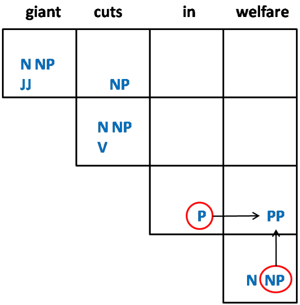
S	$\rightarrow$	$NP\ VP$	N	$\rightarrow$	*giant*
NP	$\rightarrow$	$JJ\ NP$	N	$\rightarrow$	*cuts*
NP	$\rightarrow$	$NP\ PP$	N	$\rightarrow$	*welfare*
NP	$\rightarrow$	*giant*	V	$\rightarrow$	*cuts*
NP	$\rightarrow$	*cuts*	JJ	$\rightarrow$	*giant*
NP	$\rightarrow$	*welfare*	P	$\rightarrow$	*in*
PP	$\rightarrow$	$P\ NP$	VP	$\rightarrow$	$V\ PP$

CKY example



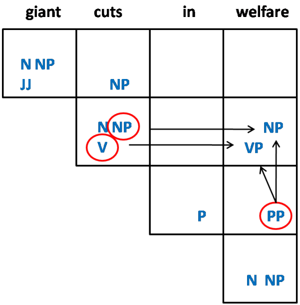
S	$\rightarrow$	$NP\ VP$	N	$\rightarrow$	*giant*
NP	$\rightarrow$	$JJ\ NP$	N	$\rightarrow$	*cuts*
NP	$\rightarrow$	$NP\ PP$	N	$\rightarrow$	*welfare*
NP	$\rightarrow$	*giant*	V	$\rightarrow$	*cuts*
NP	$\rightarrow$	*cuts*	JJ	$\rightarrow$	*giant*
NP	$\rightarrow$	*welfare*	P	$\rightarrow$	*in*
PP	$\rightarrow$	$P\ NP$	VP	$\rightarrow$	$V\ PP$

CKY example



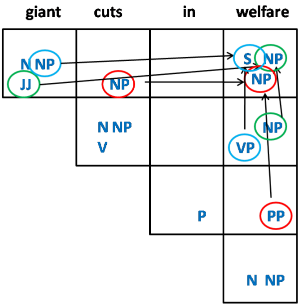
$S \rightarrow NP\ VP$	$N \rightarrow giant$
$NP \rightarrow JJ\ NP$	$N \rightarrow cuts$
$NP \rightarrow NP\ PP$	$N \rightarrow welfare$
$NP \rightarrow giant$	$V \rightarrow cuts$
$NP \rightarrow cuts$	$JJ \rightarrow giant$
$NP \rightarrow welfare$	$P \rightarrow in$
$PP \rightarrow P\ NP$	$VP \rightarrow V\ PP$

CKY example



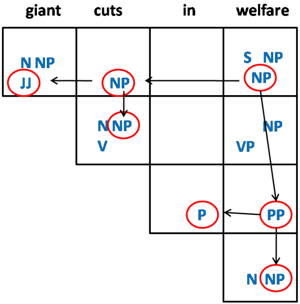
$S \rightarrow NP\ VP$	$N \rightarrow giant$
$NP \rightarrow JJ\ NP$	$N \rightarrow cuts$
$NP \rightarrow NP\ PP$	$N \rightarrow welfare$
$NP \rightarrow giant$	$V \rightarrow cuts$
$NP \rightarrow cuts$	$JJ \rightarrow giant$
$NP \rightarrow welfare$	$P \rightarrow in$
$PP \rightarrow P\ NP$	$VP \rightarrow V\ PP$

CKY example



$S \rightarrow NP\ VP$	$N \rightarrow giant$
$NP \rightarrow JJ\ NP$	$N \rightarrow cuts$
$NP \rightarrow NP\ PP$	$N \rightarrow welfare$
$NP \rightarrow giant$	$V \rightarrow cuts$
$NP \rightarrow cuts$	$JJ \rightarrow giant$
$NP \rightarrow welfare$	$P \rightarrow in$
$PP \rightarrow P\ NP$	$VP \rightarrow V\ PP$

Extracting trees from the CKY table



Summary so far

- Syntax can be described by means of context-free grammars
- Such grammars are generative
- Syntax trees describe the structure of sentences
- Ambiguous sentences have more than one tree
- By parsing (e.g. using the CKY algorithm), we can construct all trees for a given sentence (and grammar)

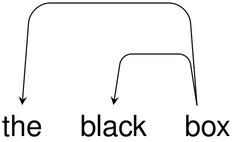
DD2417 Language Engineering
2e: Dependency syntax

Johan Boye, KTH

Dependency syntax

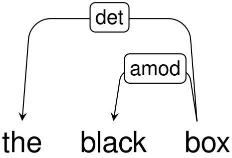
Dependency syntax provides an alternative view

- Binary relations (dependencies) between words
- A head (governor) word is related to its modifiers (dependents)
- The dependencies form a tree.



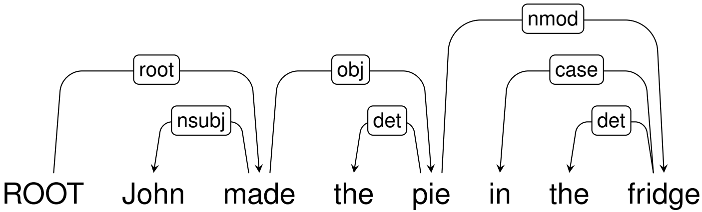
Dependency syntax

Usually the arcs are labeled with grammatical functions.

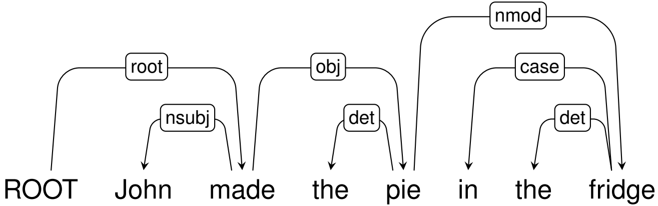


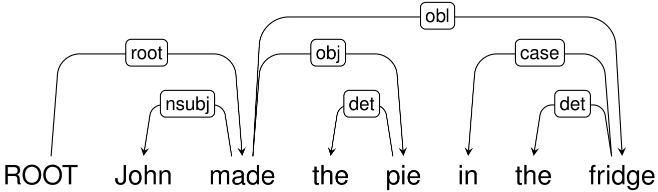
Dependency syntax

- The root of the tree of a whole sentence is almost always the main verb.
- Often a ROOT node is added (so every word is the child of some node).



Dependency syntax

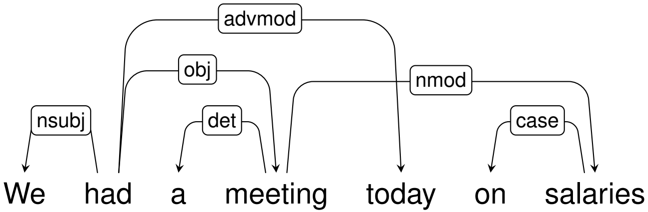




Projectivity

A tree is projective if no edges cross.

In English, most sensible sentences and phrases are projective. Some exceptions:



Universal dependencies

Where do the labels (nsubj, obj, det, ...) come from?

The Universal Dependencies initiative provides a common set of part-of-speech tags, morphological features, and dependency relations.

	Nominals	Clauses	Modifier words	Function Words
Core arguments	nsubj obj iobj	csubj ccomp xcomp		
Non-core dependents	obl vocative expl dislocated	advcl	advmod* discourse	aux cop mark
Nominal dependents	nmod appos nummod	acl	amod	det clf case
Coordination	**MWE**	**Loose**	**Special**	**Other**
conj cc	fixed flat compound	list parataxis	orphan goeswith reparandum	punct root dep

Treebanks and annotated data

A treebank is a corpus in which each sentence has exactly one parse tree.

- Manual annotation by linguists
- Provides frequency information: How often are various relations/rules used?
- Reusability: Many parsers, part-of-speech taggers can be built upon it
- Provides a way to evaluate systems.

The Universal Dependencies website provides links to treebanks in may languages, using the same tags and relations.

Dependency parsing

Parsing can be done using dynamic programming (Eisner's algorithm).

- This will give all parse trees of a given sentence in $O(n^3)$ time
- Much like the CKY algorithm for context-free grammars

However, much more popular (and more efficient) is to use transition-based parsing

- Greedy, deterministic algorithm
- Returns a single, projective, tree
- By using statistics from a treebank, the returned tree is most often (but not always) the "best" tree (the most probable interpretation of the sentence)

DD2417 Language Engineering
2f: Dependency parsing

Johan Boye, KTH

Transition-based dependency parsing

A parser configuration consists of:

- the buffer, initially containing the token ROOT + all the words of the sentence
- the stack, initially empty
- the (partial) dependency tree being contructed, initially containing all the words but no arcs.

The goal is to reach a final configuration, in which:

- the buffer is empty
- the stack only contains ROOT
- the tree is complete (there is an arc to every word)

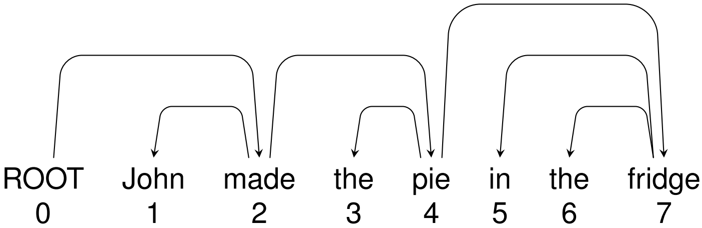
Transition-based dependency parsing

Three possible transitions:

- shift (SH): take the next word from the buffer, and push it onto the stack
- left arc (LA): create an arc from the topmost word to the second topmost word on the stack, then remove the second topmost word
- right arc (RA): create an arc from the second topmost word to the topmost word on the stack, then remove the topmost word

Representing dependency trees

Unlabeled trees can be represented by a list of head positions.



node	0	1	2	3	4	5	6	7
head	0	2	0	4	2	7	7	4

Transition-based parsing example

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0

STACK

ROOT John made the pie in the fridge

BUFFER

Next move: SH

Transition-based parsing example

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0

STACK	BUFFER
ROOT	John made the pie in the fridge

Next move: SH

Transition-based parsing example

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0

ROOT John

STACK

made the pie in the fridge

BUFFER

Next move: SH

Transition-based parsing example

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0

STACK	BUFFER
ROOT John made	the pie in the fridge

Next move: LA

Transition-based parsing example

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	2	0	0	0	0	0	0

STACK: ROOT made

BUFFER: the pie in the fridge

Next move: SH

Transition-based parsing example

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	2	0	0	0	0	0	0

ROOT made the

STACK

pie in the fridge

BUFFER

Next move: SH

Transition-based parsing example

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	2	0	0	0	0	0	0

ROOT made the pie

STACK

in the fridge

BUFFER

Next move: LA

Transition-based parsing example

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	2	0	4	0	0	0	0

ROOT made pie

STACK

in the fridge

BUFFER

Next move: SH

Transition-based parsing example

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	2	0	4	0	0	0	0

STACK: ROOT made pie in

BUFFER: the fridge

Next move: SH

Transition-based parsing example

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	2	0	4	0	0	0	0

STACK: ROOT made pie in the

BUFFER: fridge

Next move: SH

Transition-based parsing example

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	2	0	4	0	0	0	0

ROOT made pie in the fridge

STACK

BUFFER

Next move: LA

Transition-based parsing example

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	2	0	4	0	0	7	0

ROOT made pie in fridge

STACK

BUFFER

Next move: LA

Transition-based parsing example

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	2	0	4	0	7	7	0

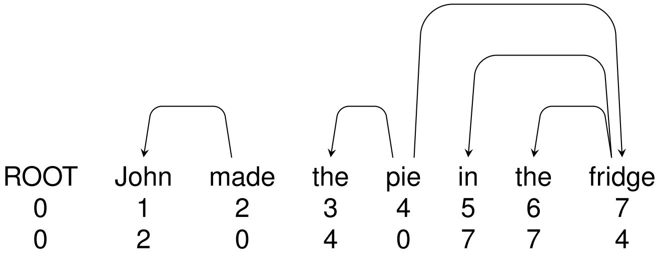
ROOT made pie fridge

STACK

BUFFER

Next move: RA

Transition-based parsing example



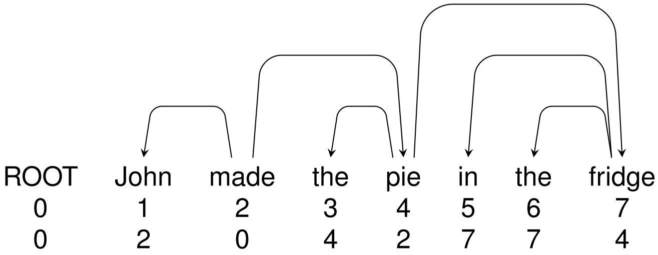
ROOT made pie

STACK

BUFFER

Next move: RA

Transition-based parsing example



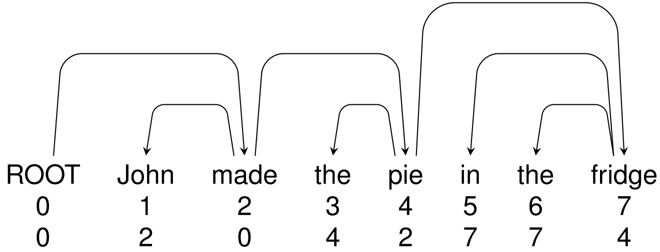
ROOT made

STACK

BUFFER

Next move: RA

Transition-based parsing example



STACK	BUFFER
ROOT	

Terminal state

Arc-standard algorithm

The algorithm we just looked at is called the arc-standard algorithm.

- possible moves are SH, LA, RA.
- in every parser configuration, one move needs to be selected
- algorithm is greedy, choices can not be undone
- we need an oracle to choose the correct action

Creating an oracle

An automatic oracle can be trained through machine learning.

We need a data material to use a training set:

- data points are parser configurations, with associated correct moves

Given such a training set, we can train a classifier:

- 3 classes: SH, LA, RA for unlabeled trees
- For labeled trees (with n labels) we would have $2n + 1$ classes: n LA moves, n RA moves, and 1 SH move.

The moves can be extracted from treebanks with correct dependency trees.

Determining the correct move

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0							
0	2	0	4	2	7	7	4

STACK	BUFFER
	ROOT John made the pie in the fridge

Next move: SH

Determining the correct move

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0							
0	2	0	4	2	7	7	4

ROOT John

STACK

made the pie in the fridge

BUFFER

Next move: SH

Determining the correct move

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0							
0	2	0	4	2	7	7	4

STACK	BUFFER
ROOT John made	the pie in the fridge

Next move: LA

Determining the correct move

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	2						
0	2	0	4	2	7	7	4

ROOT made

STACK

the pie in the fridge

BUFFER

Next move: SH

Determining the correct move

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	2		4				
0	2	0	4	2	7	7	4

STACK	BUFFER
ROOT made pie	in the fridge

Next move: SH

Determining the correct move

ROOT	John	made	the	pie	in	the	fridge
0	1	2	3	4	5	6	7
0	2		4		7	7	
0	2	0	4	2	7	7	4

ROOT made pie fridge

STACK

BUFFER

Next move: RA

Extracted datapoints

An example of a datapoint:

- Stack: $\boxed{\text{ROOT made pie}}$
- Buffer: $\boxed{\text{in the fridge}}$
- Correct move: SH

Extracted datapoints

An example of a datapoint:

- Stack: ROOT made pie
- Buffer: **in** the fridge
- Correct move: SH

Extracted datapoints

An example of a datapoint:

- Stack: ROOT VERB NOUN
- Buffer: ADP the fridge
- Correct move: SH

Extracted datapoints

An example of a datapoint:

- Stack: ROOT VERB NOUN
- Buffer: in the fridge
- Correct move: SH

Evaluation

Common metrics:

- **Labeled attachment score (LAS)**: Percentage of words that get correct head **and** label
- **Unlabled attachment score (UAS)**: Percentage of words that get correct head