Australian National University

COMP4650/6490 Document Analysis

Syntactic Parsing

ANU School of Computing

Administrative matters

- Assignment 3
  - Due: 5pm Thursday 12 October
  - Extension application:
    24 hours before due date + supporting documents
- Assignment 2
  - Results will be released this week

Australian National University

Outline

- Syntactic Parsing
- Context-Free Grammars & Constituency Parsing
- Dependency grammars & Dependency Parsing

Outline

- Syntactic Parsing
- Context-Free Grammars & Constituency Parsing
- Dependency grammars & Dependency Parsing

Parsing

Syntax: how words combine to form phrases and sentences

Syntactic analysis / parsing: determining the syntactic structure of text by analysing the underlying grammar (of the language)

- Gives a deeper understanding of word groups and their grammatical relationships

- Sentences are not simply bags of words:

  Mary bought John a coffee

  vs

  John bought Mary a coffee

Parsing

Formally tries to resolve structural ambiguity in text

e.g. *Mary saw a cat with binoculars*

Typically, in the broad context of the NLP Pipeline:

Tokenise → ··· → POS Tag → **Parse** → ···

Applications:

- Machine Translation
- Question Answering
- Text Summarisation
- Grammar Checking
- Information Extraction

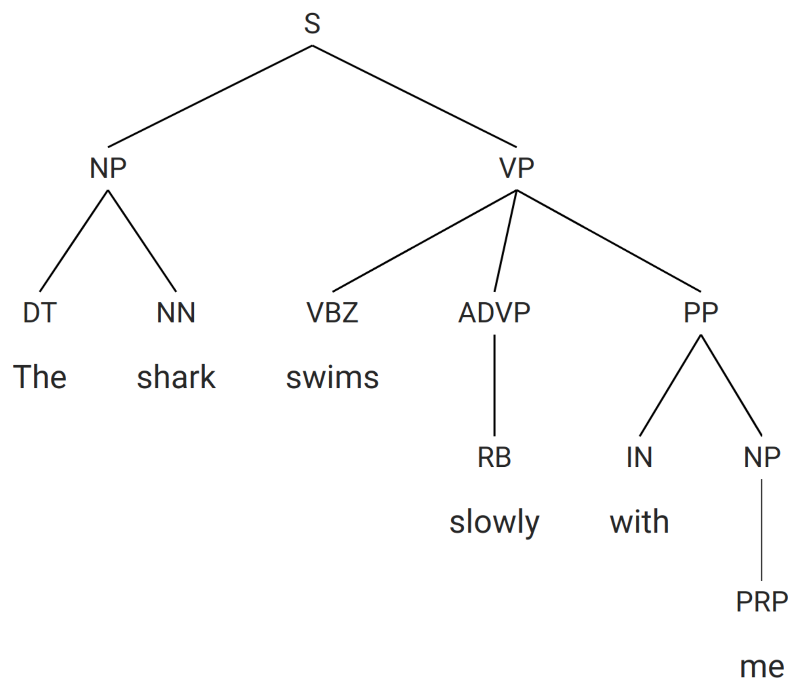
Parsing

- Constituency parsing
  - Phrases represented as nodes in a tree
- Dependency parsing
  - Dependencies between words
- Constituency parsing vs Dependency parsing
  - Dependency parsing is typically faster and works for many languages
  - Constituency parsing tends to favour languages with somewhat fixed word order patterns, and clear constituency structures, e.g. English

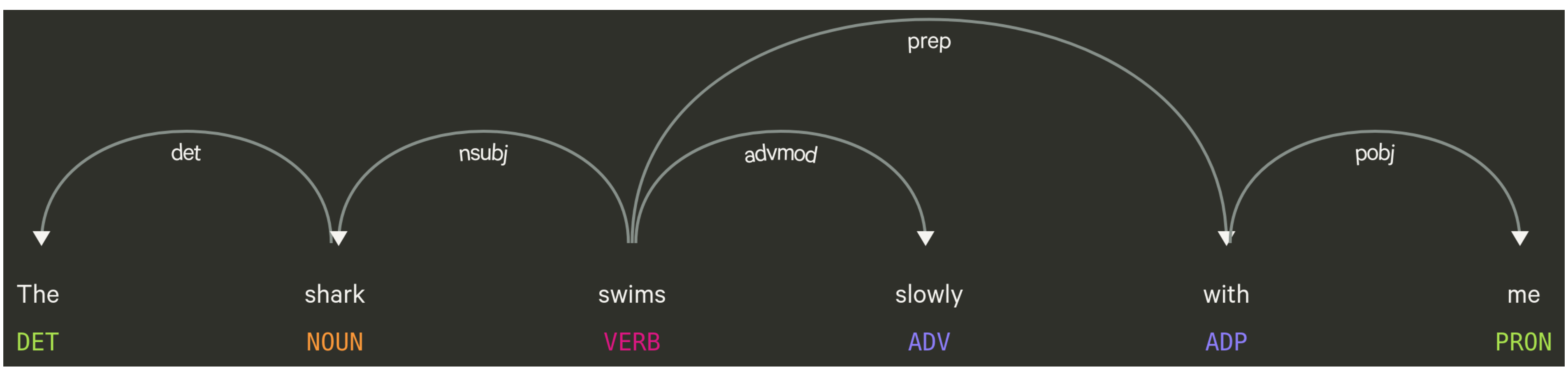
How to represent sentence structure

Constituency tree
(phrase structure tree)

https://parser.kitaev.io/



Dependency tree https://explosion.ai/demos/displacy



Outline

- Syntactic Parsing
- Context-Free Grammars & Constituency Parsing
- Dependency grammars & Dependency Parsing

Constituency Parsing

Splits sentences into sub-phrases or constituents

- **Constituent**: a word or a group of words that behaves as a single unit
- Why do these words group together?
  - Appear in similar syntactic environments

three parties from Sydney *arrive* … Drunk driver *fled* … they *sit* …	✔
from *arrive* … the *fled* … as *sit* …	✘

  - Preposed or postposed constructions

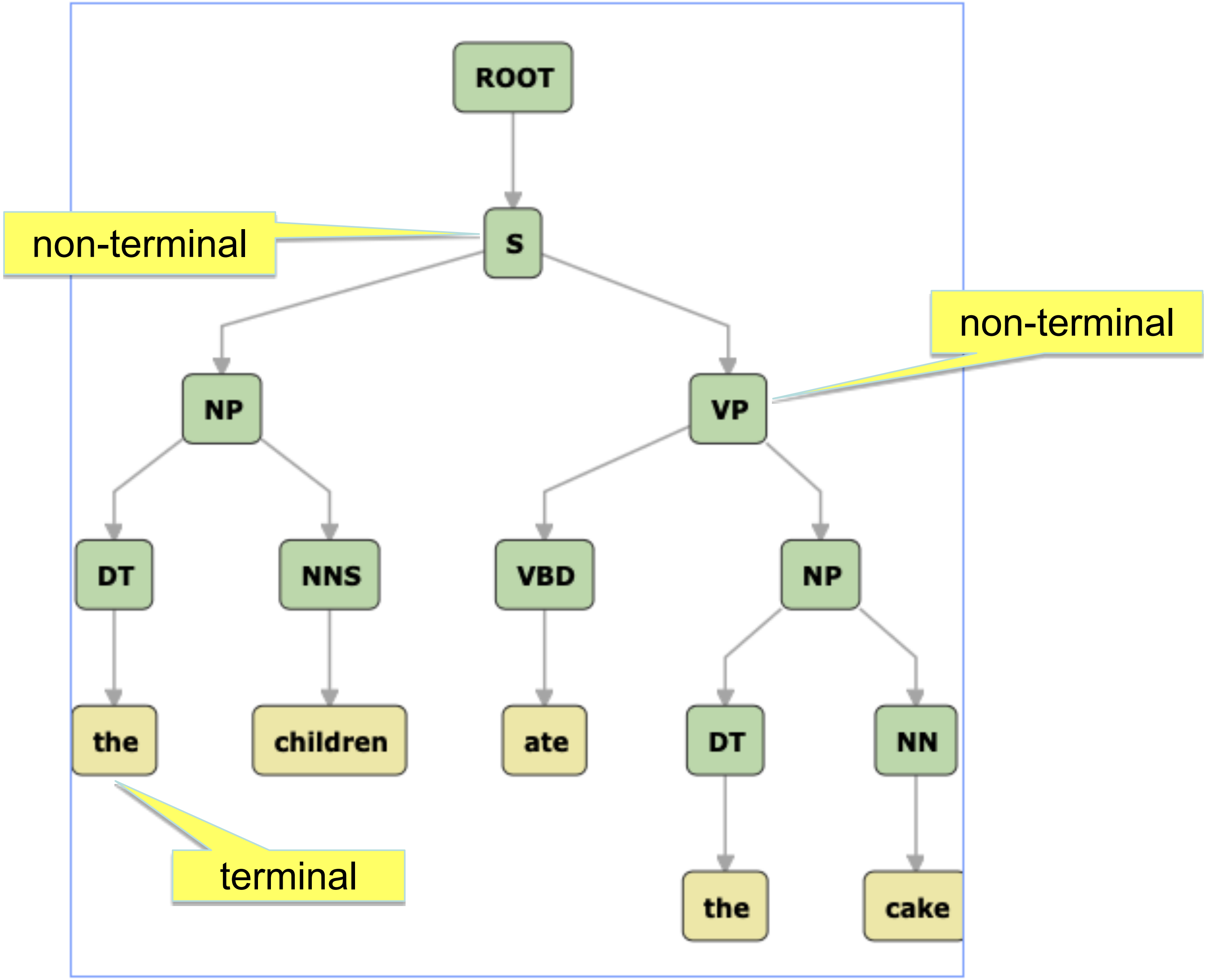
On August 30th, I'd like to fly from Canberra to Sydney.
I'd like to fly on August 30th from Canberra to Sydney.
I'd like to fly from Canberra to Sydney on August 30th.

- Adds more structure to part-of-speech (POS) tagged sentences

Constituency tree (also known as *phrase structure tree*) form:

- Types of phrases: *non-terminals*
- Words in the sentence: *terminals*

Constituency Parsing



Stanford CoreNLP Parser https://corenlp.run/

Context-Free Grammars

A context free grammar consists of:

- a set of *context-free rules* (i.e. productions), each of which expresses the ways that symbols of the language can be grouped and ordered together

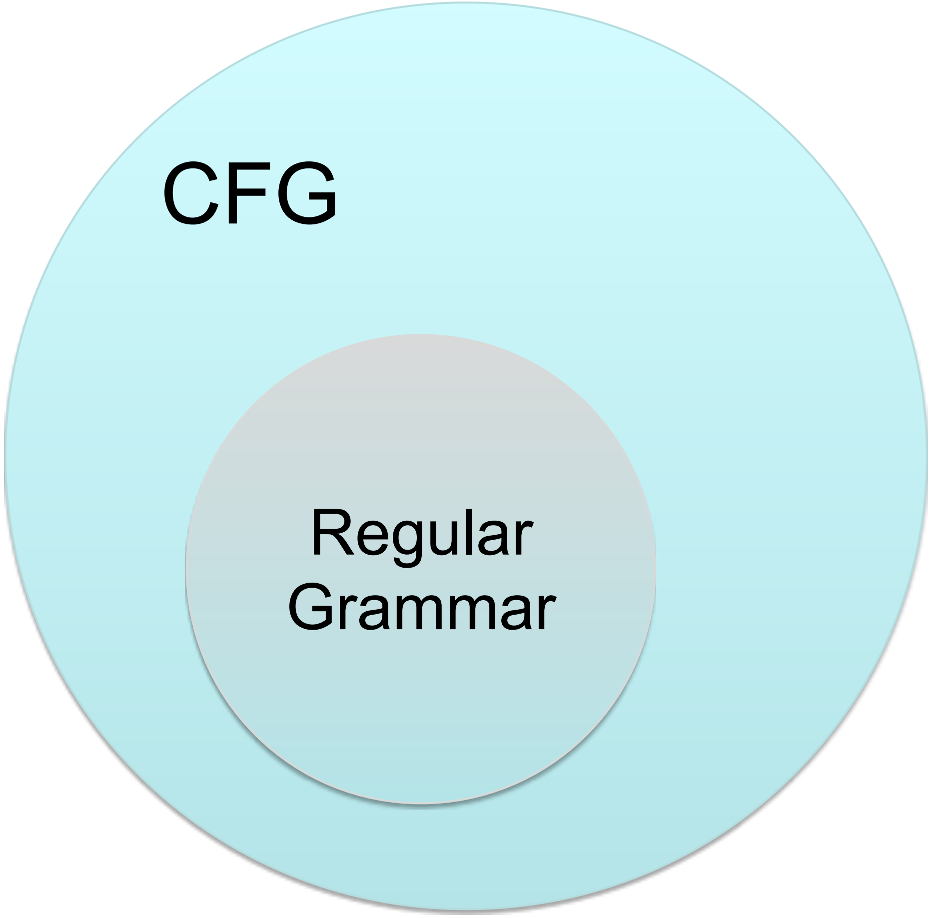
S	→	NP VP
VP	→	Verb NP
NP	→	Det Nominal
NP	→	Noun

- a *lexicon* of words and symbols, and a set of rules which express facts about the lexicon.

Noun	→	bus
Noun	→	stop
Verb	→	stop
Det	→	the
Verb	→	fled

These are the building blocks of a Constituency Parser

Context-Free Grammars



Regular Grammar:

$$A \rightarrow a$$
$$A \rightarrow a \ B$$
$$A \rightarrow \epsilon$$

Context-Free Grammars (CFGs) are more general than Regular Grammars

Formal Definition of CFG

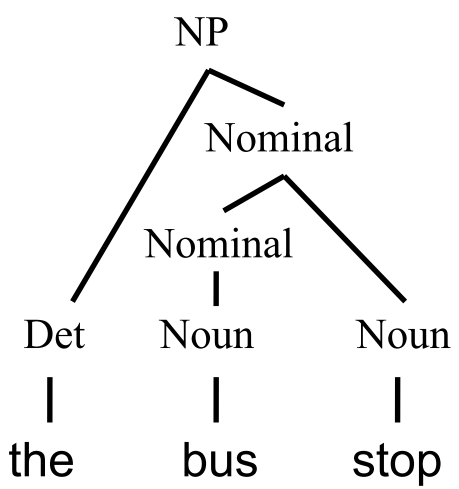
A context-free grammar $G = \{N, \Sigma, R, S\}$

- N is a set of non-terminals e.g. NP, VP, Nominal, …
- Σ is a set of terminal symbols, $N \cap \Sigma = \varnothing$, e.g. Mary, like, …
- R is a set of rules (productions), each of the form $A \rightarrow B$, where A is a non-terminal, B is a string of symbols from the infinite set of strings $\{\Sigma \cup N\}*$
- S is a designated start symbol

Context-Free Grammars

The sequence of rule expansions is called a *derivation* of the string of words

Parse tree



Bracketed notation

[*NP* [*Det* the][*Nom* [*Nom* [*Noun* bus]][*Noun* stop]]

A Toy Example

Noun $\longrightarrow$ bus

Noun $\longrightarrow$ stop

Det $\longrightarrow$ the | a | an

Nominal $\longrightarrow$ Noun

NP $\longrightarrow$ Det Nominal

Nominal $\longrightarrow$ Nominal Noun

Sentence: the bus stop

A Toy Example

Noun → bus

Noun → stop

Det → the | a | an

Nominal → Noun

NP → Det Nominal

Nominal → Nominal Noun

Sentence: the bus stop

Det	Noun	Noun
|	|	|
the	bus	stop

A Toy Example

Noun → bus

Noun → stop

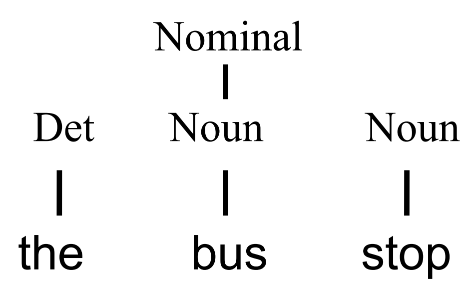
Det → the | a | an

Nominal → Noun

NP → Det Nominal

Nominal → Nominal Noun

Sentence: the bus stop



A Toy Example

Noun → bus

Noun → stop

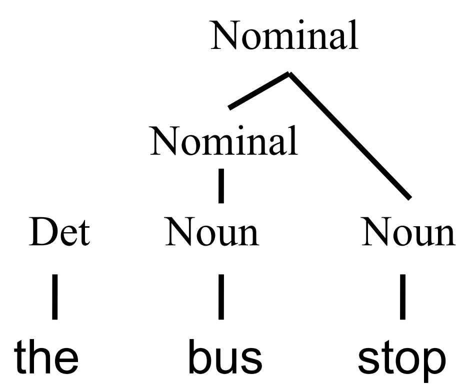
Det → the | a | an

Nominal → Noun

NP → Det Nominal

Nominal → Nominal Noun

Sentence: the bus stop



A Toy Example

Noun → bus

Noun → stop

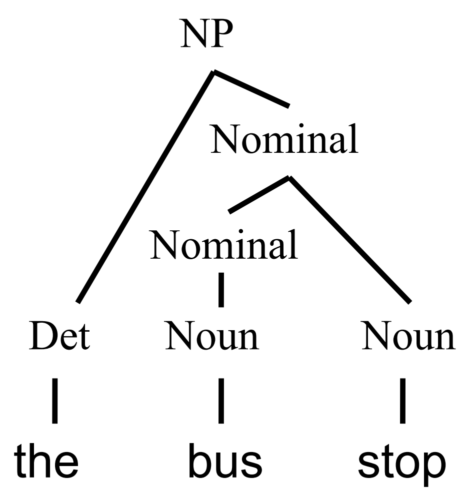
Det → the | a | an

Nominal → Noun

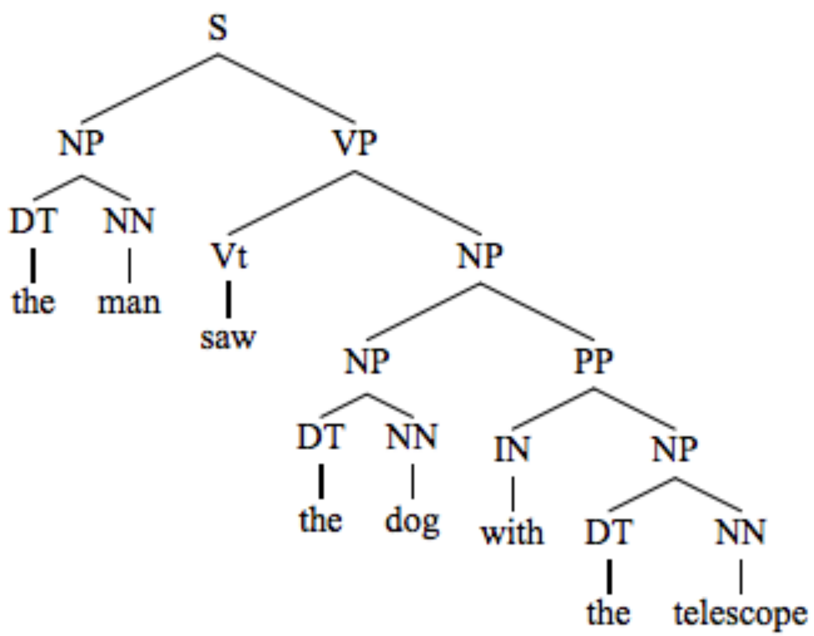
NP → Det Nominal

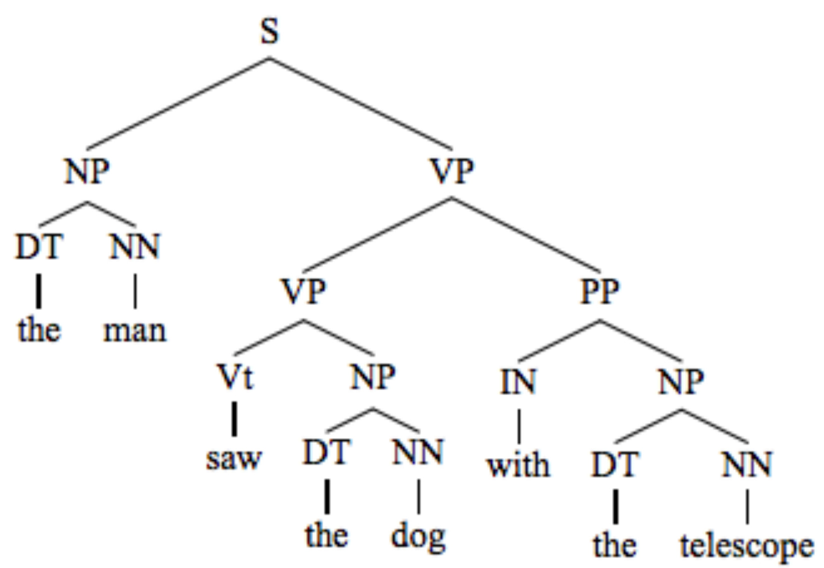
Nominal → Nominal Noun

Sentence: the bus stop



Ambiguity of Parsing: structural ambiguity





Probabilistic context-free grammar (PCFG)

A parameter for each grammar rule

α		β		$q(\alpha \to \beta)$
S	$\Rightarrow$	NP	VP	1.0
VP	$\Rightarrow$	Vi		0.4
VP	$\Rightarrow$	Vt	NP	0.4
VP	$\Rightarrow$	VP	PP	0.2
NP	$\Rightarrow$	DT	NN	0.3
NP	$\Rightarrow$	NP	PP	0.7
PP	$\Rightarrow$	P	NP	1.0

α		β	$q(\alpha \to \beta)$
Vi	$\Rightarrow$	sleeps	1.0
Vt	$\Rightarrow$	saw	1.0
NN	$\Rightarrow$	man	0.7
NN	$\Rightarrow$	woman	0.2
NN	$\Rightarrow$	telescope	0.1
DT	$\Rightarrow$	the	1.0
IN	$\Rightarrow$	with	0.5
IN	$\Rightarrow$	in	0.5

α: non-terminal

β: string of non-terminals and terminals

Find the most likely parse tree (T_G is the set of all possible trees)

$$\text{argmax}_{t \in T_G} p_G(t) \quad \text{where} \quad p_G(t) = \prod_{i=1}^{n} q_i(\alpha \to \beta)$$

Learning PCFG from Treebanks

- Penn treebank and English Web treebank

```
((S (NP-SBJ-1 Jones)
    (VP followed)
    (NP him)
    (PP-DIR into
      (NP the front room))
,
(S-ADV (NP-SBJ *-1)
  (VP closing
    (NP the door)
    (PP behind
      (NP him)))))
.))
```

- Maximum-Likelihood estimation:

$$q^*(\alpha \rightarrow \beta) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

Grammar Equivalence

- Two grammars are *equivalent* if they generate the same language (set of strings)
- Chomsky Normal Form (CNF)
  - Allow only two types of rules.
  - The right-hand side of each rule either has two non-terminals or one terminal,
  - except $S \rightarrow \epsilon$ (where ϵ is the empty string)

Examples of **valid** types of rules (if in CNF):

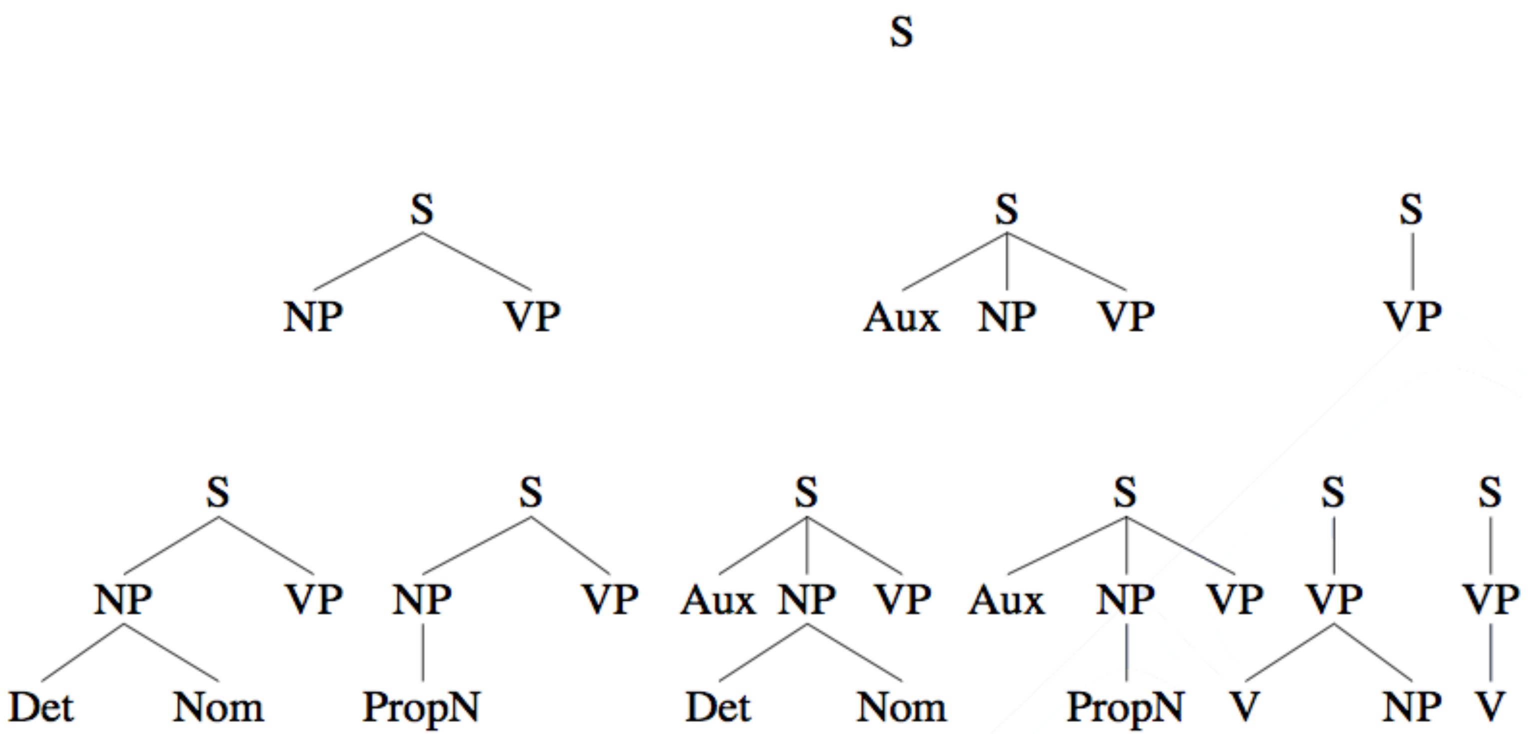
$$A \rightarrow B\ D$$
$$C \rightarrow a$$
$$S \rightarrow \epsilon$$

Examples of **invalid** types of rules (if in CNF):

$$A \rightarrow B\ a\ D$$
$$C \rightarrow \epsilon$$
$$E \rightarrow A$$

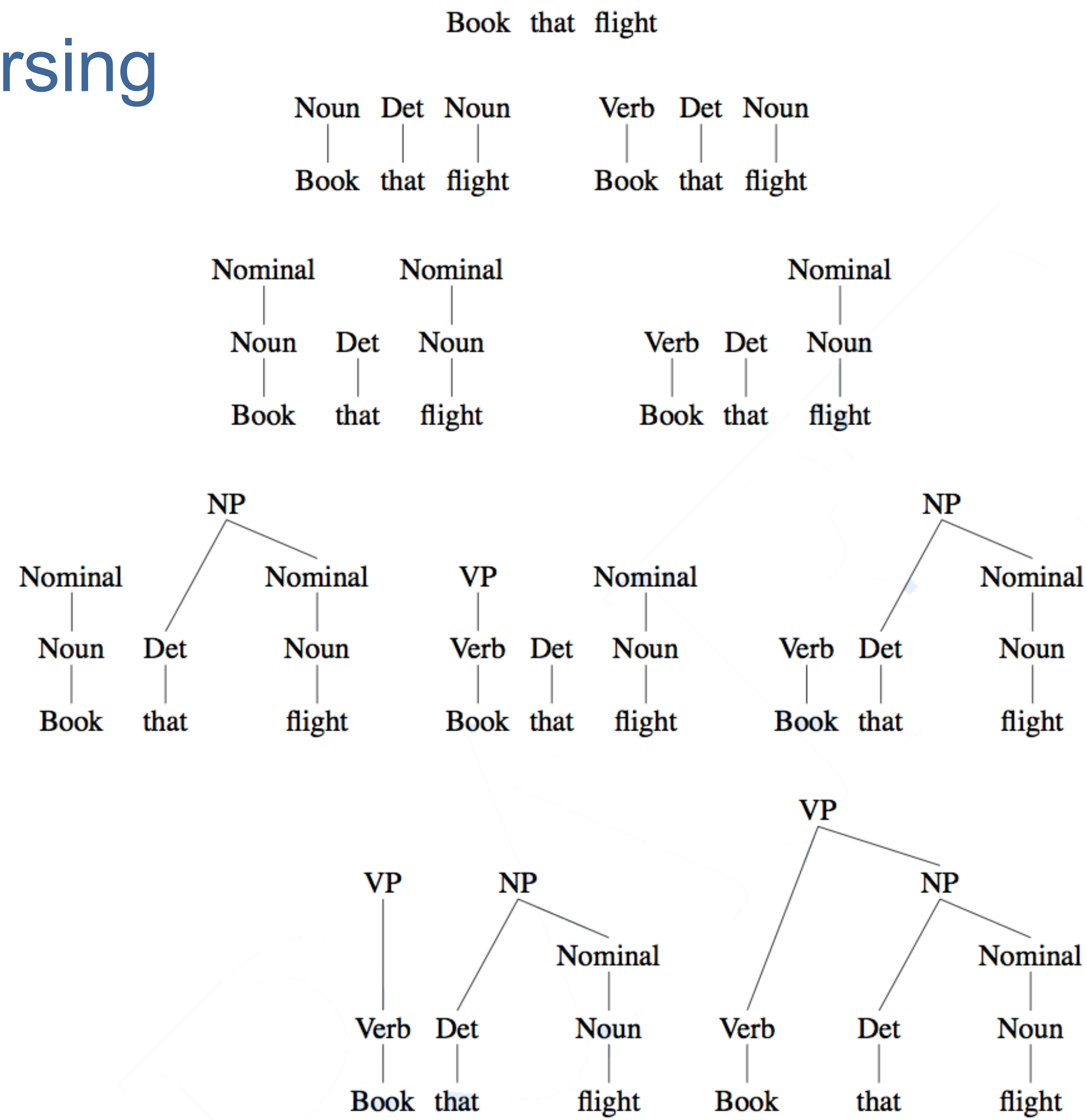
where $A, B, C, D, E \in N$ and $a \in \Sigma$

Top Down Parsing



PCFGs

Bottom Up Parsing



Available Constituency Parsers

Stanford Parser

http://nlp.stanford.edu/software/srparser.shtml

Berkley Neural Parser

https://spacy.io/universe/project/self-attentive-parser

UCSD Rethinking Self-Attention

https://github.com/KhalilMrini/LAL-Parser

And many more...

Outline

- Syntactic Parsing
- Context-Free Grammars & Constituency Parsing
- Dependency grammars & Dependency Parsing

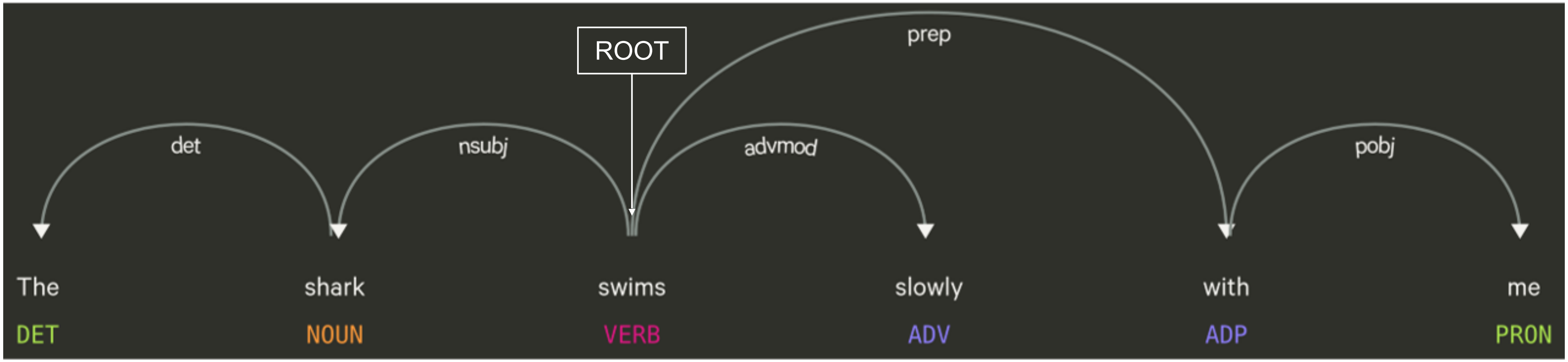
Dependency grammars

- Use binary asymmetric relations between words (rather than phrase-structure rules) to describe the syntactic structure of a sentence
- Binary asymmetric relation: head → dependent
- Head (governor): grammatically most important
- Dependent (modifier): modifier, object, or complement
- Dependency grammar can deal with languages that have a relatively free word order, e.g. Czech
- Head-dependent relations can represent predicate-argument structure in sentence

Dependency parsing

- Analyse the grammatical structure of sentences by establishing relations (i.e. the type of dependency) between “head” words and "dependent" words
- Explain how all the words in a sentence relate to each other
- Build a tree (called dependency tree) that assigns a single “head” or “parent” word to each word in the sentence
    - Nodes represent words, edges represent dependencies (from head to dependent)
    - The root of the tree is (typically) the main verb in the sentence
    - All the words, except one, are dependent on another word in the sentence
- Examples of dependency types: http://universaldependencies.org/docsv1/en/dep/index.html

Dependency Parsing

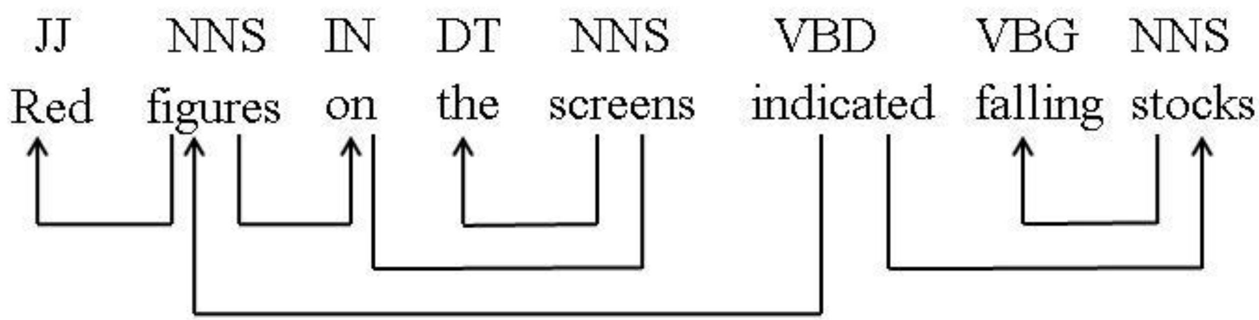


In a dependency tree

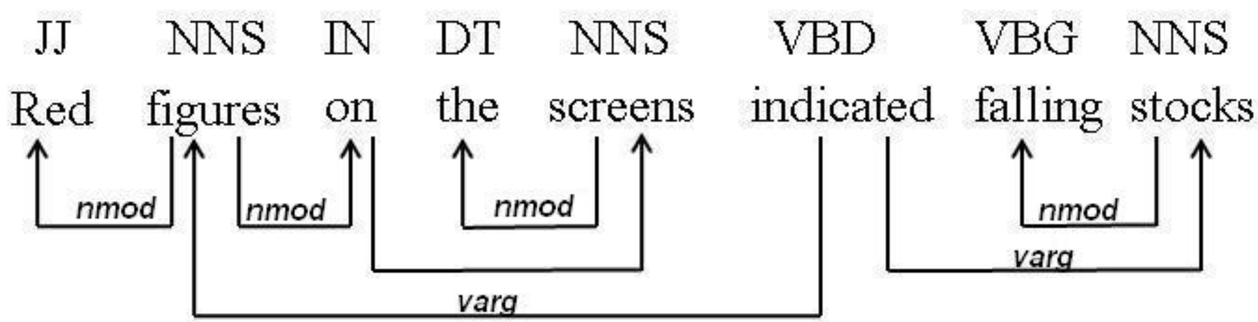
- Nodes represent words
- Edges represent dependencies (from head to dependent)
- Root = special token which is not a dependent. Sometimes omitted.
- Dependents = all other words, which are directly or indirectly linked to the root word

Dependency Parsing

Without labels



With labels



Dependency Parsing

Formal definition for unlabelled dependency trees:

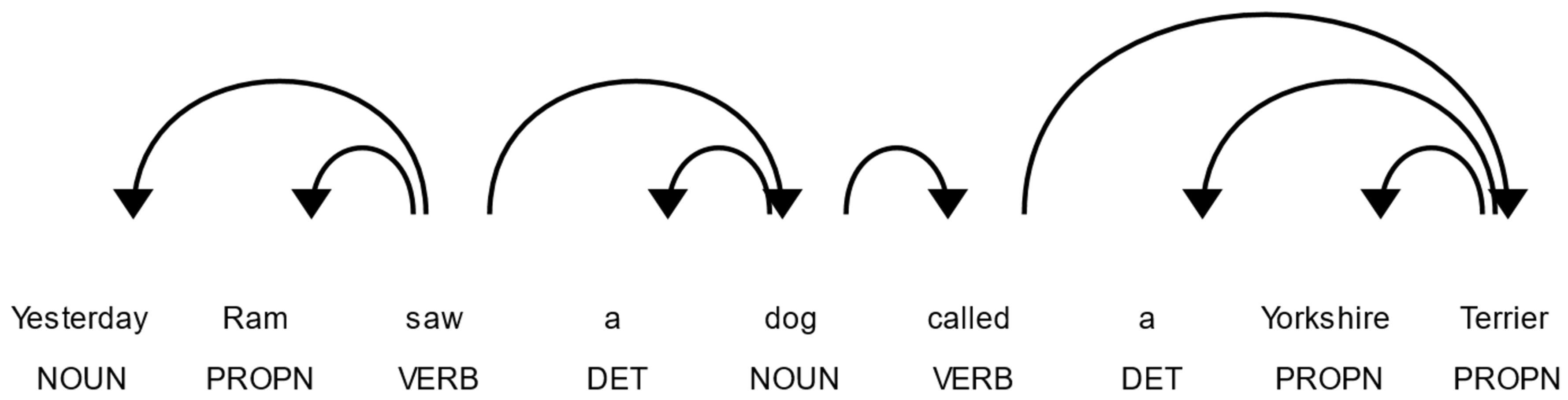
Dependency graph $D = (V, E)$ where

- V is the set of nodes (words in a input sequence).
- E is the set of arcs indicating grammatical relations.
- $v_i \to v_j$ or $(v_i, v_j) \in E$ denotes an arc from head v_i to dependent v_j.

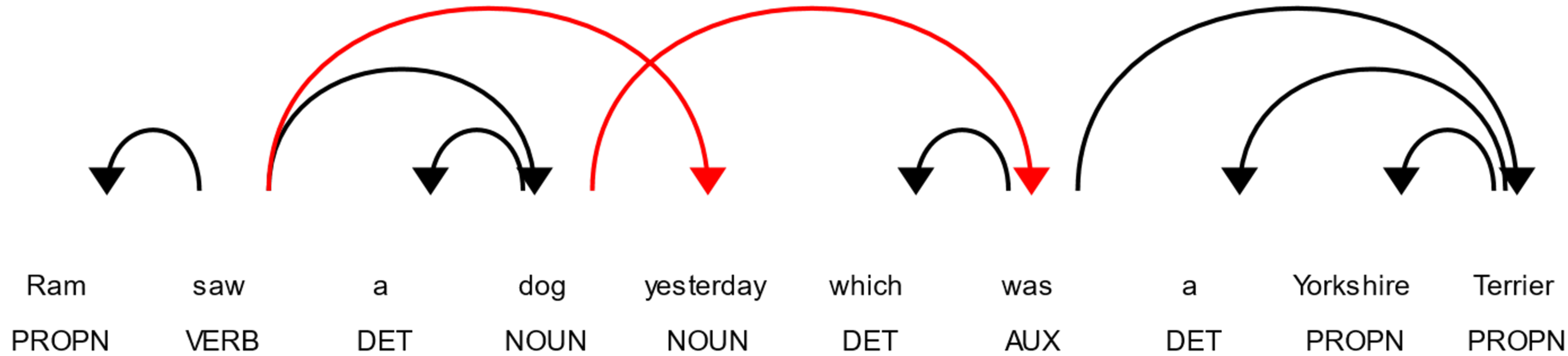
Dependency parsing: the task of mapping an input string to a dependency graph satisfying certain conditions (see following slides)

Dependency Parsing

Projective Dependency Tree



Non-Projective Dependency Tree



Crossing lines!

English has very few non-projective cases

Which is good because methods that only allow projective dependency trees/forests tend to be simpler/faster

Well-Formedness

A dependency tree is well-formed *iff*

- **Single head**: Each word has only one head
- **Acyclic**: The graph should be acyclic i.e. has no cycles
- **Connected**: There is a path between any pair of nodes
- **Projective**: if an edge from word A to word B implies that there exists a directed path in the graph from A to every word between A and B in the sentence

Note: the graph may be a forest rather than a single tree. In which case it need not be connected, and not every node has a head.

Parsing Algorithms

Transition-based parsing

- Similar to the shift-reduce parsing in compilers
  - stack (for building the parse)
  - buffer (with tokens to be parsed)
  - parser (takes actions/transitions on the parse)
- Tends to work best for local dependencies
- e.g. Covington, Yamada & Matsumuto, Nivre Arc-eager

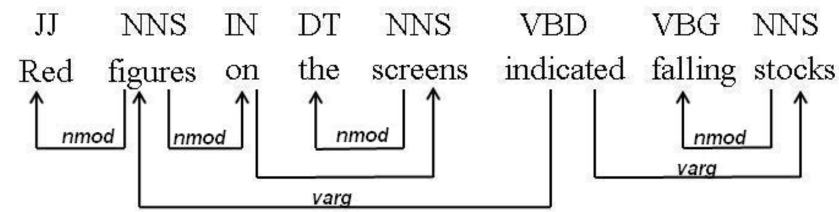
Graph-based parsing

- Search through the space of possible trees for a given sentence for optimal solutions
- More accurate than transition-based parsers for long sentences (where heads are likely far from the dependents)
- e.g. Eisner (CKY variant), McDonald (Maximum Spanning Tree)

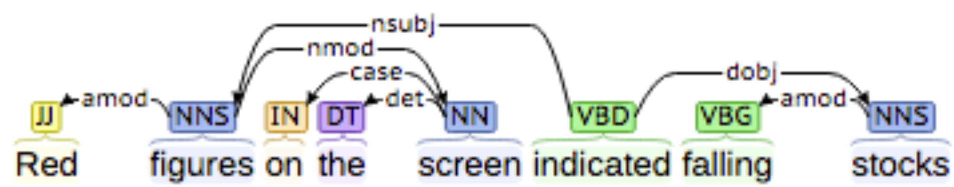
Dependency Parsing

Dependency Corpora

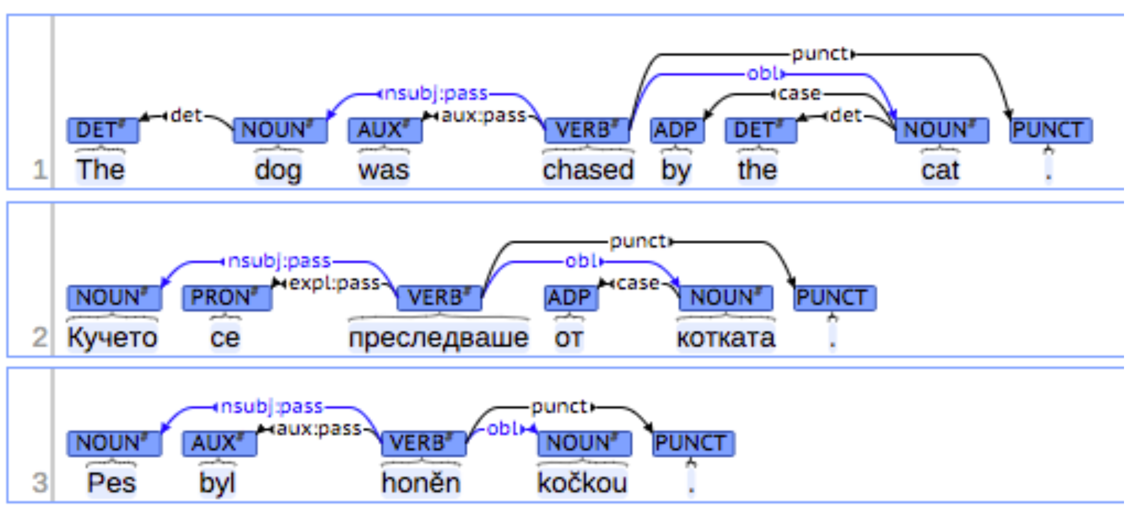
CoNLL dependencies
https://aclanthology.org/D07-1096/



Stanford typed dependencies
http://nlp.stanford.edu/software/dependencies_manual.pdf



Universal dependencies
https://universaldependencies.org/u/overview/syntax.html



Off-the-Shelf Dependency Parsers

spaCy https://spacy.io/

Stanza (from Stanford) https://stanfordnlp.github.io/stanza/

MaltParser (Nivre) https://www.maltparser.org

Demos:
https://explosion.ai/demos/displacy
https://demo.allennlp.org/dependency-parsing

Dependency Structures vs. Phrase Structures

- Dependency structures explicitly represent
  - Head-dependent relations (directed arcs)
  - Functional categories (arc labels)
  - Predicate-argument structure
- Dependency structure independent of word order
  - Suitable for free word order languages, such as Indian languages
- Phrase structures explicitly represent
  - Phrases (non-terminal nodes)
  - Structural categories (non-terminal labels)
  - Fragments are directly interpretable

References

- Chapters 17, 18. Speech and Language Processing (3rd ed. draft)
- Appendix C. Speech and Language Processing (3rd ed. draft). Separate PDF.