CS918: LECTURE 9

Grammars and Parsing

Arkaitz Zubiaga, 31st October, 2018

LECTURE 9: CONTENTS

- What is parsing?
- What are constituencies and dependency structures?
- Probabilistic parsing: Context Free Grammars (CFG and PCFG).
- Lexicalised parsing.
- Dependency parsing.

GRAMMARS AND PARSING

- **Parsing:** process of recognising a sentence and assigning **syntactic structure** to it.
- **Syntactic structure** includes:
  - **Constituents:** how words group together to behave as single units.
  - **Grammatical relations, dependencies, subcategorisation:** how words and constituents relate to each other.

WARWICK

- **Examples of constituents:**
  - Noun Phrase (NP):
    the flights, some flights, any flights,…
  - Prepositional Phrase (PP):
    in the morning, at home,…

GRAMMARS AND PARSING: EXAMPLE

- **Examples of constituents:**
  - Verb Phrase (VP):

VP → Verb	e.g. disappear
VP → Verb NP	e.g. prefer a morning flight
VP → Verb NP PP	e.g. leave London in the morning
VP → Verb PP	e.g. leaving on Thursday

GRAMMARS AND PARSING: EXAMPLE

- **Grammatical relations:**

  She ate a large breakfast

  “She”: SUBJECT, “large breakfast”: OBJECT

WHY IS SYNTAX IMPORTANT?

WARWICK

- Parsing is key for applications needing **deep understanding of language**, e.g.:
  - Grammar checkers.
  - Information extraction.
  - Machine translation.
  - Dialogue management.
  - Question answering.

SYNTACTIC CONCEPTS THAT WE WILL EXPLORE

- Constituency
- Grammatical relations
- Dependencies and Heads
- Subcategorisation
- Key formalism: Context Free Grammars, Dependency Grammars
- Resources: Treebanks
- Algorithms for parsing

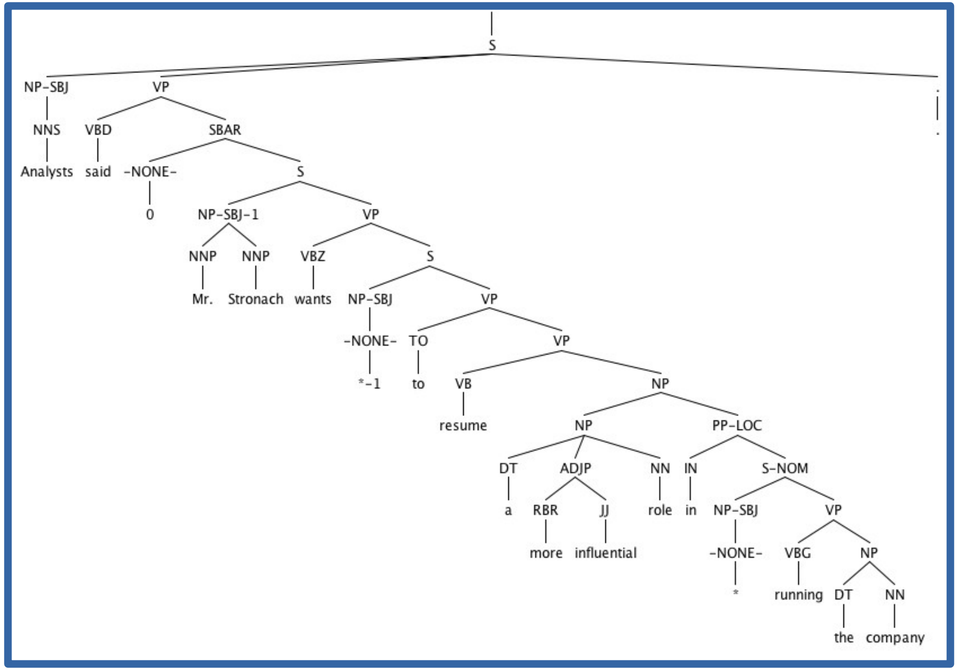
CONSTITUENCY

- In general, words forming constituents can move around together, e.g.:
  - **On 7th September** I'd like to fly from London to New York.
  - I'd like to fly from London to New York **on 7th September**.
  - **On September** I'd like to fly from London to New York **7th**.

- e.g. a **prepositional phrase (PP)** would need a **preposition followed by a noun** (e.g. "In September", "on time")

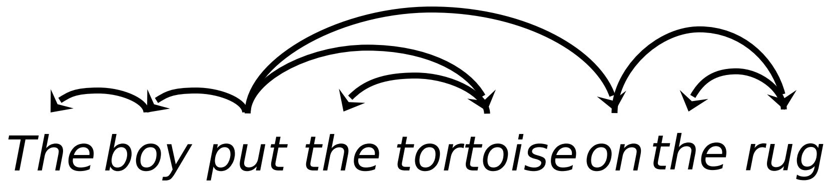
CONSTITUENCY (PHRASE STRUCTURE)

- Analysts said Mr. Stronach wants to resume a more influential role in running the company.



DEPENDENCY STRUCTURE

- **Dependency structure** shows which **words depend on (modify or are arguments of)** which **other words**.



HOW CAN PARSING AND STRUCTURE HELP?

- The computer on the 3rd floor has crashed.

- What has crashed?
  - The computer.
  - The 3rd floor.

AMBIGUITY: KEY CHALLENGE IN PARSING

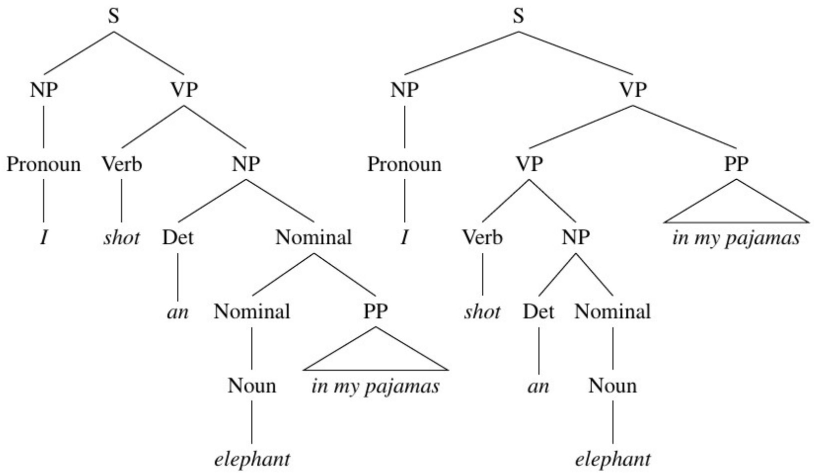
WARWICK

- I shot an elephant in my pijamas.

  What do we understand here?

AMBIGUITY: KEY CHALLENGE IN PARSING

- I shot an elephant in my pijamas.



Who’s wearing ‘my pijamas’? The elephant or me? :-)

APPROACHES TO PARSING

WARWICK

- Three different ways of parsing texts:
    - Probabilistic parsing: context-free grammars.
    - Lexicalised parsing.
    - Dependency parsing.

PROBABILISTIC PARSING: CONTEXT-FREE GRAMMARS

CONTEXT-FREE GRAMMARS (CFG)

- Context-free grammars (CFGs) consist of:
  - Terminals.
  - Non-terminals.
  - Rules.

which allow us to produce grammatical sentences.

CFG EXAMPLE

Rules

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

Non-terminals

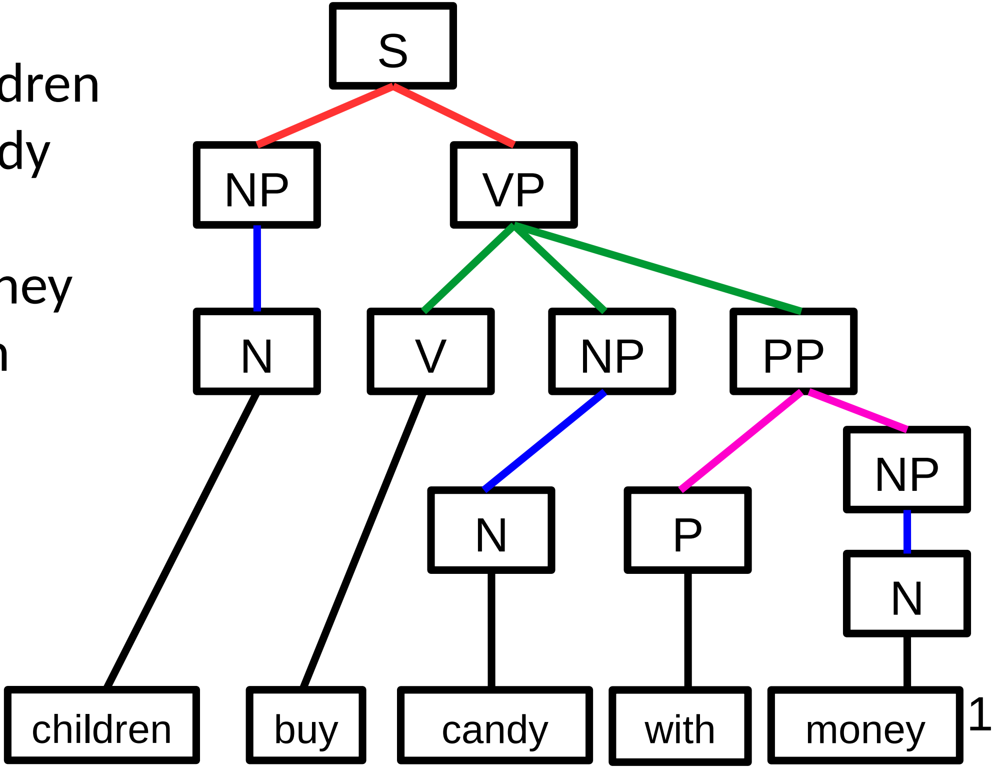
N → children
N → candy
V → buy
N → money
P → with
V → eat

Terminals

CFG EXAMPLE

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

$N \rightarrow$ children
$N \rightarrow$ candy
$V \rightarrow$ buy
$N \rightarrow$ money
$P \rightarrow$ with
$V \rightarrow$ eat



WARWICK

PROBABILISTIC CFG (PCFG)

- Like CFG, but each rule has a probability.

$$\left.\begin{array}{l} S \to NP\ VP\ (1.0) \\ VP \to V\ NP\ (0.6) \\ VP \to V\ NP\ PP\ (0.4) \\ \ldots \end{array}\right.$$

$$\left.\begin{array}{l} VP \to V\ NP\ (0.6) \\ VP \to V\ NP\ PP\ (0.4) \end{array}\right\} 1.0$$

$$\left.\begin{array}{l} N \to children\ (0.5) \\ N \to candy\ (0.3) \\ N \to money\ (0.2) \\ \ldots \end{array}\right\} 1.0$$

PARSING

WARWICK

- We may need to work in both directions:

  - Top-down approach from rules to producing sentence
    → **Natural Language Generation**.

  - Bottom-up approach from sentence to generating tree
    → **Parsing**.

PARSING

WARWICK

- We are given a sentence, and we need to generate the tree based on the rules we have.

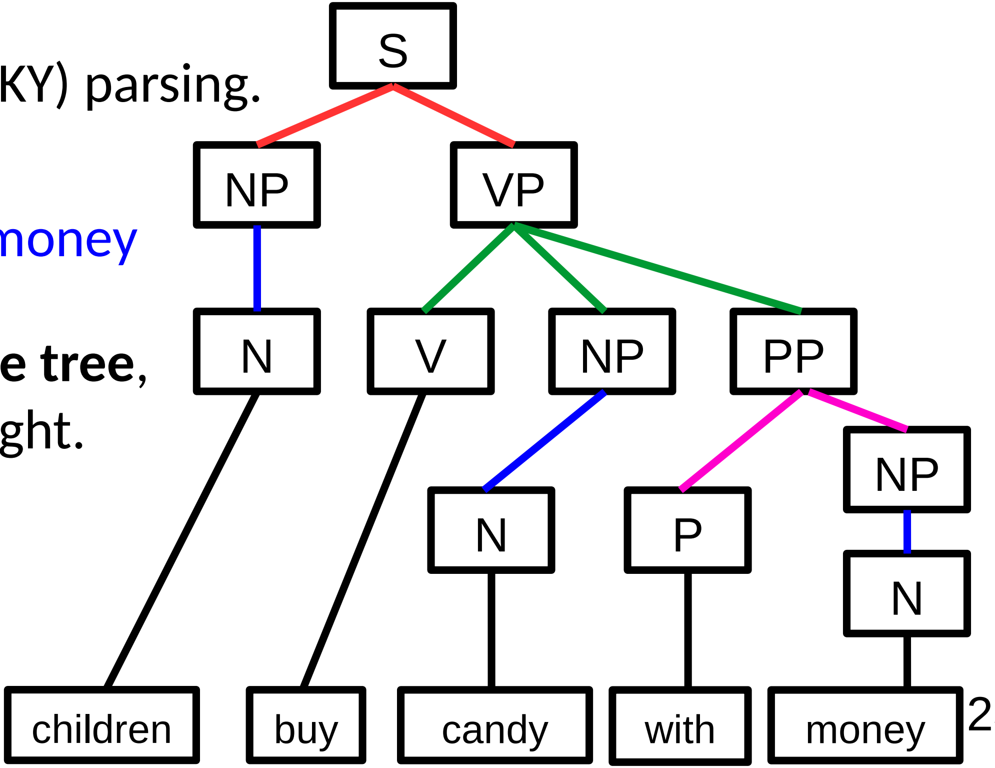
Cocke-Kasami-Younger (CKY) algorithm for parsing.

CKY PARSING ALGORITHM

WARWICK

- Cocke-Kasami-Younger (CKY) parsing.
- We have the sentence:
  children buy candy with money

  We aim to **infer the whole tree**, as in the picture on the right.

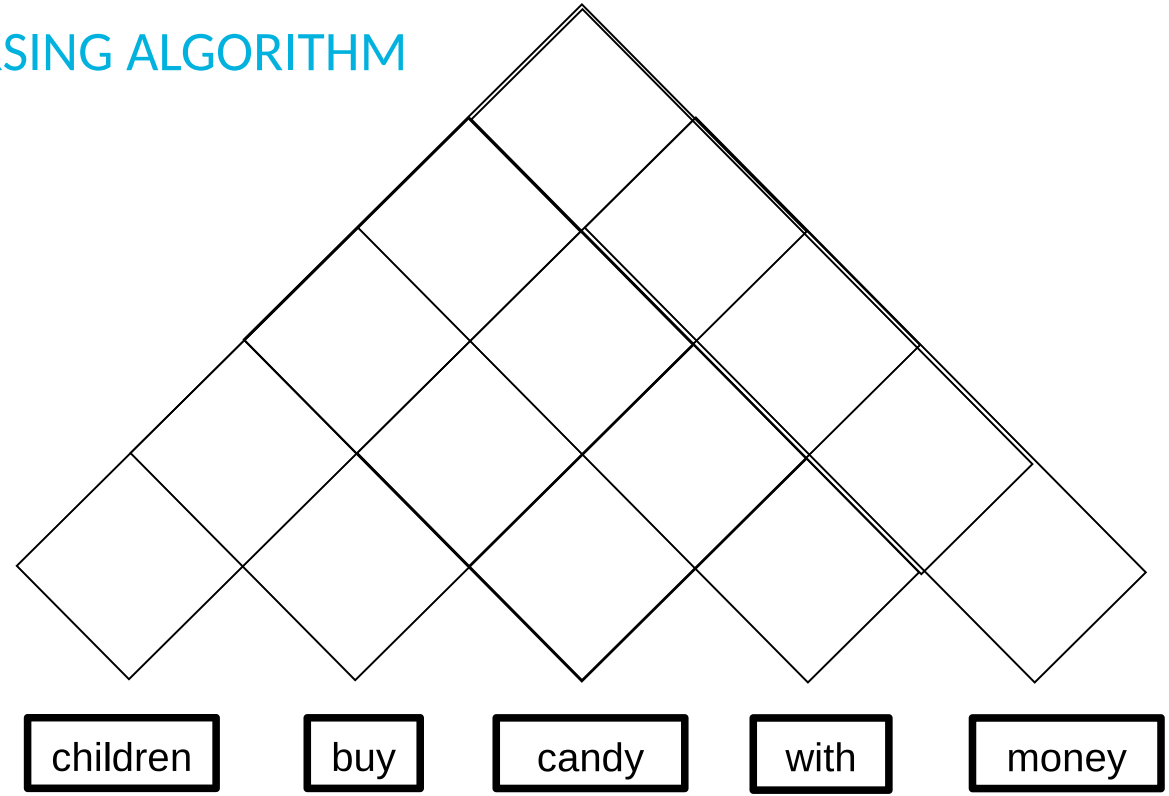


CKY PARSING ALGORITHM

WARWICK

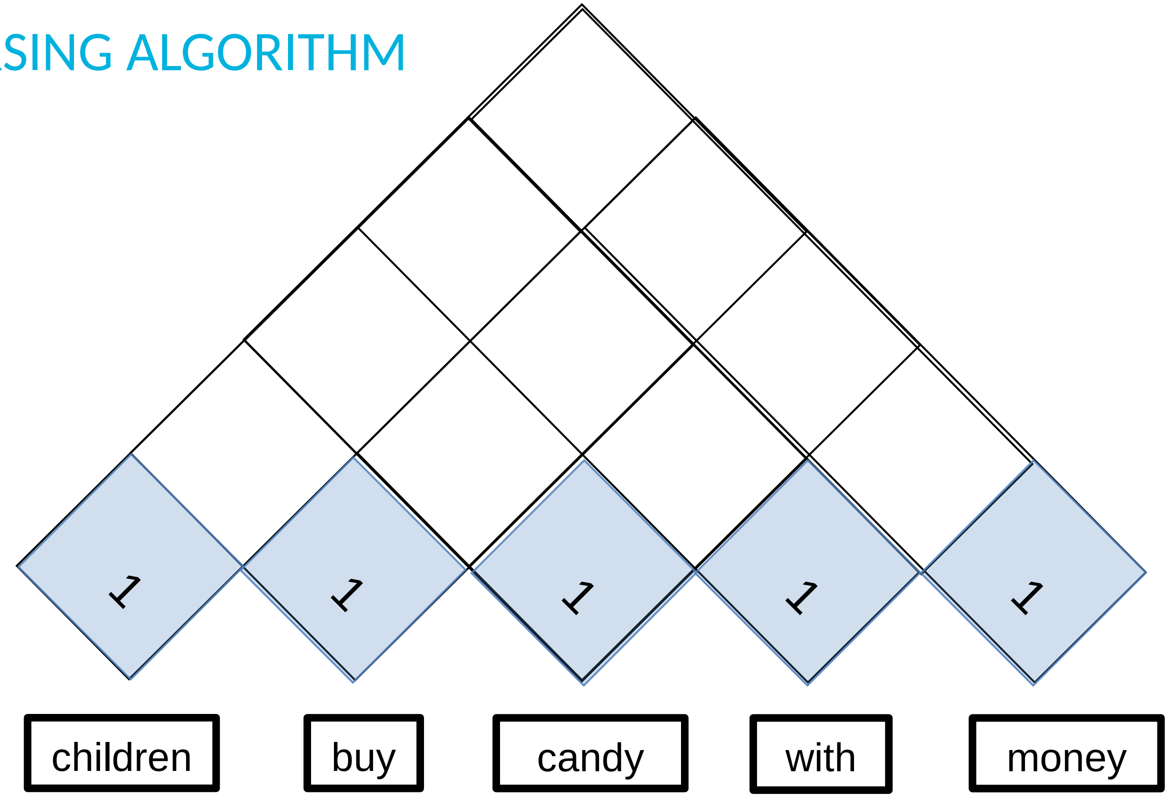
- Having the entire constituency list (T, N, R).
  - Use **bottom-up approach** to fill in the tree.

CKY PARSING ALGORITHM

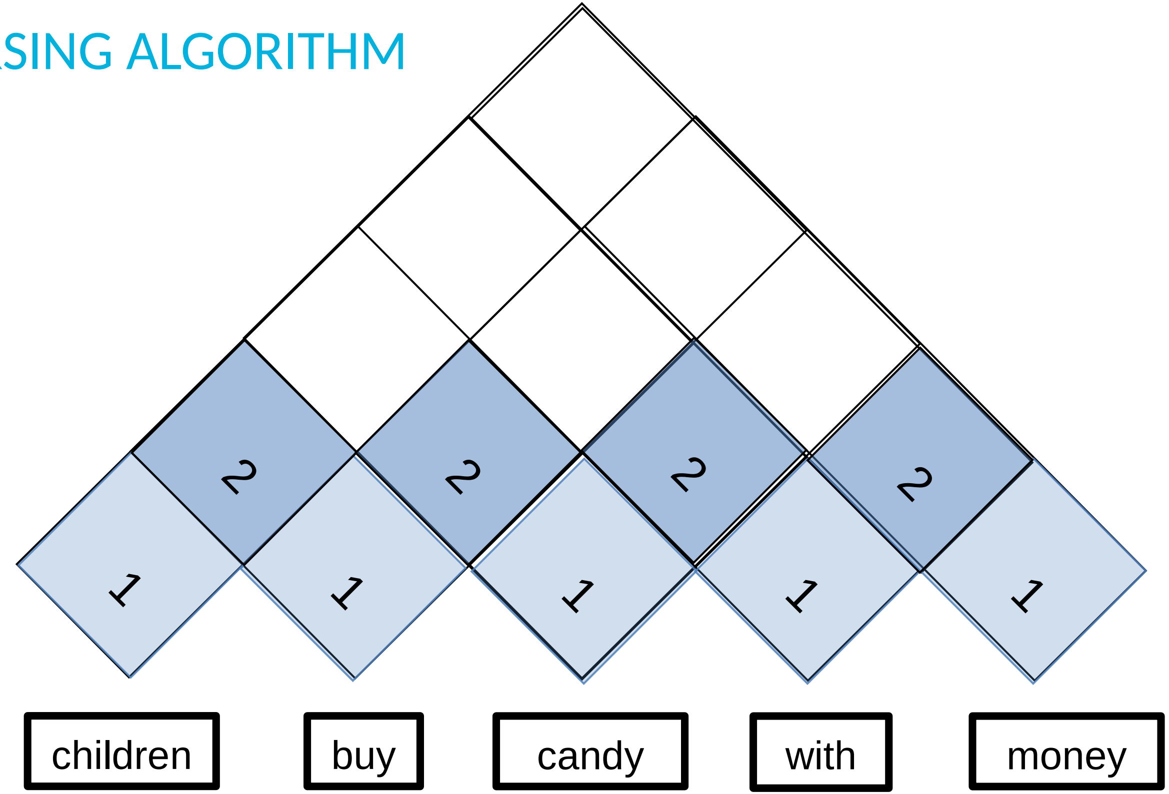


CKY PARSING ALGORITHM

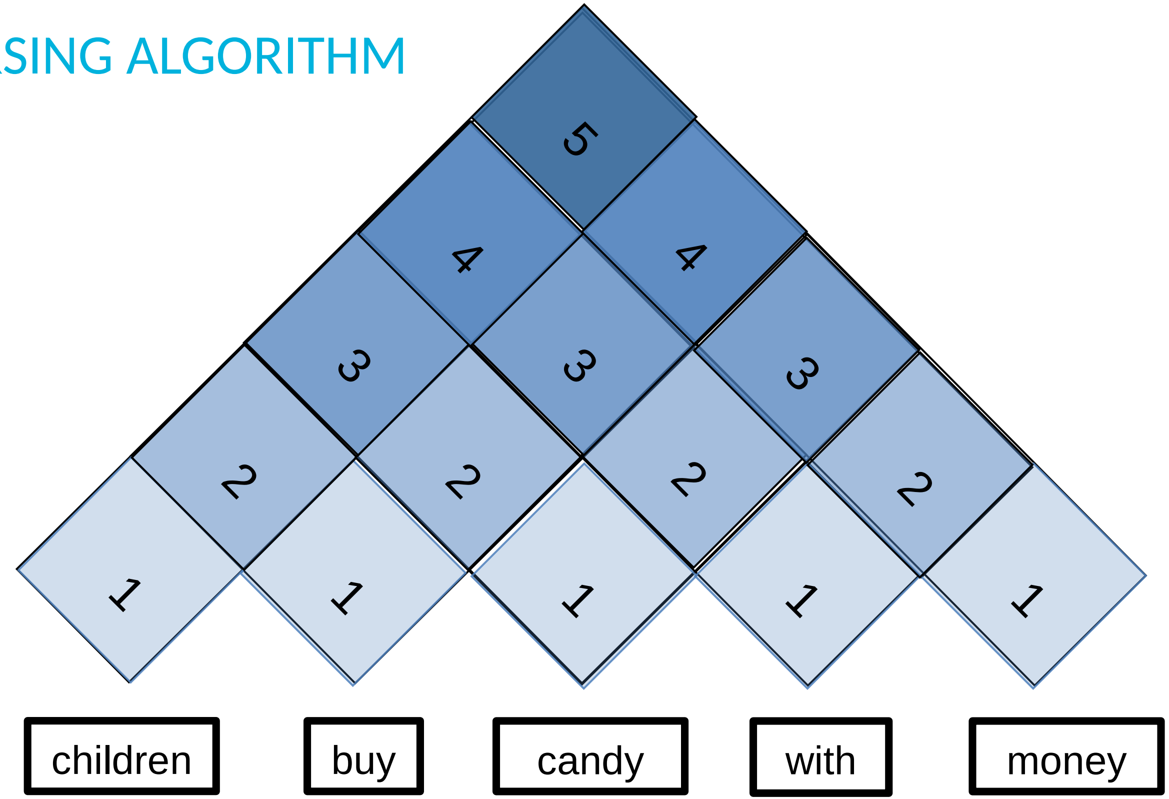
WARWICK



CKY PARSING ALGORITHM

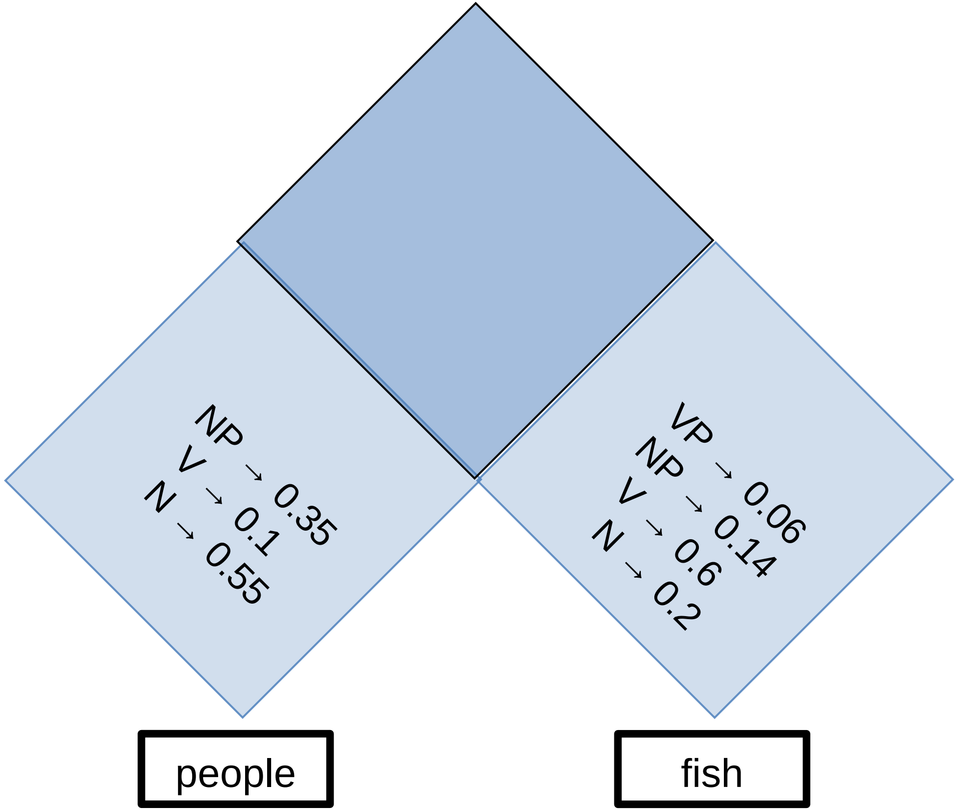


CKY PARSING ALGORITHM



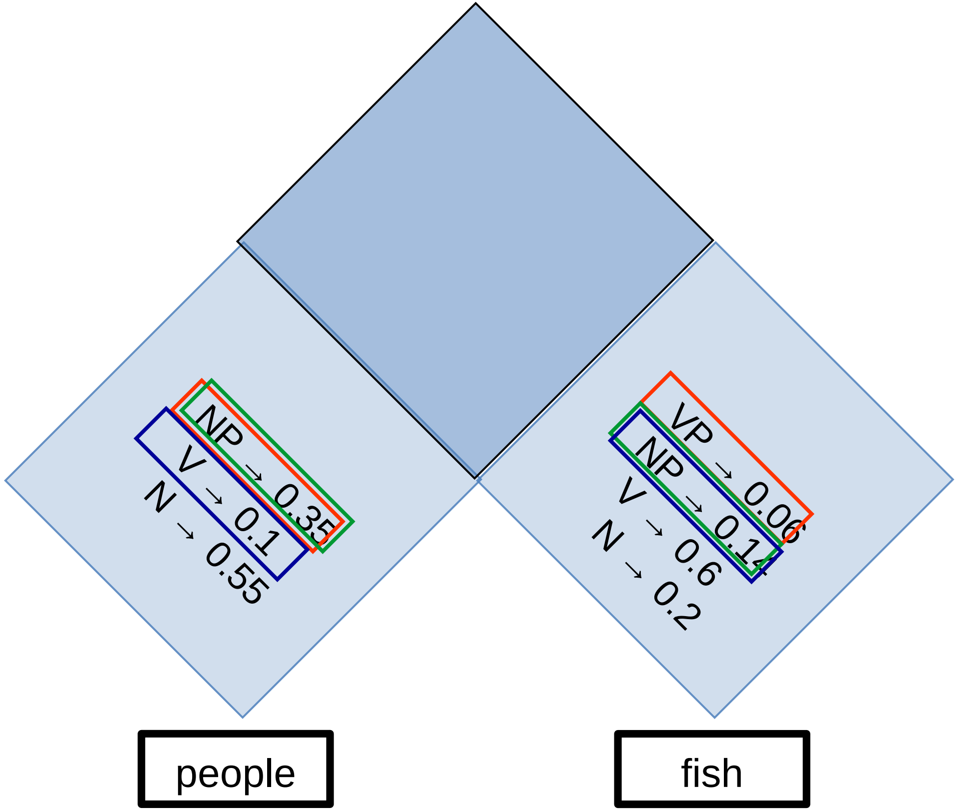
WARWICK

CKY PARSING ALGORITHM



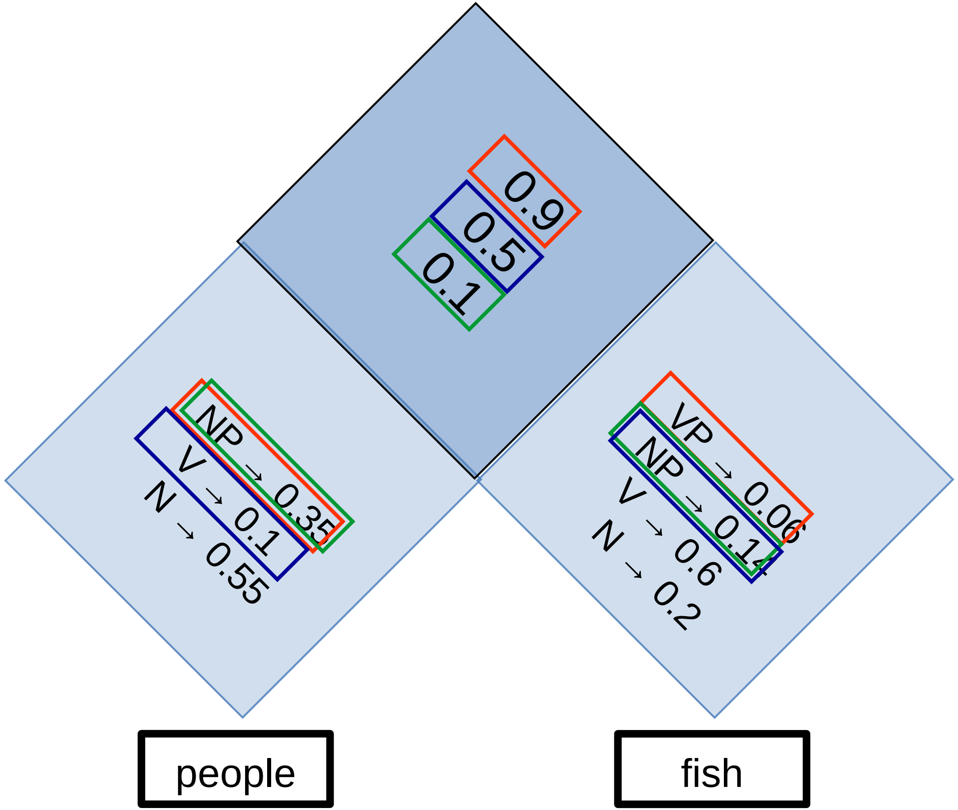
S → NP VP 0.9
S → VP 0.1
VP → V NP 0.5
VP → V 0.1
VP → V @VP_V 0.3
VP → V PP 0.1
@VP_V → NP PP 1.0
NP → NP NP 0.1
NP → NP PP 0.2
NP → N 0.7
PP → P NP 1.0

CKY PARSING ALGORITHM



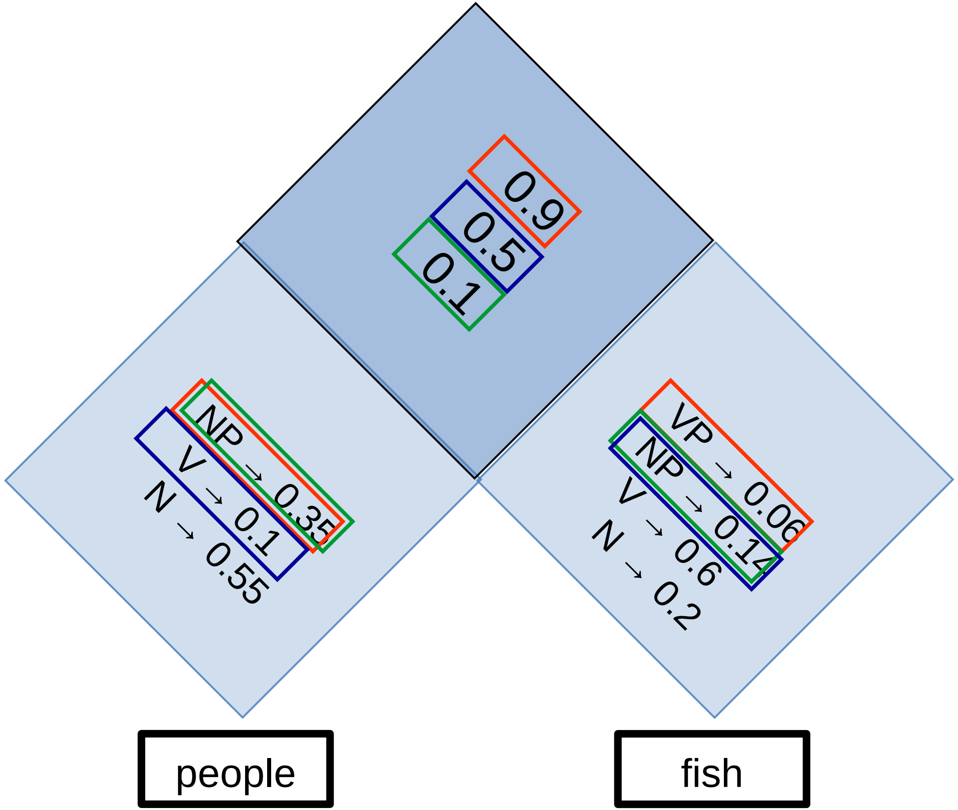
S	→	NP VP	0.9
S	→	VP	0.1
VP	→	V NP	0.5
VP	→	V	0.1
VP	→	V @VP_V	0.3
VP	→	V PP	0.1
@VP_V	→	NP PP	1.0
NP	→	NP NP	0.1
NP	→	NP PP	0.2
NP	→	N	0.7
PP	→	P NP	1.0

CKY PARSING ALGORITHM



S → NP VP	0.9
S → VP	0.1
VP → V NP	0.5
VP → V	0.1
VP → V @VP_V	0.3
VP → V PP	0.1
@VP_V → NP PP	1.0
NP → NP NP	0.1
NP → NP PP	0.2
NP → N	0.7
PP → P NP	1.0

CKY PARSING ALGORITHM

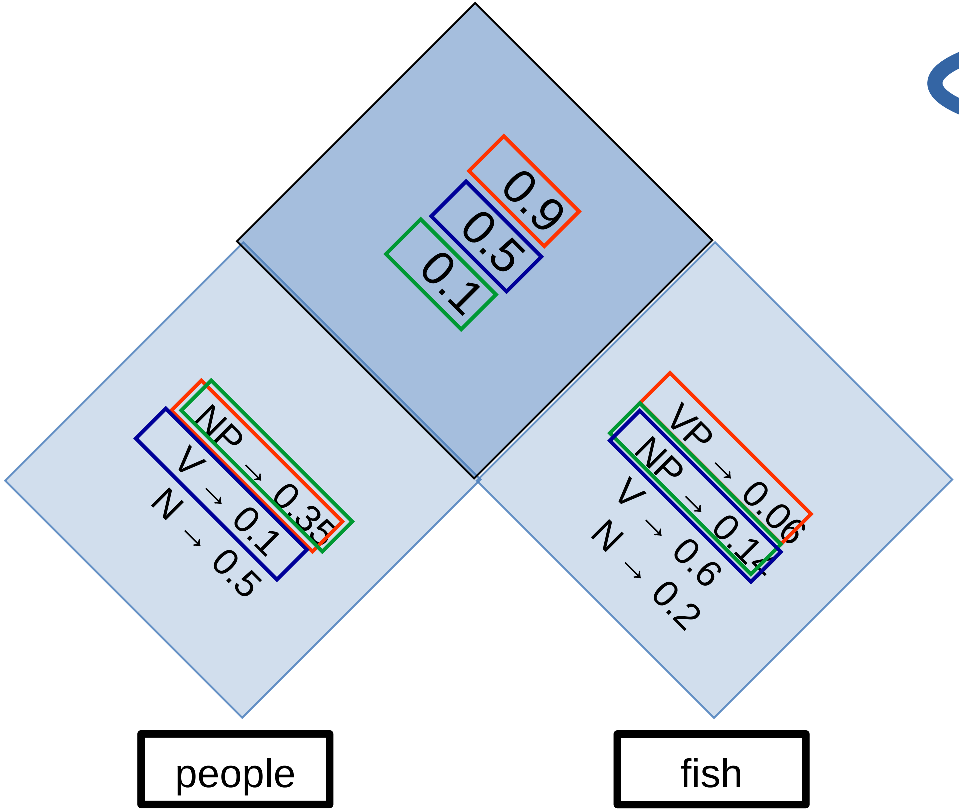


0.9*0.06*0.35 = 0.0189

0.5*0.14*0.1 = 0.007

0.1*0.14*0.35 = 0.049

CKY PARSING ALGORITHM



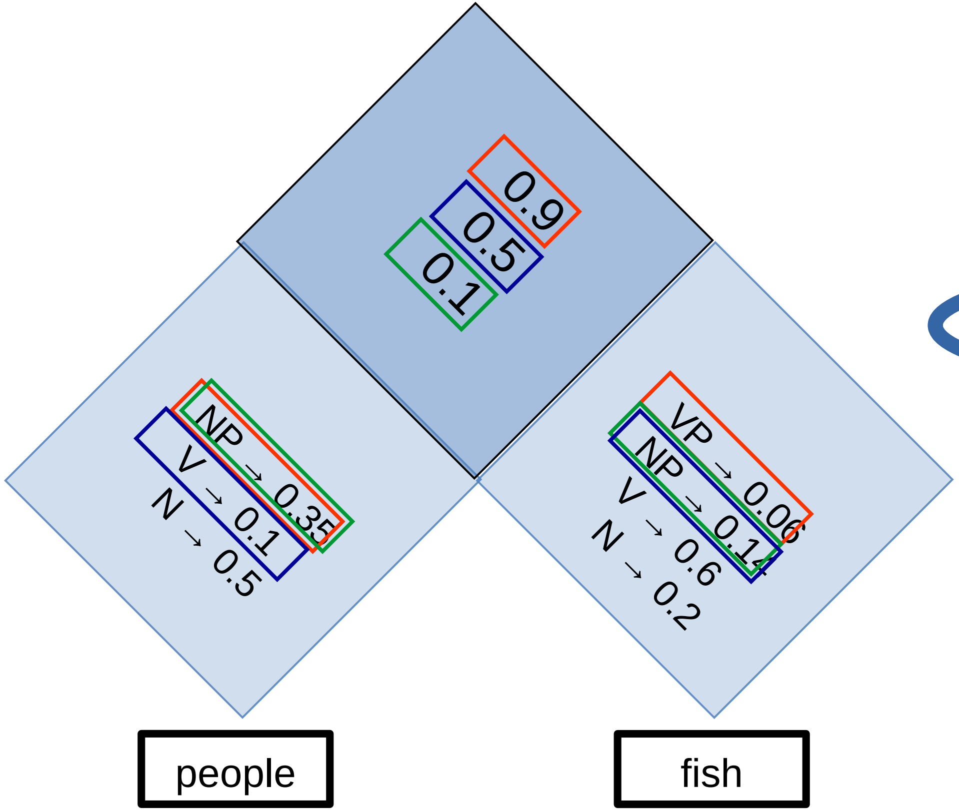
0.9*0.06*0.35 = 0.0189

0.5*0.14*0.1 = 0.007

0.1*0.14*0.35 = 0.049

The Viterbi (maximum) score determines the predicted parsed tree via CKY.

CKY PARSING ALGORITHM



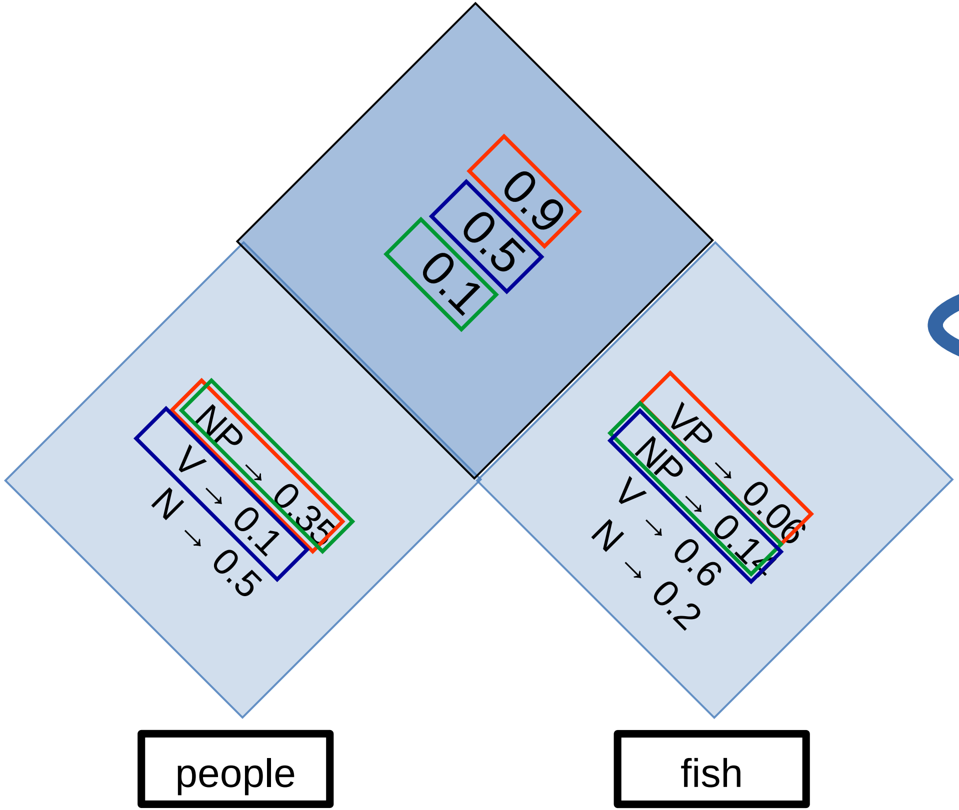
0.9*0.06*0.35 = 0.0189

0.5*0.14*0.1 = 0.007

0.1*0.14*0.35 = 0.049

It's larger, but doesn't belong to a "S → ..." rule (i.e. whole sentence)

CKY PARSING ALGORITHM



0.9*0.06*0.35 = 0.0189

0.5*0.14*0.1 = 0.007

0.1*0.14*0.35 = 0.049

If there were more levels (upwards) in the tree, then we'd choose this one, and carry on.

LEXICALISED PARSING

LEXICALISATION OF PCFGs

WARWICK

- So far we have **only considered the probabilities of POS-based rules**, e.g. P(JJ NP) → 0.3
- We can **"lexicalise" that, by considering the probabilities of words** occurring in different constituents, e.g.:
  - "money": noun (most common, P=0.9) or adjective (P=0.1).
  - "money laundering": can be JJ NP or NP NP.
    - However, "money" more likely to occur in NP NP than in JJ NP → then increases P(NP NP).

LEXICALISATION OF PCFGs

- Probability of different verbal complement frames (i.e., "subcategorisations") depends on the verb:

Local Tree	***come***	***take***	***think***	***want***
VP → V	9.5%	2.6%	4.6%	5.7%
VP → V NP	1.1%	32.1%	0.2%	13.9%
VP → V PP	34.5%	3.1%	7.1%	0.3%
VP → V SBAR	6.6%	0.3%	73.0%	0.2%
VP → V S	2.2%	1.3%	4.8%	70.8%
VP → V NP S	0.1%	5.7%	0.0%	0.3%
VP → V PRT NP	0.3%	5.8%	0.0%	0.0%
VP → V PRT PP	6.1%	1.5%	0.2%	0.0%

EXAMPLE OF PCFG

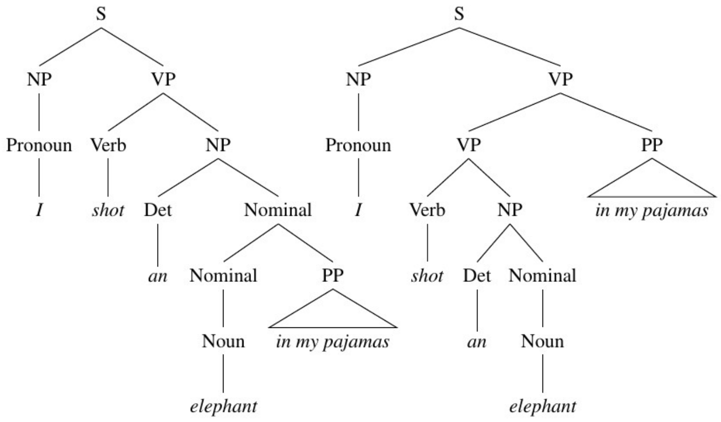
WARWICK

Grammar		Lexicon
$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.10] | a [.30] | the [.60]
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$ [.10] | $flight$ [.30]
$S \rightarrow VP$	[.05]	| $meal$ [.015] | $money$ [.05]
$NP \rightarrow Pronoun$	[.35]	| $flight$ [.40] | $dinner$ [.10]
$NP \rightarrow Proper\text{-}Noun$	[.30]	$Verb \rightarrow book$ [.30] | $include$ [.30]
$NP \rightarrow Det\ Nominal$	[.20]	| $prefer$ [.40]
$NP \rightarrow Nominal$	[.15]	$Pronoun \rightarrow I$ [.40] | she [.05]
$Nominal \rightarrow Noun$	[.75]	| me [.15] | you [.40]
$Nominal \rightarrow Nominal\ Noun$	[.20]	$Proper\text{-}Noun \rightarrow Houston$ [.60]
$Nominal \rightarrow Nominal\ PP$	[.05]	| NWA [.40]
$VP \rightarrow Verb$	[.35]	$Aux \rightarrow does$ [.60] | can [40]
$VP \rightarrow Verb\ NP$	[.20]	$Preposition \rightarrow from$ [.30] | to [.30]
$VP \rightarrow Verb\ NP\ PP$	[.10]	| on [.20] | $near$ [.15]
$VP \rightarrow Verb\ PP$	[.15]	| $through$ [.05]
$VP \rightarrow Verb\ NP\ NP$	[.05]	
$VP \rightarrow VP\ PP$	[.15]	
$PP \rightarrow Preposition\ NP$	[1.0]	

HELP WITH DISAMBIGUATION

- Lexicalised PCFGs can deal with disambiguation better:

  I shot an elephant in my pijamas.



  P("shot an elephant")
  >> P("elephant in my pijamas")

WARWICK

- Lexicalised PCFGs and language models can be linked:
  - We can learn language models informed by syntactic structure, e.g. "flying planes" as VB NN or JJ NN.

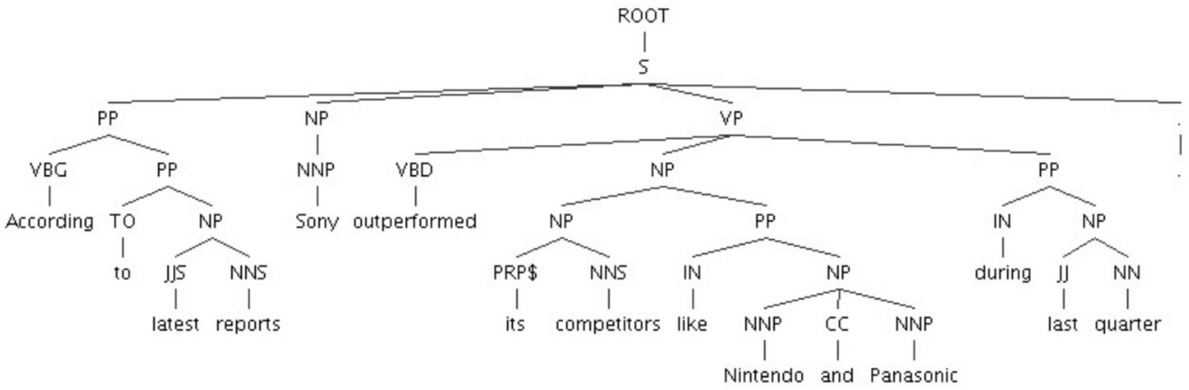
    piloting planes

    planes that fly

  - Language models can assist decision-making in PCFGs.

CONSTITUENT TREE

- Trees can be more complex.
- CKY: break it down, bottom-up; POS tagging for leaves, then go up.



According to latest reports Sony outperformed its competitors like Nintendo and Panasonic during last quarter.

PENN TREEBANK NON-TERMINALS

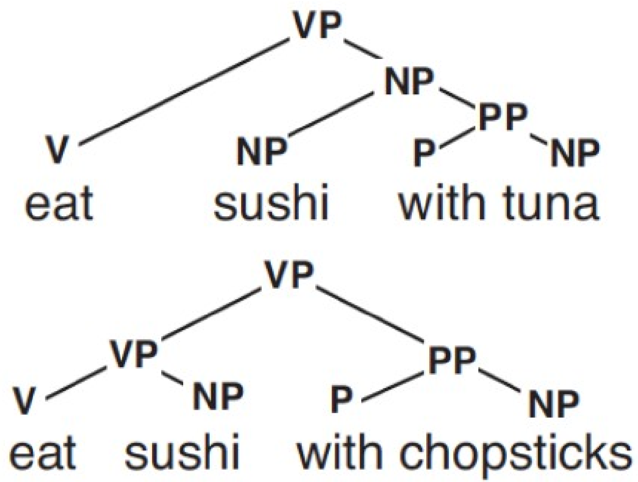
WARWICK

S	Sentence or clause.
SBAR	Clause introduced by a (possibly empty) subordinating conjunction.
SBARQ	Direct question introduced by a *wh*-word or *wh*-phrase.
SINV	Inverted declarative sentence.
SQ	Inverted yes/no question, or main clause of a *wh*-question.
ADJP	Adjective Phrase.
ADVP	Adverb Phrase.
CONJP	Conjunction Phrase.
FRAG	Fragment.
INTJ	Interjection.
LST	List marker. Includes surrounding punctuation.
NAC	Not A Constituent; used within an NP.
NP	Noun Phrase.
NX	Used within certain complex NPs to mark the head.

PP	Prepositional Phrase.
PRN	Parenthetical.
PRT	Particle.
QP	Quantity Phrase (i.e., complex measure/amount) within NP.
RRC	Reduced Relative Clause.
UCP	Unlike Coordinated Phrase.
VP	Verb Phrase.
WHADJP	*Wh*-adjective Phrase, as in *how hot*.
WHADVP	*Wh*-adverb Phrase.
WHNP	*Wh*-noun Phrase, e.g. *who*, *which book*, *whose daughter*, *none of which*, or *how many leopards*.
WHPP	*Wh*-prepositional Phrase, e.g., *of which* or *by whose authority*.
X	Unknown, uncertain, or unbracketable.

STATE-OF-THE-ART

- Dependent on entire tree, e.g. “eat sushi” is different in the below examples.



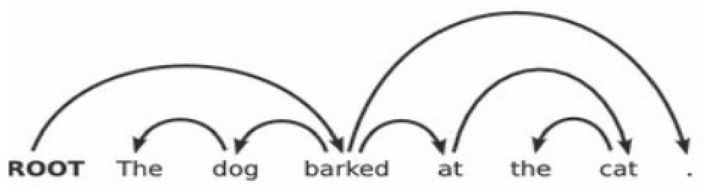
STATE-OF-THE-ART

- PCFG achieves ~73% on Penn TreeBank.
- State-of-the art ~92%: Lexicalised PCFG.

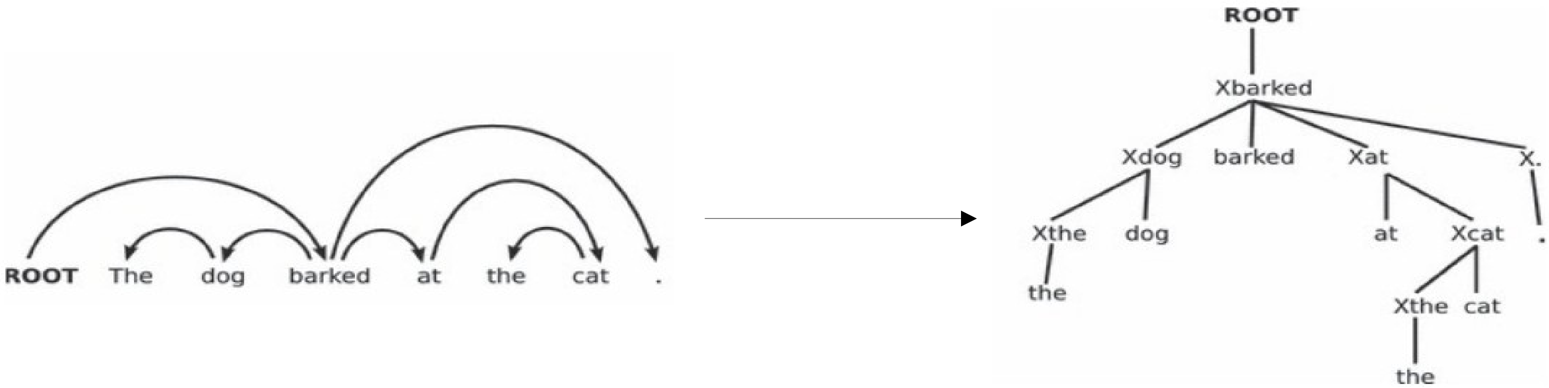
DEPENDENCY PARSING

DEPENDENCY PARSING

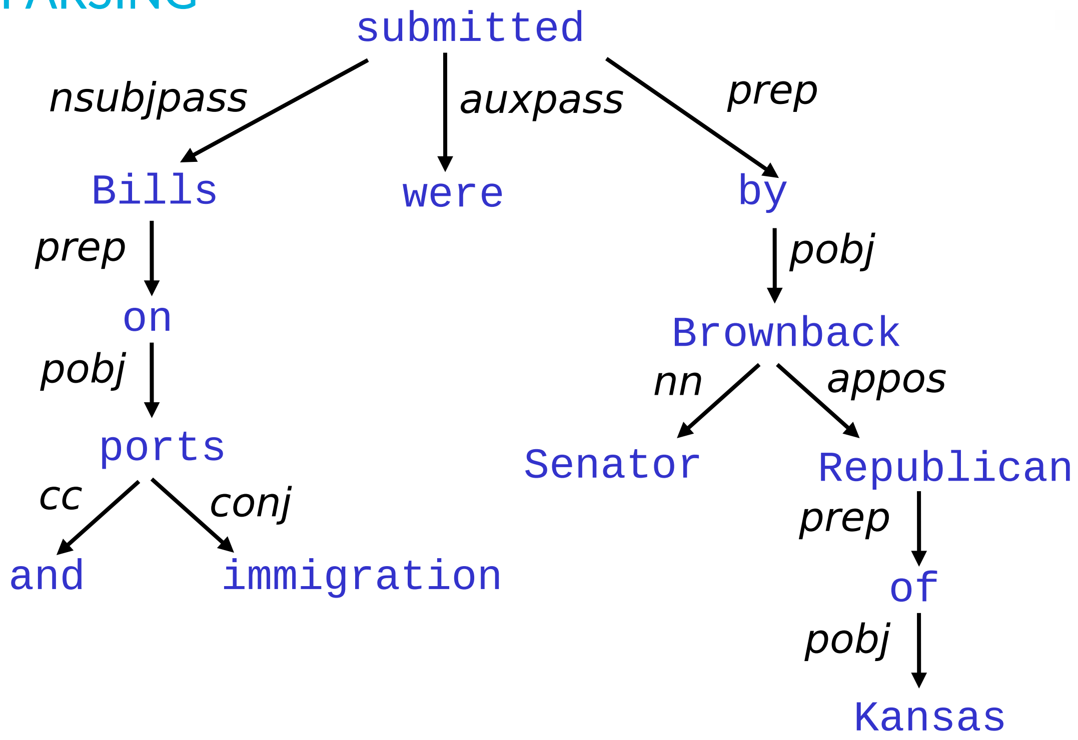
- Dependency syntax postulates that syntactic structure consists of **lexical items linked by binary asymmetric relations** (“arrows”) called dependencies.



DEPENDENCY PARSING



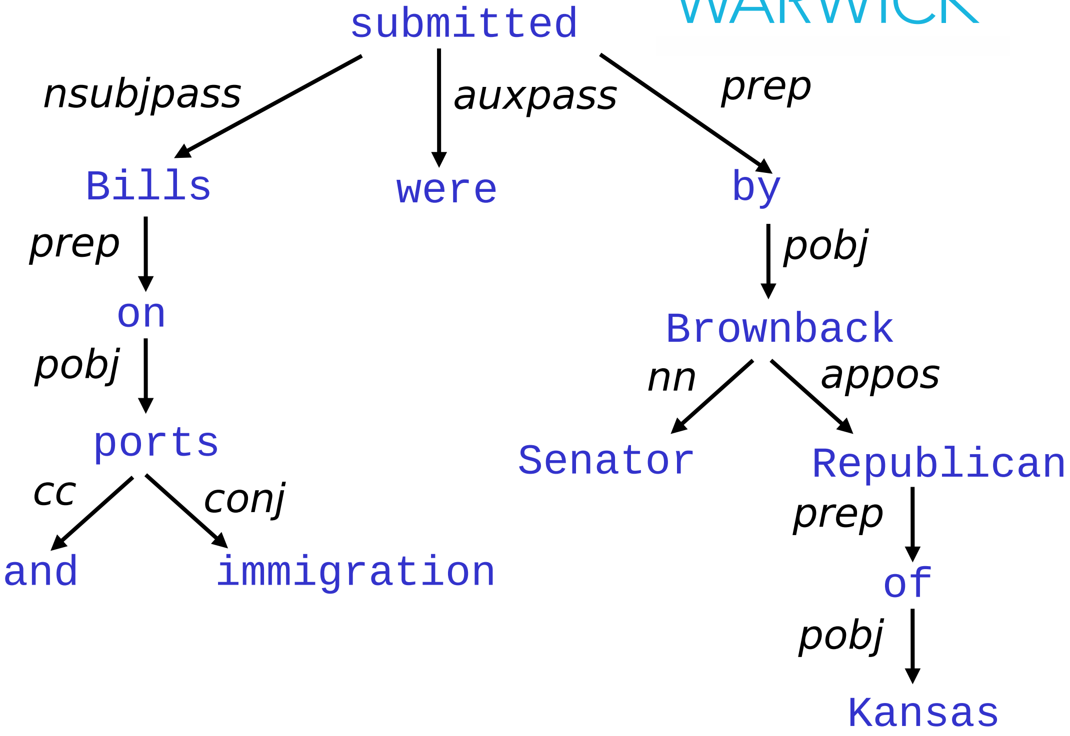
DEPENDENCY PARSING



Bills on ports and immigration were submitted by Brownback Senator Republican of Kansas

DEPENDENCY PARSING

- The arrow connects a **head** (governor, superior, regent) with a **dependent** (modifier, inferior, subordinate)



- How do we decide which one's the head?
- Usually define heads in PCFG, e.g.:
  - S → <u>NP</u> VP
  - VP → <u>VBD</u> NP PP
  - NP → DT JJ <u>NN</u>

- **Graph Algorithms:**
  - Consider all word pairs.
  - Create a Maximum Spanning Tree for a sentence.
- **Transition-based Approaches:**
  - Similar to how we parse a program:
  - Shift-Reduce Parser: MaltParser.

MALTPARSER (Nivre et al. 2008)

- Simple form of greedy discriminative dependency parser.
- The parser does a sequence of bottom up actions.
- The parser has:
  - a stack σ, written with top to the right
    - starts with the ROOT symbol
  - a buffer β, written with top to the left
    - starts with the input sentence
  - a set of dependency arcs A, which starts off empty
  - a set of actions

MALTPARSER (Nivre et al. 2008)

WARWICK

Start: σ = [ROOT], $\beta = w_1, \ldots, w_n$, $A = \varnothing$

1. Left-Arc$_r$ $\sigma|w_i, w_j|\beta, A \rightarrow \sigma, w_j|\beta, A \cup \{r(w_j,w_i)\}$

   Precondition: $r'(w_k, w_i) \notin A$, $w_i \neq$ ROOT

2. Right-Arc$_r$ $\sigma|wi, wj|\beta, A \rightarrow \sigma|w_i|w_j, \beta, A \cup \{r(w_i,w_j)\}$
3. Reduce $\sigma|w_i, \beta, A \rightarrow \sigma, \beta, A$
   1. Precondition: $r'(w_k, w_i) \in A$
4. Shift $\sigma, w_i|\beta, A \rightarrow \sigma|w_i, \beta, A$

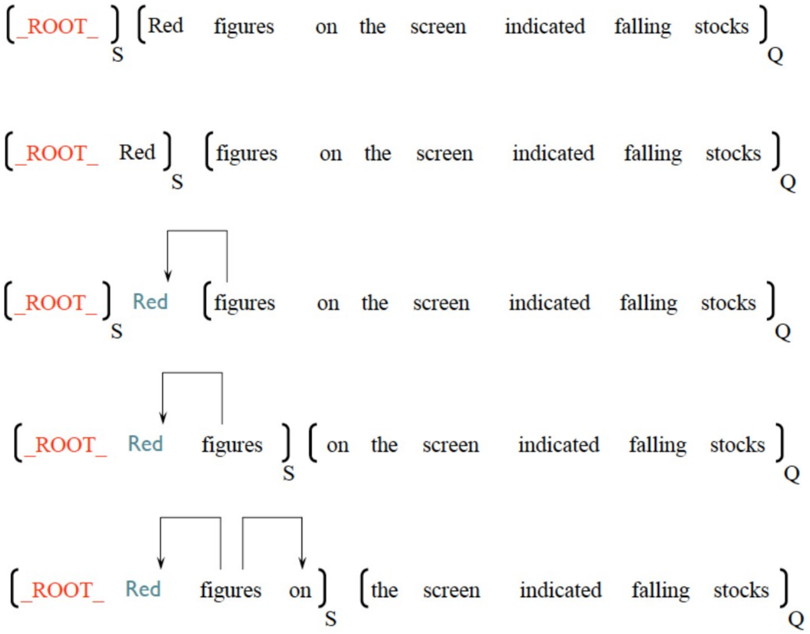
Finish: $\beta = \varnothing$

MALTPARSER

1. Left-Arc$_r$ $\sigma|w_i, w_j|\beta, A \Rightarrow \sigma, w_j|\beta, A\cup\{r(w_j,w_i)\}$
   Precondition: $(w_k, r', w_i) \notin A$, $w_i \neq$ ROOT
2. Right-Arc$_r$ $\sigma|w_i, w_j|\beta, A \Rightarrow \sigma|w_i|w_j, \beta, A\cup\{r(w_i,w_j)\}$
3. Reduce $\sigma|w_i, \beta, A \Rightarrow \sigma, \beta, A$
   Precondition: $(w_k, r', w_i) \in A$
4. Shift $\sigma, w_i|\beta, A \Rightarrow \sigma|w_i, \beta, A$

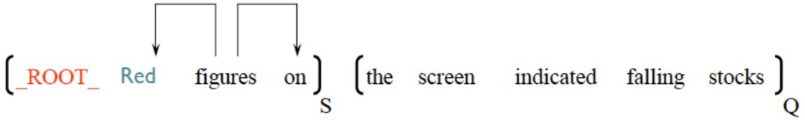
WARWICK

- 1. Shift:
- 2. Left-arc:
- 3. Shift:
- 4. Right-arc:

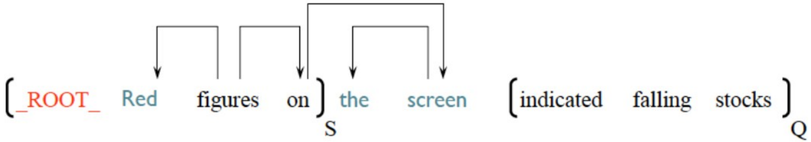


MALTPARSER

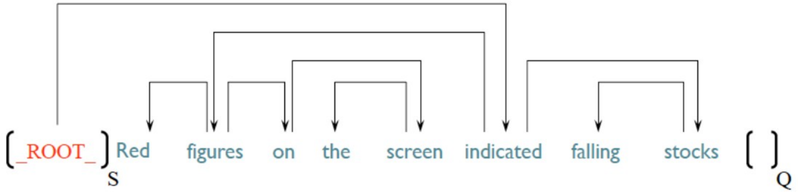
1. Left-Arc$_r$ $\sigma|w_i, w_j|\beta, A \Rightarrow \sigma, w_j|\beta, A\cup\{r(w_j,w_i)\}$
   Precondition: $(w_k, r', w_i) \notin A$, $w_i \neq$ ROOT
2. Right-Arc$_r$ $\sigma|w_i, w_j|\beta, A \Rightarrow \sigma|w_i|w_j, \beta, A\cup\{r(w_i,w_j)\}$
3. Reduce $\sigma|w_i, \beta, A \Rightarrow \sigma, \beta, A$
   Precondition: $(w_k, r', w_i) \in A$
4. Shift $\sigma, w_i|\beta, A \Rightarrow \sigma|w_i, \beta, A$

- 4. Right-arc: 

  …

  8. Reduce: 

  …

  16. Reduce: 

RESOURCES

- There is a long list of Treebanks available for a wide range of languages, a good list can be found here:

  https://en.wikipedia.org/wiki/Treebank

ASSOCIATED READING

WARWICK

- Jurafsky, Daniel, and James H. Martin. 2009. Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. 3rd edition. **Chapters 10-13.**