Advanced Artificial Intelligence

# Part II. Statistical NLP

***Introduction and Grammar Models***

**Wolfram Burgard, Luc De Raedt, Bernhard Nebel, Lars Schmidt-Thieme**

Some slides taken from Helmut Schmid, Rada Mihalcea, Bonnie Dorr, Leila Kosseim, Peter Flach and others

# Contents

- Some natural language processing tasks
- Non-probabilistic NLP models
  - Regular grammars and finite state automata
  - Context-Free Grammars
  - Definite Clause Grammars
- Motivation for statistical NLP
- Overview of the rest of this part

# Language and sequences

- Natural language processing
  - Is concerned with the analysis of sequences of words / sentences
  - Construction of language models
- Two types of models
  - Non-probabilistic
  - Probabilistic

# Key NLP Problem: Ambiguity

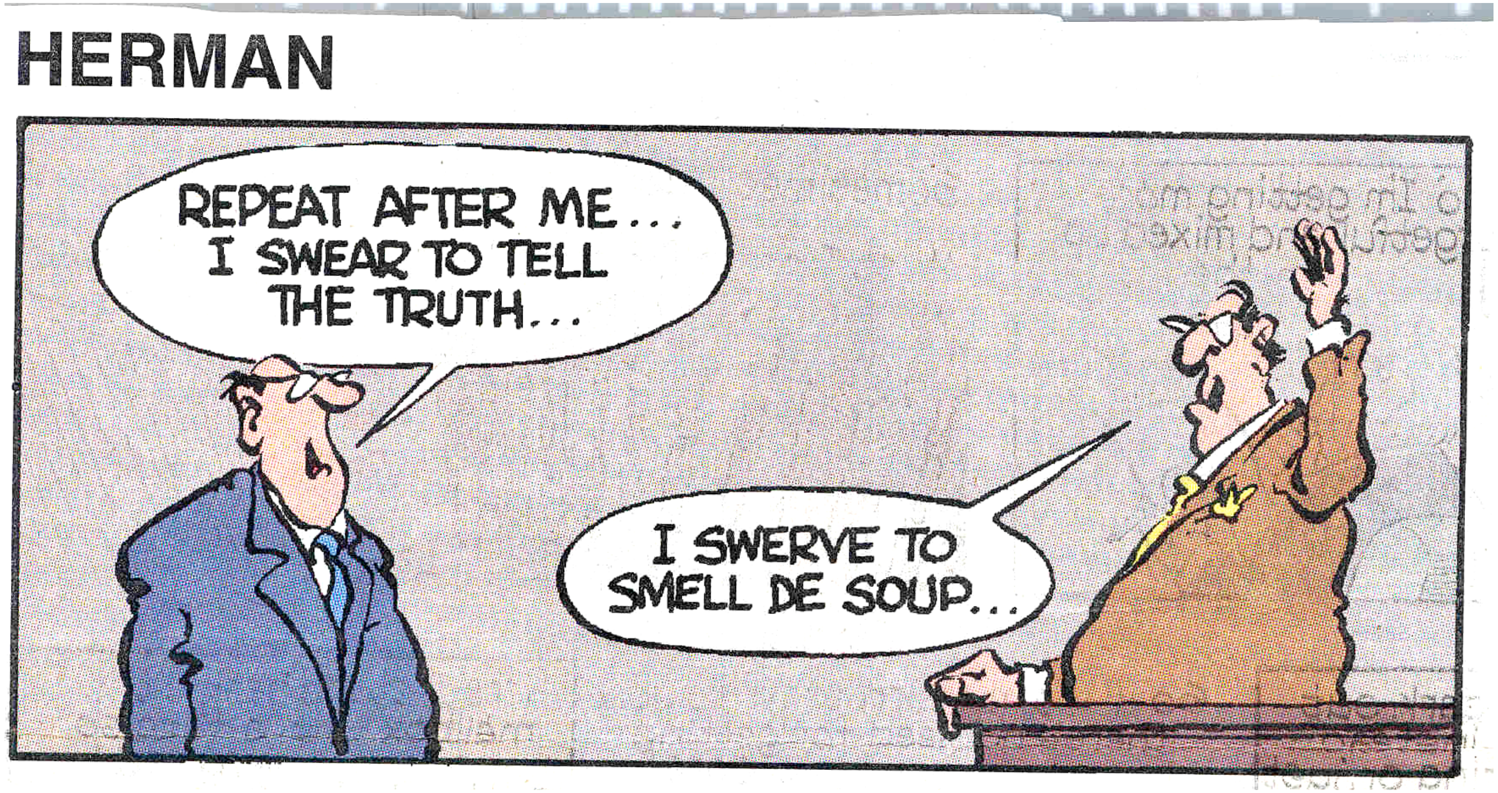
Human Language is highly ambiguous at all levels

- acoustic level
  *recognize speech   vs.   wreck a nice beach*
- morphological level
  *saw:  to see (past), saw (noun), to saw (present, inf)*
- syntactic level
  *I saw the man on the hill with a telescope*
- semantic level
  *One book has to be read by every student*

# Language Model

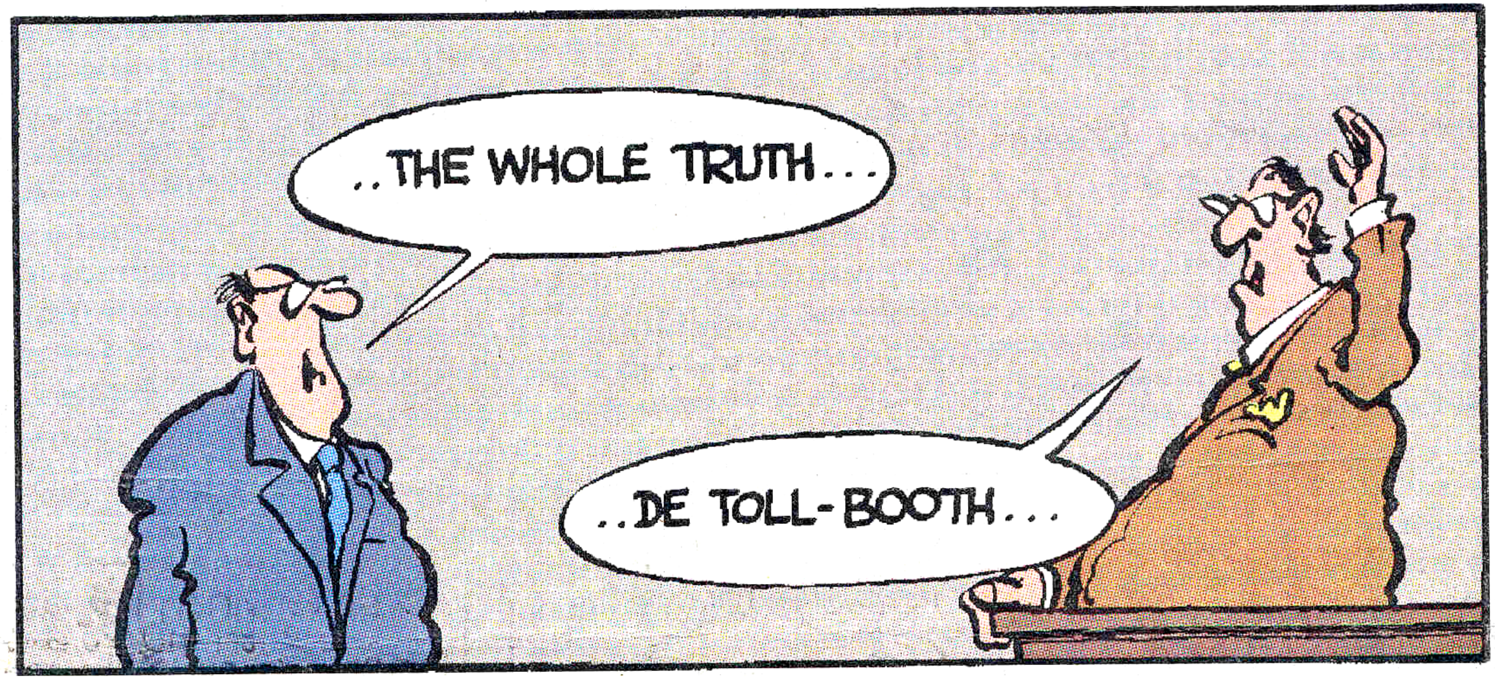
- A formal model about language
- Two types
  - Non-probabilistic
    - Allows one to compute whether a certain sequence (sentence or part thereof) is possible
    - Often grammar based
  - Probabilistic
    - Allows one to compute the probability of a certain sequence
    - Often extends grammars with probabilities

# Example of bad language model

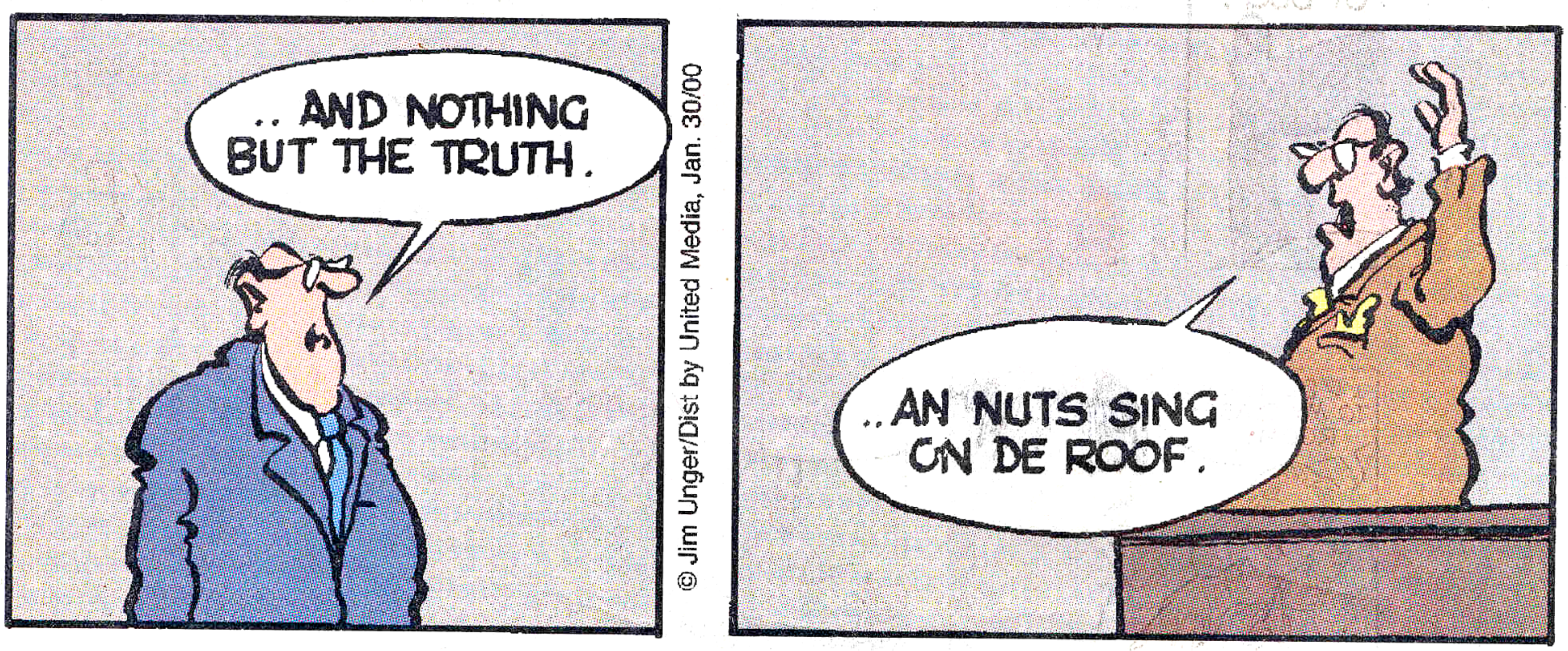


# A bad language model

by Jim Unger



# A bad language model



# A good language model

- Non-Probabilistic
  - “I swear to tell the truth” is possible
  - “I swerve to smell de soup” is impossible
- Probabilistic
  - P(I swear to tell the truth) ~ .0001
  - P(I swerve to smell de soup) ~ 0

# Why language models ?

- Consider a Shannon Game
  - Predicting the next word in the sequence
    - Statistical natural language ….
    - The cat is thrown out of the …
    - The large green …
    - Sue swallowed the large green …
    - …
- Model at the sentence level

# Applications

- Spelling correction
- Mobile phone texting
- Speech recognition
- Handwriting recognition
- Disabled users
- …

# Spelling errors

- They are leaving in about fifteen *minuets* to go to her house.
- The study was conducted mainly *be* John Black.
- Hopefully, all *with* continue smoothly in my absence.
- Can they *lave* him my messages?
- I need to *notified* the bank of….
- He is trying to *fine* out.

# Handwriting recognition

- Assume a note is given to a bank teller, which the teller reads as I have a gub. (cf. Woody Allen)
- NLP to the rescue ….
  - gub is not a word
  - gun, gum, Gus, and gull are words, but gun has a higher probability in the context of a bank

# For Spell Checkers

- Collect list of commonly substituted words
  - piece/peace, whether/weather, their/there ...

- Example:
  "On Tuesday, the whether …’’
  "On Tuesday, the weather …"

# Another dimension in language models

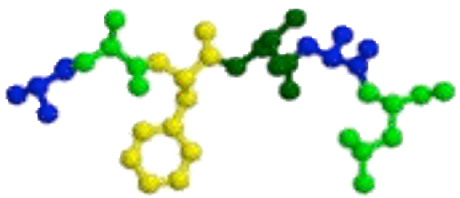
- Do we mainly want to infer (probabilities) of legal sentences / sequences ?
  - So far
- Or, do we want to infer properties of these sentences ?
  - E.g., parse tree, part-of-speech-tagging
  - Needed for understanding NL
- Let's look at some tasks

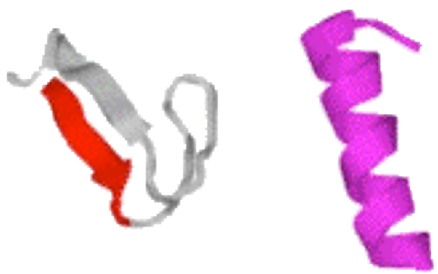
# Sequence Tagging

- Part-of-speech tagging
  - *He drives with his bike*
  - *N V PR PN N*
    `noun, verb, preposition, pronoun, noun`
- Text extraction
  - The job is that of a programmer
  - X X X X X X JobType
  - The seminar is taking place from 15.00 to 16.00
  - X X X X X X Start End

# Sequence Tagging

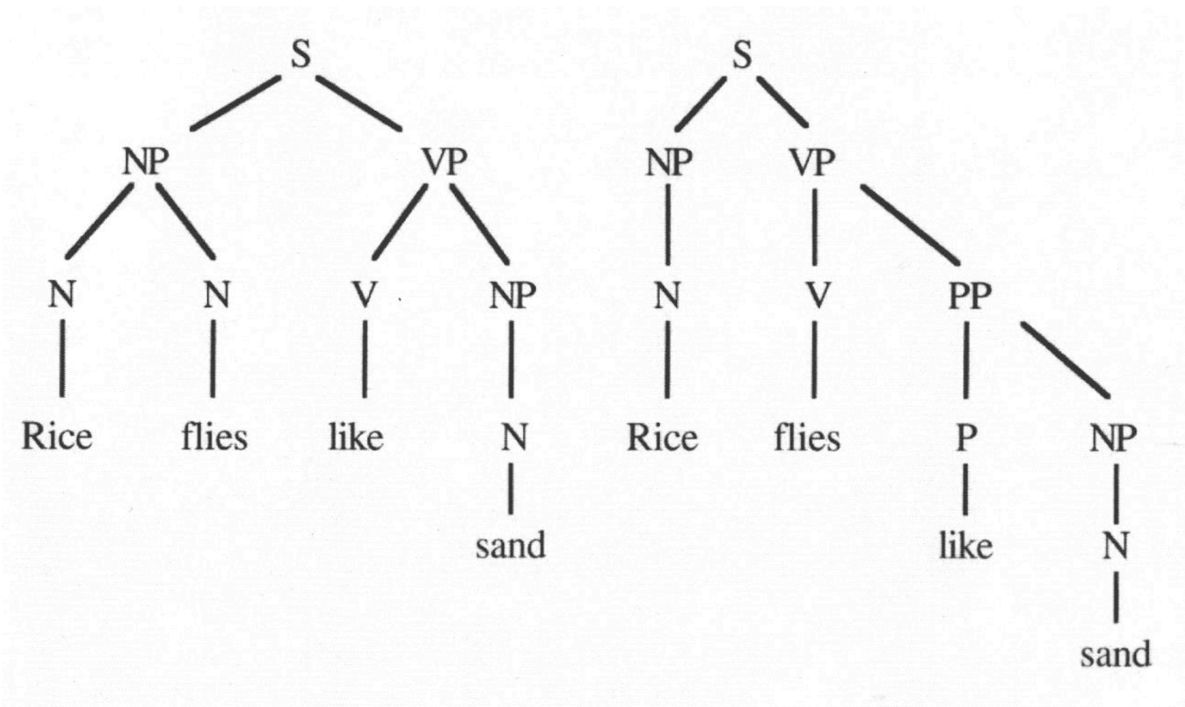
- Predicting the secondary structure of proteins, mRNA, …
  - *X* = `A,F,A,R,L,M,M,A,…`
  - *Y* = `he,he,st,st,st,he,st,he, …`





# Parsing

- Given a sentence, find its parse tree
- Important step in understanding NL



# Parsing

- In bioinformatics, allows to predict (elements of) structure from sequence

# Language models based on Grammars

- Grammar Types
  - Regular grammars and Finite State Automata
  - Context-Free Grammars
  - Definite Clause Grammars
    - A particular type of Unification Based Grammar (Prolog)
- Distinguish lexicon from grammar
  - Lexicon (dictionary) contains information about words, e.g.
    - word - possible tags (and possibly additional information)
    - flies - V(erb) - N(oun)
  - Grammar encode rules

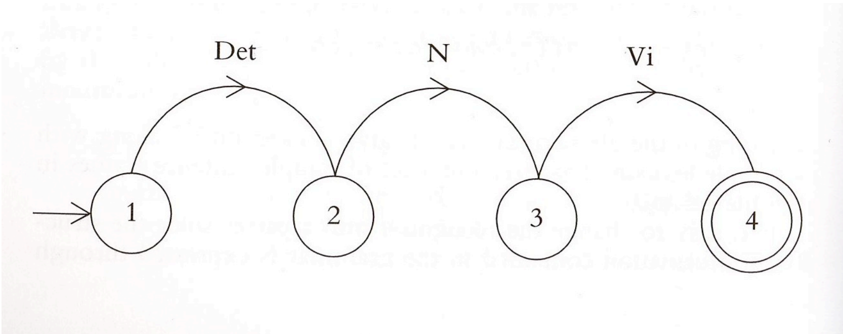
# Grammars and parsing

- Syntactic level best understood and formalized
- Derivation of grammatical structure: *parsing* (more than just *recognition*)
- Result of parsing mostly *parse tree*: showing the *constituents* of a sentence, e.g. verb or noun phrases
- Syntax usually specified in terms of a *grammar* consisting of *grammar rules*

# Regular Grammars and Finite State Automata

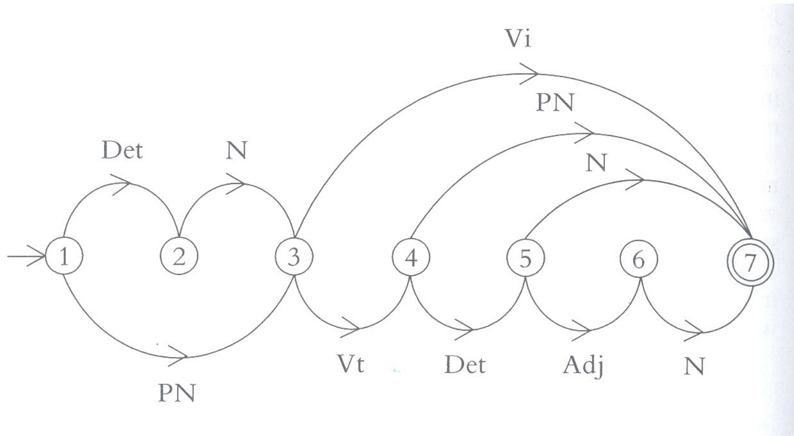
- Lexical information - which words are ?
  - Det(erminer)
  - N(oun)
  - Vi (intransitive verb) - no argument
  - Pn (pronoun)
  - Vt (transitive verb) - takes an argument
  - Adj (adjective)
- Now accept
  - The cat slept
  - Det N Vi

- As regular grammar
  - S -> [Det] S1     *% [ ] : terminal*
  - S1 -> [N] S2
  - S2 -> [Vi]
- Lexicon
  - The - Det
  - Cat - N
  - Slept - Vi
  - …

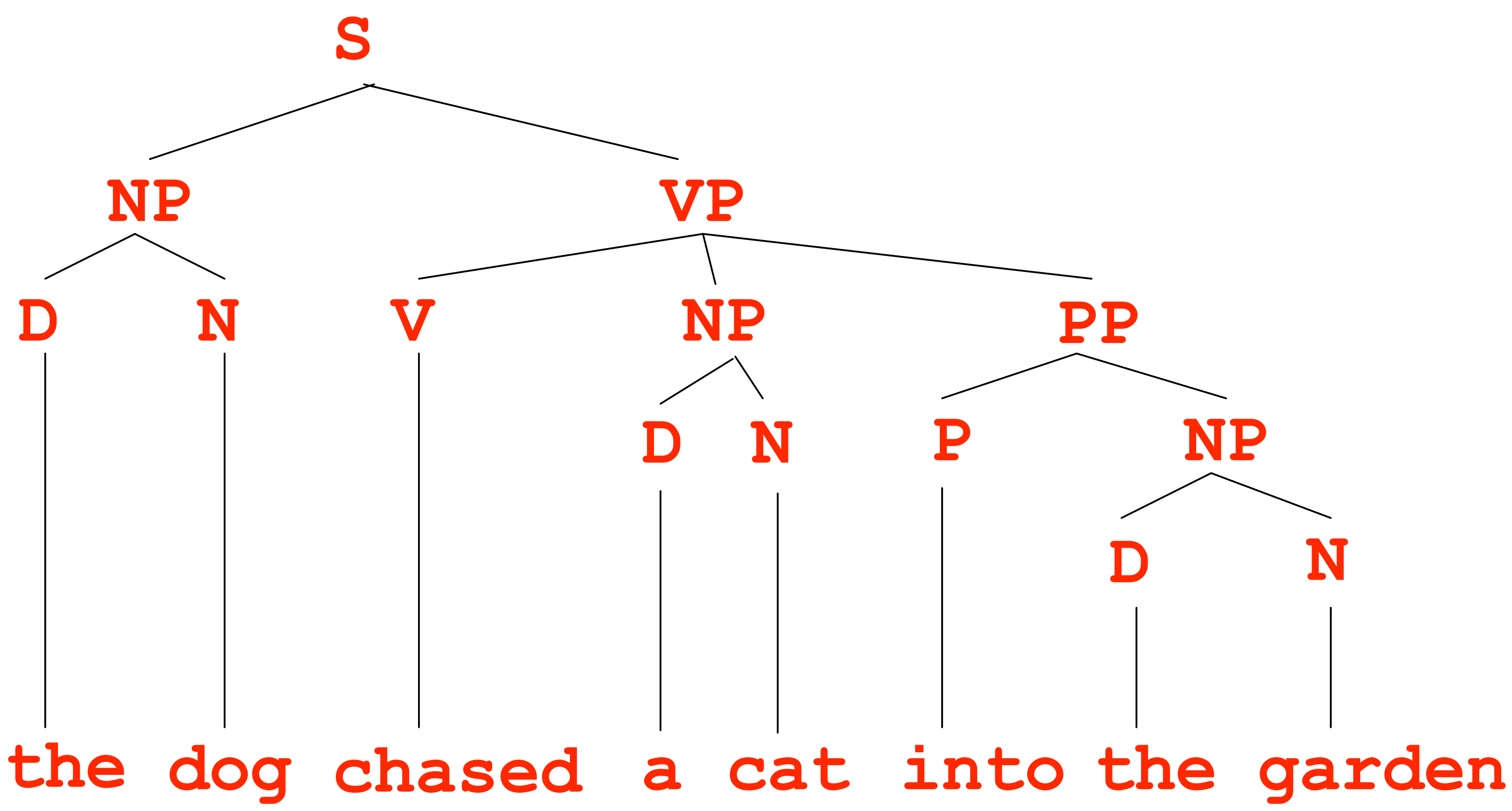


# Finite State Automaton

- Sentences
  - John smiles - Pn Vi
  - The cat disappeared  - Det N Vi
  - These new shoes hurt - Det Adj N Vi
  - John liked the old cat PN Vt Det Adj N



# Phrase structure



# Notation

- `S`: sentence
- `D` or `Det`: Determiner (e.g., articles)
- `N`: noun
- `V`: verb
- `P`: preposition
- `NP`: noun phrase
- `VP`: verb phrase
- `PP`: prepositional phrase

# Context Free Grammar

```
S     -> NP VP
NP    -> D N
VP    -> V NP
VP    -> V NP PP
PP    -> P NP
D     -> [the]
D     -> [a]
N     -> [dog]
N     -> [cat]
N     -> [garden]
V     -> [chased]
V     -> [saw]
P     -> [into]
```

Terminals ~ Lexicon

# Phrase structure

- Formalism of context-free grammars
  - Nonterminal symbols: S, NP, VP, ...
  - Terminal symbols: dog, cat, saw, the, ...
- Recursion
  - „The girl thought the dog chased the cat“

```
VP    -> V, S
N     -> [girl]
V     -> [thought]
```

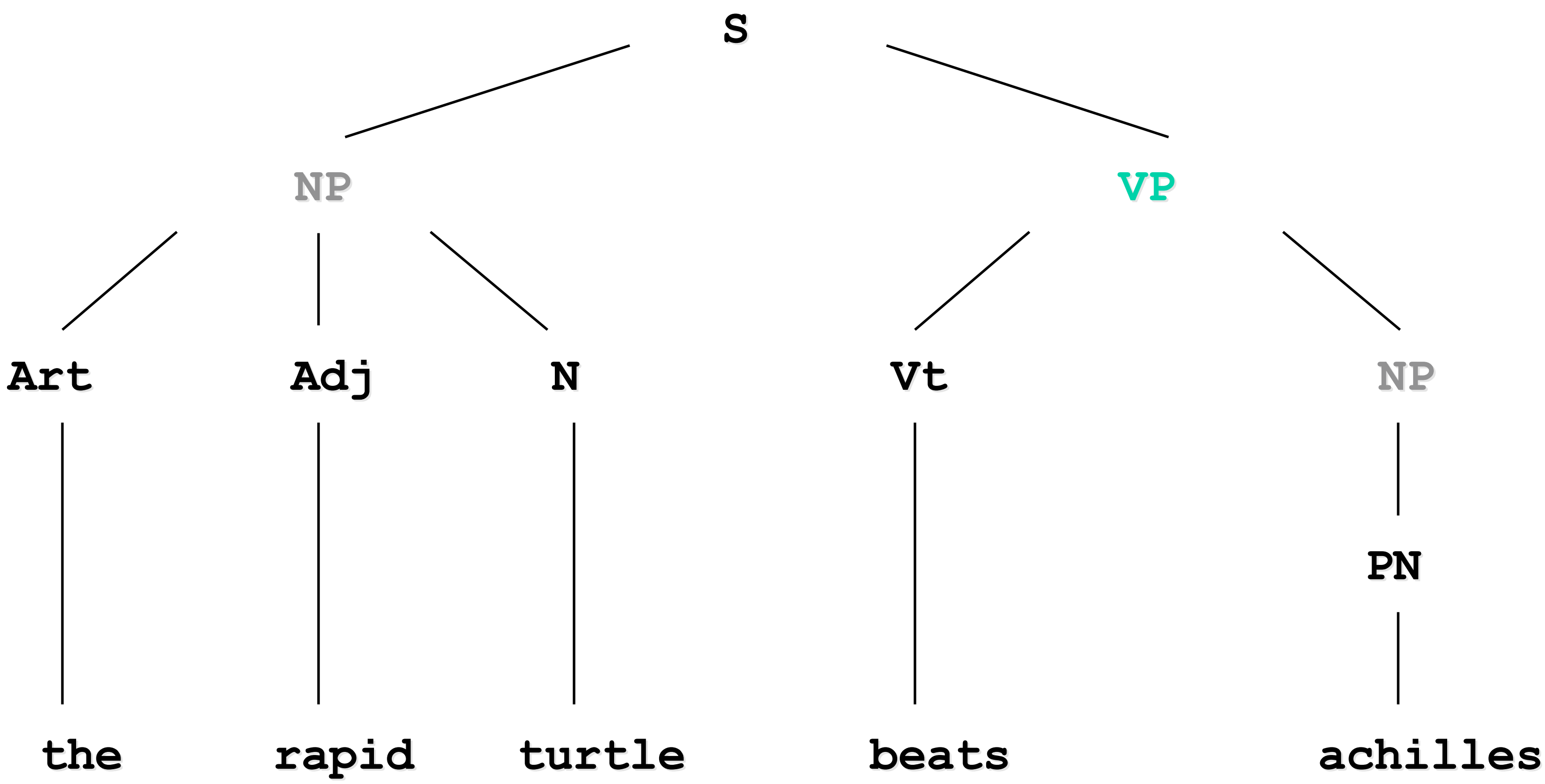
# Top-down parsing

- S -> NP VP
- S -> Det N VP
- S -> The N VP
- S -> The dog VP
- S -> The dog V NP
- S -> The dog chased NP
- S -> The dog chased Det N
- S-> The dog chased the N
- S-> The dog chased the cat

# Context-free grammar

```
S      --> NP,VP.
NP     --> PN.          %Proper noun
NP     --> Art, Adj, N.
NP     --> Art,N.
VP     --> VI.          %intransitive verb
VP     --> VT, NP.      %transitive verb
Art    --> [the].
Adj    --> [lazy].
Adj    --> [rapid].
PN     --> [achilles].
N      --> [turtle].
VI     --> [sleeps].
VT     --> [beats].
```

# Parse tree



# Definite Clause Grammars
# Non-terminals may have arguments

```
S                  --> NP(N),VP(N).
NP(N)              --> Art(N),N(N).
VP(N)              --> VI(N).
Art(singular)      --> [a].
Art(singular)      --> [the].
Art(plural)        --> [the].
N(singular)        --> [turtle].
N(plural)          --> [turtles].
VI(singular)       --> [sleeps].
VI(plural)         --> [sleep].
```

Number Agreement

# DCGs

- Non-terminals may have arguments
  - Variables (start with capital)
    - E.g. Number, Any
  - Constants (start with lower case)
    - E.g. singular, plural
  - Structured terms (start with lower case, and take arguments themselves)
    - E.g. vp(V,NP)
- Parsing needs to be adapted
  - Using unification

# Unification in a nutshell (cf. AI course)

- Substitutions

$$\theta = \{V_1/t_1, ..., V_n/t_n\}$$

with Variables $V_i$; Terms $t_i$

E.g. {Num / singular }

{T / vp(V,NP)}

- Applying substitution
  - Simultaneously replace variables by corresponding terms
  - S(Num) {Num / singular } = S(singular)

# Unification

- Take two non-terminals with arguments and compute (most general) substitution that makes them identical, e.g.,
  - Art(singular) and Art(Num)
    - Gives { Num / singular }
  - Art(singular) and Art(plural)
    - Fails
  - Art(Num1) and Art(Num2)
    - {Num1 / Num2}
  - PN(Num, accusative) and PN(singular, Case)
    - {Num/singular, Case/accusative}

# Parsing with DCGs

- Now require successful unification at each step
- S -> NP(N), VP(N)
- S -> Art(N), N(N), VP(N)    {N/singular}
- S -> a  N(singular), VP(singular)
- S -> a turtle VP(singular)
- S -> a turtle sleeps

- S-> a turtle sleep      fails

# Case Marking

```
PN(singular,nominative)      --> [he];[she]
PN(singular,accusative)      --> [him];[her]
PN(plural,nominative)        --> [they]
PN(plural,accusative)        --> [them]
S        --> NP(Number,nominative), NP(Number)
VP(Number) --> V(Number), VP(Any,accusative)
VP(Number,Case)   --> PN(Number,Case)
VP(Number,Any)      --> Det, N(Number)
```

He sees her.  She sees him. They see her.

But not Them see he.

# DCGs

- Are strictly more expressive than CFGs
- Can represent for instance $a^n b^n c^n$
  - S(N) -> A(N), B(N), C(N)
  - A(0) -> []
  - B(0) -> []
  - C(0) -> []
  - A(s(N)) -> A(N), [A]
  - B(s(N)) -> B(N), [B]
  - C(s(N)) -> C(N), [C]

# Probabilistic Models

- Traditional grammar models are very rigid,
  - essentially a yes / no decision
- Probabilistic grammars
  - Define a probability models for the data
  - Compute the probability of each alternative
  - Choose the most likely alternative
- Ilustrate on
  - Shannon Game
  - Spelling correction
  - Parsing

# Sequences are omni-present

- Therefore the techniques we will see also apply to
  - Bioinformatics
    - DNA, proteins, mRNA, … can all be represented as strings
  - Robotics
    - Sequences of actions, states, …
  - …

# Rest of the Course

- Limitations traditional grammar models motivate probabilistic extensions
  - Regular grammars and Finite State Automata
    - All use principles of Part I on Graphical Models
    - Markov Models using n-gramms
    - (Hidden) Markov Models
    - Conditional Random Fields
      - As an example of using undirected graphical models
  - Probabilistic Context Free Grammars
  - Probabilistic Definite Clause Grammars