STATISTICAL STRUCTURED PREDICTION

José Miguel Benedí **e-mail**: `jmbenedi@prhlt.upv.es`

Joan Andreu Sánchez **e-mail**: `jandreu@prhlt.upv.es`

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

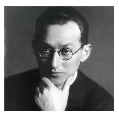
DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

MIARFID Official Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

Técnicas Fundamentales	**R.F. y Aprendizaje Computacional** **Lingüística Computacional**
Reconocimiento de Formas	**Predicción Estructurada Estadística** **Redes Neuronales Artificiales** **Aplicaciones de Reconocimiento de Formas**
Tecnologías del Lenguaje	**Traducción Automática** **Reconocimiento Automático del Habla** **Aplicaciones de la Lingüística Computacional**
Técnicas Complementarias	**Aprendizaje Automático Avanzado** **Reconocimiento de Escritura** **Biometría** **Visión por Computador**

Focus of Course

- Give students an overview of the different topics that allow them to understand the basic concepts:
  - What is structured prediction?
  - What problems will we try to address?
  - How to make predictions and learn the models in structured output spaces?
  - What are the different computational challenges for structured prediction?
- We will deal with the rigorous design of algorithms and make intensive use of mathematics, but nothing too hard.



"There is nothing more practical than a good theory".

Kurt Lewin

Syllabus

1. Introduction
2. Models for Statistical Structured Prediction
3. Making Prediction: Decoding and Inference
4. Model Parameter Estimation

Pattern Recognition and Machine Learning

- Christopher M. Bishop: Pattern Recognition and Machine Learning. Springer.
- Kevin P. Murphy: Machine Learning: A Probabilistic Perpective. The MIT Press.
- Daphne Koller and Nir Friedman: Probabilistic Graphical Models: Principles and Techniques. MIT Press.
- Richard O. Duda, Peter E. Hart and David G. Stork: Pattern Classification (2nd ed.). Wiley Interscience.

Statistical Models for Natural Language Processing

- Christopher D. Manning and Hinrich Schütze: Foundations of Statistical Natural Language Processing. The MIT Press.
- Noah A. Smith: Linguistic Structure Prediction. Morgan & Claypool.
- Daniel Jurafsky and James H. Martin: Speech and Language Processing. Prentice Hall (2^{a} ed).

Lectures Monday 18:00 - 21:00 (10 sessions of 3 hours)

J.M. Benedí November 06, first session

J.A. Sánchez December 11, first session

Assessment

2 Collections of selected exercises (theoretical and practical)

1 Multiple-choice test 12-02-2024

1 Recovery exam 19-02-2024

[Multiple-choice test and theoretical/practical exercises]

	delivery	**deadline**
Q1	04 - 12 - 2023	22 - 12 - 2023

Tutoring Tutoring available by previous appointment

José Miguel Benedí <jmbenedi@prhlt.upv.es> (office: 1D13)

1. Introduction

1.1. Structured Prediction

1.2. Predicting Sequences

- Weighted Finite-State Transducers and Automata

1.3. Syntactic Parsing

- Context-Free Grammars

- Input observation; $\boldsymbol{x} \in \mathcal{X}$ can be any kind of object.
- Output hypothesis; $y \in \mathcal{Y}$ is a real number: $\mathcal{Y} = \{1, \ldots, K\}$ or $y \in \mathbb{R}$.
- (Non-structured) prediction function; $f : \mathcal{X} \to \mathbb{R}$ assigns a hypothesis $y = f(\boldsymbol{x})$ to each entry $\boldsymbol{x}$.
  - Binary classification: $y \in \{-1, 1\}$.
  - Multiclass classification: $y \in \{1, \ldots, K\}$.
  - Regression: $y \in \mathbb{R}$.

Algorithms for predicting non-structured output data

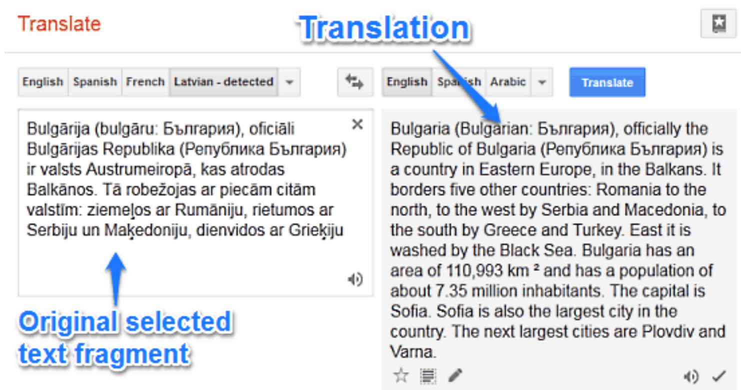
Naive Bayes classifier	Regression
Logistic Regression	Fisher's linear discriminant
Perceptron algorithm	K-Nearest Neighbor
Support Vector Machines	Random Forests
Neural Networks	...

What if the space of outputs is much larger and more structured?
Say trees, or in general, graphs.

Machine Translation

Input: Text sequences

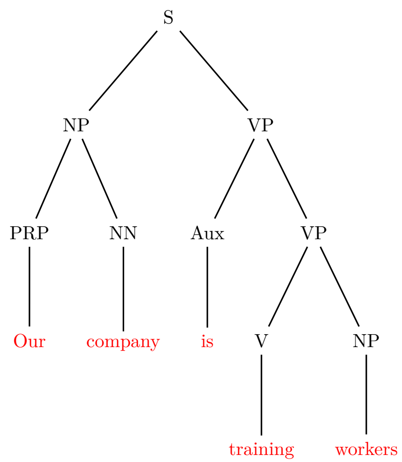
Output: Text sequences



Syntactic Parsing

Input: Text sequences

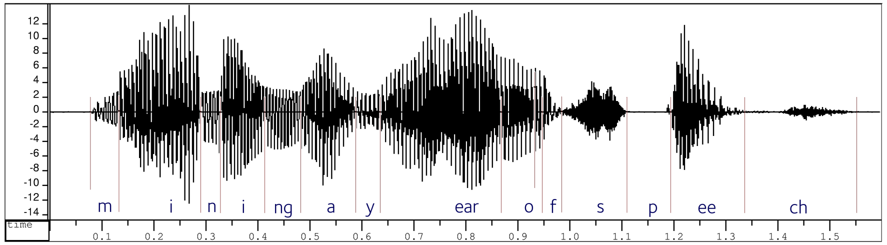
Output: Parse trees



Automatic Speech Recognition

Input: Speech signals

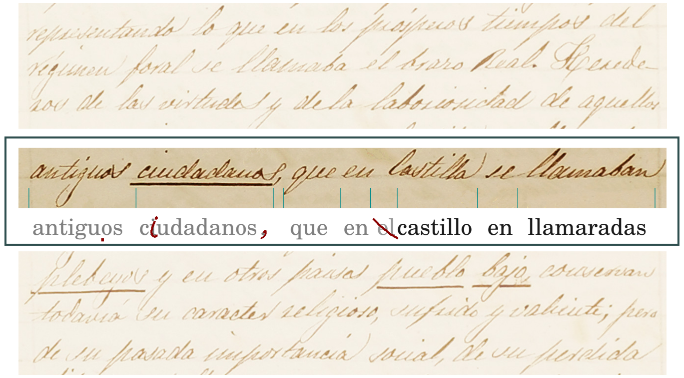
Output: Transcribed text sequences



Handwriting Text Recognition

Input: Images

Output: Transcribed text sequences



Scene Analysis

Input: Images

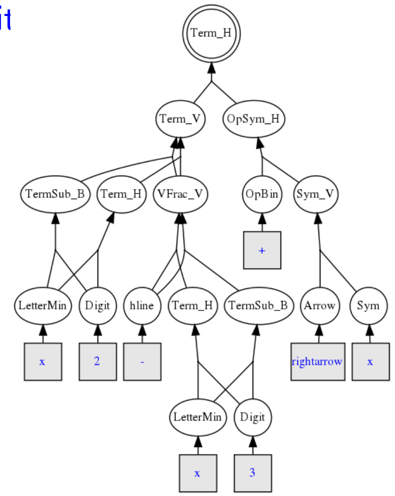
Output: Scene layout graphs

Mathematical Expressions Recognit

Input: Images

Output: Hypergrphs

$$\boxed{\frac{x_2}{x_3} + \vec{x}}.$$



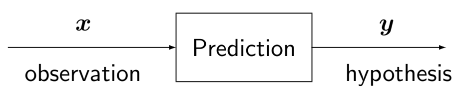
⇒ Mathematical Expression Recognition

STRUCTURED PREDICTION: APPLICATIONS

Outputs

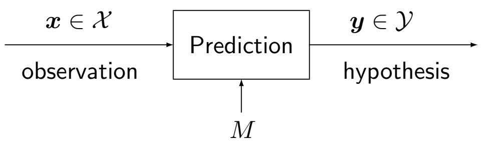
- Natural Language Processing:
  - Part-of-Speech tagging (sentences)
  - Parsing (parse trees)
  - Machine Translation (sentences or hypergraphs)
  - Information Extraction (sentences)
- Image Processing:
  - Visual Scene Analysis (sentences or relationship graphs)
  - Handwritten Text Recognition (sentences or word graphs)
- Speech Processing:
  - Automatic transcription (sentences or word graphs)
  - Text-to-Speech (audio signal)
- Bioinformatics:
  - Protein Structure Prediction (graphs)
- Robotics:
  - Planning (sequence of actions)

STRUCTURED PREDICTION



- Input observation; $\boldsymbol{x} \in \mathcal{X}$ can be any kind of objects
- Output hypothesis; $\boldsymbol{y} \in \mathcal{Y}$ is a complex structured object.
- (Structured) prediction function; $f : \mathcal{X} \to \mathcal{Y}$ which assigns a hypothesis $\boldsymbol{y} = f(\boldsymbol{x})$ to each entry $\boldsymbol{x}$.
  - $\boldsymbol{y}$ can be a sequence,
  - $\boldsymbol{y}$ can be a parse tree,
  - $\boldsymbol{y}$ can be a graph.

STRUCTURED PREDICTION: SEARCH SPACE



- Typically, $\mathcal{Y}$ is potentially infinite $\Longrightarrow$ $f(\boldsymbol{x}) = \boldsymbol{y} \in \mathcal{Y}$ will often be intractable.
- (Structural) Solution: We assume a specific structure in $\mathcal{Y}$, and we define a (finite) model M that allows us to characterize this structure: $Y(M) \subseteq \mathcal{Y}$

$$f(\boldsymbol{x}) = \boldsymbol{y} \in Y(M)$$

Depending on M, we can find polynomial solutions.

PREDICTING SEQUENCES

Sequences

- Text is a sequence of words or even letters,
- A spoken utterance is a sequence of parameter vectors,
- A video is a sequence of frames, . . .

Models and motivation

Finite-State Acceptors: Compact representations of regular sets that are efficient to search, e.g. pattern matching, tokenization, compression.

Finite-State Transducers: Compact representations of rational binary relations that are efficient to search and combine, e.g. dictionaries, context-dependent rules.

Weighted Automata: Weights typically encode uncertainty (probabilities), e.g. n-gram language models, Hidden Markov Models.

Preliminaries

WFSA and WFST-based operations are underpinned by algebraic objects called **semirings**.

Definition. A semiring is an algebraic system $(\mathbb{K},\ \oplus,\ \otimes,\ \overline{0},\ \overline{1})$ such that,

- $(\mathbb{K},\ \oplus,\ \overline{0})$ is a **commutative monoid**[a] with $\overline{0}$ as the identity element for $\oplus$,
- $(\mathbb{K},\ \otimes,\ \overline{1})$ is a **monoid** with $\overline{1}$ as the identity element for $\otimes$,
- $\otimes$ distributes over $\oplus$: for all $a, b, c \in \mathbb{K}$,

  $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$

  $c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$
- $\overline{0}$ is an annihilator for $\otimes$: for all $a \in \mathbb{K}$,

  $a \otimes \overline{0} = \overline{0} \otimes a = \overline{0}$

This has implications for optimization, search, and combination algorithms such as determinization, shortest-path, and composition.

[a] A monoid is an algebraic structure that supports a single associative binary operation and an identity element.

Preliminaries

- **Product** $\otimes$: to compute the weight of a path
  (product of the weights of constituent transitions).
- **Sum** $\oplus$: to compute the weight of a sequence
  (sum of the weights of the paths labeled with that sequence).

Semiring	Set	$\oplus$	$\otimes$	$\overline{0}$	$\overline{1}$
Boolean	$\{\ 0,\ 1\ \}$	$\vee$	$\wedge$	0	1
Probability	$\mathbb{R}_+ \cup \{\ +\infty\ \}$	$+$	$\times$	0	1
Log	$\mathbb{R} \cup \{\ -\infty,\ +\infty\ \}$	$\oplus_{log}$	$+$	$+\infty$	0
Tropical	$\mathbb{R}_+ \cup \{\ +\infty\ \}$	min	$+$	$+\infty$	0

- The **log semiring** is isomorphic to the **probability semiring** and
  $\oplus_{log}$ is defined by: $x \oplus_{log} y = -\log(e^{-x} + e^{-y})$
- The **tropical semiring** is derived from the **log semiring** using the *Viterbi* approximation

[M. Mohri. *Semiring frameworks and algorithms for shortest-distance problems*. 2002], for more details.

Preliminaries

Tropical Semiring example

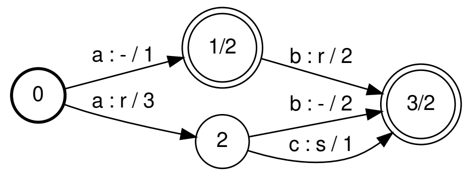
Definitions
$a \oplus b \stackrel{\text{def}}{=} \min(a,\ b)$
$a \otimes b \stackrel{\text{def}}{=} a + b$
$\overline{0} \stackrel{\text{def}}{=} +\infty$
$\overline{1} \stackrel{\text{def}}{=} 0$

Examples
$7 \oplus 4 = 4$
$1 \otimes 7 = 8$
$7 \oplus \overline{0} = 7$
$3 \otimes \overline{1} = 3$
$(4 \otimes 3) \oplus (2 \otimes 3) = 5$
$4 \oplus \overline{1} = 0$

Weighted finite-state transducer

Definition. A weighted transducer T over a semiring $(\mathbb{K},\ \oplus,\ \otimes,\ \overline{0},\ \overline{1})$ is a tuple $T = (\Sigma,\ \Delta,\ Q,\ I,\ F,\ \delta,\ \lambda,\ \rho)$, where

- Finite input alphabet Σ and finite output alphabet Δ.
- States Q, initial states $I \subseteq Q$, and final states $F \subseteq Q$.
- Transiticon function $\delta :\ Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{K} \times Q$.
- Initial weight function $\lambda : I \to \mathbb{K}$ and final weight function $\rho : F \to \mathbb{K}$.



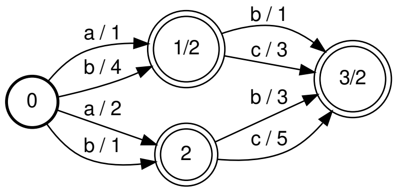
Probability semiring $(\mathbb{K}, +, \times, 0, 1)$	**Tropical semiring** $(\mathbb{K}, \min, +, \infty, 0)$
$[[T]](ab,\ r) = 16$	$[[T]](ab,\ r) = 5$

[Obtained from the OpenFst tool documentation]

WEIGHTED FINITE-STATE AUTOMATON

Definition. A weighted automaton A over a semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ is a tuple $A = (\Sigma, Q, I, F, \delta, \lambda, \rho)$, where

- Finite input alphabet Σ.
- States Q, initial states $I \subseteq Q$, and final states $F \subseteq Q$.
- Transiticon function $\delta :\ Q \times (\Sigma \cup \{\epsilon\}) \times \mathbb{K} \times Q$.
- Initial weight function $\lambda : I \rightarrow \mathbb{K}$ and final weight function $\rho : F \rightarrow \mathbb{K}$.



Probability semiring $(\mathbb{K}, +, \times, 0, 1)$	**Tropical semiring** $(\mathbb{K}, \min, +, \infty, 0)$
$[[T]](ab) = 14$	$[[T]](ab) = 4$

[Obtained from the OpenFst tool documentation]

DEFINITIONS

Path π
- $p[\pi]$ Path origen state
- $n[\pi]$ Path destination state
- $w[\pi]$ Path weight

Sets of paths
- $P(I, x, F)$ Set of all paths from I to F with input label $x \in \Sigma^*$
- $P(I, x, y, F)$ Set of all paths from I to F with input label $x \in \Sigma^*$ and output label $y \in \Delta^*$

Automata and transducers

Given an **automaton** $A = (\Sigma, Q, I, F, \delta, \lambda, \rho)$, for all $x \in \Sigma^*$

$$[[A]]\ (x) \ = \bigoplus_{\pi \in P(I,x,F)} \lambda(p(\pi)) \otimes w(\pi) \otimes \rho(n(\pi))$$

Given a **transducer** $T = (\Sigma, \Delta, Q, I, F, \delta, \lambda, \rho)$, for all $x \in \Sigma^*$, $y \in \Delta^*$

$$[[T]]\ (x, y) \ = \bigoplus_{\pi \in P(I,x,y,F)} \lambda(p(\pi)) \otimes w(\pi) \otimes \rho(n(\pi))$$

OPERATIONS

Rational operations

Sum (union) $$[[T_1 \oplus T_2]](x, y) = [[T_1]](x, y) \oplus [[T_2]](x, y)$$

Product (Concat.) $$[[T_1 \otimes T_2]](x, y) = \bigoplus_{\substack{x=x_1x_2 \\ y=y_1y_2}} [[T_1]](x_1, y_2) \otimes [[T_2]](x_2, y_2)$$

Closure $$[[T^*]](x, y) = \bigoplus_{n=0}^{\infty} [[T^n]](x, y)$$

Unary operations

Reversal $$[[T^R]](x, y) = [[T]](x^R, y^R)$$

Inversion $$[[T^{-1}]](x, y) = [[T]](y, x)$$

Projection $$[[\downarrow T]](x) = \bigoplus_{y} [[T]](x, y)$$

OPERATIONS

Binary Operations

Composition $$[[T_1 \circ T_2]](x, y) = \bigoplus_{z} [[T_1]](x, z) \oplus [[T_2]](z, y)$$

Intersection $$[[A_1 \cap A_2]](x) = [[A_1]](x) \oplus [[A_2]](x)$$

Difference $$[[A_1 - A_2]](x) = [[A_1 \cap \overline{A_2}]](x)$$

Optimization algorithms

ϵ-Removal: Creates an equivalent ϵ-free transducer.

Determinization: Creates an equivalent deterministic transducer.

Pushing: Pushes arc weights forward or backward, accumulating and/or distributing them according to the semiring

Minimization: Creates an equivalent minimal deterministic transducer.

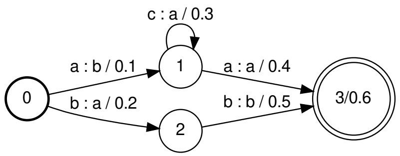
Shortest-distance algorithms: **Shortest path**, and **N-Shortest paths**.

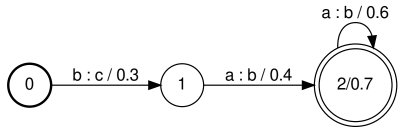
[M. Mohri. *Semiring frameworks and algorithms for shortest-distance problems*. 2002], for more details.

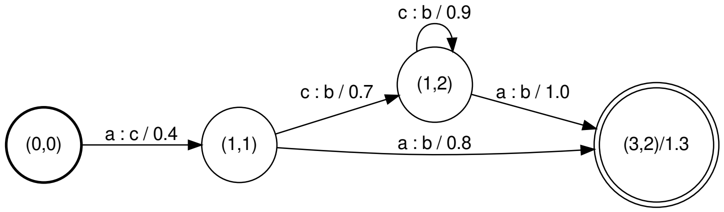
COMPOSITION

Definition: $[[T_i \circ T_2]](x,y) = \bigoplus_z [[T_1]](x,z) \otimes [[T_2]](z,y)$

Example: Weighted automaton over the tropical semiring



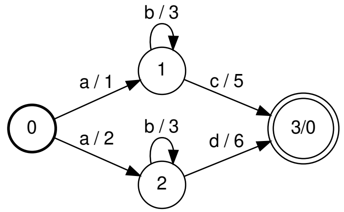


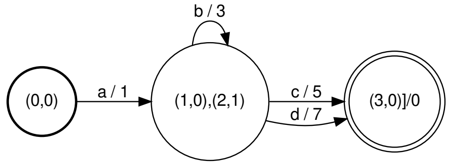


DETERMINIZATION

Definition: Creates an equivalent deterministic weighted automaton/transducer

Example: Weighted automaton over the tropical semiring

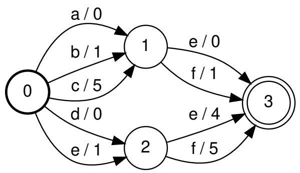


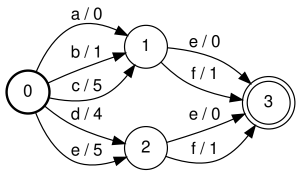


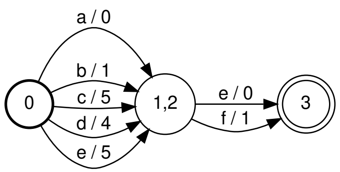
MINIMIZATION

Definition: Computes a minimal equivalent deterministic machine while preserving the input language and and weight/path properties of the original

Example: Weighted automaton over the tropical semiring







REFERENCES

General Background

- J.E.Hopcroft, J.D.Ullman: *Introduction to Automata Theory, Languages, and Computation.* Addison Wesley, 1979.
- T.H. Cormen, C.E.Leiserson, R.L.Rivest: *Introduction to Algorithms.* The MIT Press, 1992.

WFST and WFSA applications

- M.Mohri, F.Pereira, M.Riley: *Speech Recognition with Weighted Finite-State Transducers.* In *Springer Handbook of Speech Processing.* Springer, 2008.
- M.Mohri: *Weighted Automata Algorithms.* In *Handbook of Weighted Automata.* Monographs in Theoretical Computer Science. Springer, 2009.
- A.Argueta, D.Chiang: *Composing Finite State Transducers on GPUs.* Proc. of the ACL, p.pages 2697–2705. Melbourne, Australia, 2018.

Software

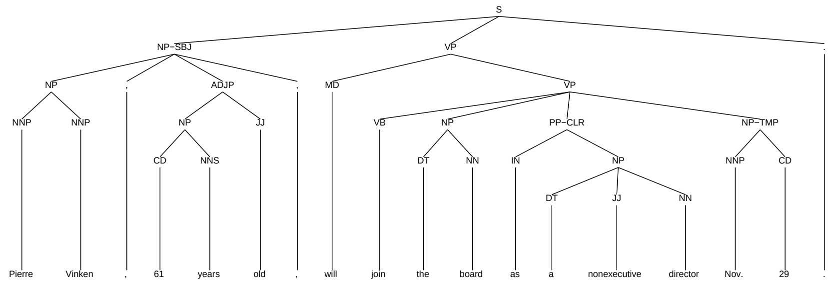
OpenFst is a open-source C++ library for weighted finite state transducers developed at Google. More information available at `http://www.openfst.org`

- C.Allauzen, M.Riley, J.Schalkwyk, W.Skut, M.Mohri: *OpenFst: A General and Efficient Weighted Finite-State Transducer Library.* Proc. of the CIAA. Prague, CZ, 2007.

SYNTACTIC PARSING

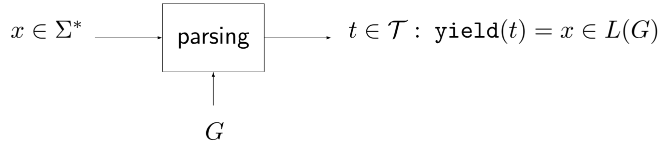
Example: *"Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29."* [Marcus et al., 1993]

((S (NP-SBJ (NP (NNP Pierre) (NNP Vinken)) (,,) (ADJP (NP (CD 61) (NNS years)) (JJ old)) (,,)) (VP (MD will) (VP (VB join) (NP (DT the) (NN board)) (PP-CLR (IN as) (NP (DT a) (JJ nonexecutive) (NN director))) (NP-TMP (NNP Nov.) (CD 29)))) (..)))



SYNTACTIC PARSING

- Parsing as a search problem



- Grammatical models: reasons for use
  - Grammatical models are the simplest and most natural model for tree structures
  - Formal (mathematical) framework is well known
  - Compact models (small number of free parameters)
  - Good behavior against the problem of ambiguity
  - Context-free grammatical models represent well long-term dependencies of syntactic and semantic constraints of Natural Language

CONTEXT-FREE GRAMMARS

- Context-Free Grammars $G = (\Sigma, N, S, \mathcal{P})$

  Σ a set of **terminal symbols**

  N a set of **non-terminals symbols** (or variables): $N \cap \Sigma = \emptyset$

  S a distinguished **start symbol**: $S \in N$

  $\mathcal{P}$ a set of **rules** (or productions): $(A \to \alpha) \in \mathcal{P}$; $A \in N$; $\alpha \in (N \cup \Sigma)^*$

- Direct derivation: $\delta A \gamma$ **directly derives** $\delta \alpha \gamma$ or

  $\delta\ A\ \gamma \ \Rightarrow\ \delta\ \alpha\ \gamma$ **iff** $\exists (A \to \alpha) \in \mathcal{P}$; $\delta, \gamma \in (N \cup \Sigma)^*$

- Derivation: α **derives** β or

  $\alpha \overset{*}{\Rightarrow} \beta$ **iff** $\exists\ \alpha_0, \ldots \alpha_m \in (N \cup \Sigma)^*$: $\alpha = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \ldots \alpha_{m-1} \Rightarrow \alpha_m = \beta$

- Language generated by a grammar: $L(G) = \{x \mid x \in \Sigma^* : \ S \overset{+}{\Rightarrow} x\}$

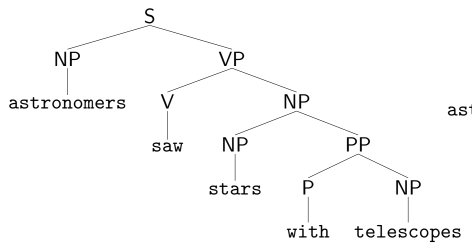
- Theorem $x \in L(G)$ **iff** $S \overset{+}{\Rightarrow} x$ **iff** $\exists\ t \in \mathcal{T}: \ \texttt{yield}(t) = x$

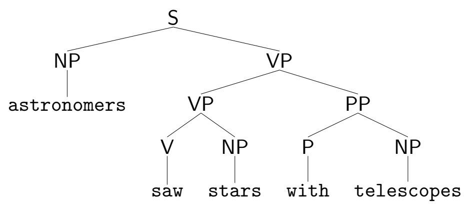
CONTEXT-FREE GRAMMARS: EXAMPLE

Example: A simple Context-Free Grammars [Manning and Schütze, 2002]

S $\to$ NP VP	VP $\to$ V NP	V $\to$ `saw`	NP $\to$ `saw`
NP $\to$ NP PP	VP $\to$ VP PP	NP $\to$ `astronomers`	NP $\to$ `stars`
PP $\to$ P NP	P $\to$ `with`	NP $\to$ `ears`	NP $\to$ `telescopes`

S $\Rightarrow$ NP VP $\Rightarrow$ `astronomers` VP $\Rightarrow$ `astronomers` V NP $\Rightarrow$ `astronomers saw` NP $\Rightarrow$ `astronomers saw` NP PP $\Rightarrow$ `astronomers saw stars` PP $\Rightarrow$ `astronomers saw stars` P NP $\Rightarrow$ `astronomers saw stars with` NP $\Rightarrow$ `astronomers saw stars with telescopes`





CKY PARSING

Algorithm 1: Cocke-Kasami-Younger

Input: $G = (\Sigma, N, S, \mathcal{P})$ in FNC and
$\mathbf{x} = x_1 \ldots x_T \in \Sigma^*$

Output: Parsing table $t[i,j]$ $\quad (1 \leq i, j \leq T)$;
$A \in t[i, i+l]$ **iff** $A \stackrel{*}{\Longrightarrow} x_{i+1} \ldots x_{i+l}$

for $i :\ 0 \ldots T-1$ **do**
  $t[i, i+1] = t[i, i+1] \cup \{A \ :\ (A \to b) \in P;\ \ b = x_{i+1}\}$

for $l :\ 2 \ldots T$ **do**
  for $i :\ 0 \ldots T-l$ **do**
    for $k :\ 1 \ldots l-1$ **do**
      $t[i, i+l] = t[i, i+l] \ \cup\ \{A \ :\ (A \to BC) \in P$;
      $B \in t[i, i+k];\ \ C \in t[i+k, i+l]\}$;

if $S \in t[0, T]$ **then** $x \in L(G)$ **else** $x \notin L(G)$;

CKY PARSING: EXAMPLE

S ⇒ A S
S ⇒ b
A ⇒ A S
A ⇒ A A
A ⇒ a

a a b

	$j=1$	$j=2$	$j=3$	
$i=0$	1: (A, 0, 0)	4: (A, 1, 2)	7: (S, 1, 5) 8: (A, 1, 5) 9: (A, 1, 6) 10: (S, 4, 3) 11: (A, 4, 3)	$l=3$
	$i=1$	2: (A, 0, 0)	5: (S, 2, 3) 6: (A, 2, 3)	$l=2$
		$i=2$	3: (S, 0, 0)	$l=1$