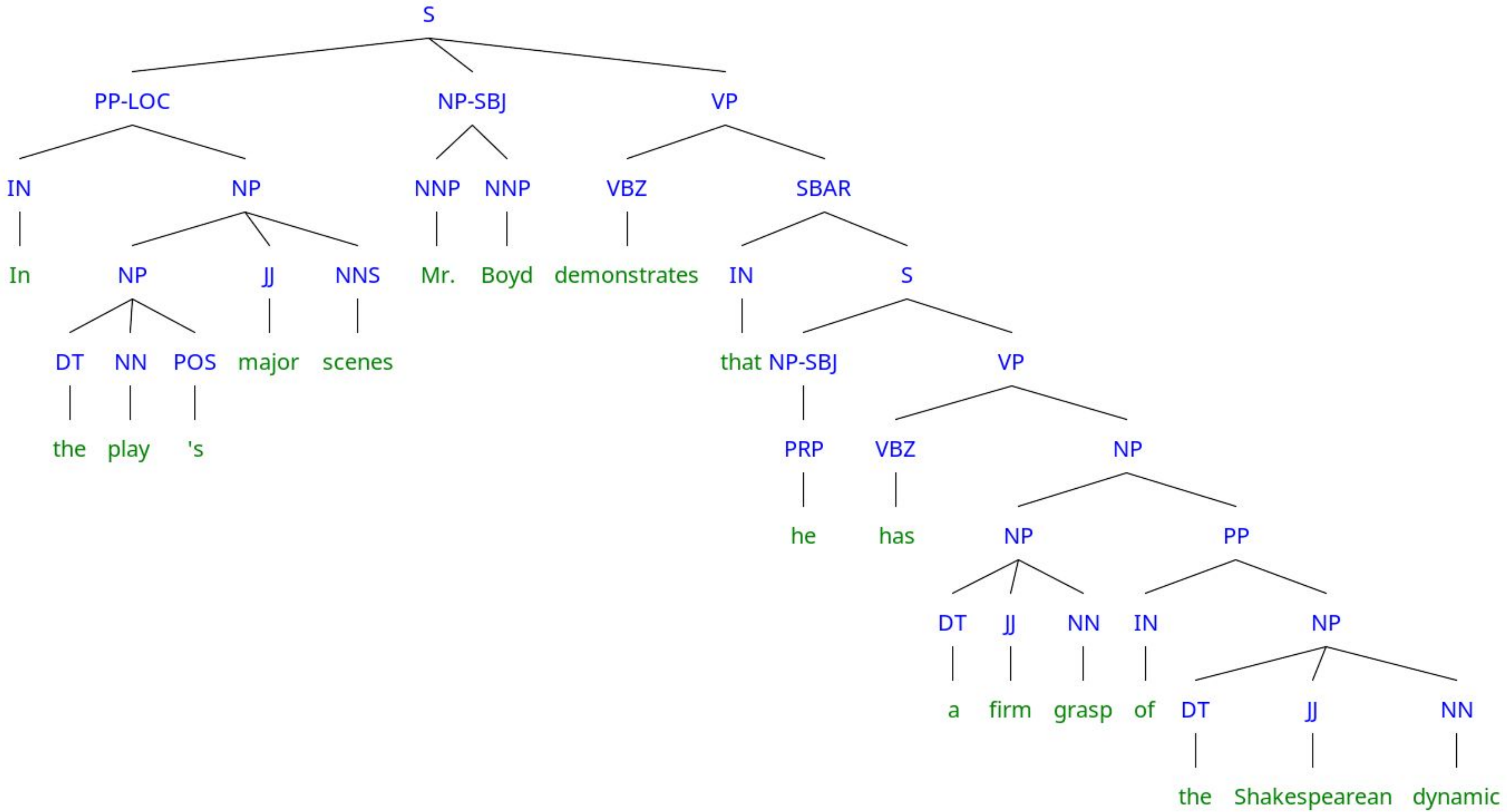
(Unsupervised) Parsing & Grammar Induction

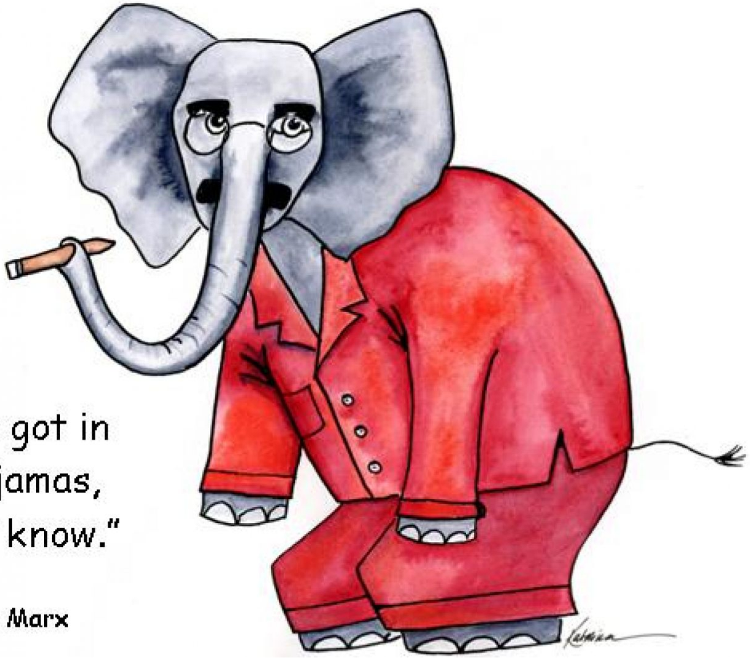
Outline

- **Motivation**
- Review: Formal grammars
- Probabilistic grammar induction
- Recent approaches for grammar induction & unsupervised parsing
- Conclusion

Language has hierarchical structure

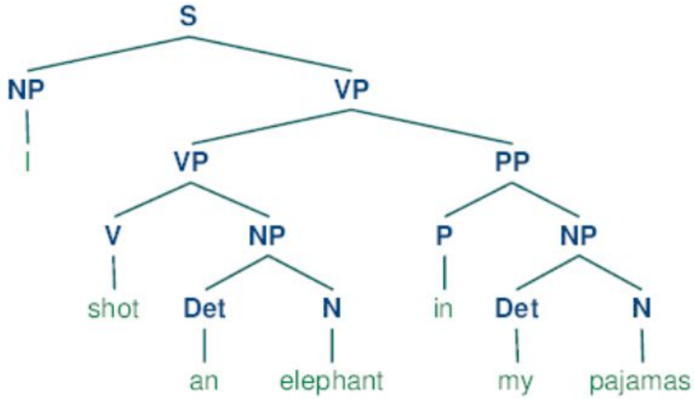


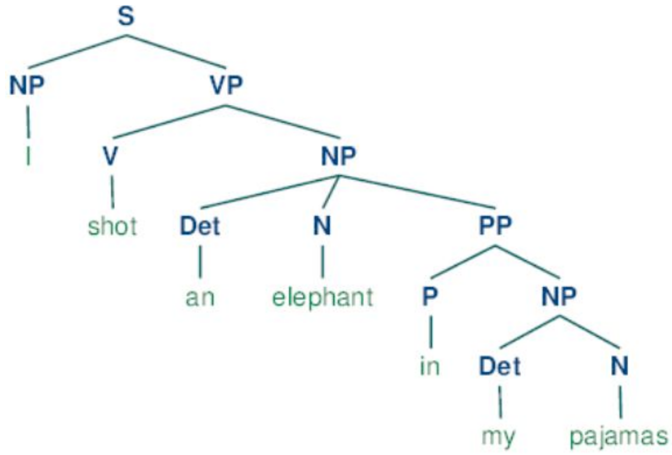
“One morning I shot an elephant in my pajamas.



How he got in my pajamas, I don’t know.”

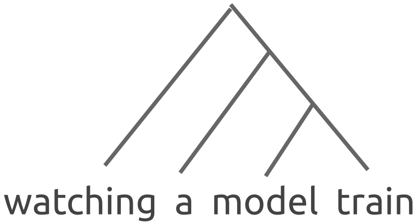
Groucho Marx



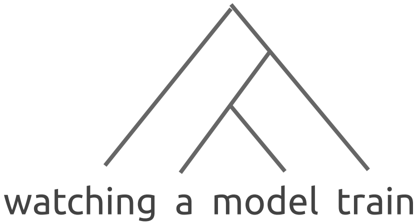


Bojan Tunguz
@tunguz

Watching a model train can be very calming and satisfying.


watching a model train


Bojan Tunguz
@tunguz
Watching a model train can be very calming and satisfying.
Watching a model train
Watching a model train
wluper


watching a model train

Human Language Competence

- Robust intuitions about grammaticality of novel but meaningless sentences:

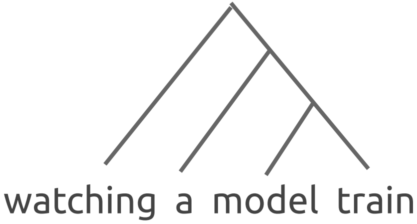
✔ *“colorless green ideas sleep furiously”*

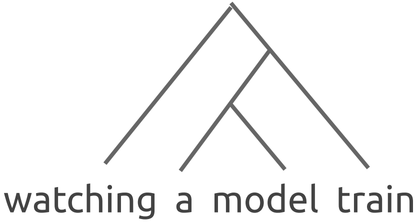
✘ *“furiously sleep ideas green colorless”*

- What is the underlying structure governing human language that allows us to generate/recognize an infinite number well-formed sentences?

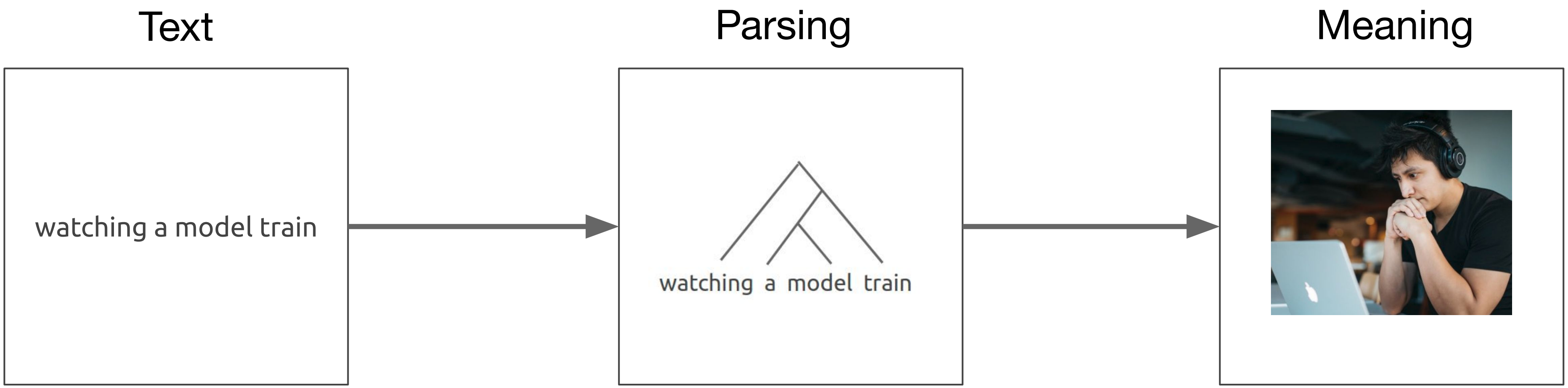
Parse Trees

- **Linguistics**: Human language understanding is mediated by compositional tree-like structures [Chomsky ’57].

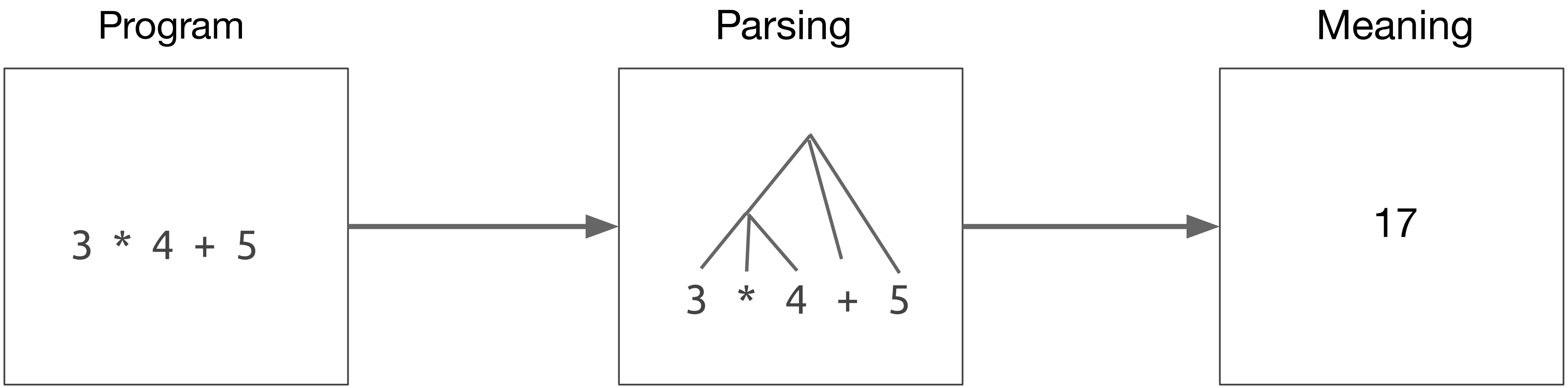




Parsing as a step towards meaning

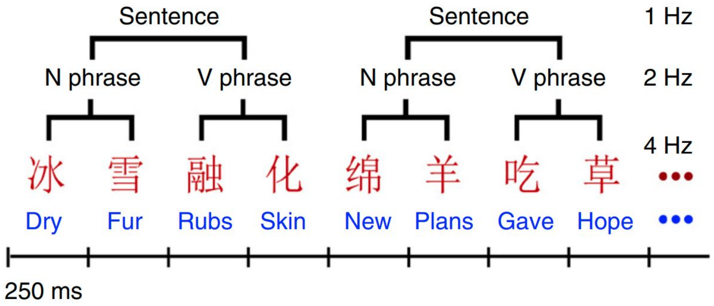


Parsing as a step towards meaning



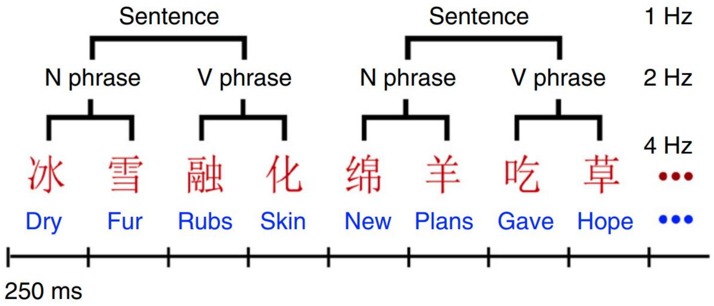
Human Parsing

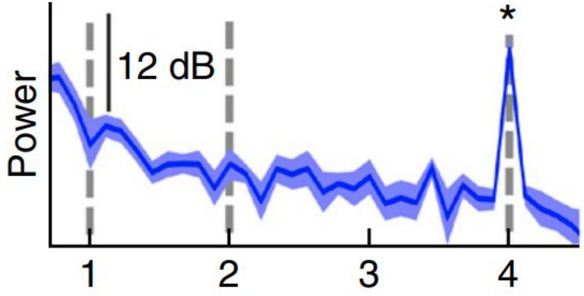
- **Neuroscience**: different neural activity for English vs. Chinese listeners when listening to Chinese [Ding et al ’15]



Human Parsing

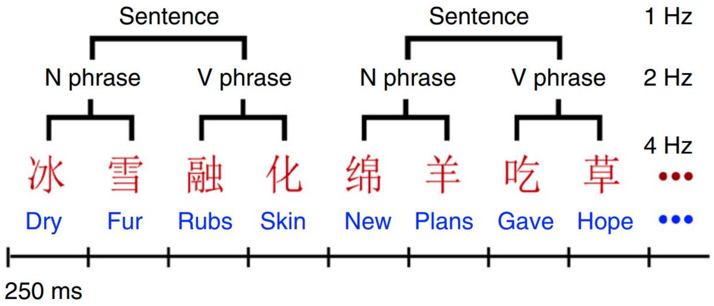
- **Neuroscience**: different neural activity for English vs. Chinese listeners when listening to Chinese [Ding et al ’15]

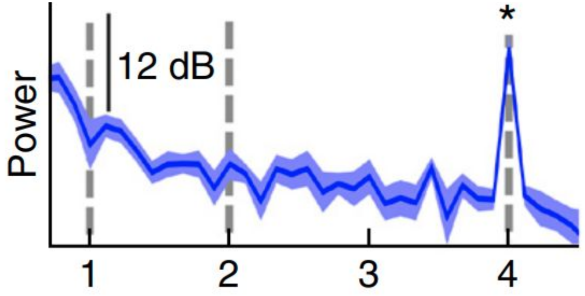


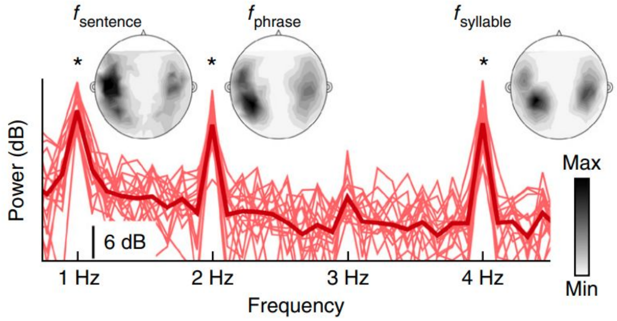


Human Parsing

- **Neuroscience**: different neural activity for English vs. Chinese listeners when listening to Chinese [Ding et al ’15]

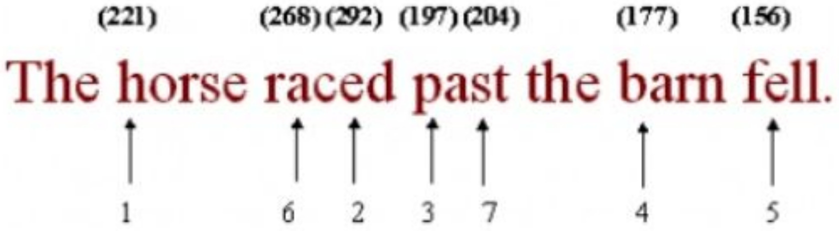






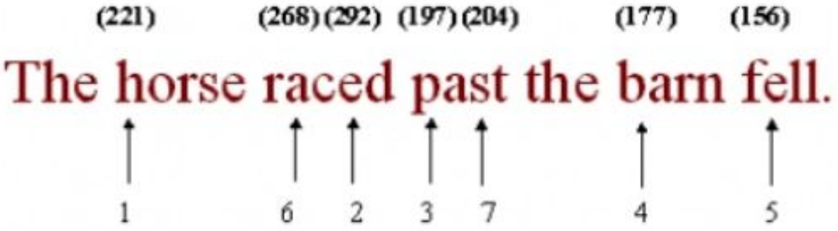
Human Parsing

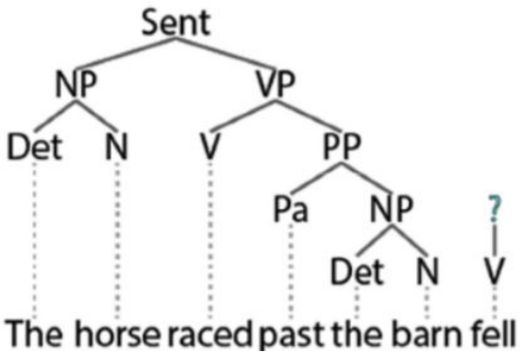
- **Psycholinguistics**: Eye movement in garden path sentences



Human Parsing

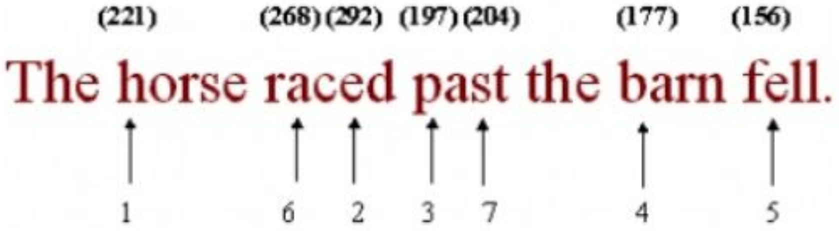
- **Psycholinguistics**: Eye movement in garden path sentences

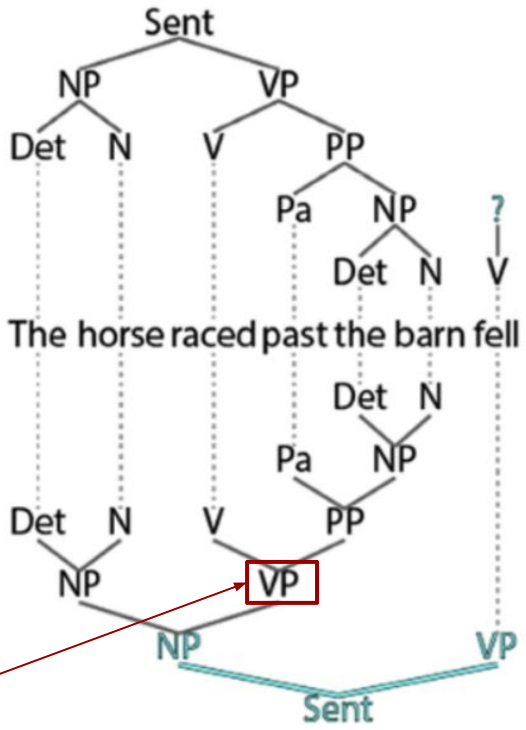




Human Parsing

- **Psycholinguistics**: Eye movement in garden path sentences





Increased processing time to reanalyze this verb phrase into a relative clause

Computational Approaches to Parsing

- **Rules-based**:
  - Ask a really smart linguist to come up rules for parsing (e.g. based on a grammar).
  - Hard to capture the complexities of natural language.

- **Statistical**:
  - Ask linguists to annotate sentences with their corresponding parse trees.
  - Treat it as a supervised learning problem.

Supervised Parsing

- Core task in NLP with lots of different approaches.
  - Graph-based
  - Transition-based
- Parse trees often used as part of a larger NLP pipeline for a downstream task.
- … until deep learning came long.

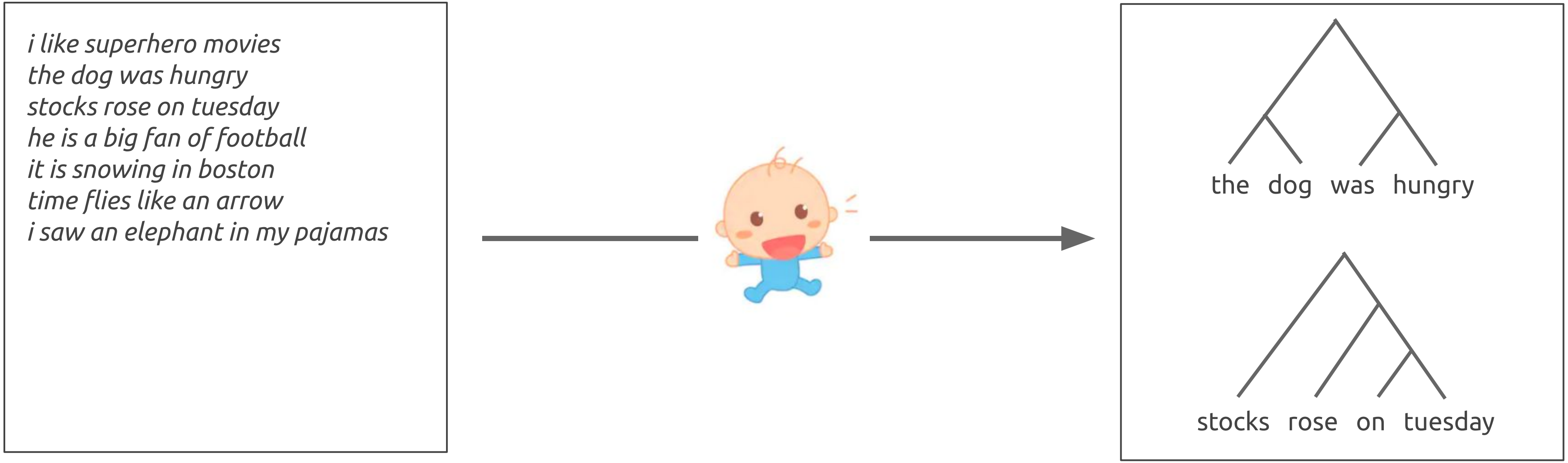
Model	F_1
Non-Neural Models	
Collins (1997)	87.8
Charniak (1999)	89.6
Petrov and Klein (2007)	90.1
McClosky et al. (2006)	92.1
Neural Models	
Dyer et al. (2016)	93.3
Fried et al. (2017)	94.7
Kitaev and Klein (2019)	95.8

(on WSJ Penn Treebank)

Unsupervised Parsing

- Parse trees may not be as useful from an engineering perspective, but still interesting from a cognitive science standpoint.

- How do humans learn to parse? (Mostly) Unsupervised!



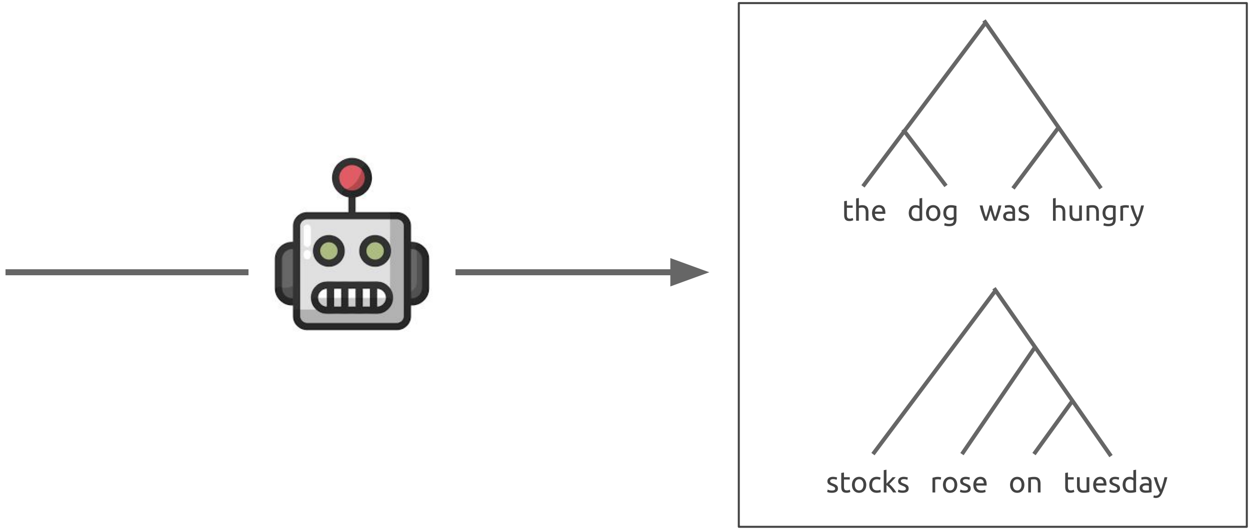
Unsupervised Parsing

- Parse trees may not be as useful from an engineering perspective, but still interesting from a cognitive science standpoint.

- Can we train machine to do the same?

i like superhero movies
the dog was hungry
stocks rose on tuesday
he is a big fan of football
it is snowing in boston
time flies like an arrow
i saw an elephant in my pajamas

⋮



Grammars for Parsing

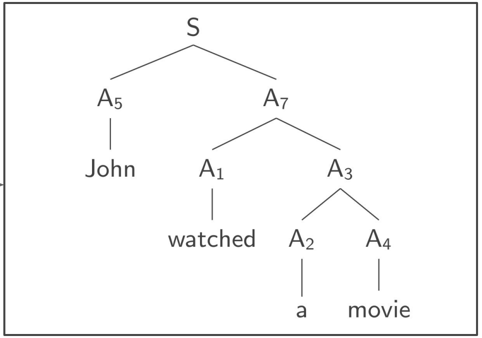
- Classic approach: hypothesize a **formal grammar** that generates human language.

$$N = \{\mathsf{S}, \mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_T\}$$

$$\Sigma = \{\text{a, aardvark, able, are}, \ldots, \text{zyzzyva}\}$$

$$\begin{aligned} P = \ & \mathsf{S} \to \mathsf{A}_1\ \mathsf{A}_1 \\ & \mathsf{S} \to \mathsf{A}_1\ \mathsf{A}_2 \\ & \mathsf{S} \to \mathsf{A}_1\ \mathsf{A}_3 \\ & \quad \vdots \end{aligned}$$

Grammar



(Parse tree structure implied by the grammar.)

Outline

- Motivation
- Review: Formal grammars
- Probabilistic grammar induction
- Recent approaches for grammar induction & unsupervised parsing
- Conclusion

Grammars

- A set of production rules for deriving strings in a formal language.

$$G = (N, \Sigma, P, S)$$

N : set of nonterminal symbols

Σ : set of terminal symbols

P : set of production rules

S : start symbol $(S \in N)$

Grammars

- A set of production rules for deriving strings in a formal language.

$$G = (N, \Sigma, P, S)$$

$$\begin{aligned} N &: \text{ set of nonterminal symbols} \\ \Sigma &: \text{ set of terminal symbols} \\ P &: \text{ set of production rules} \\ S &: \text{ start symbol } \ (S \in N) \end{aligned}$$

- $L(G) = \{w \in \Sigma^\star \mid S \to_G w\}$

(set of strings that can be generated from S by applying rules in G)

Grammars

$$N = \{S\}$$

$$\Sigma = \{a, b\}$$

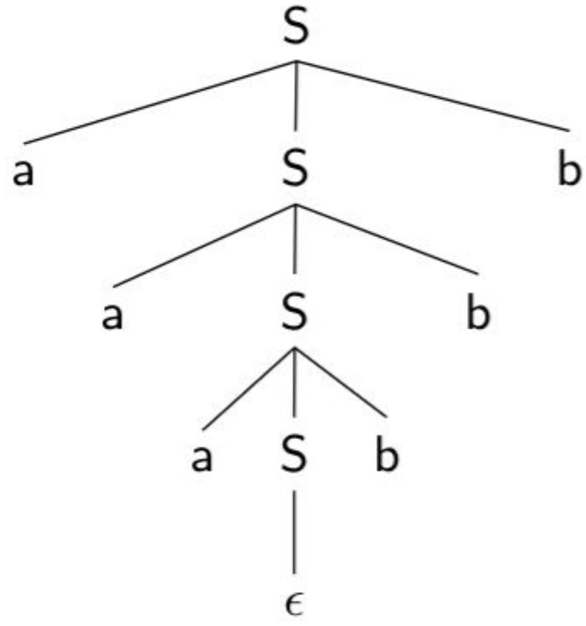
$$P = \{S \to aSb, S \to \epsilon\}$$

Grammars

$N = \{S\}$

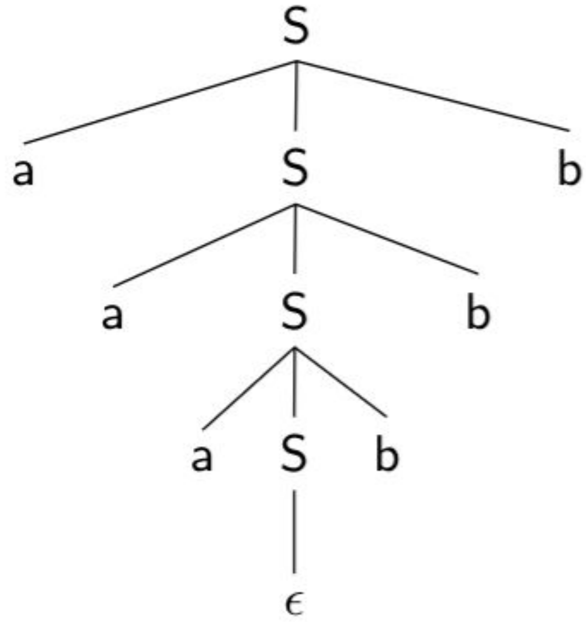
$\Sigma = \{a, b\}$

$P = \{S \to aSb, S \to \epsilon\}$



Grammars

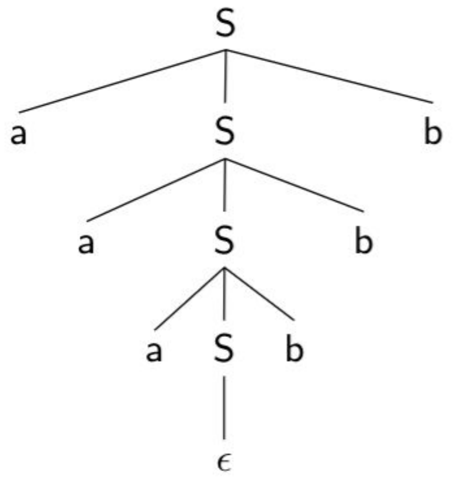
$$
\begin{aligned}
N &= \{S\} \\
\Sigma &= \{a, b\} \\
P &= \{S \to aSb, S \to \epsilon\} \\
L(G) &= \{\epsilon, ab, aabb, aaabbb \dots\} \\
&= \{a^n b^n : n \geq 0\}
\end{aligned}
$$



Grammars

- Is “aaaabbbb” in L(G)? What about “abbb”?

- Given a grammar G:
    - Can check if a string belongs to L(G) by parsing.
    - Parsing also gives gives underlying sentence structure.



Grammars for Natural Language

$N = \{\mathsf{S}, \mathsf{NP}, \mathsf{VP}, \mathsf{CP}, \mathsf{C}, \mathsf{N}, \mathsf{D}, \mathsf{V}_t, \mathsf{V}_i\}$

$\Sigma = \{\mathsf{a}, \mathsf{the}, \mathsf{that}, \mathsf{said}, \mathsf{meows}, \mathsf{barks}, \mathsf{noticed}, \mathsf{cat}, \mathsf{dog}, \mathsf{zyzzyva}\}$

$$
\begin{aligned}
P = \;& \mathsf{S} \rightarrow \mathsf{NP}\ \mathsf{VP} \\
& \mathsf{VP} \rightarrow \mathsf{V}_t\ \mathsf{CP} \mid \mathsf{V}_i \\
& \mathsf{CP} \rightarrow \mathsf{C}\ \mathsf{S} \\
& \mathsf{NP} \rightarrow \mathsf{D}\ \mathsf{N} \\
& \mathsf{N} \rightarrow \mathsf{cat} \mid \mathsf{dog} \mid \mathsf{zyzzyva} \\
& \mathsf{V}_t \rightarrow \mathsf{said} \mid \mathsf{noticed} \\
& \mathsf{V}_i \rightarrow \mathsf{meows} \mid \mathsf{barks} \\
& \mathsf{C} \rightarrow \mathsf{that} \\
& \mathsf{D} \rightarrow \mathsf{the} \mid \mathsf{a}
\end{aligned}
$$

Grammars for Natural Language

$$N = \{\mathsf{S}, \mathsf{NP}, \mathsf{VP}, \mathsf{CP}, \mathsf{C}, \mathsf{N}, \mathsf{D}, \mathsf{V}_t, \mathsf{V}_i\}$$

$$\Sigma = \{\text{a}, \text{the}, \text{that}, \text{said}, \text{meows}, \text{barks}, \text{noticed}, \text{cat}, \text{dog}, \text{zyzzyva}\}$$

$$\begin{aligned} P = \;& \mathsf{S} \rightarrow \mathsf{NP}\ \mathsf{VP} \\ & \mathsf{VP} \rightarrow \mathsf{V}_t\ \mathsf{CP} \mid \mathsf{V}_i \\ & \mathsf{CP} \rightarrow \mathsf{C}\ \mathsf{S} \\ & \mathsf{NP} \rightarrow \mathsf{D}\ \mathsf{N} \\ & \mathsf{N} \rightarrow \text{cat} \mid \text{dog} \mid \text{zyzzyva} \\ & \mathsf{V}_t \rightarrow \text{said} \mid \text{noticed} \\ & \mathsf{V}_i \rightarrow \text{meows} \mid \text{barks} \\ & \mathsf{C} \rightarrow \text{that} \\ & \mathsf{D} \rightarrow \text{the} \mid \text{a} \end{aligned}$$

$$\begin{aligned} L(G) = \;& \text{the cat meows} \\ & \text{a zyzzyva barks} \\ & \text{the dog said that a cat barks} \\ & \text{the zyzzyva noticed that a dog meows} \\ & \vdots \end{aligned}$$

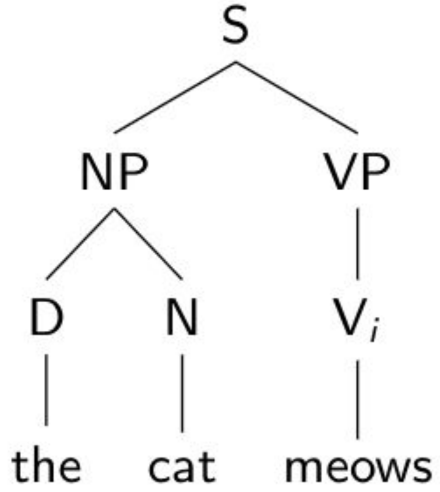
Grammars for Natural Language

$N = \{\mathsf{S}, \mathsf{NP}, \mathsf{VP}, \mathsf{CP}, \mathsf{C}, \mathsf{N}, \mathsf{D}, \mathsf{V}_t, \mathsf{V}_i\}$

$\Sigma = \{\text{a, the, that, said, meows, barks, noticed, cat, dog, zyzzyva}\}$

$$
\begin{aligned}
P = \;& \mathsf{S} \rightarrow \mathsf{NP}\ \mathsf{VP} \\
& \mathsf{VP} \rightarrow \mathsf{V}_t\ \mathsf{CP} \mid \mathsf{V}_i \\
& \mathsf{CP} \rightarrow \mathsf{C}\ \mathsf{S} \\
& \mathsf{NP} \rightarrow \mathsf{D}\ \mathsf{N} \\
& \mathsf{N} \rightarrow \text{cat} \mid \text{dog} \mid \text{zyzzyva} \\
& \mathsf{V}_t \rightarrow \text{said} \mid \text{noticed} \\
& \mathsf{V}_i \rightarrow \text{meows} \mid \text{barks} \\
& \mathsf{C} \rightarrow \text{that} \\
& \mathsf{D} \rightarrow \text{the} \mid \text{a}
\end{aligned}
$$

$L(G) = \text{the cat meows}$



Grammar Induction

- What is G such that L(G) = Human Language?

  (A huge chunk of linguistics is devoted to finding this G)

- Data-driven approach: can we learn G from observed sentences alone?

Grammar Induction

- Learn an underlying grammar (syntax) from observed sentences alone

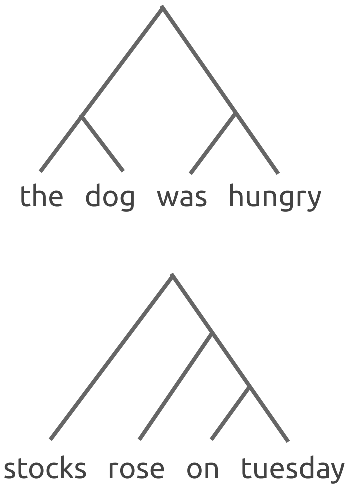
i like superhero movies
the dog was hungry
stocks rose on tuesday
schools should use interfolio
it is snowing in boston
time flies like an arrow
i saw an elephant in my pajamas

⋮

$$N = \{\mathsf{S}, \mathsf{NT}_1, \mathsf{NT}_2, \ldots, \mathsf{NT}_T\}$$

$$\Sigma = \{\mathsf{a}, \mathsf{aardvark}, \mathit{are}, \ldots, \mathsf{zyzzyva}\}$$

$$\begin{aligned} P = \ & \mathsf{S} \to \mathsf{NT}_1\ \mathsf{NT}_2 \\ & \mathsf{NT}_1 \to \mathsf{NT}_1\ \mathsf{NT}_4 \\ & \mathsf{NT}_3 \to \mathsf{NT}_2\ \mathsf{NT}_5 \\ & \mathsf{NT}_5 \to \mathsf{NT}_4\ \mathsf{NT}_1 \\ & \quad \vdots \\ & \mathsf{NT}_{10} \to \mathsf{dog} \\ & \mathsf{NT}_{10} \to \mathsf{is} \\ & \mathsf{NT}_{32} \to \mathsf{in} \\ & \mathsf{NT}_7 \to \mathsf{the} \end{aligned}$$



Tangent #1: “The Poverty of the Stimulus”

1. Children acquire the syntax of their native language by 4~5 years and can generalize in sophisticated ways.

2. They have not been exposed to enough data to learn such generalizations.

3. Therefore, there must be an innate “language acquisition device” (“universal grammar”) that children are equipped with at birth.

Tangent #1: “The Poverty of the Stimulus”

“If a Martian linguist were to visit Earth, he would deduce from the evidence that there was only one language, with a number of local variants.”

Tangent #1: “The Poverty of the Stimulus”

- Grammar induction ⇒ Data-driven acquisition of syntax is possible!

- Depending on how much bias one builds into the learning system, successful grammar induction can be used as an empirical argument against the poverty of the stimulus.

Tangent #2: Grammar vs. Parsing

- Grammar $\Rightarrow$ Parser, but Parser $\not\Rightarrow$ Grammar.
- Can learn an unsupervised parser without learning a grammar.
  (most prior work in unsupervised parsing has been in this vein)
- Robust neurobiological evidence for human parsing, much less for grammars.

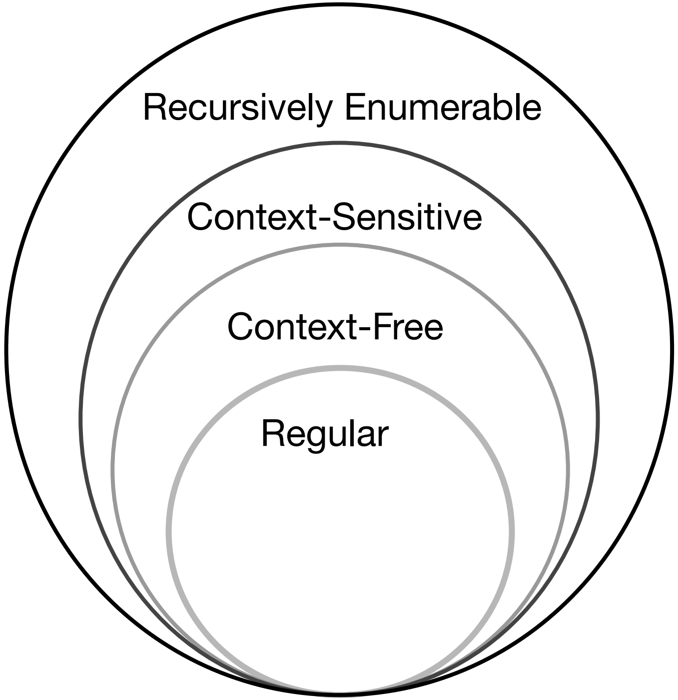
Tangent #2: Grammar vs. Parsing

Why learn a grammar?

- Theoretically appealing.
- Grammar can explain how humans can recognize (i.e. parse) *and* generate an infinite number of sentences.
- Some experimental evidence that Grammar = Parser.

  Children can start using syntactic rules for generation immediately after learning to recognize it [McKee et al ’93]

Tangent #3: Grammars & Formal Languages



N : set of nonterminal symbols

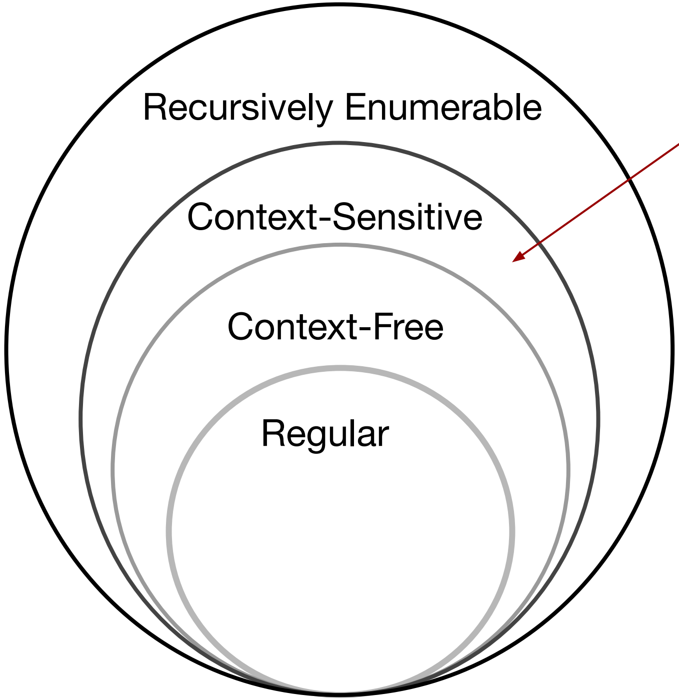
Σ : set of terminal symbols

P : set of production rules

$L(G)$	Rules (P)
Recursively Enumerable	$\gamma \to \beta$
Context-Sensitive	$\alpha A \beta \to \alpha \gamma \beta$
Context-Free	$A \to \alpha$
Regular	$A \to a, A \to aB, A \to \epsilon$

$\gamma \in (N \cup \Sigma)^+ \quad \alpha, \beta \in (N \cup \Sigma)^* \quad A, B \in N \quad a \in \Sigma$

Tangent #3: Grammars & Formal Languages

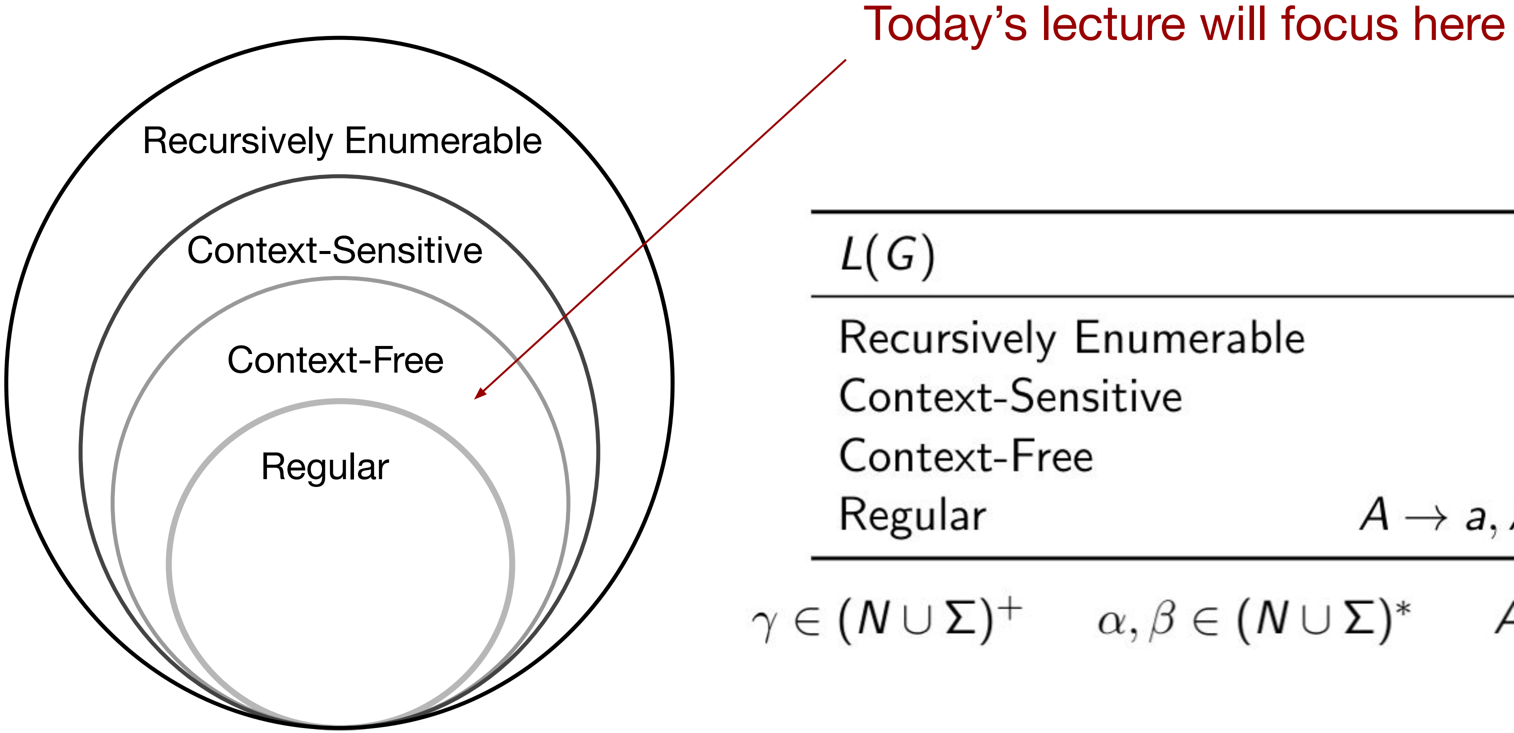


Human language thought to be here
(mildly context-sensitive)

$L(G)$	Rules (P)
Recursively Enumerable	$\gamma \rightarrow \beta$
Context-Sensitive	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Context-Free	$A \rightarrow \alpha$
Regular	$A \rightarrow a, A \rightarrow aB, A \rightarrow \epsilon$

$\gamma \in (N \cup \Sigma)^+ \quad \alpha, \beta \in (N \cup \Sigma)^* \quad A, B \in N \quad a \in \Sigma$

Tangent #3: Grammars & Formal Languages



$L(G)$	Rules (P)
Recursively Enumerable	$\gamma \rightarrow \beta$
Context-Sensitive	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Context-Free	$A \rightarrow \alpha$
Regular	$A \rightarrow a, A \rightarrow aB, A \rightarrow \epsilon$

$\gamma \in (N \cup \Sigma)^+ \quad \alpha, \beta \in (N \cup \Sigma)^* \quad A, B \in N \quad a \in \Sigma$

Outline

- Motivation
- Review: Formal grammars
- **Probabilistic grammar induction**
- Recent approaches for grammar induction & unsupervised parsing
- Conclusion

Probabilistic Modeling

- Probabilistic grammars: associate a probability to each rule.

$$\mathsf{S} \rightarrow \mathsf{A}_5\ \mathsf{A}_7 \qquad\qquad p_\pi(\mathsf{S} \rightarrow \mathsf{A}_5\ \mathsf{A}_7)$$

Probabilistic Modeling

- Probabilistic grammars: associate a probability to each rule.

$$\mathsf{S} \rightarrow \mathsf{A}_5\ \mathsf{A}_7 \qquad\qquad p_\pi(\mathsf{S} \rightarrow \mathsf{A}_5\ \mathsf{A}_7)$$

- Induces a distribution over surface strings (language model)

$$x \in \Sigma^* \qquad\qquad p_{\boldsymbol{\pi}}(x)$$

$G = (N, \Sigma, P, S)$

N : set of nonterminal symbols

Σ : set of terminal symbols

P : set of production rules

Probabilistic Modeling

- Probabilistic grammars: associate a probability to each rule.

$$\mathsf{S} \rightarrow \mathsf{A}_5\ \mathsf{A}_7 \qquad\qquad p_\pi(\mathsf{S} \rightarrow \mathsf{A}_5\ \mathsf{A}_7)$$

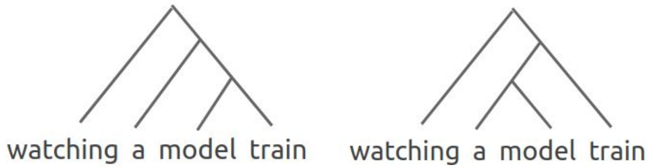
- Induces a distribution over surface strings (language model)

$$x \in \Sigma^* \qquad\qquad p_{\boldsymbol{\pi}}(x)$$

- **π** := rule probabilities (probabilistic grammars)
  := RNN / Transformer parameters (neural language models)
  := n-gram probabilities (count-based language models)

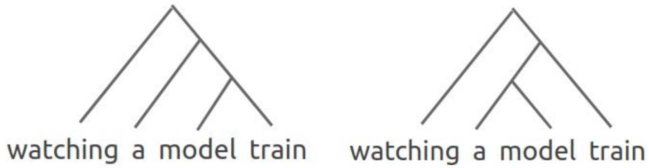
Probabilistic Modeling

- Why probabilistic grammars?
  - Naturally model uncertainty/ambiguity



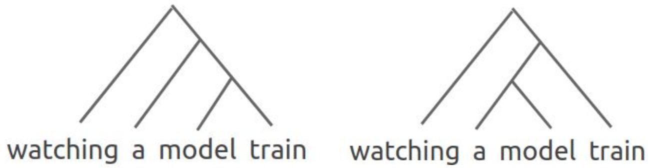
Probabilistic Modeling

- Why probabilistic grammars?
    - Naturally model uncertainty/ambiguity
    - Favorable learnability results:
      [Gold ’67]: Cannot even learn regular grammars from positive samples alone
      [Horning ’69]: Probabilistic grammars are learnable from positive samples



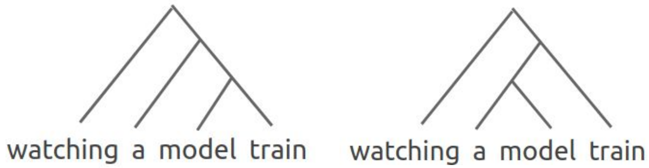
Probabilistic Modeling

- Why probabilistic grammars?
  - Naturally model uncertainty/ambiguity
  - Favorable learnability results:
    [Gold ’67]: Cannot even learn regular grammars from positive samples alone
    [Horning ’69]: Probabilistic grammars are learnable from positive samples
  - Information-theoretic interpretation: grammar induction corresponds to finding a grammar that can best compress the data statistically



Probabilistic Modeling

- Why probabilistic grammars?

  - Naturally model uncertainty/ambiguity
  - Favorable learnability results:
    [Gold ’67]: Cannot even learn regular grammars from positive samples alone
    [Horning ’69]: Probabilistic grammars are learnable from positive samples
  - Information-theoretic interpretation: grammar induction corresponds to finding a grammar that can best compress the data statistically
  - Natural objective to optimize: likelihood of corpus

PCFG Induction

- Classic approach: assume each sentence is generated from a probabilistic context-free grammar (PCFG).

$$\mathbf{x} \sim \mathrm{PCFG}(N, \Sigma, P, \boldsymbol{\pi})$$

PCFG Induction

- Classic approach: assume each sentence is generated from a probabilistic context-free grammar (PCFG).

$$\mathbf{x} \sim \mathrm{PCFG}(N, \Sigma, P, \pi)$$

Nonterminal symbols (50 - 100)

$$N = \{\mathsf{S}, \mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_T\}$$

PCFG Induction

- Classic approach: assume each sentence is generated from a probabilistic context-free grammar (PCFG).

$$\mathbf{x} \sim \mathrm{PCFG}(N, \Sigma, P, \pi)$$

Nonterminal symbols (50 - 100)

$$N = \{\mathrm{S}, \mathrm{A}_1, \mathrm{A}_2, \ldots, \mathrm{A}_T\}$$

Special “Start” symbol

PCFG Induction

- Classic approach: assume each sentence is generated from a probabilistic context-free grammar (PCFG).

$$\mathbf{x} \sim \mathrm{PCFG}(N, \Sigma, P, \pi)$$

$$N = \{\mathsf{S}, \mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_T\}$$

Terminal symbols (10K - 100K)

$$\Sigma = \{\mathsf{a}, \mathsf{aardvark}, \mathsf{able}, \mathsf{are}, \ldots, \mathsf{zyzzyva}\}$$

PCFG Induction

- Classic approach: assume each sentence is generated from a probabilistic context-free grammar (PCFG).

$$\mathbf{x} \sim \mathrm{PCFG}(N, \Sigma, P, \pi)$$

$N = \{\mathsf{S}, \mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_T\}$

$\Sigma = \{\mathsf{a}, \mathsf{aardvark}, \mathsf{able}, \mathsf{are}, \ldots, \mathsf{zyzzyva}\}$

Context-free rules
(all possible unary/binary rules: O(|N||∑| + |N|^3)

$$P = \begin{array}{lcl} \mathsf{S} \to \mathsf{A}_1\ \mathsf{A}_1 & & \mathsf{A}_1 \to \mathsf{zyzzyva} \\ \mathsf{S} \to \mathsf{A}_1\ \mathsf{A}_2 & \cdots & \mathsf{A}_2 \to \mathsf{a} \\ \mathsf{S} \to \mathsf{A}_1\ \mathsf{A}_3 & & \mathsf{A}_2 \to \mathsf{aardvark} \end{array}$$

PCFG Induction

- Classic approach: assume each sentence is generated from a probabilistic context-free grammar (PCFG).

$$\mathbf{x} \sim \mathrm{PCFG}(N, \Sigma, P, \pi)$$

$$N = \{\mathsf{S}, \mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_T\}$$

$$\Sigma = \{\mathsf{a}, \mathsf{aardvark}, \mathsf{able}, \mathsf{are}, \ldots, \mathsf{zyzzyva}\}$$

$$P = \begin{array}{lcl} \mathsf{S} \to \mathsf{A}_1\ \mathsf{A}_1 & & \mathsf{A}_1 \to \mathsf{zyzzyva} \\ \mathsf{S} \to \mathsf{A}_1\ \mathsf{A}_2 & \cdots & \mathsf{A}_2 \to \mathsf{a} \\ \mathsf{S} \to \mathsf{A}_1\ \mathsf{A}_3 & & \mathsf{A}_2 \to \mathsf{aardvark} \end{array}$$

Rule probabilities

$$\pi = \{p_\pi(r) \mid r \in P\}$$

$$p_\pi(\mathsf{S} \to \mathsf{A}_5\ \mathsf{A}_7)$$

$$p_\pi(\mathsf{A}_5 \to \mathsf{John})$$

PCFG Induction

- Classic approach: assume each sentence is generated from a probabilistic context-free grammar (PCFG).

$$\mathbf{x} \sim \mathrm{PCFG}(N, \Sigma, P, \boxed{\pi})$$

$$N = \{\mathsf{S}, \mathsf{A}_1, \mathsf{A}_2, \dots, \mathsf{A}_T\}$$

$$\Sigma = \{\mathsf{a}, \mathsf{aardvark}, \mathsf{able}, \mathsf{are}, \dots, \mathsf{zyzzyva}\}$$

$$P = \begin{array}{lcl} \mathsf{S} \to \mathsf{A}_1\ \mathsf{A}_1 & & \mathsf{A}_1 \to \mathsf{zyzzyva} \\ \mathsf{S} \to \mathsf{A}_1\ \mathsf{A}_2 & \cdots & \mathsf{A}_2 \to \mathsf{a} \\ \mathsf{S} \to \mathsf{A}_1\ \mathsf{A}_3 & & \mathsf{A}_2 \to \mathsf{aardvark} \end{array}$$

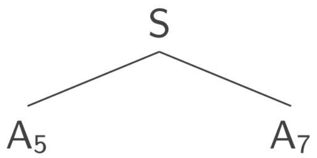
$$\pi = \{p_\pi(r) \mid r \in P\}$$

Probabilistic grammar induction
⇒ Learning rule probabilities from data

PCFG Example

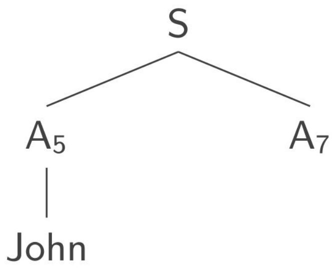
S

PCFG Example



$$S \rightarrow A_5\ A_7$$

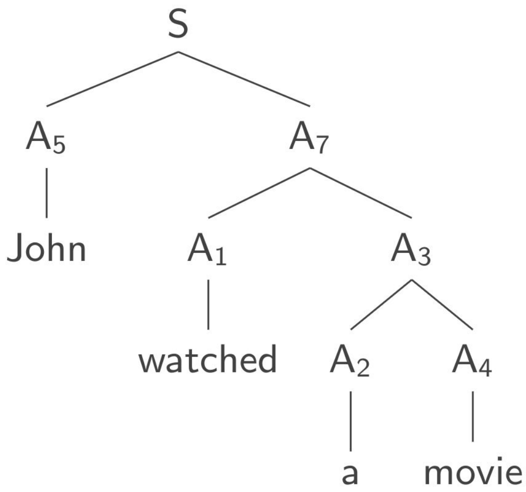
PCFG Example



$$S \rightarrow A_5\ A_7$$

$$A_5 \rightarrow \text{John}$$

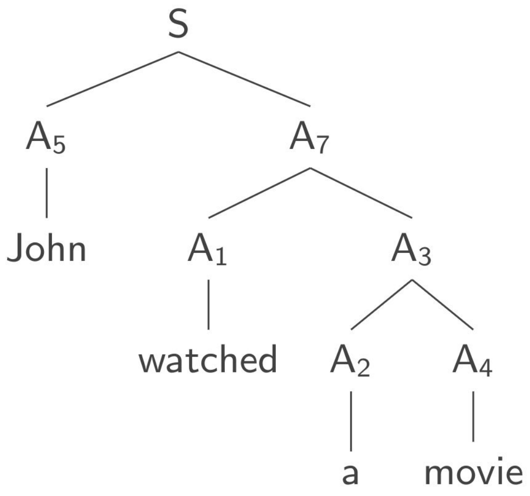
PCFG Example



$$S \rightarrow A_5\ A_7$$
$$A_5 \rightarrow \text{John}$$
$$A_7 \rightarrow A_1\ A_3$$
$$A_1 \rightarrow \text{watched}$$
$$A_3 \rightarrow A_2\ A_4$$
$$A_2 \rightarrow \text{a}$$
$$A_4 \rightarrow \text{movie}$$

PCFG Example

p_π



$$= p_\pi(\mathsf{S} \to \mathsf{A}_5\ \mathsf{A}_7) \times p_\pi(\mathsf{A}_5 \to \mathsf{John}) \times p_\pi(\mathsf{A}_7 \to \mathsf{A}_1\ \mathsf{A}_3) \times p_\pi(\mathsf{A}_1 \to \mathsf{watched}) \times p_\pi(\mathsf{A}_3 \to \mathsf{A}_2\ \mathsf{A}_4) \times p_\pi(\mathsf{A}_2 \to \mathsf{a}) \times p_\pi(\mathsf{A}_4 \to \mathsf{movie})$$

Probability of a tree

$$\boxed{p_\pi(\boldsymbol{t})} = \prod_{r \in \boxed{P_t}} p_\pi(r)$$

Set of rules used to derive the tree

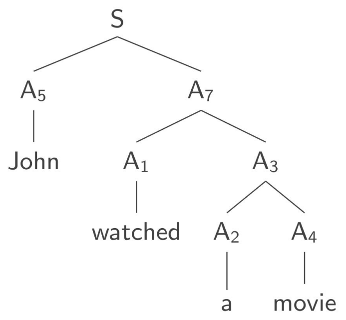
PCFG as a generative model of language

Autoregressive Language Model

$$p_\theta\left(\text{John watched a movie}\right) = \begin{array}{l} p_\theta(\text{John}) \times \\ p_\theta(\text{watched} \mid \text{John}) \times \\ p_\theta(\text{a} \mid \text{John watched}) \times \\ p_\theta(\text{movie} \mid \text{John watched a}) \end{array}$$

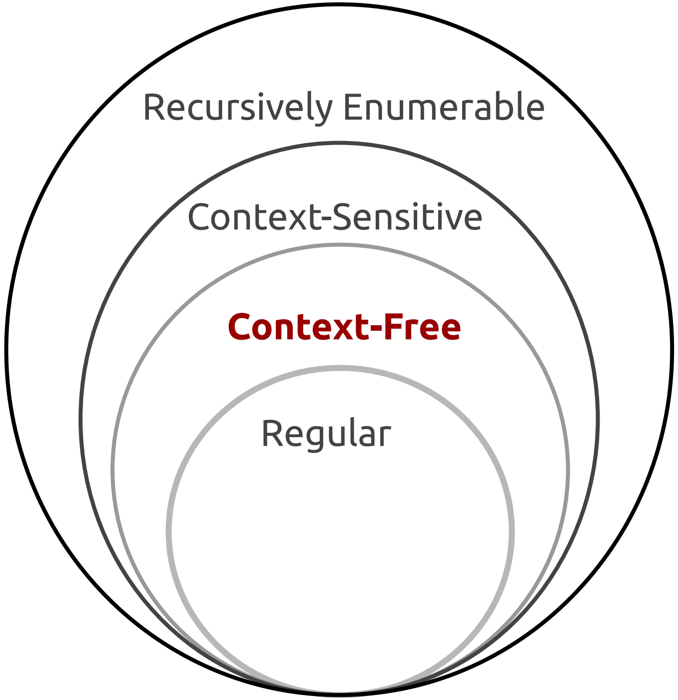
PCFG

$$p_{\boldsymbol{\pi}}\left(\quad \right)$$



$$= \begin{array}{l} p_\pi(\text{S} \to \text{A}_5\ \text{A}_7) \times \\ p_\pi(\text{A}_5 \to \text{John}) \times \\ p_\pi(\text{A}_7 \to \text{A}_1\ \text{A}_3) \times \\ p_\pi(\text{A}_1 \to \text{watched}) \times \\ p_\pi(\text{A}_3 \to \text{A}_2\ \text{A}_4) \times \\ p_\pi(\text{A}_2 \to \text{a}) \times \\ p_\pi(\text{A}_4 \to \text{movie}) \end{array}$$

Why context-free grammars?



- Reasonably fast (cubic) algorithms for learning.
- Many human language phenomena can be captured by context-free grammars [Pullum and Gazdar '82].
- … though not all [Shieber '85].

PCFG Training

- Supervised case: **t** is observed, so can just maximize likelihood:

$$\sum_{m=1}^{M} \log p_{\boldsymbol{\pi}}(\boldsymbol{t}^{(m)})$$

- MLE solution corresponds to just counting and dividing observed rules, as in n-gram language models.

PCFG Training

- Unsupervised case: only the leaves **x** are observed

- MLE: maximize the likelihood of **x**

$$\max_{\pi} \sum_{m=1}^{M} \log p_{\pi}(\mathrm{x}^{(m)})$$

PCFG Training

- Unsupervised case: only the leaves **x** are observed

- MLE: maximize the likelihood of **x**

$$\max_{\pi} \sum_{m=1}^{M} \log p_{\pi}(\mathrm{x}^{(m)}) = \max_{\pi} \sum_{m=1}^{M} \log \left(\sum_{\boldsymbol{t} \in \mathcal{T}(\mathrm{x}^{(m)})} p_{\pi}(\boldsymbol{t}) \right)$$

$$\mathcal{T}(\mathrm{x}) = \{\boldsymbol{t} \text{ such that its leaves are } \mathrm{x}\}$$

- Marginalize out unseen structure

PCFG Training

- Marginalization with dynamic programming

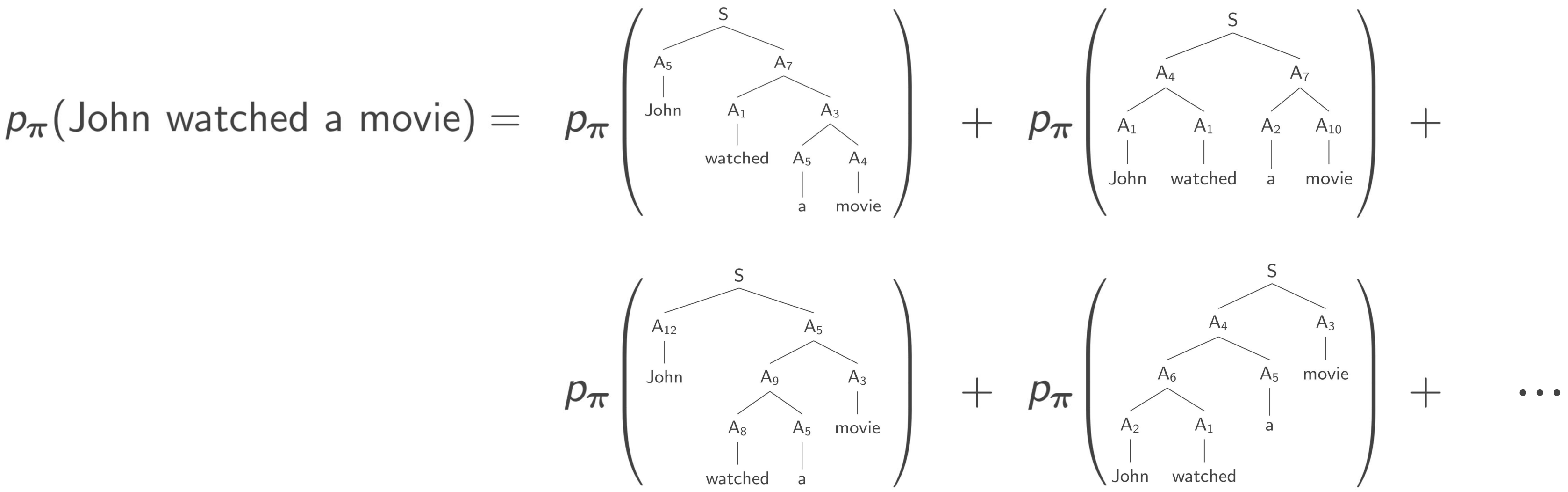
$$p_{\boldsymbol{\pi}}(x) = \sum_{\boldsymbol{t} \in \mathcal{T}(x)} p_{\boldsymbol{\pi}}(\boldsymbol{t})$$

PCFG Training

- Marginalization with dynamic programming

$$p_\pi(x) = \sum_{\boldsymbol{t} \in \mathcal{T}(x)} p_\pi(\boldsymbol{t})$$

Sum over an exponentially-sized with dynamic programming



PCFG Training: Inside Algorithm

- Inside algorithm to calculate marginal likelihood (generalization of backward algorithm in HMMs) $p_{\boldsymbol{\pi}}(x) = \sum_{\boldsymbol{t} \in \mathcal{T}(x)} p_{\boldsymbol{\pi}}(\boldsymbol{t})$

- Define the "inside" variables as

$$\beta[s, t, A] = \text{Prob(nonterminal } A \text{ expands to } x_{s:t})$$

- Then marginal likelihood given by

$$p_{\boldsymbol{\pi}}(x) = \beta[1, L, S]$$

PCFG Training: Inside Algorithm

$G = (N, \Sigma, P, S)$

N : set of nonterminal symbols

Σ : set of terminal symbols

P : set of production rules

S : start symbol $(S \in N)$

- Bottom-up dynamic programming

  Initialization:

$$
\begin{aligned}
&\text{for } i = 1, \ldots L \\
&\quad \text{for } C \in N \\
&\qquad \beta[i, i, C] = p_\pi(C \to x_i)
\end{aligned}
$$

Initialize “width-0” span probabilities (i.e. words)
ith unary expansion probabilities

PCFG Training: Inside Algorithm

$G = (N, \Sigma, P, S)$

N : set of nonterminal symbols

Σ : set of terminal symbols

P : set of production rules

S : start symbol $(S \in N)$

- Bottom-up dynamic programming

  Recursion:

  $$\text{for } w = 1, \ldots, L-1$$

For all spans with width w

PCFG Training: Inside Algorithm

$G = (N, \Sigma, P, S)$

N : set of nonterminal symbols

Σ : set of terminal symbols

P : set of production rules

S : start symbol $(S \in N)$

- Bottom-up dynamic programming

  Recursion:

$$\begin{array}{l} \text{for } w = 1, \ldots, L-1 \\ \quad \text{for } s = 1, \ldots, L-w \end{array}$$

For all spans with width w

For all spans starting at position s

PCFG Training: Inside Algorithm

$G = (N, \Sigma, P, S)$

N : set of nonterminal symbols

Σ : set of terminal symbols

P : set of production rules

S : start symbol $(S \in N)$

- Bottom-up dynamic programming

  Recursion:

$$
\begin{array}{l}
\text{for } w = 1, \ldots, L-1 \\
\quad \text{for } s = 1, \ldots, L-w \\
\quad\quad t = s + w \\
\quad\quad \text{for } k = s, \ldots, t-1
\end{array}
$$

For all spans with width w

For all spans starting at position s

Span end position t

For all possible ways to break up span (s,t)

PCFG Training: Inside Algorithm

$G = (N, \Sigma, P, S)$

N : set of nonterminal symbols

Σ : set of terminal symbols

P : set of production rules

S : start symbol $(S \in N)$

- Bottom-up dynamic programming

Recursion:

for $w = 1, \ldots, L-1$ — For all spans with width w

for $s = 1, \ldots, L-w$ — For all spans starting at position s

$t = s + w$ — Span end position t

for $k = s, \ldots, t-1$ — For all possible ways to break up span (s,t)

for all rules $A \rightarrow BC \in P$ — For all possible rules of the form A ⇒ BC

$$\beta[s,t,A] \mathrel{+}= \beta[s,k,B]\beta[k+1,t,C]p_\pi(A \rightarrow BC)$$

PCFG Training: Inside Algorithm

- A version of the CKY algorithm: $O(|P|L^3)$

- Each step of the dynamic program to calculate $p_{\pi}(x) = \beta[1, L, S]$ is just a series of multiplications / additions

  ⇒ we can easily calculate $\nabla_{\pi} \log p_{\pi}(x)$

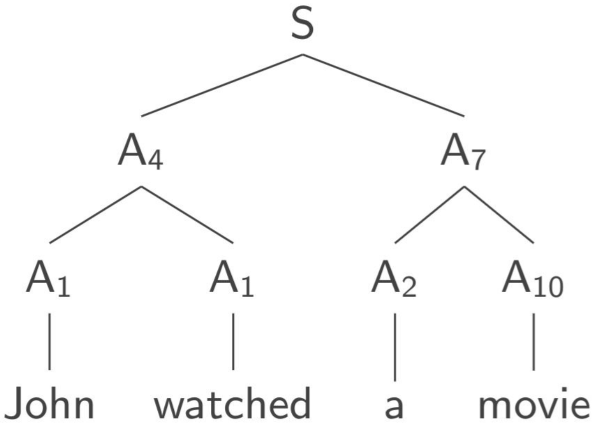
- Can ensure **π** are probabilities by working in logit space:

$$\boldsymbol{\pi} = \text{softmax}(\boldsymbol{\theta})$$

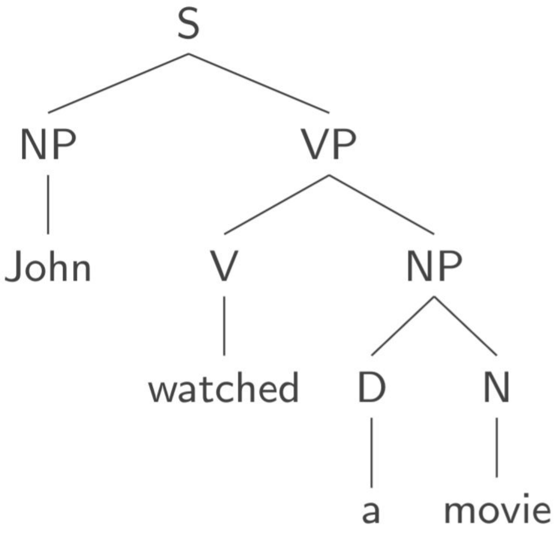
- Gradient ascent on log marginal likelihood = (one version of the) expectation maximization algorithm [Dempster ’77].

Unsupervised Parsing Evaluation

- Predicted trees compared against linguistic trees ignoring label alignment.



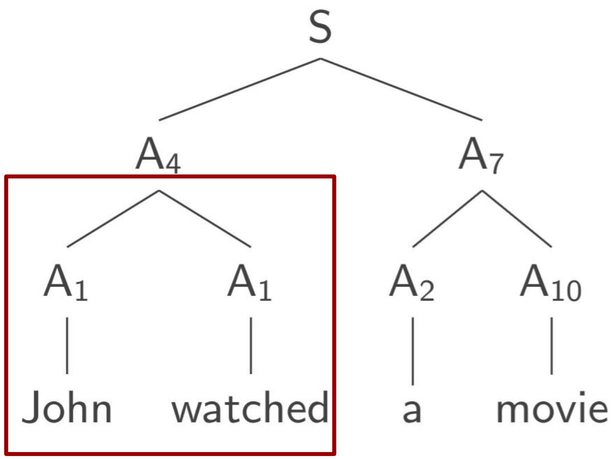
Predicted

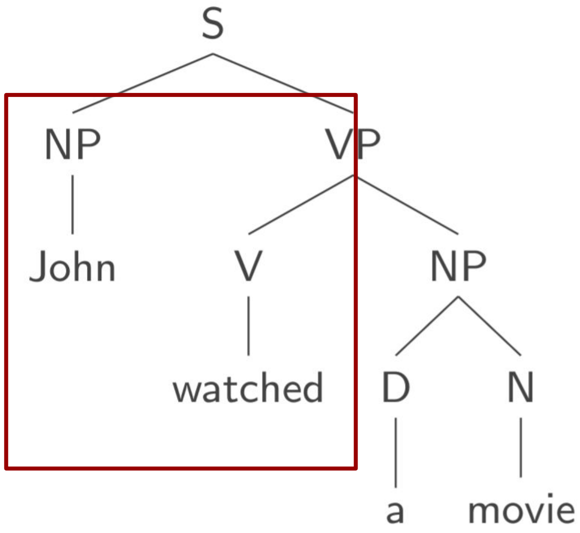


Linguistic Annotation

Unsupervised Parsing Evaluation

- Predicted trees compared against linguistic trees ignoring label alignment.





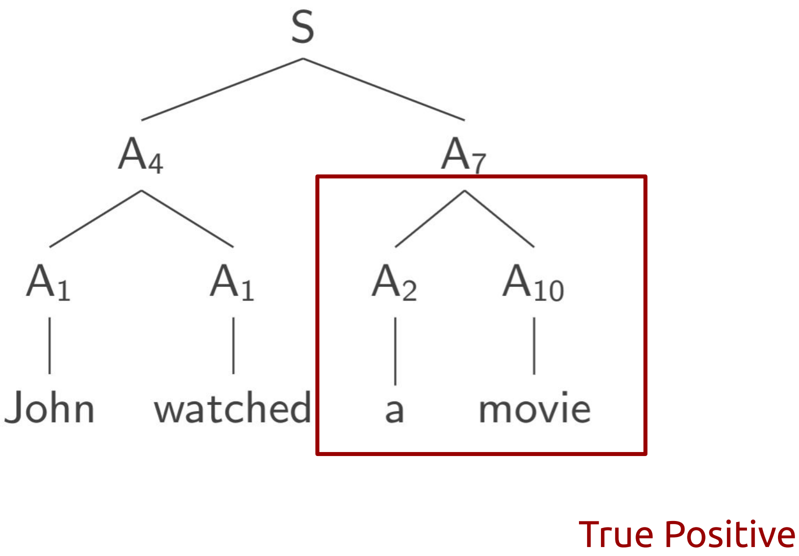
False Positive

Predicted

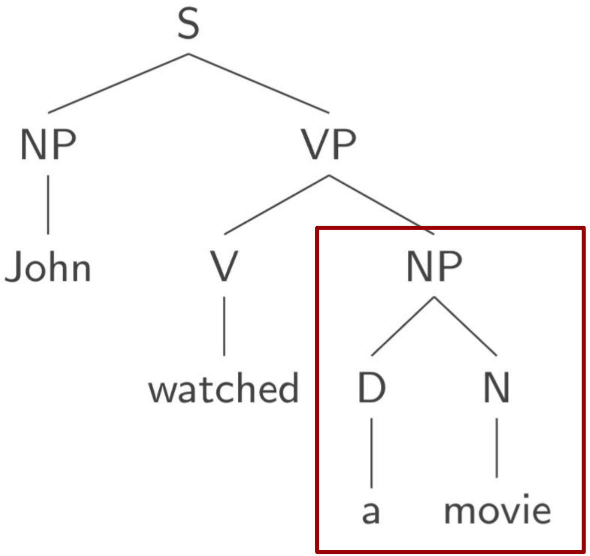
Linguistic Annotation

Unsupervised Parsing Evaluation

- Predicted trees compared against linguistic trees ignoring label alignment.



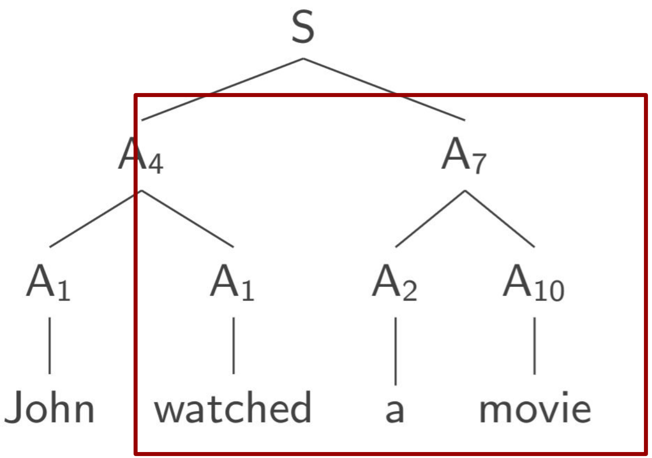
Predicted



Linguistic Annotation

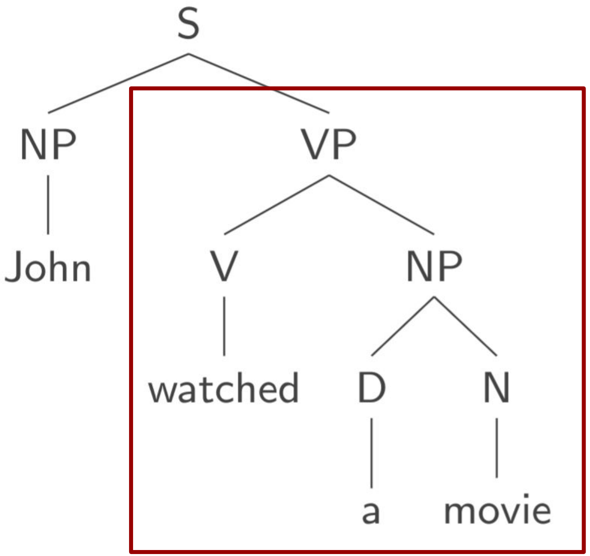
Unsupervised Parsing Evaluation

- Predicted trees compared against linguistic trees ignoring label alignment.



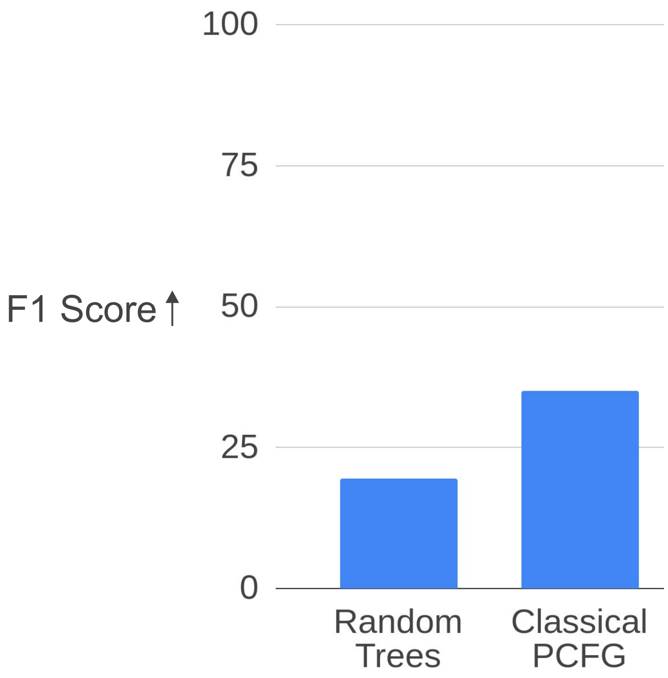
Predicted

False Negative

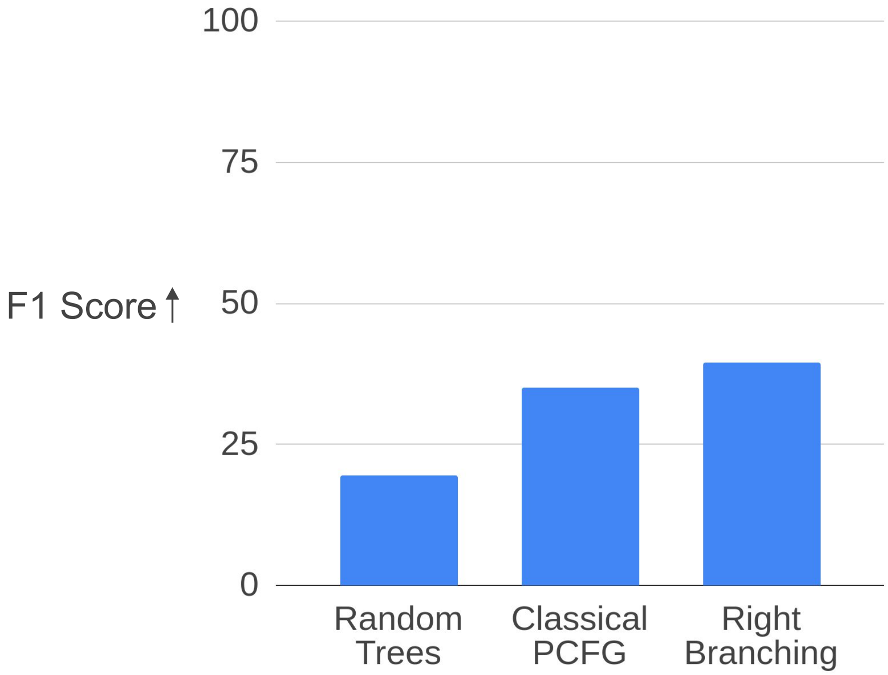


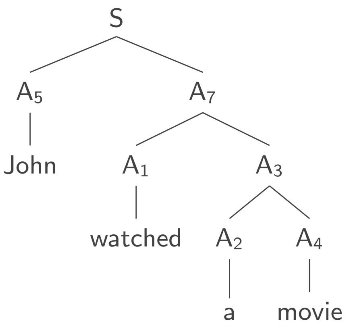
Linguistic Annotation

Results with Simple PCFG induction



Results with Simple PCFG induction





Why doesn’t PCFG induction “work”?

- Complex optimization landscape (non-convex)
- PCFG model is too simple
- But no one really knows… ¯_(ツ)_/¯

History of Unsupervised Constituency Parsing

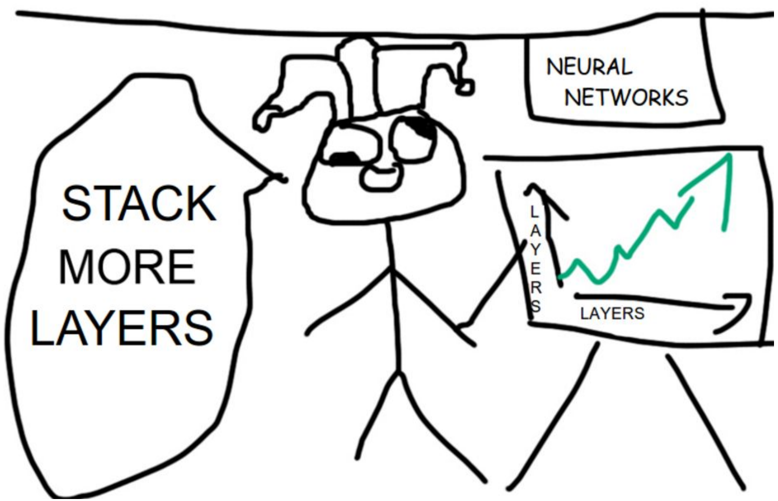
- Decades of negative results from 90s [Carroll and Charniak ’92]
- Inspired rich line of work on alternative approaches to unsupervised parsing [Clark ’01, Klein and Manning ’02, Bod ’06, Seginer ’07]...
- Some success on unsupervised parsing with simple setups:
    - Part-of-speech tags as inputs
    - Train/evaluate on short sentences (<20 words)
    - Incorporate heuristics (e.g. based on punctuation)

Outline

- Motivation
- Review: Formal grammars
- Probabilistic grammar induction
- **Recent approaches for grammar induction & unsupervised parsing**
- Conclusion

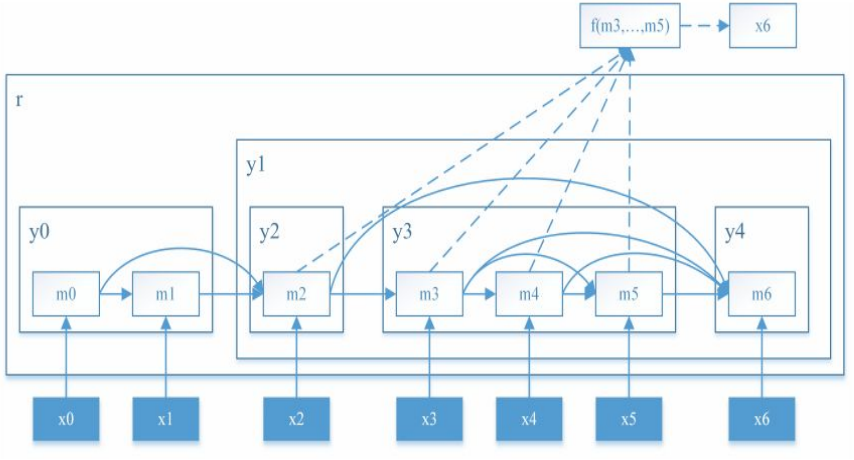
Recent Work

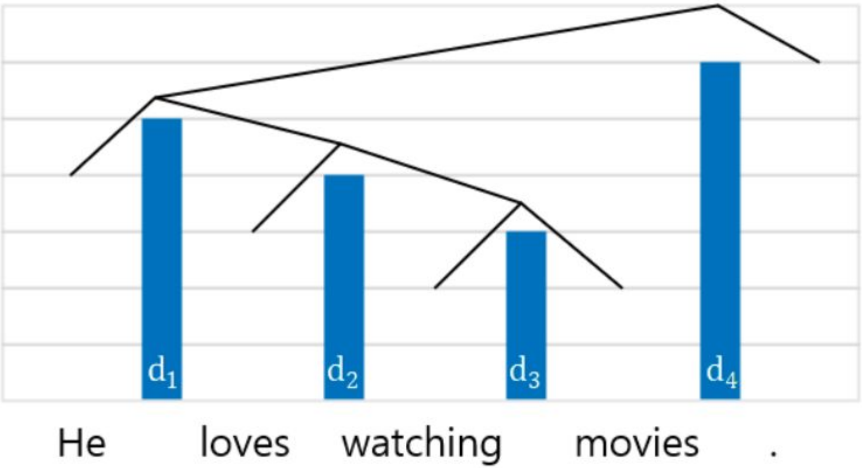
- Induce parse trees directly from words on full-length sentences.
- Employ neural networks “somewhere” in the pipeline



Gating Functions within Neural Language Models

- Parsing-Reading-Predict Network [Shen et al ’18]

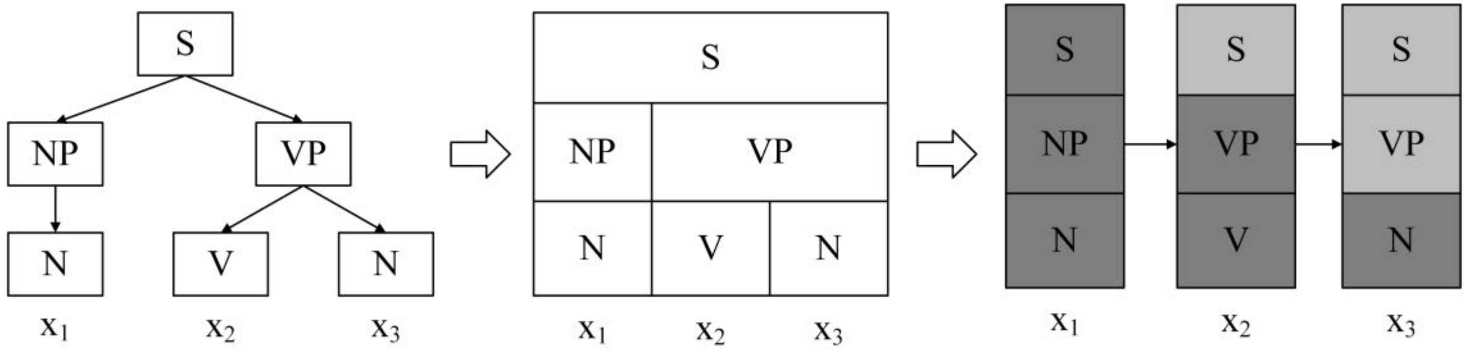




Attention over previous words mediated by (predicted) “syntactic distance” between two words ⇒ use syntactic distance to derive trees

Gating Functions within Neural Language Models

- Ordered Neurons [Shen et al '19]

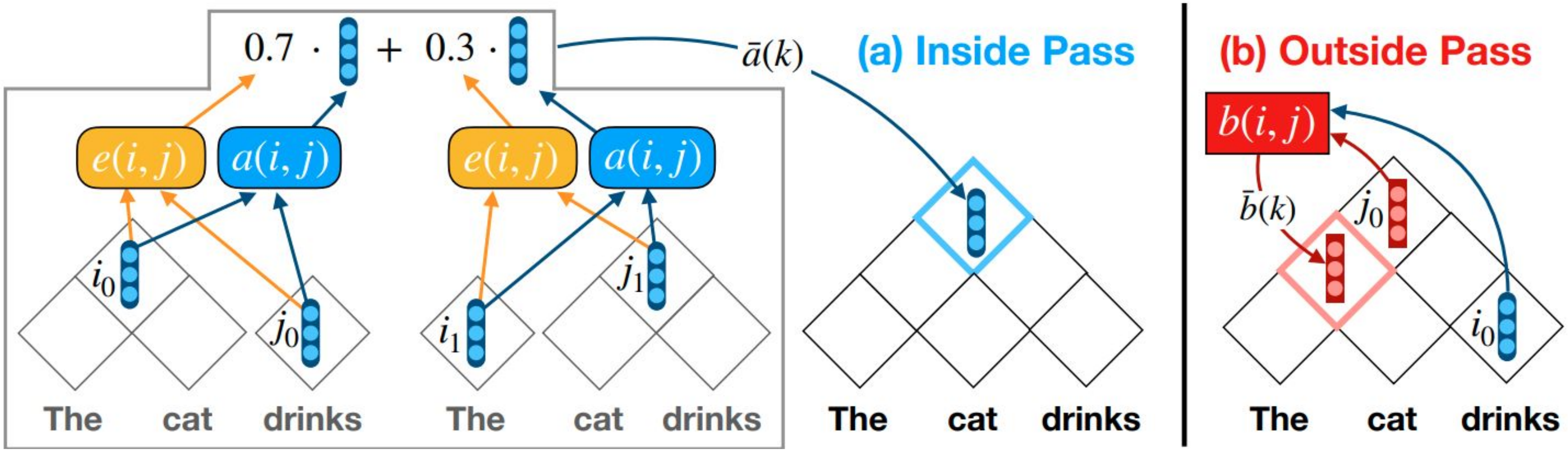


(a) Constituency tree (b) Block view (c) ON-LSTM cell states

Softly partition the hidden states of an LSTM into "blocks" which represent constituents

Structured Autoencoder

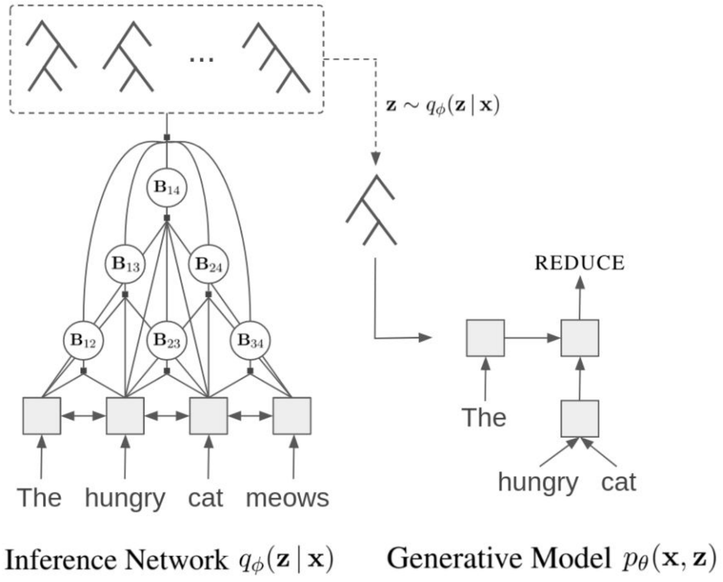
- Deep Inside-Outside Recursive Autoencoder [Drozdov et al ’19]



Autoencoder tries to obtain (soft) spans that best reconstruct the leaf word embeddings (pretrained)

Structured Autoencoder

- Unsupervised Recurrent Neural Network Grammars [Kim et al ’19]



Variational autoencoder where the latent variable is a parse tree, and the generative model is a syntax-aware language model.

“Neuralizing” Classic PCFGs

- Neural PCFG [Kim et al ’19]: Use neural networks over symbol embeddings parameterize rule probabilities

“Neural” Language Models

$$
\begin{array}{r}
\textit{a} \\
\textit{aardvark} \\
\textit{able} \\
\textit{are} \\
\vdots \\
\textit{zyzzyva}
\end{array}
\begin{bmatrix}
1.2 & -0.1 & 0.3 & \dots & 0.1 \\
0.2 & 0.7 & -0.4 & \dots & 1.1 \\
-0.7 & 0.5 & 0.6 & \dots & -0.8 \\
0.1 & 0.9 & 0.8 & \dots & 0.7 \\
 & & \vdots & & \\
0.3 & -0.2 & 0.7 & \dots & 0.4
\end{bmatrix}
$$

$$\boldsymbol{e} = \textsc{Embed}(\text{trading})$$
$$\boldsymbol{h} = \textsc{NeuralNet}(\boldsymbol{e})$$
$$p_\theta(\text{high} \mid \text{trading}) \propto \exp(\boldsymbol{h}^\top \mathbf{w}_{\text{high}})$$

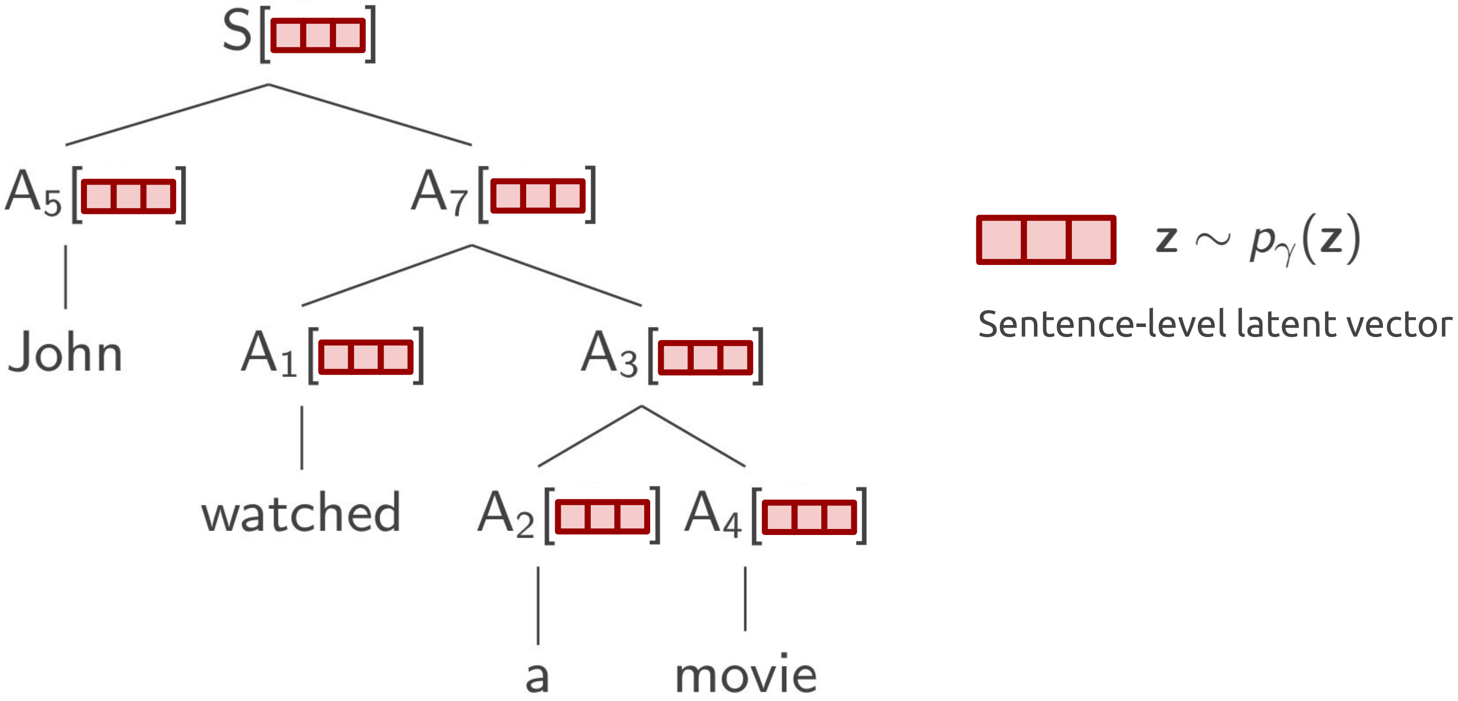
“Neural” PCFG

$$
\begin{array}{r}
\mathrm{S} \\
\mathrm{A}_1 \\
\mathrm{A}_2 \\
\mathrm{A}_3 \\
\vdots \\
\mathrm{A}_T
\end{array}
\begin{bmatrix}
0.2 & -0.6 & 0.3 & \dots & 1.8 \\
1.4 & 0.7 & -0.4 & \dots & 0.1 \\
-0.7 & 0.3 & 0.6 & \dots & -0.9 \\
0.4 & 0.9 & 0.8 & \dots & 0.7 \\
 & & \vdots & & \\
0.5 & -0.1 & 0.3 & \dots & 0.8
\end{bmatrix}
$$

$$\boldsymbol{e} = \textsc{Embed}(\mathrm{S})$$
$$\boldsymbol{h} = \textsc{NeuralNet}(\boldsymbol{e})$$
$$p_\pi(\mathrm{S} \to \mathrm{A}_1\ \mathrm{A}_4) \propto \exp(\boldsymbol{h}^\top \boldsymbol{w}_{\mathrm{A}_1, \mathrm{A}_4})$$

“Neuralizing” Classic PCFGs

- Compound PCFG [Kim et al ’19]: Learn richer grammars with neural variational inference



Linguistically Motivated Grammaticality Tests

- Create new sentences via “constituency tests” [Cao et al ’20]

  The quick brown fox jumped over the lazy dog.

  The quick brown fox jumped over it.

  The lazy dog, the quick brown fox jumped over.

Linguistically Motivated Grammaticality Tests

- Create new sentences via “constituency tests” [Cao et al ’20]

  The quick brown fox jumped over the lazy dog.

  *The quick brown fox it the lazy dog.

  *Jumped over the, the quick brown fox lazy dog.

- These tests indicate that “the lazy dog” is a constituent while “jumped over the” is not.

Linguistically Motivated Grammaticality Tests

- Create new sentences via these transformations [Cao et al '20]

- Derive a score for each span by performing these tests and giving it to a pretrained LM

  score("the lazy dog") = *model*("The quick brown fox jumped over it")

- *model*() is a "grammaticality" model pretrained on real/fake sentences (outputs higher score for real sentences)

- Use these scores as input into the CKY algorithm:

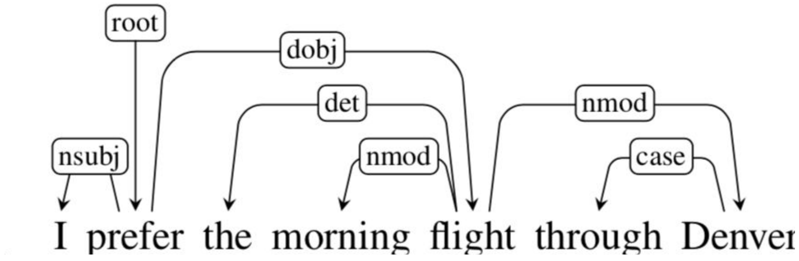
$$\beta[s,t] = \sum_{k=s}^{t-1} \beta[s,k]\beta[k+1,t]\mathit{score}(x_{s:t})$$

Comparison of recent work

Approach	**F1 Score**↑
Random Trees	19.5
Right Branching Trees	39.5
Classic PCFG	35.0
Parsing Reading Predict Network [Shen et al. '18]	47.9
Ordered Neurons [Shen et al. '19]	50.0
Unsupervised RNNG [Kim et al. '19]	45.4
Deep Inside-Outside Autoencoders [Drozdov et al. '19]	58.9
Neural PCFG [Kim et al '19]	52.6
Compound PCFG [Kim et al '19]	60.1
Linguistic Constituency Tests [Cao et al '20]	**65.9**
Supervised Neural Binary Parser	71.9
Binary Tree Oracle (upper bound)	84.3

What we haven't covered today

- Other formalisms: dependency grammars, tree substitution grammars

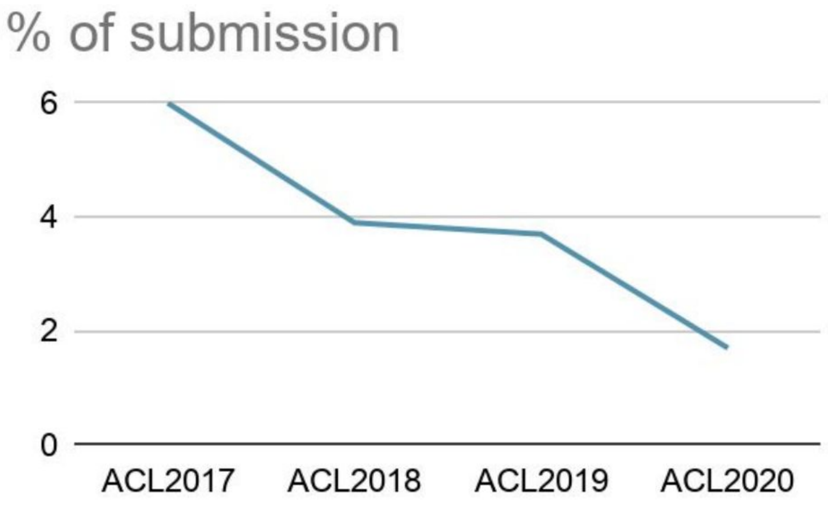


- Non-probabilistic approaches

Why should we care about grammars/trees in modern NLP?

Grammar Induction / Parsing in Contemporary NLP

- "We assume that the goal of learning a context-free grammar needs no justification." [Carroll and Charniak '92]
- Parsing becoming less important in deep learning-based NLP

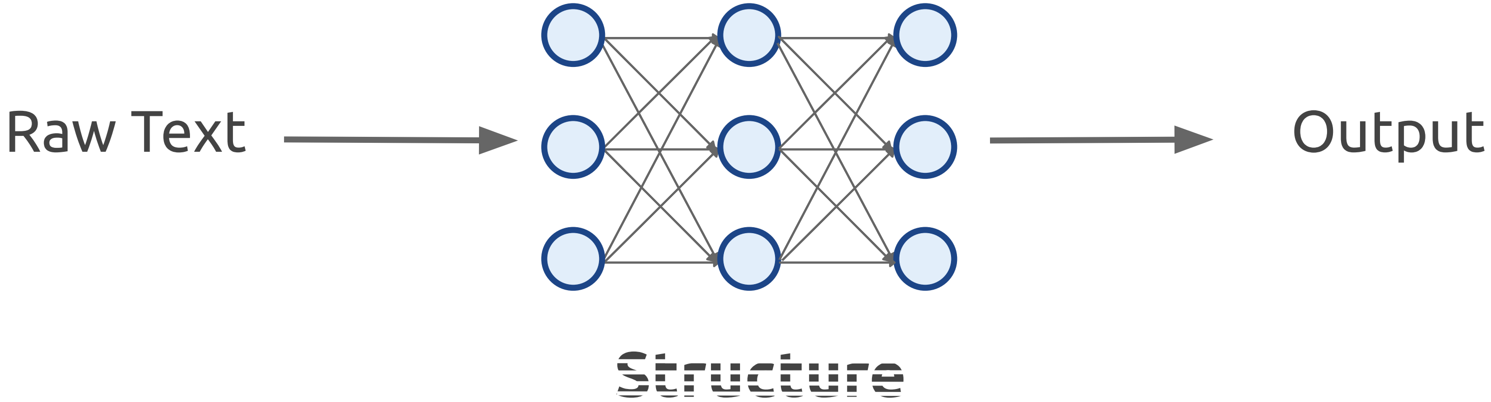


Grammar Induction / Parsing in Contemporary NLP

- Much evidence that ELMo/BERT etc. capture many language phenomena (including syntax!) implicitly in their hidden layers [Liu et al. ’19, Tenney et al. ’19].

Grammar Induction / Parsing in Contemporary NLP

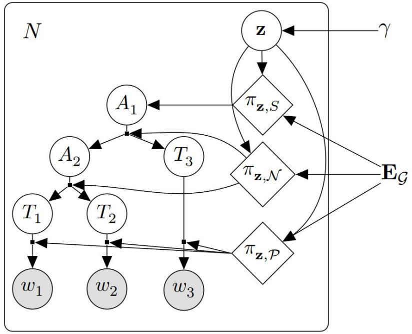
- A case for latent variables: **implicit** vs. explicit modeling of structure



("sort of" captured through hidden layers)

Grammar Induction / Parsing in Contemporary NLP

- A case for latent variables: implicit vs. **explicit** modeling of structure

Raw Text →  → Output

Structure

(explicitly captured through latent variables)

Grammar Induction / Parsing in Contemporary NLP

- Would be ideal to have explicit access to such structures from the perspective of
    - Controllability
    - Interpretability
    - Transfer learning
- Grammars can readily operationalize various notions of compositionality.