Natural Language Processing (CSE 517 & 447): Linguistic Structure Prediction

Noah Smith

University of Washington
nasmith@cs.washington.edu

Winter 2022

Readings: Eisenstein (2019) 10 and 12 (11 and 13 suggested)

Motivation

As data, we tend to view natural language text as sequences (of words, wordpieces, or characters, depending on the NLP application).

But language obeys implicit rules of grammar, and it carries meaning.

- It's helpful to consider an analogy to programming languages, which have *syntax* and *semantics*; well-formed programs can be compiled and executed to carry out a task.
- Well-formed natural language strings can be understood by others.

Computational models that analyze natural language syntax and semantics typically map into **structures** like trees, graphs, and more.

Big Abstraction: Linguistic Analysis

Every linguistic analyzer is comprised of:

1. Theoretical motivation from linguistics and/or the text domain
2. An algorithm that maps $\mathcal{V}^\dagger$ to some output space $\mathcal{Y}$.
   - Some $\mathcal{Y}$ are very specialized, but others, like label sequences we saw earlier, show up again and again.
3. An implementation of the algorithm
   - Once upon a time: rule systems and crafted rules
   - More robust: supervised learning from annotated data
   - Today: unsupervised pretraining followed by supervised finetuning

What Theories?

The field of linguistics offers a huge range of theories that can inform our design of $\mathcal{Y}$.

- **Syntax**: rules governing grammaticality or well-formedness of strings, relative to a language
- **Semantics**: how the meaning of an utterance is constructed, grounded in "the world" (or a proxy to the world)
- **Pragmatics**: the intended meaning by a speaker, in a given social context

Each has many theories, and none of them is complete!

Theory: Constituents

Noun Phrases: Groups of Tokens that "Act Like" Nouns

What, exactly makes a noun phrase? Examples (Jurafsky and Martin, forthcoming):

- Harry the Horse
- the Broadway coppers
- they
- a high-class spot such as Mindy's
- the reason he comes into the Hot Box
- three parties from Brooklyn

Constituents

More general than noun phrases: **constituents** are groups of words with certain (possible) behaviors.

Linguists characterize constituents in a number of ways, including:

Constituents

More general than noun phrases: **constituents** are groups of words with certain (possible) behaviors.

Linguists characterize constituents in a number of ways, including:

- where they occur (e.g., “NPs can occur before verbs”)
- where they can *move* in variations of a sentence
  - On September 17th, I’d like to fly from Atlanta to Denver
  - I’d like to fly on September 17th from Atlanta to Denver
  - I’d like to fly from Atlanta to Denver on September 17th

Constituents

More general than noun phrases: **constituents** are groups of words with certain (possible) behaviors.

Linguists characterize constituents in a number of ways, including:

- where they occur (e.g., “NPs can occur before verbs”)
- where they can *move* in variations of a sentence
  - On September 17th, I’d like to fly from Atlanta to Denver
  - I’d like to fly on September 17th from Atlanta to Denver
  - I’d like to fly from Atlanta to Denver on September 17th
- what parts can move and what parts can’t
  - *On September I’d like to fly 17th from Atlanta to Denver

Constituents

More general than noun phrases: **constituents** are groups of words with certain (possible) behaviors.

Linguists characterize constituents in a number of ways, including:

- where they occur (e.g., “NPs can occur before verbs”)
- where they can *move* in variations of a sentence
  - On September 17th, I’d like to fly from Atlanta to Denver
  - I’d like to fly on September 17th from Atlanta to Denver
  - I’d like to fly from Atlanta to Denver on September 17th
- what parts can move and what parts can’t
  - *On September I’d like to fly 17th from Atlanta to Denver
- what they can be conjoined with
  - I’d like to fly from Atlanta to Denver on September 17th and in the morning

Recursion and Constituents

this is the house

this is the house that Jack built

this is the cat that lives in the house that Jack built

this is the dog that chased the cat that lives in the house that Jack built

this is the flea that bit the dog that chased the cat that lives in the house the Jack built

this is the virus that infected the flea that bit the dog that chased the cat that lives in the house that Jack built

Reflection

Can you think of some subsequences that are *not* constituents?

At the end of the slides, there's a list of book titles that arguably do not form constituents.

Toward a Theory

Take constituents to be the main building block of natural language syntax, we can attempt to formalize what makes a string grammatical in a language.

Context-Free Grammar

A **context-free grammar** consists of:

- A finite set of nonterminal symbols $\mathcal{N}$ (sometimes called "categories")
  - A start symbol $S \in \mathcal{N}$
- A finite alphabet Σ, called "terminal" symbols, distinct from $\mathcal{N}$
- Production rule set $\mathcal{R}$, each of the form "$N \rightarrow \boldsymbol{\alpha}$" where
  - The lefthand side N is a nonterminal from $\mathcal{N}$
  - The righthand side $\boldsymbol{\alpha}$ is a sequence of zero or more terminals and/or nonterminals: $\boldsymbol{\alpha} \in (\mathcal{N} \cup \Sigma)^*$
    - Special case: **Chomsky normal form** constrains $\boldsymbol{\alpha}$ to be either a single terminal symbol or two nonterminals

An Example CFG for a Tiny Bit of English

From Jurafsky and Martin (forthcoming)

S $\rightarrow$ NP VP
S $\rightarrow$ Aux NP VP
S $\rightarrow$ VP
NP $\rightarrow$ Pronoun
NP $\rightarrow$ Proper-Noun
NP $\rightarrow$ Det Nominal
Nominal $\rightarrow$ Noun
Nominal $\rightarrow$ Nominal Noun
Nominal $\rightarrow$ Nominal PP
VP $\rightarrow$ Verb
VP $\rightarrow$ Verb NP
VP $\rightarrow$ Verb NP PP
VP $\rightarrow$ Verb PP
VP $\rightarrow$ VP PP
PP $\rightarrow$ Preposition NP

Det $\rightarrow$ that | this | a
Noun $\rightarrow$ book | flight | meal | money
Verb $\rightarrow$ book | include | prefer
Pronoun $\rightarrow$ I | she | me
Proper-Noun $\rightarrow$ Houston | NWA
Aux $\rightarrow$ does
Preposition $\rightarrow$ from | to | on | near | through

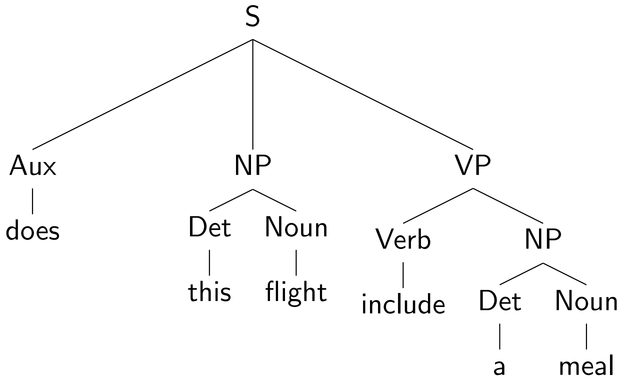
“Lexicon”

This term is used in NLP to refer to an object that associates information with words.

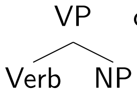
In a CFG, the “lexicon rules” are the rules that map a nonterminal (usually a part of speech) to a single word.

(In an earlier lecture, we encountered WordNet, which is a semantic lexicon.)

Example Phrase Structure Tree



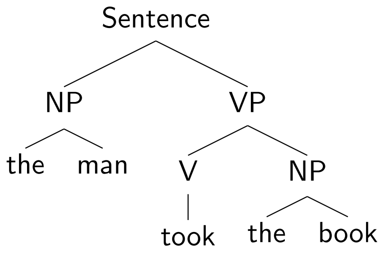
The phrase-structure tree represents both the syntactic structure of the sentence and the **derivation** of the sentence under the grammar. E.g.,



corresponds to the rule VP $\rightarrow$ Verb NP.

The First Phrase-Structure Tree

(Chomsky, 1956)



Where do natural language CFGs come from?

Building a CFG for a natural language by hand is really hard (Jurafsky and Martin, forthcoming, chapter 10).

Where do natural language CFGs come from?

Building a CFG for a natural language by hand is really hard (Jurafsky and Martin, forthcoming, chapter 10).

- Need lots of categories to make sure all and only grammatical sentences are included.

Where do natural language CFGs come from?

Building a CFG for a natural language by hand is really hard (Jurafsky and Martin, forthcoming, chapter 10).

- Need lots of categories to make sure all and only grammatical sentences are included.
- Categories tend to start exploding combinatorially.

Where do natural language CFGs come from?

Building a CFG for a natural language by hand is really hard (Jurafsky and Martin, forthcoming, chapter 10).

- Need lots of categories to make sure all and only grammatical sentences are included.
- Categories tend to start exploding combinatorially.
- Alternative grammar formalisms are typically used for manual grammar construction; these are often based on constraints and a powerful algorithmic tool called *unification*.

Where do natural language CFGs come from?

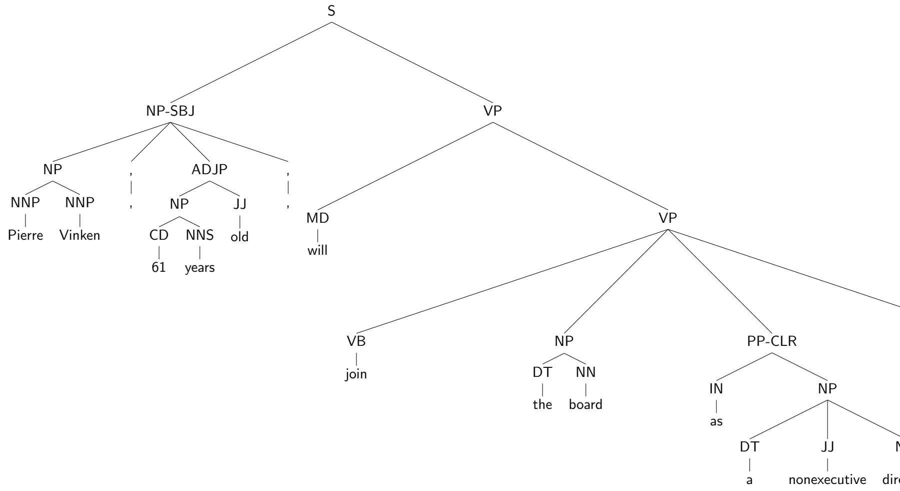
Building a CFG for a natural language by hand is really hard (Jurafsky and Martin, forthcoming, chapter 10).

- Need lots of categories to make sure all and only grammatical sentences are included.
- Categories tend to start exploding combinatorially.
- Alternative grammar formalisms are typically used for manual grammar construction; these are often based on constraints and a powerful algorithmic tool called *unification*.

A data-driven approach:

1. Build a corpus of annotated sentences, called a **treebank**. (e.g., the Penn Treebank, Marcus et al., 1993.)
2. Extract rules from the treebank.
3. Optionally, use statistical models to generalize the rules.

Example from the Penn Treebank



LISP Encoding in the Penn Treebank

```
( (S
    (NP-SBJ-1
      (NP (NNP Rudolph) (NNP Agnew) )
      (, ,)
      (UCP
        (ADJP
          (NP (CD 55) (NNS years) )
          (JJ old) )
        (CC and)
        (NP
          (NP (JJ former) (NN chairman) )
          (PP (IN of)
            (NP (NNP Consolidated) (NNP Gold) (NNP Fields) (NNP PLC) ))))
      (, ,) )
    (VP (VBD was)
      (VP (VBN named)
        (S
          (NP-SBJ (-NONE- *-1) )
          (NP-PRD
            (NP (DT a) (JJ nonexecutive) (NN director) )
            (PP (IN of)
              (NP (DT this) (JJ British) (JJ industrial) (NN conglomerate) ))))
    (. .) ))
```

Some Penn Treebank Rules with Counts

40717 PP → IN NP
33803 S → NP-SBJ VP
22513 NP-SBJ → -NONE-
21877 NP → NP PP
20740 NP → DT NN
14153 S → NP-SBJ VP .
12922 VP → TO VP
11881 PP-LOC → IN NP
11467 NP-SBJ → PRP
11378 NP → -NONE-
11291 NP → NN
. . .
989 VP → VBG S
985 NP-SBJ → NN
983 PP-MNR → IN NP
983 NP-SBJ → DT
969 VP → VBN VP
. . .

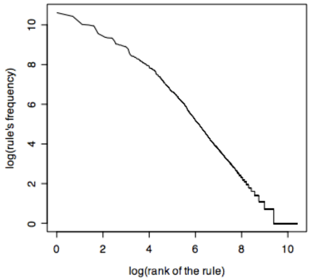
100 VP → VBD PP-PRD
100 PRN → : NP :
100 NP → DT JJS
100 NP-CLR → NN
99 NP-SBJ-1 → DT NNP
98 VP → VBN NP PP-DIR
98 VP → VBD PP-TMP
98 PP-TMP → VBG NP
97 VP → VBD ADVP-TMP VP
. . .
10 WHNP-1 → WRB JJ
10 VP → VP CC VP PP-TMP
10 VP → VP CC VP ADVP-MNR
10 VP → VBZ S , SBAR-ADV
10 VP → VBZ S ADVP-TMP

Penn Treebank Rules: Statistics

32,728 rules in the training section (not including 52,257 lexicon rules)

4,021 rules in the development section

overlap: 3,128



(Phrase-Structure) Recognition and Parsing

Given a CFG $(\mathcal{N}, S, \Sigma, \mathcal{R})$ and a sentence $\boldsymbol{x}$, the **recognition** problem is:

Is $\boldsymbol{x}$ in the language of the CFG?

Related problem: **parsing**:

Show one or more derivations for $\boldsymbol{x}$, using $\mathcal{R}$.

(Phrase-Structure) Recognition and Parsing

Given a CFG $(\mathcal{N}, S, \Sigma, \mathcal{R})$ and a sentence $\boldsymbol{x}$, the **recognition** problem is:

Is $\boldsymbol{x}$ in the language of the CFG?

The proof is a derivation of $\boldsymbol{x}$ using the rules $\mathcal{R}$.

Related problem: **parsing**:

Show one or more derivations for $\boldsymbol{x}$, using $\mathcal{R}$.

(Phrase-Structure) Recognition and Parsing

Given a CFG $(\mathcal{N}, S, \Sigma, \mathcal{R})$ and a sentence $\boldsymbol{x}$, the **recognition** problem is:

Is $\boldsymbol{x}$ in the language of the CFG?

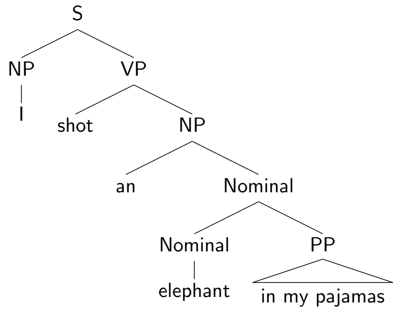
The proof is a derivation of $\boldsymbol{x}$ using the rules $\mathcal{R}$.

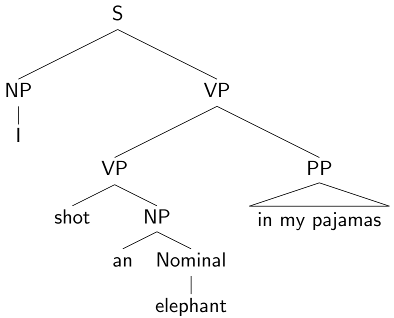
Related problem: **parsing**:

Show one or more derivations for $\boldsymbol{x}$, using $\mathcal{R}$.

With reasonable grammars, the number of parses is exponential in $|\boldsymbol{x}|$.

Syntactic Ambiguity





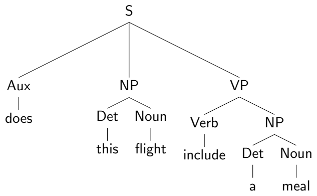
NLP Task: Parsing

Parser Evaluation

Represent a parse tree as a collection of tuples $\langle\langle \ell_1, i_1, j_1\rangle, \langle \ell_2, i_2, j_2\rangle, \ldots, \langle \ell_n, i_n, j_n\rangle\rangle$, where

- ℓ_k is the nonterminal labeling the kth phrase
- i_k is the index of the first word in the kth phrase
- j_k is the index of the last word in the kth phrase

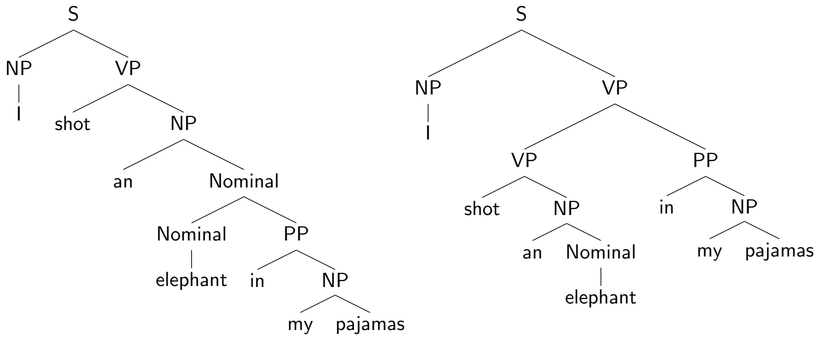
Example:



$$\longrightarrow \left\langle \begin{array}{ll} \langle \text{S}, 1, 6\rangle, & \langle \text{NP}, 2, 3\rangle, \\ \langle \text{VP}, 4, 6\rangle, & \langle \text{NP}, 5, 6\rangle \end{array} \right\rangle$$

Convert gold-standard tree and system hypothesized tree into this representation, then estimate precision, recall, and F_1.

Tree Comparison Example



$$\underbrace{\left\langle \begin{matrix}\langle \mathsf{NP}, 3, 7\rangle, \\ \langle \mathsf{Nominal}, 4, 7\rangle\end{matrix} \right\rangle}_{\text{only in left tree}} \underbrace{\left\langle \begin{matrix}\langle \mathsf{NP}, 1, 1\rangle \\ \langle \mathsf{S}, 1, 7\rangle, \langle \mathsf{VP}, 2, 7\rangle, \\ \langle \mathsf{PP}, 5, 7\rangle, \langle \mathsf{NP}, 6, 7\rangle \\ \langle \mathsf{Nominal}, 4, 4\rangle\end{matrix} \right\rangle}_{\text{in both trees}} \underbrace{\left\langle \begin{matrix}\langle \mathsf{VP}, 2, 4\rangle, \\ \langle \mathsf{NP}, 3, 4\rangle\end{matrix} \right\rangle}_{\text{only in right tree}}$$

Two Views of Parsing

Two Views of Parsing

1. Incremental search: the state of the search is the partial structure built so far; each action incrementally extends the tree.

Two Views of Parsing

1. Incremental search: the state of the search is the partial structure built so far; each action incrementally extends the tree.
   - Often **greedy**, with a statistical classifier deciding what action to take in every state.

Two Views of Parsing

1. Incremental search: the state of the search is the partial structure built so far; each action incrementally extends the tree.
    - Often **greedy**, with a statistical classifier deciding what action to take in every state.
2. Discrete optimization: define a scoring function and seek the tree with the highest score.

Probabilistic Context-Free Grammar

A **probabilistic context-free grammar** consists of:

- A finite set of nonterminal symbols $\mathcal{N}$
  - A start symbol $S \in \mathcal{N}$
- A finite alphabet Σ, called "terminal" symbols, distinct from $\mathcal{N}$
- Production rule set $\mathcal{R}$, each of the form "$N \rightarrow \boldsymbol{\alpha}$" where
  - The lefthand side N is a nonterminal from $\mathcal{N}$
  - The righthand side $\boldsymbol{\alpha}$ is a sequence of zero or more terminals and/or nonterminals: $\boldsymbol{\alpha} \in (\mathcal{N} \cup \Sigma)^*$
    - Special case: **Chomsky normal form** constrains $\boldsymbol{\alpha}$ to be either a single terminal symbol or two nonterminals
- For each $N \in \mathcal{N}$, a probability distribution over the rules where N is the lefthand side, $p(* \mid N)$.

PCFGs Score Trees

We can write the parsing problem as finding the best-scoring tree:

$$\hat{\boldsymbol{t}} = \operatorname*{argmax}_{\boldsymbol{t} \in \mathcal{T}_{\boldsymbol{x}}} \text{Score}(\boldsymbol{t})$$

PCFGs view each tree $\boldsymbol{t}$ as a "bag of rules" (from $\mathcal{R}$), and define:

$$\begin{aligned} \text{Score}(\boldsymbol{t}) &= p(\boldsymbol{t}) \\ &= \prod_{(N \to \alpha) \in \mathcal{R}} p(\alpha \mid N)^{\text{count}(N \to \alpha; \boldsymbol{t})} \end{aligned}$$

PCFG Example

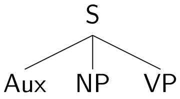
S

Write down the start symbol. Here: S

Probability:

$$1$$

PCFG Example

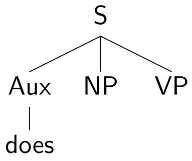


Choose a rule from the “S” distribution. Here: S $\rightarrow$ Aux NP VP

Probability:

$$p(\text{Aux NP VP} \mid \text{S})$$

PCFG Example

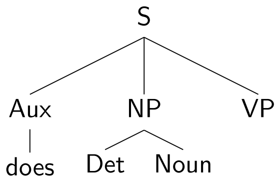


Choose a rule from the “Aux” distribution. Here: Aux $\rightarrow$ does

Probability:

$$p(\text{Aux NP VP} \mid \text{S}) \cdot p(\text{does} \mid \text{Aux})$$

PCFG Example

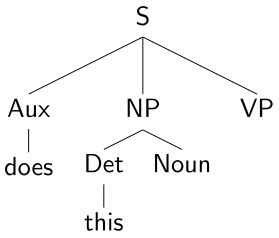


Choose a rule from the "NP" distribution. Here: NP $\rightarrow$ Det Noun

Probability:

$$p(\text{Aux NP VP} \mid \text{S}) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP})$$

PCFG Example

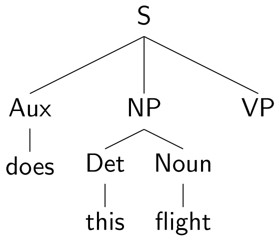


Choose a rule from the "Det" distribution. Here: Det $\rightarrow$ this

Probability:

$p(\text{Aux NP VP} \mid \text{S}) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det})$

PCFG Example

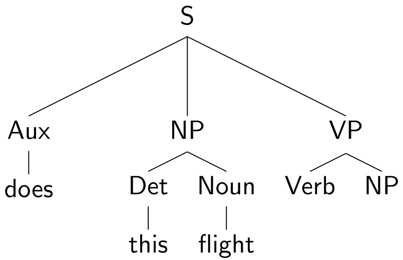


Choose a rule from the “Noun” distribution. Here: Noun $\rightarrow$ flight

Probability:

$p(\text{Aux NP VP} \mid \text{S}) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det})$
$\cdot\, p(\text{flight} \mid \text{Noun})$

PCFG Example

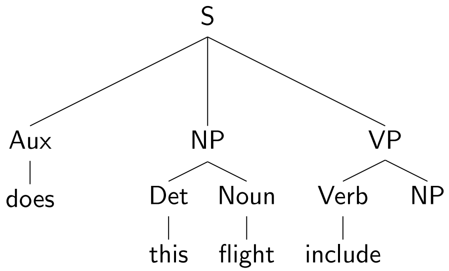


Choose a rule from the “VP” distribution. Here: VP $\rightarrow$ Verb NP

Probability:

$p(\text{Aux NP VP} \mid \text{S}) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det})$
$\cdot\, p(\text{flight} \mid \text{Noun}) \cdot p(\text{Verb NP} \mid \text{VP})$

PCFG Example

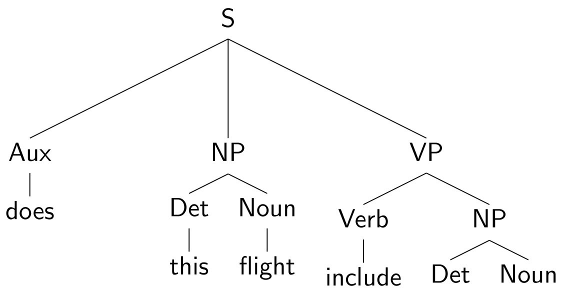


Choose a rule from the “Verb” distribution. Here: Verb $\rightarrow$ include

Probability:

p(Aux NP VP | S) $\cdot\, p$(does | Aux) $\cdot\, p$(Det Noun | NP) $\cdot\, p$(this | Det) $\cdot\, p$(flight | Noun) $\cdot\, p$(Verb NP | VP) $\cdot\, p$(include | Verb)

PCFG Example

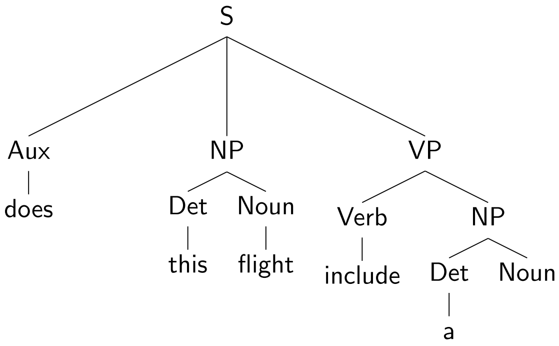


Choose a rule from the "NP" distribution. Here: NP $\rightarrow$ Det Noun

Probability:

$p(\text{Aux NP VP} \mid \text{S}) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det})$
$\cdot\, p(\text{flight} \mid \text{Noun}) \cdot p(\text{Verb NP} \mid \text{VP}) \cdot p(\text{include} \mid \text{Verb})$
$\cdot\, p(\text{Det Noun} \mid \text{NP})$

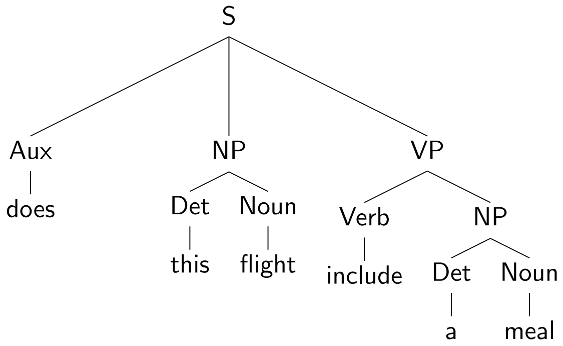
PCFG Example



Choose a rule from the "Det" distribution. Here: Det $\rightarrow$ a
Probability:

$$p(\text{Aux NP VP} \mid \text{S}) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det})$$
$$\cdot\, p(\text{flight} \mid \text{Noun}) \cdot p(\text{Verb NP} \mid \text{VP}) \cdot p(\text{include} \mid \text{Verb})$$
$$\cdot\, p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{a} \mid \text{Det})$$

PCFG Example



Choose a rule from the "Noun" distribution. Here: Noun $\rightarrow$ meal
Probability:

$$p(\text{Aux NP VP} \mid \text{S}) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det})$$
$$\cdot\, p(\text{flight} \mid \text{Noun}) \cdot p(\text{Verb NP} \mid \text{VP}) \cdot p(\text{include} \mid \text{Verb})$$
$$\cdot\, p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{a} \mid \text{Det}) \cdot p(\text{meal} \mid \text{Noun})$$

Parsing with PCFGs

- How to set the probabilities $p(\text{righthand side} \mid \text{lefthand side})$?
- How to decode/parse?

Probabilistic CKY

(Cocke and Schwartz, 1970; Kasami, 1965; Younger, 1967)

Input:

- a PCFG $(\mathcal{N}, S, \Sigma, \mathcal{R}, p(* \mid *))$, in **Chomsky normal form**
- a sentence $\boldsymbol{x}$ (let n be its length)

Output: If $\boldsymbol{x}$ is in the language of the grammar.

$$\operatorname*{argmax}_{\boldsymbol{t} \in \mathcal{T}_{\boldsymbol{x}}} \log p(\boldsymbol{t});$$

undefined if not.

Notation

Probabilistic CKY is closely related to the Viterbi algorithm; it is a dynamic programming algorithm.
The recurrence is defined around

$$\heartsuit_{i:j}(N),$$

which will store the best score (log probability) found (so far) for constructing an N-rooted constituent that spans $\langle x_i, \ldots, x_j \rangle$.

In Viterbi, we used conditional independence to collapse all prefix label sequences that ended in the same label into one stored item; here we collapse all trees spanning words i to j with the same root into a single item.

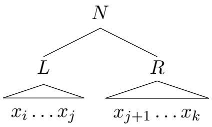
Probabilistic CKY

Base case: for $i \in \{1, \ldots, n\}$ and for each $N \in \mathcal{N}$:

$$\heartsuit_{i:i}(N) = \log p(x_i \mid N)$$

For each i, k such that $1 \leq i < k \leq n$ and each $N \in \mathcal{N}$:

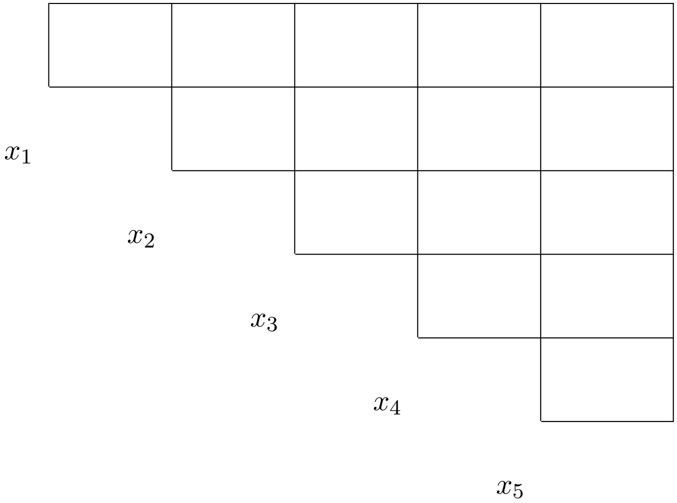
$$\heartsuit_{i:k}(N) = \max_{L,R \in \mathcal{N}, j \in \{i,\ldots,k-1\}} \log p(L \ R \mid N) + \heartsuit_{i:j}(L) + \heartsuit_{(j+1):k}(R)$$



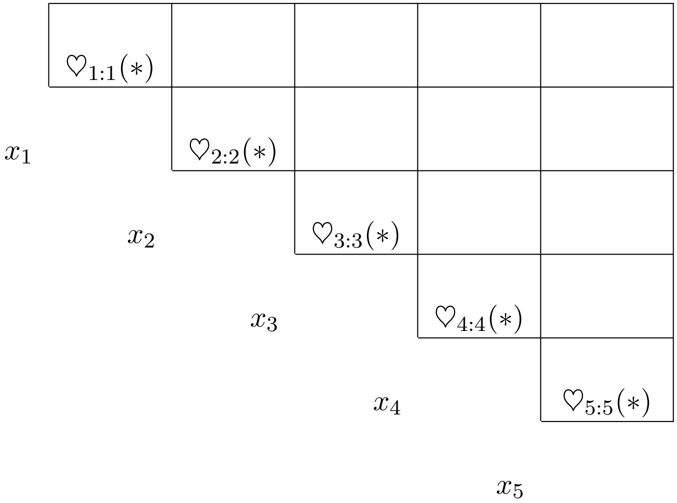
Solution:

$$\heartsuit_{1:n}(S) = \max_{\boldsymbol{t} \in \mathcal{T}_{\boldsymbol{x}}} \log p(\boldsymbol{t})$$

Parse Chart



Parse Chart


$\heartsuit_{1:1}(*)$
$\heartsuit_{2:2}(*)$
$\heartsuit_{3:3}(*)$
$\heartsuit_{4:4}(*)$
$\heartsuit_{5:5}(*)$
x_1
x_2
x_3
x_4
x_5

Parse Chart

	$\heartsuit_{1:1}(*)$	$\heartsuit_{1:2}(*)$			
x_1		$\heartsuit_{2:2}(*)$	$\heartsuit_{2:3}(*)$		
	x_2		$\heartsuit_{3:3}(*)$	$\heartsuit_{3:4}(*)$	
		x_3		$\heartsuit_{4:4}(*)$	$\heartsuit_{4:5}(*)$
			x_4		$\heartsuit_{5:5}(*)$
				x_5	

Parse Chart

	$\heartsuit_{1:1}(*)$	$\heartsuit_{1:2}(*)$	$\heartsuit_{1:3}(*)$		
x_1		$\heartsuit_{2:2}(*)$	$\heartsuit_{2:3}(*)$	$\heartsuit_{2:4}(*)$	
	x_2		$\heartsuit_{3:3}(*)$	$\heartsuit_{3:4}(*)$	$\heartsuit_{3:5}(*)$
		x_3		$\heartsuit_{4:4}(*)$	$\heartsuit_{4:5}(*)$
			x_4		$\heartsuit_{5:5}(*)$
				x_5	

Parse Chart

	$\heartsuit_{1:1}(*)$	$\heartsuit_{1:2}(*)$	$\heartsuit_{1:3}(*)$	$\heartsuit_{1:4}(*)$	
x_1		$\heartsuit_{2:2}(*)$	$\heartsuit_{2:3}(*)$	$\heartsuit_{2:4}(*)$	$\heartsuit_{2:5}(*)$
	x_2		$\heartsuit_{3:3}(*)$	$\heartsuit_{3:4}(*)$	$\heartsuit_{3:5}(*)$
		x_3		$\heartsuit_{4:4}(*)$	$\heartsuit_{4:5}(*)$
			x_4		$\heartsuit_{5:5}(*)$
				x_5	

Parse Chart

	$\heartsuit_{1:1}(*)$	$\heartsuit_{1:2}(*)$	$\heartsuit_{1:3}(*)$	$\heartsuit_{1:4}(*)$	$\heartsuit_{1:5}(*)$
x_1		$\heartsuit_{2:2}(*)$	$\heartsuit_{2:3}(*)$	$\heartsuit_{2:4}(*)$	$\heartsuit_{2:5}(*)$
	x_2		$\heartsuit_{3:3}(*)$	$\heartsuit_{3:4}(*)$	$\heartsuit_{3:5}(*)$
		x_3		$\heartsuit_{4:4}(*)$	$\heartsuit_{4:5}(*)$
			x_4		$\heartsuit_{5:5}(*)$
				x_5	

Remarks

- Space and runtime requirements?

- Analogous to (neural) CRFs for sequence labeling, modern practice replaces the log-probabilities with $s(\boldsymbol{x}, i, j, N)$, built out of a neural network with word vectors at the bottom (e.g., Stern et al., 2017).

Remarks

- Space and runtime requirements? $O(|\mathcal{N}|n^2)$ space, $O(|\mathcal{R}|n^3)$ runtime.
- Analogous to (neural) CRFs for sequence labeling, modern practice replaces the log-probabilities with $s(\boldsymbol{x}, i, j, N)$, built out of a neural network with word vectors at the bottom (e.g., Stern et al., 2017).

Remarks

- Space and runtime requirements? $O(|\mathcal{N}|n^2)$ space, $O(|\mathcal{R}|n^3)$ runtime.
- Recovering the best tree?

- Analogous to (neural) CRFs for sequence labeling, modern practice replaces the log-probabilities with $s(\boldsymbol{x}, i, j, N)$, built out of a neural network with word vectors at the bottom (e.g., Stern et al., 2017).

Remarks

- Space and runtime requirements? $O(|\mathcal{N}|n^2)$ space, $O(|\mathcal{R}|n^3)$ runtime.
- Recovering the best tree? Backpointers.

- Analogous to (neural) CRFs for sequence labeling, modern practice replaces the log-probabilities with $s(\boldsymbol{x}, i, j, N)$, built out of a neural network with word vectors at the bottom (e.g., Stern et al., 2017).

Remarks

- Space and runtime requirements? $O(|\mathcal{N}|n^2)$ space, $O(|\mathcal{R}|n^3)$ runtime.
- Recovering the best tree? Backpointers.
- Probabilistic **Earley's** algorithm does not require the grammar to be in Chomsky normal form.
- Analogous to (neural) CRFs for sequence labeling, modern practice replaces the log-probabilities with $s(\boldsymbol{x}, i, j, N)$, built out of a neural network with word vectors at the bottom (e.g., Stern et al., 2017).

Demo of Recent State of the Art

`https://demo.allennlp.org/constituency-parsing`

Beyond PCFGs

The “bag of rules” assumption in our scoring function is very limiting.

- Making the rules “second-order,” e.g., $p(\alpha \mid \mathit{Parent}, \mathit{Grandparent})$ has a huge benefit (Johnson, 1998)
- “Decorating” the trees in various ways, e.g., with lexical “heads” that relate a each category back to the most important word in its yield
- Recurrent neural network grammars: encode the entire history of derivation steps in a vector (Dyer et al., 2016)
- Many richer syntactic formalisms from (computational) linguistics, as well, venturing into (mild) context-sensitivity and beyond!

Alternatives

A different family of theories of syntax focuses on **dependencies** between words, so that parse trees are directed graphs over word-vertices (Tesnière, 1959; Mel'čuk, 1987). Dependency parsing is largely based on directed spanning tree algorithms (McDonald et al., 2005).

Dependency syntax has, arguably, a stronger tradition of application across a more diverse set of languages. The Universal Dependencies project (`https://universaldependencies.org`), for example, includes treebanks in over 100 languages.

Switching Gears: Semantics

Toward Semantics

Semantics is about formally capturing (some of) what a piece of text means.

Desiderata for a **meaning representation language**:

Toward Semantics

Semantics is about formally capturing (some of) what a piece of text means.
Desiderata for a **meaning representation language**:

- represent the state of the world, i.e., a knowledge base

Toward Semantics

Semantics is about formally capturing (some of) what a piece of text means.
Desiderata for a **meaning representation language**:

- represent the state of the world, i.e., a knowledge base
- query the knowledge base (e.g., verify that a statement is true, or answer a question)

Toward Semantics

Semantics is about formally capturing (some of) what a piece of text means.

Desiderata for a **meaning representation language**:

- represent the state of the world, i.e., a knowledge base
- query the knowledge base (e.g., verify that a statement is true, or answer a question)
- handle ambiguity, vagueness, and non-canonical forms
  - "I wanna eat someplace that's close to UW"
  - "something not too spicy"

Toward Semantics

Semantics is about formally capturing (some of) what a piece of text means.

Desiderata for a **meaning representation language**:

- represent the state of the world, i.e., a knowledge base
- query the knowledge base (e.g., verify that a statement is true, or answer a question)
- handle ambiguity, vagueness, and non-canonical forms
  - "I wanna eat someplace that's close to UW"
  - "something not too spicy"
- support inference and reasoning
  - "can Karen eat at Schultzy's?"

Toward Semantics

Semantics is about formally capturing (some of) what a piece of text means.
Desiderata for a **meaning representation language**:

- represent the state of the world, i.e., a knowledge base
- query the knowledge base (e.g., verify that a statement is true, or answer a question)
- handle ambiguity, vagueness, and non-canonical forms
  - "I wanna eat someplace that's close to UW"
  - "something not too spicy"
- support inference and reasoning
  - "can Karen eat at Schultzy's?"

Eventually (but not today):

- deal with non-literal meanings
- expressiveness across a wide range of subject matter

First Steps

Today we'll create a computational model of a tiny world and use first-order logic as our meaning representation language.

A (Tiny) World Model

- **Domain:** Adrian, Brook, Chris, Donald, Schultzy's Sausage, Din Tai Fung, Banana Leaf, American, Chinese, Thai
- **Property:** Din Tai Fung has a long wait, Schultzy's is noisy; Adrian, Brook, and Chris are human
- **Relations:** Schultzy's serves American, Din Tai Fung serves Chinese, and Banana Leaf serves Thai

Simple questions are easy:

- Is Schultzy's noisy?
- Does Din Tai Fung serve Thai?

A (Tiny) World Model

- **Domain:** Adrian, Brook, Chris, Donald, Schultzy's Sausage, Din Tai Fung, Banana Leaf, American, Chinese, Thai
  $a, b, c, d, ss, dtf, bl, am, ch, th$
- **Property:** Din Tai Fung has a long wait, Schultzy's is noisy; Adrian, Brook, and Chris are human
  $Longwait = \{dtf\}, Noisy = \{ss\}, Human = \{a, b, c\}$
- **Relations:** Schultzy's serves American, Din Tai Fung serves Chinese, and Banana Leaf serves Thai
  $Serves = \{(ss, am), (dtf, ch), (bl, th)\}, Likes = \{(a, ss), (a, dtf), \dots\}$

Simple questions are easy:

- Is Schultzy's noisy?
- Does Din Tai Fung serve Thai?

A Quick Tour of First-Order Logic

- **Term:** a constant (ss) or a variable
- **Formula:** defined inductively ...
  - If R is an n-ary relation and $t_1, \ldots, t_n$ are terms, then $R(t_1, \ldots, t_n)$ is a formula.
  - If ϕ is a formula, then its negation, $\neg\phi$, is a formula.
  - If ϕ and ψ are formulas, then binary logical connectives can be used to create formulas:
    - $\phi \wedge \psi$
    - $\phi \vee \psi$
    - $\phi \Rightarrow \psi$
  - If ϕ is a formula and v is a variable, then quantifiers can be used to create formulas:
    - Universal quantifier: $\forall v, \phi$
    - Existential quantifier: $\exists v, \phi$

Note: Leaving out functions, because we don't need them in a single lecture on FOL for NL.

Translating Between FOL and NL

1. Schultzy's is not loud
2. Some human likes Chinese
3. If a person likes Thai, then they aren't friends with Donald
4. $\forall x, Restaurant(x) \Rightarrow (Longwait(x) \vee \neg Likes(a, x))$
5. $\forall x, \exists y, \neg Likes(x, y)$
6. $\exists y, \forall x, \neg Likes(x, y)$

Translating Between FOL and NL

1. Schultzy's is not loud $\neg Noisy(ss)$
2. Some human likes Chinese
3. If a person likes Thai, then they aren't friends with Donald
4. $\forall x, Restaurant(x) \Rightarrow (Longwait(x) \vee \neg Likes(a, x))$
5. $\forall x, \exists y, \neg Likes(x, y)$
6. $\exists y, \forall x, \neg Likes(x, y)$

Translating Between FOL and NL

1. Schultzy's is not loud $\neg Noisy(ss)$
2. Some human likes Chinese $\exists x, Human(x) \wedge Likes(x, ch)$
3. If a person likes Thai, then they aren't friends with Donald
4. $\forall x, Restaurant(x) \Rightarrow (Longwait(x) \vee \neg Likes(a, x))$
5. $\forall x, \exists y, \neg Likes(x, y)$
6. $\exists y, \forall x, \neg Likes(x, y)$

Translating Between FOL and NL

1. Schultzy's is not loud $\neg Noisy(ss)$
2. Some human likes Chinese $\exists x, Human(x) \wedge Likes(x, ch)$
3. If a person likes Thai, then they aren't friends with Donald
   $\forall x, Human(x) \wedge Likes(x, th) \Rightarrow \neg Friends(x, d)$
4. $\forall x, Restaurant(x) \Rightarrow (Longwait(x) \vee \neg Likes(a, x))$
5. $\forall x, \exists y, \neg Likes(x, y)$
6. $\exists y, \forall x, \neg Likes(x, y)$

Translating Between FOL and NL

1. Schultzy's is not loud $\neg Noisy(ss)$
2. Some human likes Chinese $\exists x, Human(x) \land Likes(x, ch)$
3. If a person likes Thai, then they aren't friends with Donald $\forall x, Human(x) \land Likes(x, th) \Rightarrow \neg Friends(x, d)$
4. $\forall x, Restaurant(x) \Rightarrow (Longwait(x) \lor \neg Likes(a, x))$ Every restaurant has a long wait or is disliked by Adrian.
5. $\forall x, \exists y, \neg Likes(x, y)$
6. $\exists y, \forall x, \neg Likes(x, y)$

Translating Between FOL and NL

1. Schultzy's is not loud $\neg Noisy(ss)$
2. Some human likes Chinese $\exists x, Human(x) \wedge Likes(x, ch)$
3. If a person likes Thai, then they aren't friends with Donald
   $\forall x, Human(x) \wedge Likes(x, th) \Rightarrow \neg Friends(x, d)$
4. $\forall x, Restaurant(x) \Rightarrow (Longwait(x) \vee \neg Likes(a, x))$
   Every restaurant has a long wait or is disliked by Adrian.
5. $\forall x, \exists y, \neg Likes(x, y)$
   Everybody has something they don't like.
6. $\exists y, \forall x, \neg Likes(x, y)$

Translating Between FOL and NL

1. Schultzy's is not loud $\neg Noisy(ss)$
2. Some human likes Chinese $\exists x, Human(x) \wedge Likes(x, ch)$
3. If a person likes Thai, then they aren't friends with Donald
   $\forall x, Human(x) \wedge Likes(x, th) \Rightarrow \neg Friends(x, d)$
4. $\forall x, Restaurant(x) \Rightarrow (Longwait(x) \vee \neg Likes(a, x))$
   Every restaurant has a long wait or is disliked by Adrian.
5. $\forall x, \exists y, \neg Likes(x, y)$
   Everybody has something they don't like.
6. $\exists y, \forall x, \neg Likes(x, y)$
   There exists something that nobody likes.

Logical Semantics
(Montague, 1970)

The denotation of a NL sentence is the set of conditions that must hold in the (model) world for the sentence to be true.

Every restaurant has a long wait or Adrian doesn't like it.

is true if and only if

$$\forall x, Restaurant(x) \Rightarrow (Longwait(x) \vee \neg Likes(a, x))$$

is true.

This is sometimes called the **logical form** of the NL sentence.

The Principle of Compositionality

The meaning of a NL phrase is determined by the meanings of its sub-phrases.

The Principle of Compositionality

The meaning of a NL phrase is determined by the meanings of its sub-phrases.

I.e., semantics is derived from syntax.

The Principle of Compositionality

The meaning of a NL phrase is determined by the meanings of its sub-phrases.

I.e., semantics is derived from syntax.

We need a way to express semantics of phrases, and compose them together!

λ-Calculus

(Much more powerful than what we'll see today; ask your PL professor!)

Informally, two extensions:

- λ-**abstraction** is another way to "scope" variables.
  - If ϕ is a FOL formula and v is a variable, then $\lambda v.\phi$ is a λ-term, meaning: an unnamed function from values (of v) to formulas (usually involving v)
- **application** of such functions: if we have $\lambda v.\phi$ and ψ, then $[\lambda v.\phi](\psi)$ is a formula.
  - It can be **reduced** by substituting ψ in for every instance of v in ϕ.

λ-Calculus

(Much more powerful than what we'll see today; ask your PL professor!)

Informally, two extensions:

- λ-**abstraction** is another way to "scope" variables.
  - If ϕ is a FOL formula and v is a variable, then $\lambda v.\phi$ is a λ-term, meaning: an unnamed function from values (of v) to formulas (usually involving v)
- **application** of such functions: if we have $\lambda v.\phi$ and ψ, then $[\lambda v.\phi](\psi)$ is a formula.
  - It can be **reduced** by substituting ψ in for every instance of v in ϕ.

Example:
$\lambda x.Likes(x, dtf)$ maps things to statements that they like Din Tai Fung

λ-Calculus

(Much more powerful than what we'll see today; ask your PL professor!)

Informally, two extensions:

- λ-**abstraction** is another way to "scope" variables.
  - If ϕ is a FOL formula and v is a variable, then $\lambda v.\phi$ is a λ-term, meaning: an unnamed function from values (of v) to formulas (usually involving v)
- **application** of such functions: if we have $\lambda v.\phi$ and ψ, then $[\lambda v.\phi](\psi)$ is a formula.
  - It can be **reduced** by substituting ψ in for every instance of v in ϕ.

Example:
$[\lambda x.Likes(x, dtf)](c)$ reduces to $Likes(c, dtf)$

λ-Calculus

(Much more powerful than what we'll see today; ask your PL professor!)

Informally, two extensions:

- λ-**abstraction** is another way to "scope" variables.
  - If ϕ is a FOL formula and v is a variable, then $\lambda v.\phi$ is a λ-term, meaning: an unnamed function from values (of v) to formulas (usually involving v)
- **application** of such functions: if we have $\lambda v.\phi$ and ψ, then $[\lambda v.\phi](\psi)$ is a formula.
  - It can be **reduced** by substituting ψ in for every instance of v in ϕ.

Example:
$\lambda x.\lambda y.Friends(x, y)$ maps things x to maps of things y to statements that x and y are friends

λ-Calculus

(Much more powerful than what we'll see today; ask your PL professor!)

Informally, two extensions:

- λ-**abstraction** is another way to "scope" variables.
  - If ϕ is a FOL formula and v is a variable, then $\lambda v.\phi$ is a λ-term, meaning: an unnamed function from values (of v) to formulas (usually involving v)
- **application** of such functions: if we have $\lambda v.\phi$ and ψ, then $[\lambda v.\phi](\psi)$ is a formula.
  - It can be **reduced** by substituting ψ in for every instance of v in ϕ.

Example:
$[\lambda x.\lambda y.Friends(x, y)](b)$ reduces to $\lambda y.Friends(b, y)$

λ-Calculus

(Much more powerful than what we'll see today; ask your PL professor!)

Informally, two extensions:

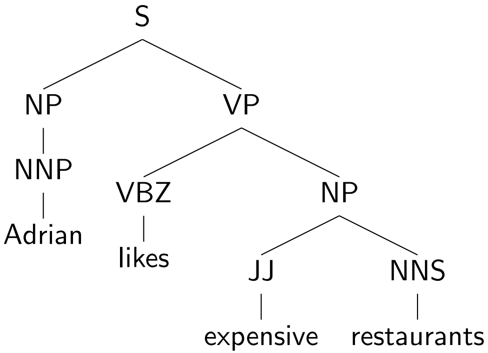
- λ-**abstraction** is another way to "scope" variables.
  - If ϕ is a FOL formula and v is a variable, then $\lambda v.\phi$ is a λ-term, meaning: an unnamed function from values (of v) to formulas (usually involving v)
- **application** of such functions: if we have $\lambda v.\phi$ and ψ, then $[\lambda v.\phi](\psi)$ is a formula.
  - It can be **reduced** by substituting ψ in for every instance of v in ϕ.

Example:
$[[\lambda x.\lambda y.Friends(x, y)](b)](a)$ reduces to $[\lambda y.Friends(b, y)](a)$, which reduces to $Friends(b, a)$

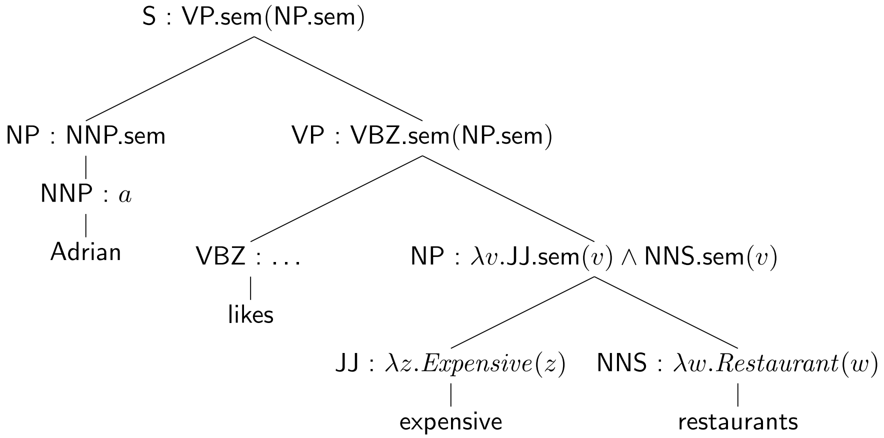
Semantic Attachments to CFG

- NNP $\rightarrow$ Adrian $\{a\}$
- VBZ $\rightarrow$ likes $\{\lambda f.\lambda y.\forall x f(x) \Rightarrow Likes(y,x)\}$
- JJ $\rightarrow$ expensive $\{\lambda x.Expensive(x)\}$
- NNS $\rightarrow$ restaurants $\{\lambda x.Restaurant(x)\}$
- NP $\rightarrow$ NNP $\{\text{NNP.sem}\}$
- NP $\rightarrow$ JJ NNS $\{\lambda x.\text{JJ.sem}(x) \wedge \text{NNS.sem}(x)\}$
- VP $\rightarrow$ VBZ NP $\{\text{VBZ.sem(NP.sem)}\}$
- S $\rightarrow$ NP VP $\{\text{VP.sem(NP.sem)}\}$

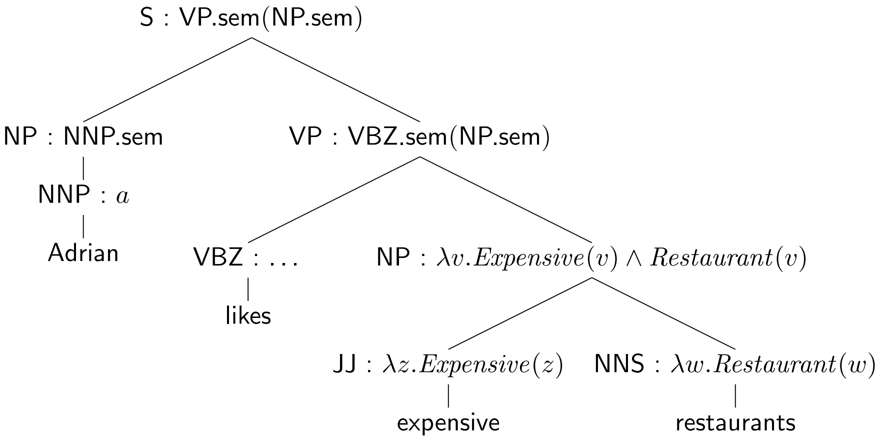
Example



Example


S : VP.sem(NP.sem)
NP : NNP.sem
VP : VBZ.sem(NP.sem)
NNP : a
Adrian
VBZ : ...
likes
NP : $\lambda v.$JJ.sem$(v) \wedge$ NNS.sem(v)
JJ : $\lambda z.Expensive(z)$
NNS : $\lambda w.Restaurant(w)$
expensive
restaurants

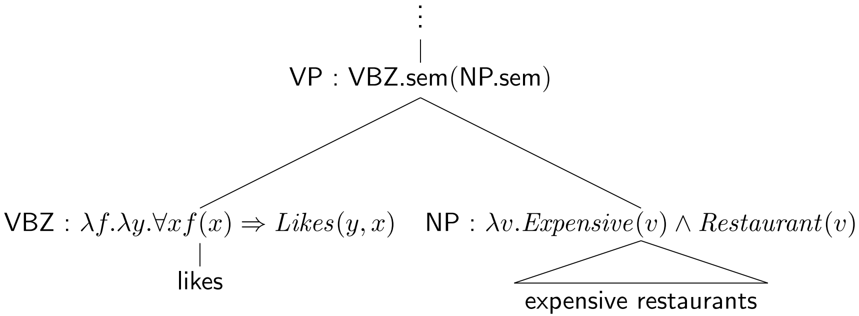
Example



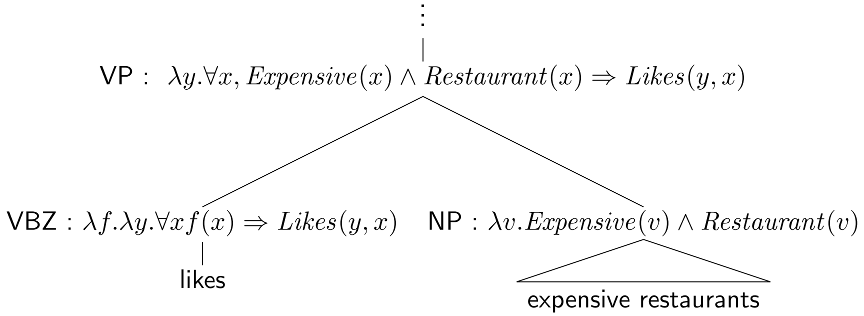
S : VP.sem(NP.sem)
NP : NNP.sem
VP : VBZ.sem(NP.sem)
NNP : a
Adrian
VBZ : ...
likes
NP : $\lambda v.Expensive(v) \wedge Restaurant(v)$
JJ : $\lambda z.Expensive(z)$
NNS : $\lambda w.Restaurant(w)$
expensive
restaurants

$$\lambda v.\left[\underbrace{\lambda z.Expensive(z)}_{\text{JJ.sem}}\right](v) \wedge \left[\underbrace{\lambda w.Restaurant(w)}_{\text{NNS.sem}}\right](v)$$

Example


⋮
VP : VBZ.sem(NP.sem)
VBZ : $\lambda f.\lambda y.\forall x\, f(x) \Rightarrow Likes(y, x)$
likes
NP : $\lambda v.Expensive(v) \wedge Restaurant(v)$
expensive restaurants

Example

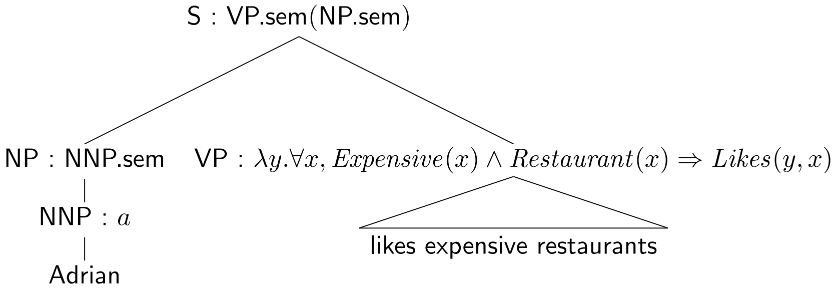

⋮
VP : $\lambda y.\forall x, Expensive(x) \wedge Restaurant(x) \Rightarrow Likes(y,x)$
VBZ : $\lambda f.\lambda y.\forall x f(x) \Rightarrow Likes(y,x)$
likes
NP : $\lambda v.Expensive(v) \wedge Restaurant(v)$
expensive restaurants

$$\left[\underbrace{\lambda f.\lambda y.\forall x f(x) \Rightarrow Likes(y,x)}_{\text{VBZ.sem}}\right]\left(\underbrace{\lambda v.Expensive(v) \wedge Restaurant(v)}_{\text{NP.sem}}\right)$$

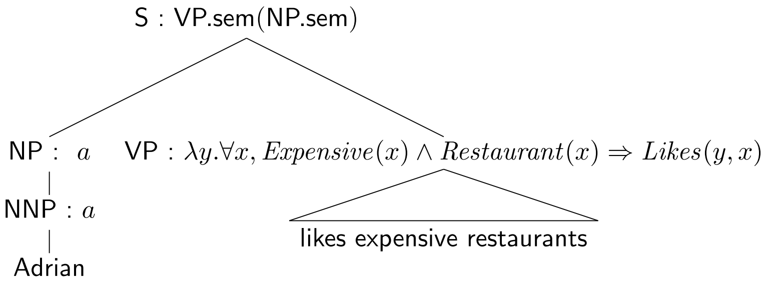
$$\lambda y.\forall x \left[\lambda v.Expensive(v) \wedge Restaurant(v)\right](x) \Rightarrow Likes(y,x)$$

$$\lambda y.\forall x, Expensive(x) \wedge Restaurant(x) \Rightarrow Likes(y,x)$$

Example

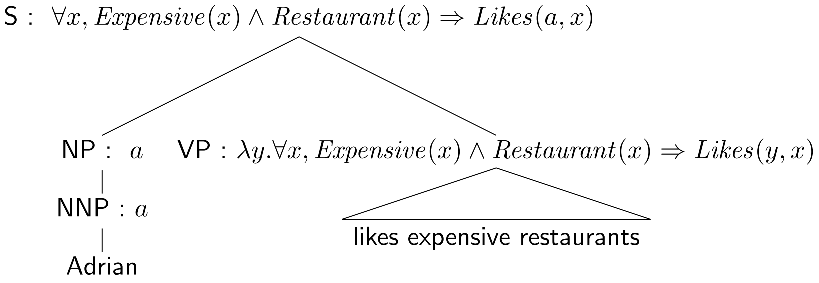

S : VP.sem(NP.sem)
NP : NNP.sem
VP : $\lambda y.\forall x, Expensive(x) \wedge Restaurant(x) \Rightarrow Likes(y, x)$
NNP : a
Adrian
likes expensive restaurants

Example



S : VP.sem(NP.sem)
NP : a
VP : $\lambda y.\forall x, Expensive(x) \wedge Restaurant(x) \Rightarrow Likes(y, x)$
NNP : a
Adrian
likes expensive restaurants

Example



S : $\forall x, \mathit{Expensive}(x) \wedge \mathit{Restaurant}(x) \Rightarrow \mathit{Likes}(a, x)$
NP : a
VP : $\lambda y.\forall x, \mathit{Expensive}(x) \wedge \mathit{Restaurant}(x) \Rightarrow \mathit{Likes}(y, x)$
NNP : a
Adrian
likes expensive restaurants

$$\left[\underbrace{\lambda y.\forall x, \mathit{Expensive}(x) \wedge \mathit{Restaurant}(x) \Rightarrow \mathit{Likes}(y, x)}_{\text{VP.sem}}\right]\left(\underbrace{a}_{\text{NP.sem}}\right)$$

$$\forall x, \mathit{Expensive}(x) \wedge \mathit{Restaurant}(x) \Rightarrow \mathit{Likes}(a, x)$$

Reflection

There are plenty of counter-examples to the idea that natural language meaning is entirely compositional. Can you think of some? Can you think of a way to account for noncompositional elements within the CFG-with-semantic-attachments framework?

The Main Dish

Combinatory Categorial Grammar
(Steedman, 2000)

CCG is a grammatical formalism that is well-suited for tying together syntax and semantics.

Formally, it is more powerful than CFG—it can represent some of the context-*sensitive* languages (which we do not have time to define formally).

CCG Types

Instead of the "$\mathcal{N}$" of CFGs, CCGs can have an infinitely large set of structured categories (called **types**).

- Primitive types: typically S, NP, N, and maybe more
- Complex types, built with "slashes," for example:
  - S/NP is "an S, except that it lacks an NP to the right"
  - S\NP is "an S, except that it lacks an NP to its left"
  - (S\NP)/NP is "an S, except that it lacks an NP to its right, and its left"

You can think of complex types as functions, e.g., S/NP maps NPs to Ss.

CCG Combinators

Instead of the production rules of CFGs, CCGs have a very small set of generic **combinators** that tell us how we can put types together.

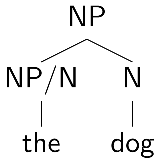
Convention writes the rule differently from CFG: "$X \quad Y \Rightarrow Z$" means that X and Y combine to form a Z (the "parent" in the tree).

Application Combinator

Forward ($X/Y \quad Y \Rightarrow X$) and backward ($Y \quad X\backslash Y \Rightarrow X$)

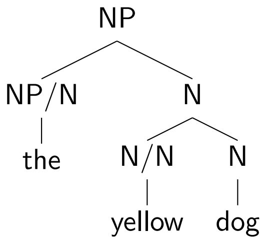
Application Combinator

Forward $(X/Y \quad Y \Rightarrow X)$ and backward $(Y \quad X\backslash Y \Rightarrow X)$



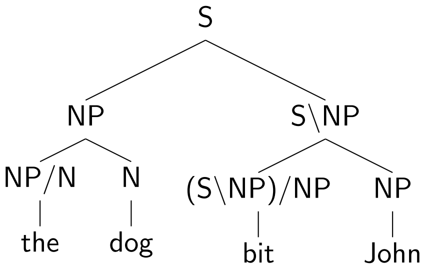
Application Combinator

Forward ($X/Y \quad Y \Rightarrow X$) and backward ($Y \quad X\backslash Y \Rightarrow X$)



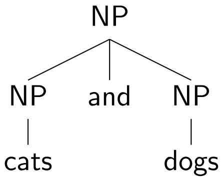
Application Combinator

Forward $(X/Y \quad Y \Rightarrow X)$ and backward $(Y \quad X\backslash Y \Rightarrow X)$



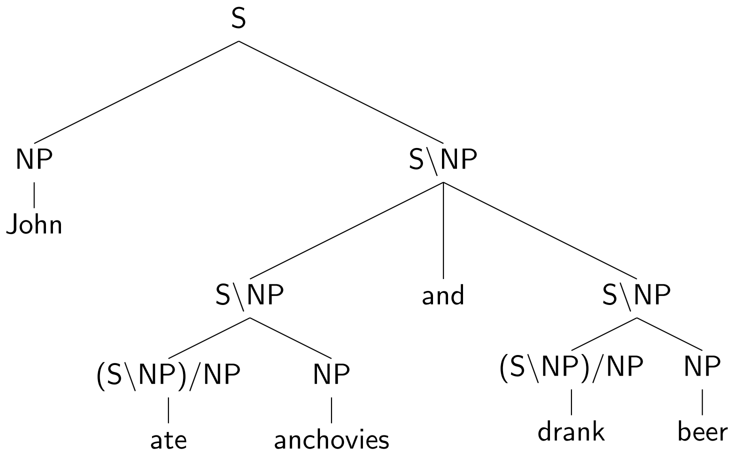
Conjunction Combinator

X and $X \Rightarrow X$



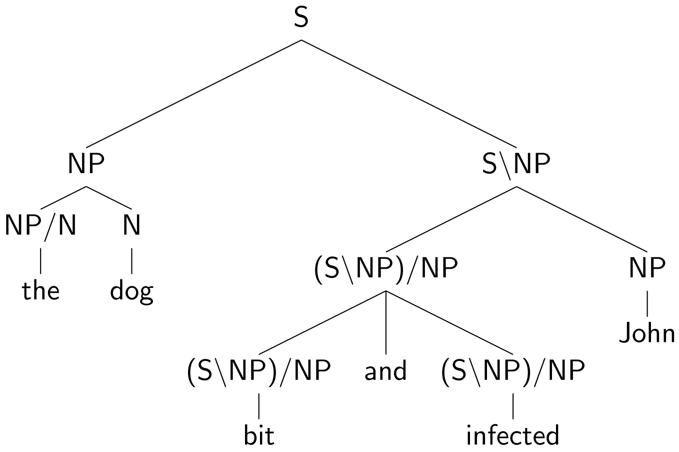
Conjunction Combinator

X and $X \Rightarrow X$



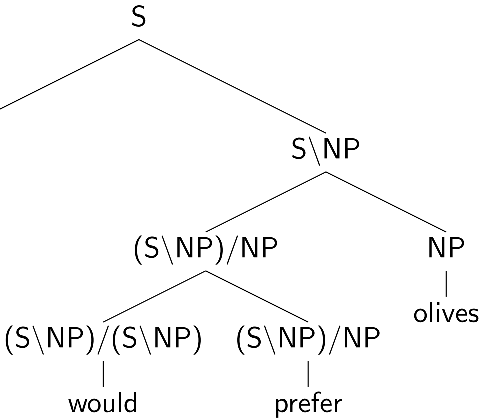
Conjunction Combinator

X and $X \Rightarrow X$



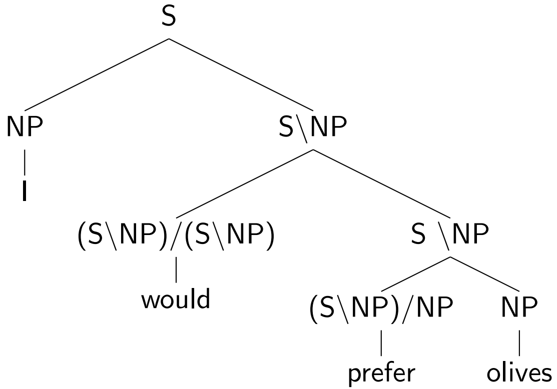
Composition Combinator

Forward $(X/Y \quad Y/Z \Rightarrow X/Z)$ and backward $(Y\backslash Z \quad X\backslash Y \Rightarrow X\backslash Z)$



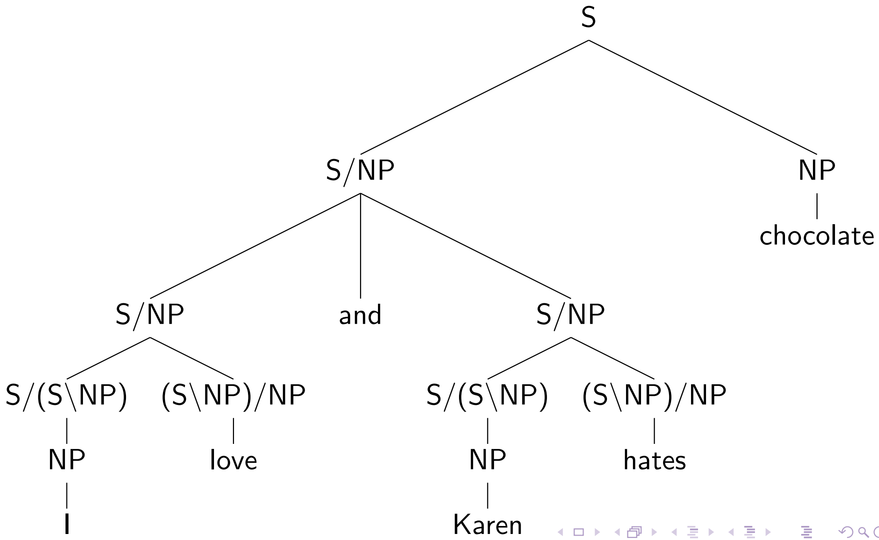
Composition Combinator

Forward ($X/Y \quad Y/Z \Rightarrow X/Z$) and backward ($Y\backslash Z \quad X\backslash Y \Rightarrow X\backslash Z$)



Type-Raising Combinator

Forward ($X \Rightarrow Y/(Y\backslash X)$) and backward ($X \Rightarrow Y\backslash(Y/X)$)



Back to Semantics

Each combinator also tells us what to do with the semantic attachments.

- Forward application: $X/Y : f \quad Y : g \Rightarrow X : f(g)$
- Forward composition:
  $X/Y : f \quad Y/Z : g \Rightarrow X/Z : \lambda x.f(g(x))$
- Forward type-raising: $X : g \Rightarrow Y/(Y\backslash X) : \lambda f.f(g)$

CCG Lexicon

Most of the work is done in the lexicon!

Syntactic and semantic information is much more formal here.

- Slash categories define where all the syntactic arguments are expected to be
- λ-expressions define how the expected arguments get “used” to build up a FOL expression

Extensive discussion: Carpenter (1997). Again, this is one theory out of many!

Semantic Parsing

Semantic parsing comprises a wide range of tasks where strings are mapped into meaning representation languages. Examples:

- Programming languages, especially query languages that can be used to answer questions using a database (Zettlemoyer and Collins, 2005, e.g.,)
- Schemas designed around real-world event-types (called "frames"); trying to extract "who did what to whom?" (Baker et al., 1998; Palmer et al., 2005)

These tasks have inspired a rich literature on *learning* for semantic parsing, which builds heavily on the techniques we've covered in this class and frequently goes beyond supervised learning (e.g., maybe we observe text inputs and semantic outputs, but no syntax that links them). Kamath and Das (2019) gives a survey.

Stepping Back: Linguistic Structure Prediction

Beyond syntactic and semantic parsing, people have applied similar ideas to:

- Coreference resolution (chapter 15)
- Discourse parsing (chapter 16)
- Extractive summarization
- Machine translation

Smith (2011) gives a general, abstract framework for thinking about these problems that predates (but combines nicely with) neural models.

Warning

Relative to other problems we have studied in this class, the tasks we discussed today have very little training data to provide supervision. Consider:

- Whose language was considered when the grammar was built, the annotation conventions were defined, and the data was selected?
- How will accuracy be affected when a parser is applied to someone else's data?

Longstanding research question: how well can linguistic structure be automatically discovered with little (or no) supervision?

Closing Discussion

Why does linguistic structure matter today?

Closing Discussion

Why does linguistic structure matter today?

- If NLP is a tool for the scientific study of language, then developing models and algorithms can advance and test computational theories of language.

Closing Discussion

Why does linguistic structure matter today?

- If NLP is a tool for the scientific study of language, then developing models and algorithms can advance and test computational theories of language.
- If NLP is a tool for the scientific study of models, then linguistic theories tell us what properties/behaviors we should look for.

Closing Discussion

Why does linguistic structure matter today?

- If NLP is a tool for the scientific study of language, then developing models and algorithms can advance and test computational theories of language.
- If NLP is a tool for the scientific study of models, then linguistic theories tell us what properties/behaviors we should look for.
- Linguistic structure is a tool for interpretability.

Closing Discussion

Why does linguistic structure matter today?

- If NLP is a tool for the scientific study of language, then developing models and algorithms can advance and test computational theories of language.
- If NLP is a tool for the scientific study of models, then linguistic theories tell us what properties/behaviors we should look for.
- Linguistic structure is a tool for interpretability.
- Theories of syntax and semantics can be used to impose inductive bias on learning procedures, reducing cost.

References I

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. In *Proc. of ACL-COLING*, 1998.

Bob Carpenter. *Type-logical semantics*. MIT Press, 1997.

Noam Chomsky. Three models for the description of language. *Information Theory, IEEE Transactions on*, 2(3):113–124, 1956.

John Cocke and Jacob T. Schwartz. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, New York University, 1970.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proc. of NAACL*, 2016.

Jacob Eisenstein. *Introduction to Natural Language Processing*. MIT Press, 2019.

Mark Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–32, 1998.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, third edition, forthcoming. URL `https://web.stanford.edu/~jurafsky/slp3/`.

Aishwarya Kamath and Rajarshi Das. A survey on semantic parsing. In *Proc. of AKBC*, 2019.

References II

Tadao Kasami. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Lab, 1965.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2): 313–330, 1993.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT-EMNLP*, 2005. URL `http://www.aclweb.org/anthology/H/H05/H05-1066.pdf`.

Igor A. Mel'čuk. *Dependency Syntax: Theory and Practice*. State University Press of New York, 1987.

Richard Montague. Universal grammar. *Theoria*, 36:373–398, 1970.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105, 2005.

Geoffrey K. Pullum. *The Great Eskimo Vocabulary Hoax and Other Irreverent Essays on the Study of Language*. University of Chicago Press, 1991.

Noah A. Smith. *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool, 2011. URL `http://www.morganclaypool.com/doi/pdf/10.2200/S00361ED1V01Y201105HLT013.pdf`.

Mark Steedman. *The Syntactic Process*. MIT Press, 2000.

References III

Mitchell Stern, Jacob Andreas, and Dan Klein. A minimal span-based neural constituency parser. In *Proc. of ACL*, 2017.

L. Tesnière. *Éléments de Syntaxe Structurale*. Klincksieck, 1959.

Daniel H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2), 1967.

Luke Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of UAI*, 2005.

Extras

Not Constituents
(Pullum, 1991)

- *If on a Winter's Night a Traveler* (by Italo Calvino)
- *Nuclear and Radiochemistry* (by Gerhart Friedlander et al.)
- *The Fire Next Time* (by James Baldwin)
- *A Tad Overweight, but Violet Eyes to Die For* (by G.B. Trudeau)
- *Sometimes a Great Notion* (by Ken Kesey)
- [how can we know the] *Dancer from the Dance* (by Andrew Holleran)