Natural Language Processing (CSE 447/547M): Phrase Structure Parsing

Noah Smith

University of Washington
nasmith@cs.washington.edu

February 20, 2019

Administrivia

- Please double-check your tarballs before you turn them in!
- Assignment 2: watch for updated version!
- Highlights from mid-quarter evaluation:
  - You are mostly happy with lectures, TAs, office hours, the switch to Piazza and TA responsiveness there, slides, question management in class, section slides, textbook.
  - You want more formal instruction on AllenNLP and more explanation of the connection between lecture and assignments.

Context-Free Grammar

A **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 "$N \to \boldsymbol{\alpha}$" 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

(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.

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?

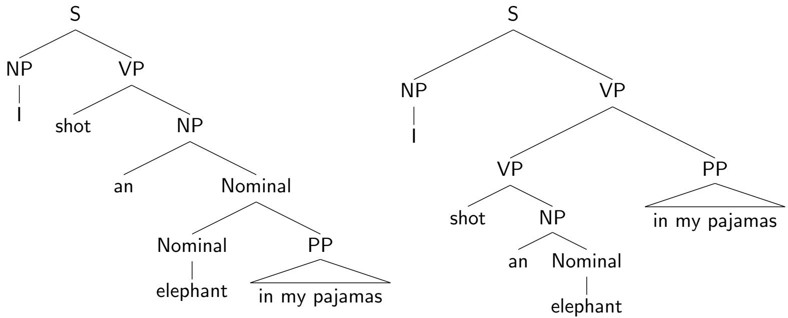
The proof is a derivation.

Related problem: **parsing**:

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

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

Ambiguity

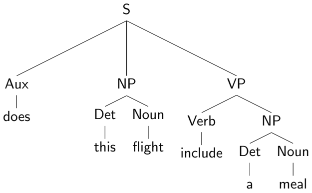

S
NP
VP
I
shot
NP
an
Nominal
Nominal
PP
elephant
in my pajamas
S
NP
VP
I
VP
PP
shot
NP
in my pajamas
an
Nominal
elephant

Parser Evaluation

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

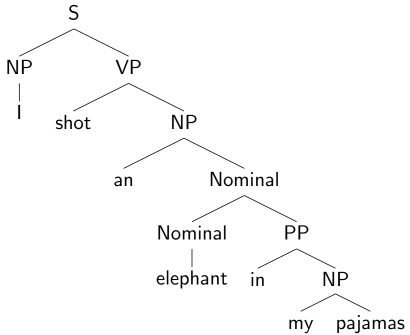
Example:

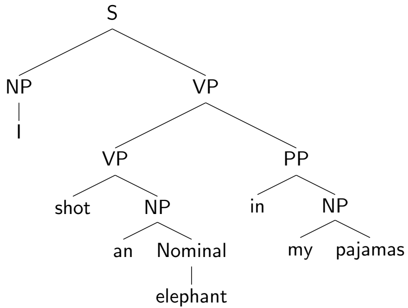


$$\longrightarrow \left\langle \begin{array}{ll} \langle \mathsf{S}, 1, 6\rangle, & \langle \mathsf{NP}, 2, 3\rangle, \\ \langle \mathsf{VP}, 4, 6\rangle, & \langle \mathsf{NP}, 5, 6\rangle \end{array} \right\rangle$$

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

Tree Comparison Example





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

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.

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.
   - Today: scores are defined using the rules.

$$\text{predict}(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{t}=\langle r_1,\ldots,r_k\rangle} \sum_{i=1}^{k} s(r_i) = \operatorname*{argmax}_{\boldsymbol{t}} \sum_{r \in \mathcal{R}} c_{\boldsymbol{t}}(r) s(r) \tag{1}$$

where $\boldsymbol{t}$ is constrained to be a grammatical tree with $\boldsymbol{x}$ as the yield, and $\langle r_1, \ldots, r_k\rangle$ denotes the bag of rules used in deriving the tree. Denote this set of grammatical trees with $\boldsymbol{x}$ as the yield $\mathcal{T}_{\boldsymbol{x}}$.

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, $p(* \mid N)$.

Note: in the notation of equation 1, $s(N \rightarrow \alpha) = \log p(\alpha \mid N)$, and the total score of a tree is its *log* probability.

PCFG Example

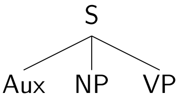
S

Write down the start symbol. Here: S

Probability:

$$1$$

PCFG Example

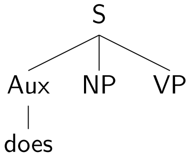


Choose a rule from the “S” distribution. Here: S $\rightarrow$ Aux NP VP

Probability:

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

PCFG Example

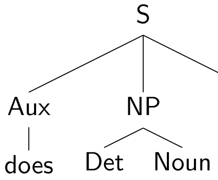


Choose a rule from the "Aux" distribution. Here: Aux $\rightarrow$ does

Probability:

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

PCFG Example

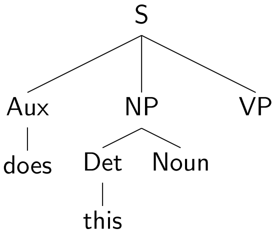


Choose a rule from the "NP" distribution. Here: NP $\rightarrow$ Det Noun

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

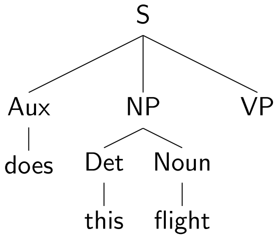


Choose a rule from the "Det" distribution. Here: Det $\rightarrow$ this

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

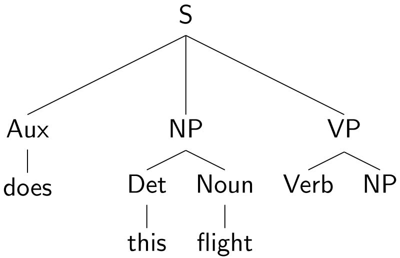


Choose a rule from the "Noun" distribution. Here: Noun $\rightarrow$ flight

Probability:

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

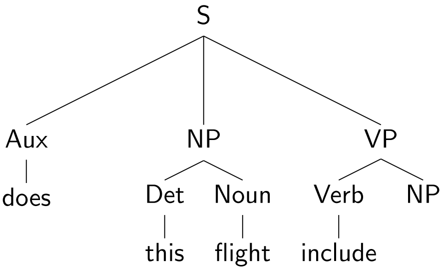
PCFG Example



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

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

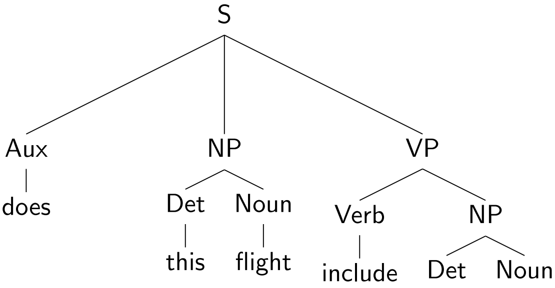
PCFG Example



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

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

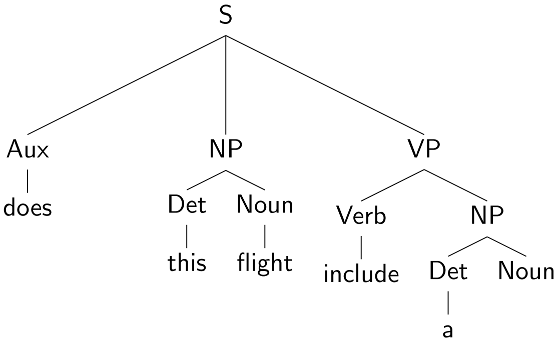
PCFG Example



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

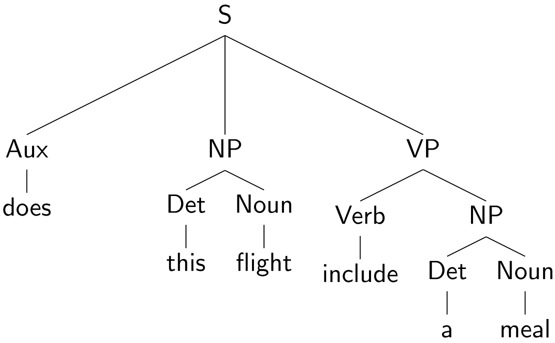


Choose a rule from the "Det" distribution. Here: Det $\rightarrow$ a

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:

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

PCFG as a Noisy Channel

$$\boxed{\text{source}} \longrightarrow \boldsymbol{T} \longrightarrow \boxed{\text{channel}} \longrightarrow \boldsymbol{X}$$

The PCFG defines the source model.

The channel is deterministic: it erases everything except the tree's leaves (the yield).

Decoding:

$$\begin{aligned} & \operatorname*{argmax}_{\boldsymbol{t}} p(\boldsymbol{t}) \cdot \begin{cases} 1 & \text{if } \boldsymbol{t} \in \mathcal{T}_{\boldsymbol{x}} \\ 0 & \text{otherwise} \end{cases} \\ = & \operatorname*{argmax}_{\boldsymbol{t} \in \mathcal{T}_{\boldsymbol{x}}} p(\boldsymbol{t}) \end{aligned}$$

Probabilistic Parsing with CFGs

- 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: $\underset{\boldsymbol{t} \in \mathcal{T}_{\boldsymbol{x}}}{\operatorname{argmax}} \log p(\boldsymbol{t} \mid \boldsymbol{x})$ (if $\boldsymbol{x}$ is in the language of the grammar)

Probabilistic CKY

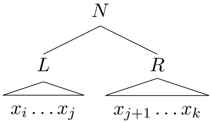
Base case: for $i \in \{1, \ldots, n\}$ and for each $N \in \mathcal{N}$:

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

In general, $\heartsuit_{i:k}(N)$ is the log probability of the best tree spanning $\langle x_i, \ldots, x_k \rangle$ rooted at N.

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

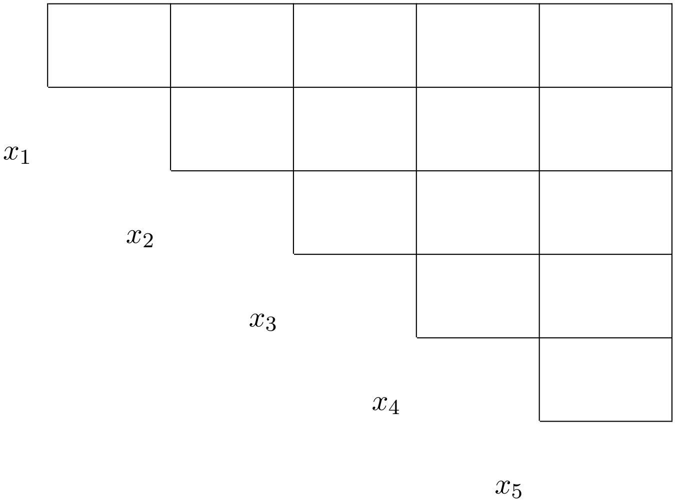
$$\heartsuit_{i:k}(N) = \max_{L,R \in \mathcal{N}, j \in \{i,\ldots,k-1\}} \log p(L\ R \mid N) + \heartsuit_{i:j}(L) + \heartsuit_{(j+1):k}(R)$$



Solution:

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

Parse Chart



Parse Chart

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

Remarks

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

Remarks

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

Remarks

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

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.

Probabilistic CKY with an Agenda

1. Initialize every item's value in the **chart** to the "default" (zero).
2. Place all initializing updates onto the **agenda**.
3. While the agenda is not empty or the goal is not reached:
   - Pop the highest-priority update from the agenda (item I with value v)
   - If $I = \mathrm{goal}$, then return v.
   - If $v > \mathrm{chart}(I)$:
     - $\mathrm{chart}(I) \leftarrow v$
     - Find all combinations of I with other items in the chart, generating new possible updates; place these on the agenda.

Any priority function will work! But smart ordering will save time.

This idea can also be applied to other algorithms (e.g., Viterbi).

Demo of Recent State of the Art

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

Machine Learning and Parsing

Machine Learning and Parsing

- Define arbitrary features on trees, based on linguistic knowledge; to parse, use a PCFG to generate a **k-best list** of parses (Huang and Chiang, 2005), then train a log-linear model to rerank (Charniak and Johnson, 2005). Or a neural model (Socher et al., 2013).

Machine Learning and Parsing

- Define arbitrary features on trees, based on linguistic knowledge; to parse, use a PCFG to generate a **k-best list** of parses (Huang and Chiang, 2005), then train a log-linear model to rerank (Charniak and Johnson, 2005). Or a neural model (Socher et al., 2013).
- Define rule-local features on trees and any part of the input sentence ($s(\boldsymbol{x}, i, k, N, L, R)$; minimize hinge or log loss. Recent state of the art is simpler, scoring labeled spans, $s(\boldsymbol{x}, i, k, N)$ (Stern et al., 2017).
  - These exploit dynamic programming algorithms for training.

Machine Learning and Parsing

- Define arbitrary features on trees, based on linguistic knowledge; to parse, use a PCFG to generate a **k-best list** of parses (Huang and Chiang, 2005), then train a log-linear model to rerank (Charniak and Johnson, 2005). Or a neural model (Socher et al., 2013).
- Define rule-local features on trees and any part of the input sentence ($s(\boldsymbol{x}, i, k, N, L, R)$; minimize hinge or log loss. Recent state of the art is simpler, scoring labeled spans, $s(\boldsymbol{x}, i, k, N)$ (Stern et al., 2017).
  - These exploit dynamic programming algorithms for training.
- Learn refinements on the constituents, as latent variables (Petrov et al., 2006).

Machine Learning and Parsing

- Define arbitrary features on trees, based on linguistic knowledge; to parse, use a PCFG to generate a **k-best list** of parses (Huang and Chiang, 2005), then train a log-linear model to rerank (Charniak and Johnson, 2005). Or a neural model (Socher et al., 2013).
- Define rule-local features on trees and any part of the input sentence ($s(\boldsymbol{x}, i, k, N, L, R)$; minimize hinge or log loss. Recent state of the art is simpler, scoring labeled spans, $s(\boldsymbol{x}, i, k, N)$ (Stern et al., 2017).
  - These exploit dynamic programming algorithms for training.
- Learn refinements on the constituents, as latent variables (Petrov et al., 2006).
- **Recurrent neural network grammars**, generative models like PCFGs that encode arbitrary previous derivation steps in a vector (Dyer et al., 2016). Parsing requires some tricks.

References I

Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of ACL*, 2005.

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, 2016. To appear.

Liang Huang and David Chiang. Better k-best parsing. In *Proc. of IWPT*, 2005.

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

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proc. of COLING-ACL*, 2006.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. In *Proc. of ACL*, 2013.

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

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