



Natural Language Processing

Anoop Sarkar

anoopsarkar.github.io/nlp-class

Simon Fraser University

September 25, 2018

Natural Language Processing

Anoop Sarkar

anoopsarkar.github.io/nlp-class

Simon Fraser University

Part 1: Ambiguity

Context Free Grammars and Ambiguity

$S \rightarrow NP VP$
 $VP \rightarrow V NP$
 $VP \rightarrow VP PP$
 $PP \rightarrow P NP$
 $NP \rightarrow NP PP$
 $NP \rightarrow Calvin$
 $NP \rightarrow monsters$
 $NP \rightarrow school$
 $V \rightarrow imagined$
 $P \rightarrow in$

What is the analysis using the above grammar for:
Calvin imagined monsters in school

Context Free Grammars and Ambiguity

Calvin imagined monsters in school

```
(S (NP Calvin)
  (VP (V imagined)
      (NP (NP monsters)
          (PP (P in)
              (NP school))))))
```

```
(S (NP Calvin)
  (VP (VP (V imagined)
          (NP monsters))
      (PP (P in)
          (NP school))))
```

Which one is more plausible?

Context Free Grammars and Ambiguity

Calvin imagined monsters in school



Calvin imagined monsters in school



Ambiguity Kills (your parser)

natural language learning course

(run demos/parsing-ambiguity.py)

((natural language) (learning course))

((natural language) learning) course)

((natural (language learning)) course)

(natural (language (learning course)))

(natural ((language learning) course))

- ▶ Some difficult issues:
 - ▶ Which one is more plausible?
 - ▶ How many analyses for a given input?
 - ▶ Computational complexity of parsing language

Number of derivations

CFG rules $\{ N \rightarrow N N, N \rightarrow a \}$

$n : a^n$	number of parses
1	1
2	1
3	2
4	5
5	14
6	42
7	132
8	429
9	1430
10	4862
11	16796

CFG Ambiguity

- ▶ Number of parses in previous table is an integer series, known as the Catalan numbers
- ▶ Catalan numbers have a closed form:

$$Cat(n) = \frac{1}{n+1} \binom{2n}{n}$$

- ▶ $\binom{a}{b}$ is the *binomial coefficient*

$$\binom{a}{b} = \frac{a!}{(b!(a-b)!)}$$

Catalan numbers

- ▶ Why Catalan numbers? $\text{Cat}(n)$ is the number of ways to parenthesize an expression of length n with two conditions:
 1. there must be equal numbers of open and close parens
 2. they must be properly nested so that an open precedes a close
- ▶ $((ab)c)d$ $(a(bc))d$ $(ab)(cd)$ $a((bc)d)$ $a(b(cd))$
- ▶ For an expression of with n ways to form constituents there are a total of $2n$ choose n parenthesis pairs. Then divide by $n + 1$ to remove invalid parenthesis pairs.
- ▶ For more details see (Church and Patil, CL Journal, 1982)

Natural Language Processing

Anoop Sarkar

anoopsarkar.github.io/nlp-class

Simon Fraser University

Part 2: Context Free Grammars

Context-Free Grammars

- ▶ A CFG is a 4-tuple: (N, T, R, S) , where
 - ▶ N is a set of non-terminal symbols,
 - ▶ T is a set of terminal symbols which can include the empty string ϵ . T is analogous to Σ the alphabet in FSAs.
 - ▶ R is a set of rules of the form $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in \{N \cup T\}^*$
 - ▶ S is a set of start symbols, $S \in N$

Context-Free Grammars

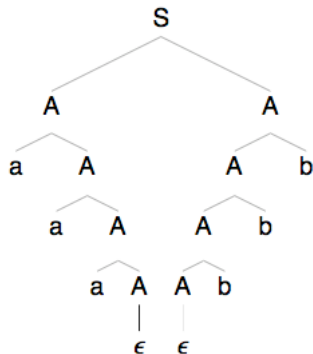
- ▶ Here's an example of a CFG, let's call this one G :
 1. $S \rightarrow a S b$
 2. $S \rightarrow \epsilon$
- ▶ What is the language of this grammar, which we will call $L(G)$, the set of strings *generated* by this grammar **How?**
Notice that there cannot be any FSA that corresponds exactly to this set of strings $L(G)$ **Why?**
- ▶ What is the *tree set* or derivations produced by this grammar?

Context-Free Grammars

- ▶ This notion of generating both the strings and the trees is an important one for Computational Linguistics
- ▶ Consider the trees for the grammar G' :
 $P = \{S \rightarrow A A, A \rightarrow aA, A \rightarrow A b, A \rightarrow \epsilon\},$
 $\Sigma = \{a, b\}, N = \{S, A\}, T = \{a, b, \epsilon\}, S = \{S\}$
- ▶ Why is it called *context-free* grammar?

Context-Free Grammars

- Can the grammar G' produce only trees with equal height subtrees on the left and right?

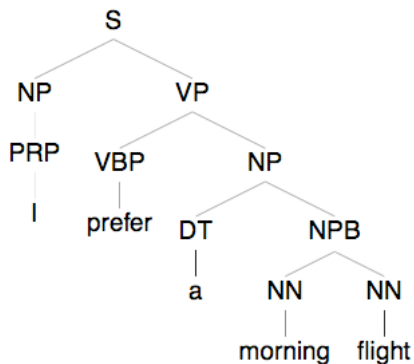


Parse Trees

Consider the grammar with rules:

$$\begin{aligned} S &\rightarrow NP VP \\ NP &\rightarrow PRP \\ NP &\rightarrow DT NPB \\ VP &\rightarrow VBP NP \\ NPB &\rightarrow NN NN \\ PRP &\rightarrow I \\ VBP &\rightarrow prefer \\ DT &\rightarrow a \\ NN &\rightarrow morning \\ NN &\rightarrow flight \end{aligned}$$

Parse Trees



Parse Trees: Equivalent Representations

- ▶ (S (NP (PRP I)) (VP (VBP prefer) (NP (DT a) (NPB (NN morning) (NN flight))))))
- ▶ [S [NP [PRP I]] [VP [VBP prefer] [NP [DT a] [NPB [NN morning] [NN flight]]]]]]

Ambiguous Grammars

- ▶ $S \rightarrow S S$
- ▶ $S \rightarrow a$
- ▶ Given the above rules, consider the input aaa , what are the valid parse trees?
- ▶ Now consider the input $aaaa$

Natural Language Processing

Anoop Sarkar

anoopsarkar.github.io/nlp-class

Simon Fraser University

Part 3: Structural Ambiguity

Ambiguity

- ▶ Part of Speech ambiguity

saw → **noun**

saw → **verb**

- ▶ Structural ambiguity: Prepositional Phrases

I saw (the man) with the telescope

I saw (the man with the telescope)

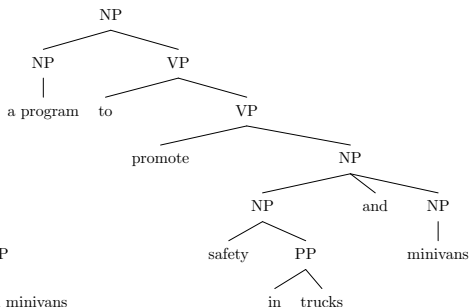
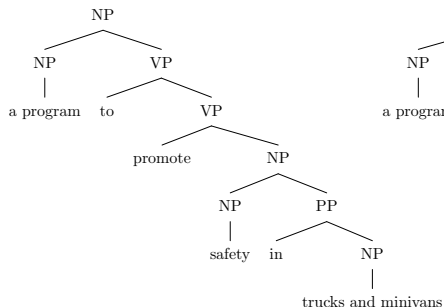
- ▶ Structural ambiguity: Coordination

a program to promote safety in ((trucks) and (minivans))

a program to promote ((safety in trucks) and (minivans))

((a program to promote safety in trucks) and (minivans))

Ambiguity \leftarrow attachment choice in alternative parses



Ambiguity in Prepositional Phrases

- ▶ noun attach: *I bought the shirt with pockets*
- ▶ verb attach: *I washed the shirt with soap*
- ▶ As in the case of other attachment decisions in parsing: it depends on the meaning of the entire sentence – needs world knowledge, etc.
- ▶ Maybe there is a simpler solution: we can attempt to solve it using heuristics or associations between words

Structure Based Ambiguity Resolution

- ▶ Right association: a constituent (NP or PP) tends to attach to another constituent immediately to its right (Kimball 1973)
- ▶ Minimal attachment: a constituent tends to attach to an existing non-terminal using the fewest additional syntactic nodes (Frazier 1978)
- ▶ These two principles make opposite predictions for prepositional phrase attachment
- ▶ Consider the grammar:

$$VP \rightarrow V NP PP \quad (1)$$

$$NP \rightarrow NP PP \quad (2)$$

for input: *I* [_{VP} *saw* [_{NP} *the man* ... [_{PP} *with the telescope*],
RA predicts that the PP attaches to the NP, i.e. use rule (2),
and MA predicts V attachment, i.e. use rule (1)

Structure Based Ambiguity Resolution

- ▶ Garden-paths look structural:
The emergency crews hate most is domestic violence
- ▶ Neither MA or RA account for more than 55% of the cases in real text
- ▶ Psycholinguistic experiments using eyetracking show that humans resolve ambiguities as soon as possible in the left to right sequence using the words to disambiguate
- ▶ Garden-paths are caused by a combination of lexical and structural effects:
The flowers delivered for the patient arrived

Ambiguity Resolution: Prepositional Phrases in English

► Learning Prepositional Phrase Attachment: Annotated Data

v	n1	p	n2	Attachment
join	board	as	director	V
is	chairman	of	N.V.	N
using	crocidolite	in	filters	V
bring	attention	to	problem	V
is	asbestos	in	products	N
making	paper	for	filters	N
including	three	with	cancer	N
⋮	⋮	⋮	⋮	⋮

Prepositional Phrase Attachment

Method	Accuracy
Always noun attachment	59.0
Most likely for each preposition	72.2
Average Human (4 head words only)	88.2
Average Human (whole sentence)	93.2

Some other studies

- ▶ **Toutanova, Manning, and Ng, 2004:** 87.54% using some external knowledge (word classes)
- ▶ **Merlo, Crocker and Berthouzoz, 1997:** test on multiple PPs
 - ▶ generalize disambiguation of 1 PP to 2-3 PPs
 - ▶ 14 structures possible for 3PPs assuming a single verb
 - ▶ all 14 are attested in the Penn WSJ Treebank
 - ▶ 1PP: 84.3% 2PP: 69.6% 3PP: 43.6%
- ▶ **Belinkov+ TACL 2014:** Neural networks for PP attachment (multiple candidate heads)
 - ▶ NN model (no extra data): 86.6%
 - ▶ NN model (lots of raw data for word vectors): 88.7%
 - ▶ NN model with parser and lots of raw data: 90.1%
- ▶ **This experiment is still only part of the real problem faced in parsing English.** Plus other sources of ambiguity in other languages

Natural Language Processing

Anoop Sarkar

anoopsarkar.github.io/nlp-class

Simon Fraser University

Part 4: Weighted Context Free Grammars

Treebanks

- ▶ What is the CFG that can be extracted from this single tree:

```
(S  (NP (Det the) (NP man))
    (VP (VP (V played)
             (NP (Det a) (NP game)))
        (PP (P with)
             (NP (Det the) (NP dog))))))
```

PCFG

<i>S</i>	→	<i>NP VP</i>	<i>c</i> = 1
<i>NP</i>	→	<i>Det NP</i>	<i>c</i> = 3
<i>NP</i>	→	<i>man</i>	<i>c</i> = 1
<i>NP</i>	→	<i>game</i>	<i>c</i> = 1
<i>NP</i>	→	<i>dog</i>	<i>c</i> = 1
<i>VP</i>	→	<i>VP PP</i>	<i>c</i> = 1
<i>VP</i>	→	<i>V NP</i>	<i>c</i> = 1
<i>PP</i>	→	<i>P NP</i>	<i>c</i> = 1
<i>Det</i>	→	<i>the</i>	<i>c</i> = 2
<i>Det</i>	→	<i>a</i>	<i>c</i> = 1
<i>V</i>	→	<i>played</i>	<i>c</i> = 1
<i>P</i>	→	<i>with</i>	<i>c</i> = 1

- ▶ We can do this with multiple trees. Simply count occurrences of CFG rules over all the trees.
- ▶ A repository of such trees labelled by a human is called a TreeBank.

Probabilistic CFG (PCFG)

S	\rightarrow	$NP VP$	1
VP	\rightarrow	$V NP$	0.9
VP	\rightarrow	$VP PP$	0.1
PP	\rightarrow	$P NP$	1
NP	\rightarrow	$NP PP$	0.25
NP	\rightarrow	$Calvin$	0.25
NP	\rightarrow	$monsters$	0.25
NP	\rightarrow	$school$	0.25
V	\rightarrow	$imagined$	1
P	\rightarrow	in	1

$$P(input) = \sum_{tree} P(tree \mid input)$$

$$P(Calvin \text{ imagined monsters in school}) = ?$$

Notice that $P(VP \rightarrow V NP) + P(VP \rightarrow VP PP) = 1.0$

Probabilistic CFG (PCFG)

$P(\textit{Calvin imagined monsters in school}) = ?$

```
(S (NP Calvin)
  (VP (V imagined)
    (NP (NP monsters)
      (PP (P in)
        (NP school))))))
```

```
(S (NP Calvin)
  (VP (VP (V imagined)
    (NP monsters))
    (PP (P in)
      (NP school))))
```


Probabilistic CFG (PCFG)

```
(S (NP Calvin)
  (VP (V imagined)
      (NP (NP monsters)
          (PP (P in)
              (NP school))))))
```

$$\begin{aligned}P(\text{tree}_1) &= P(S \rightarrow NP VP) \times P(NP \rightarrow Calvin) \times P(VP \rightarrow V NP) \times \\&\quad P(V \rightarrow imagined) \times P(NP \rightarrow NP PP) \times P(NP \rightarrow monsters) \times \\&\quad P(PP \rightarrow P NP) \times P(P \rightarrow in) \times P(NP \rightarrow school) \\&= 1 \times 0.25 \times 0.9 \times 1 \times 0.25 \times 0.25 \times 1 \times 1 \times 0.25 = .003515625\end{aligned}$$

Probabilistic CFG (PCFG)

(S (NP Calvin)
 (VP (VP (V imagined)
 (NP monsters))
 (PP (P in)
 (NP school)))))

$$\begin{aligned}P(\text{tree}_2) &= P(S \rightarrow NP VP) \times P(NP \rightarrow Calvin) \times P(VP \rightarrow VP PP) \times \\&\quad P(VP \rightarrow V NP) \times P(V \rightarrow imagined) \times P(NP \rightarrow monsters) \times \\&\quad P(PP \rightarrow P NP) \times P(P \rightarrow in) \times P(NP \rightarrow school) \\&= 1 \times 0.25 \times 0.1 \times 0.9 \times 1 \times 0.25 \times 1 \times 1 \times 0.25 = .00140625\end{aligned}$$

Probabilistic CFG (PCFG)

$$\begin{aligned}P(\textit{Calvin imagined monsters in school}) &= P(\textit{tree}_1) + P(\textit{tree}_2) \\&= .003515625 + .00140625 \\&= .004921875 \\ \text{Most likely tree is } \textit{tree}_1 &= \arg \max_{\textit{tree}} P(\textit{tree} \mid \textit{input})\end{aligned}$$

```
(S (NP Calvin)
  (VP (V imagined)
    (NP (NP monsters)
      (PP (P in)
        (NP school))))))
```

```
(S (NP Calvin)
  (VP (VP (V imagined)
    (NP monsters))
    (PP (P in)
      (NP school))))
```

Probabilistic Context-Free Grammars (PCFG)

- ▶ A PCFG is a 4-tuple: (N, T, R, S) , where
 - ▶ N is a set of non-terminal symbols,
 - ▶ T is a set of terminal symbols which can include the empty string ϵ . T is analogous to Σ the alphabet in FSAs.
 - ▶ R is a set of rules of the form $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in \{N \cup T\}^*$
 - ▶ $P(R)$ is the probability of rule $R : A \rightarrow \alpha$ such that $\sum_{\alpha} P(A \rightarrow \alpha) = 1.0$
 - ▶ S is a set of start symbols, $S \in N$

PCFG

- ▶ Central condition: $\sum_{\alpha} P(A \rightarrow \alpha) = 1$
- ▶ Called a *proper* PCFG if this condition holds
- ▶ Note that this means $P(A \rightarrow \alpha) = P(\alpha \mid A) = \frac{f(A, \alpha)}{f(A)}$
- ▶ $P(T \mid S) = \frac{P(T, S)}{P(S)} = P(T, S) = \prod_i P(RHS_i \mid LHS_i)$

- ▶ What is the PCFG that can be extracted from this single tree:

(S (NP (Det the) (NP man))
 (VP (VP (V played)
 (NP (Det a) (NP game)))
 (PP (P with)
 (NP (Det the) (NP dog))))))

- ▶ How many different rhs α exist for $A \rightarrow \alpha$ where A can be S , NP , VP , PP , Det , N , V , P

PCFG

<i>S</i>	→	<i>NP VP</i>	<i>c</i> = 1	<i>p</i> = 1/1	= 1.0
<i>NP</i>	→	<i>Det NP</i>	<i>c</i> = 3	<i>p</i> = 3/6	= 0.5
<i>NP</i>	→	<i>man</i>	<i>c</i> = 1	<i>p</i> = 1/6	= 0.1667
<i>NP</i>	→	<i>game</i>	<i>c</i> = 1	<i>p</i> = 1/6	= 0.1667
<i>NP</i>	→	<i>dog</i>	<i>c</i> = 1	<i>p</i> = 1/6	= 0.1667
<i>VP</i>	→	<i>VP PP</i>	<i>c</i> = 1	<i>p</i> = 1/2	= 0.5
<i>VP</i>	→	<i>V NP</i>	<i>c</i> = 1	<i>p</i> = 1/2	= 0.5
<i>PP</i>	→	<i>P NP</i>	<i>c</i> = 1	<i>p</i> = 1/1	= 1.0
<i>Det</i>	→	<i>the</i>	<i>c</i> = 2	<i>p</i> = 2/3	= 0.67
<i>Det</i>	→	<i>a</i>	<i>c</i> = 1	<i>p</i> = 1/3	= 0.33
<i>V</i>	→	<i>played</i>	<i>c</i> = 1	<i>p</i> = 1/1	= 1.0
<i>P</i>	→	<i>with</i>	<i>c</i> = 1	<i>p</i> = 1/1	= 1.0

- ▶ We can do this with multiple trees. Simply count occurrences of CFG rules over all the trees.
- ▶ A repository of such trees labelled by a human is called a TreeBank.

Natural Language Processing

Anoop Sarkar

anoopsarkar.github.io/nlp-class

Simon Fraser University

Part 5: Lexicalized Context Free Grammars

Ambiguity

- ▶ Part of Speech ambiguity

saw → noun

saw → verb

- ▶ Structural ambiguity: Prepositional Phrases

I saw (the man) with the telescope

I saw (the man with the telescope)

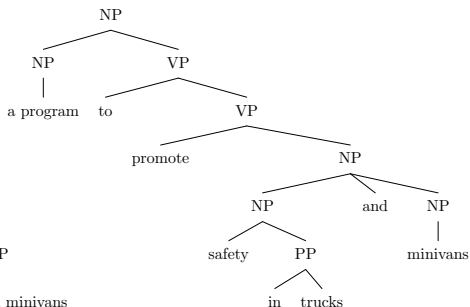
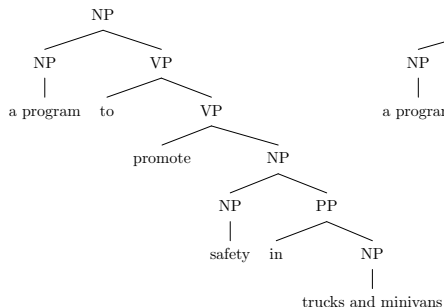
- ▶ Structural ambiguity: Coordination

a program to promote safety in ((trucks) and (minivans))

a program to promote ((safety in trucks) and (minivans))

((a program to promote safety in trucks) and (minivans))

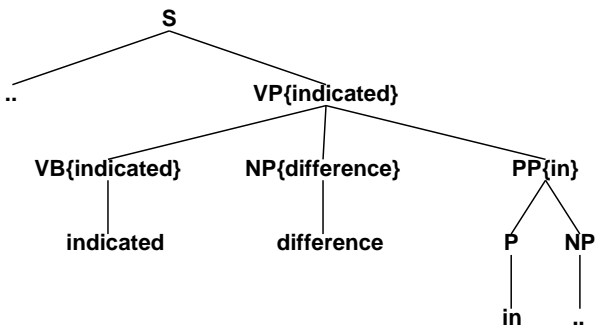
Ambiguity \leftarrow attachment choice in alternative parses



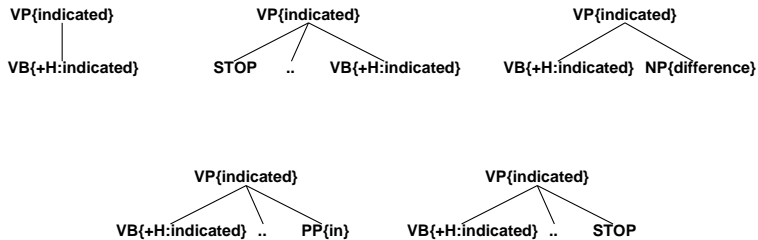
Parsing as a machine learning problem

- ▶ S = a sentence
 T = a parse tree
A statistical parsing model defines $P(T | S)$
- ▶ Find best parse: $\arg \max_T P(T | S)$
- ▶ $P(T | S) = \frac{P(T, S)}{P(S)} = P(T, S)$
- ▶ Best parse: $\arg \max_T P(T, S)$
- ▶ e.g. for PCFGs: $P(T, S) = \prod_{i=1 \dots n} P(\text{RHS}_i | \text{LHS}_i)$

Adding Lexical Information to PCFG



Adding Lexical Information to PCFG (Collins 99, Charniak 00)



$$\begin{aligned} &P_h(VB \mid VP, indicated) \times P_l(STOP \mid VP, VB, indicated) \times \\ &P_r(NP(difference) \mid VP, VB, indicated) \times \\ &P_r(PP(in) \mid VP, VB, indicated) \times \\ &P_r(STOP \mid VP, VB, indicated) \end{aligned}$$

Evaluation of Parsing

- Consider a candidate parse to be evaluated against the truth (or gold-standard parse):

candidate: (S (A (P this) (Q is)) (A (R a) (T test)))

gold: (S (A (P this)) (B (Q is) (A (R a) (T test))))

- In order to evaluate this, we list all the constituents

Candidate	Gold
(0,4,S)	(0,4,S)
(0,2,A)	(0,1,A)
(2,4,A)	(1,4,B)
	(2,4,A)

- Skip spans of length 1 which would be equivalent to part of speech tagging accuracy.

- Precision is defined as $\frac{\#correct}{\#proposed} = \frac{2}{3}$ and recall as

$$\frac{\#correct}{\#in\ gold} = \frac{2}{4}.$$

- Another measure: crossing brackets,

candidate: [an [incredibly expensive] coat] (1 CB)

gold: [an [incredibly [expensive coat]]

Evaluation of Parsing

Bracketing recall R $=$ $\frac{\text{num of correct constituents}}{\text{num of constituents in the goldfile}}$

Bracketing precision P $=$ $\frac{\text{num of correct constituents}}{\text{num of constituents in the parsed file}}$

Complete match $=$ % of sents where recall & precision are both 100%

Average crossing $=$ $\frac{\text{num of constituents crossing a goldfile constituent}}{\text{num of sents}}$

No crossing $=$ % of sents which have 0 crossing brackets

2 or less crossing $=$ % of sents which have ≤ 2 crossing brackets

Statistical Parsing Results

$$\text{F1-score} = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

System	$\leq 100\text{wds}$ F1-score
Shift-Reduce (Magerman, 1995)	84.14
PCFG with Lexical Features (Charniak, 1999)	89.54
Unlexicalized Berkeley parser (Petrov et al, 2007)	90.10
<i>n</i> -best Re-ranking (Charniak and Johnson, 2005)	91.02
Tree-insertion grammars (Carreras, Collins, Koo, 2008)	91.10
Ensemble <i>n</i> -best Re-ranking (Johnson and Ural, 2010)	91.49
Forest Re-ranking (Huang, 2010)	91.70
Unlabeled Data with Self-Training (McCloskey et al, 2006)	92.10
Self-Attention (Kitaev and Klein, 2018)	93.55
Self-Attention with unlabeled data (Kitaev and Klein, 2018)	95.13

Natural Language Processing

Anoop Sarkar

anoopsarkar.github.io/nlp-class

Simon Fraser University

Part 6: Practical Issues in Parsing

Practical Issues: Beam Thresholding and Priors

- ▶ Read the parsing slides before this section.
- ▶ Probability of nonterminal X spanning $j \dots k$: $N[X, j, k]$
- ▶ Beam Thresholding compares $N[X, j, k]$ with every other Y where $N[Y, j, k]$
- ▶ But what should be compared?
- ▶ Just the *inside probability*: $P(X \xRightarrow{*} t_j \dots t_k)$?
written as $\beta(X, j, k)$
- ▶ Perhaps $\beta(\text{FRAG}, 0, 3) > \beta(\text{NP}, 0, 3)$, but NPs are much more likely than FRAGs in general

Practical Issues: Beam Thresholding and Priors

- ▶ The correct estimate is the *outside probability*:

$$P(S \overset{*}{\Rightarrow} t_1 \dots t_{j-1} \ X \ t_{k+1} \dots t_n)$$

written as $\alpha(X, j, k)$

- ▶ Unfortunately, you can only compute $\alpha(X, j, k)$ efficiently after you finish parsing and reach $(S, 0, n)$

Practical Issues: Beam Thresholding and Priors

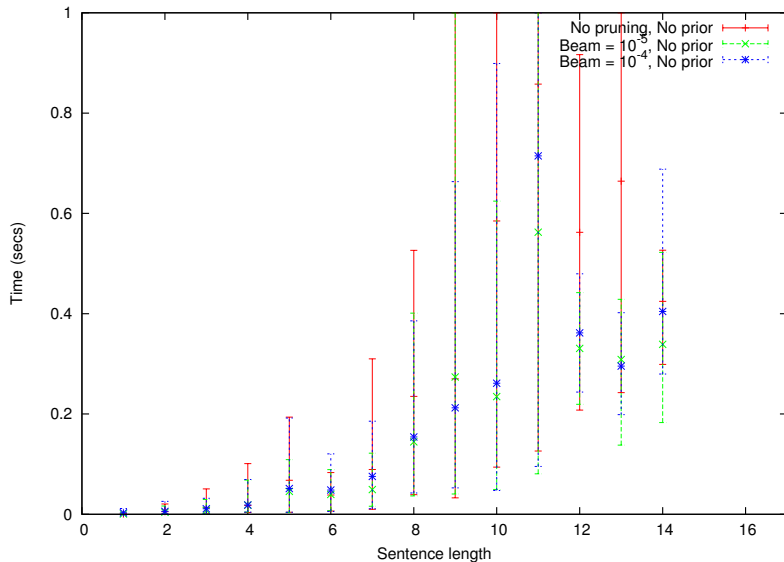
- ▶ To make things easier we multiply the prior probability $P(X)$ with the inside probability
- ▶ In beam Thresholding we compare every new insertion of X for span j, k as follows:
Compare $P(X) \cdot \beta(X, j, k)$ with the most probable Y
 $P(Y) \cdot \beta(Y, j, k)$
- ▶ Assume Y is the most probable entry in j, k , then we compare

$$\text{beam} \cdot P(Y) \cdot \beta(Y, j, k) \quad (3)$$

$$P(X) \cdot \beta(X, j, k) \quad (4)$$

- ▶ If $(4) < (3)$ then we prune X for this span j, k
- ▶ beam is set to a small value, say 0.001 or even 0.01.
- ▶ As the beam value increases, the parser speed increases (since more entries are pruned).
- ▶ A simpler (but not as effective) alternative to using the beam is to keep only the top K entries for each span j, k

Experiments with Beam Thresholding



Experiments with Beam Thresholding

