Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

September 5, 2019

Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

Part 1: Ambiguity

Context Free Grammars and Ambiguity

$$
\begin{array}{ccc}
S & \rightarrow & NP\ VP \\
VP & \rightarrow & V\ NP \\
VP & \rightarrow & VP\ PP \\
PP & \rightarrow & P\ NP \\
NP & \rightarrow & NP\ PP \\
NP & \rightarrow & Calvin \\
NP & \rightarrow & monsters \\
NP & \rightarrow & school \\
V & \rightarrow & imagined \\
P & \rightarrow & in
\end{array}
$$

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

Context Free Grammars and Ambiguity

Calvin imagined monsters in school

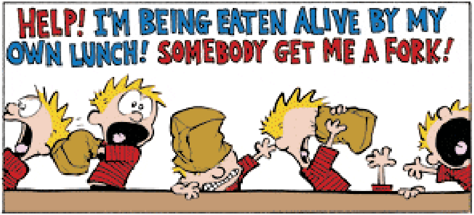
```
(S (NP Calvin)
   (VP (V imagined)
       (NP (NP monsters)
           (PP (P in)
               (NP school)))))
```

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

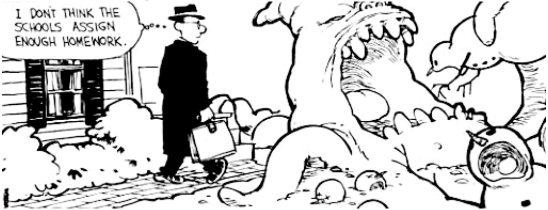
Which one is more plausible?

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? 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

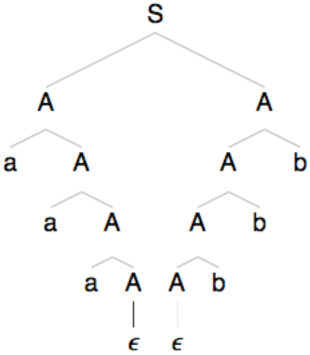
- Here's an example of a CFG, let's call this one G:
  1. $S \rightarrow a\ S\ b$
  2. $S \rightarrow \epsilon$
- What is the language of this grammar, which we will call $L(G)$, the set of strings *generated* by this grammar How? Notice that there cannot be any FSA that corresponds exactly to this set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

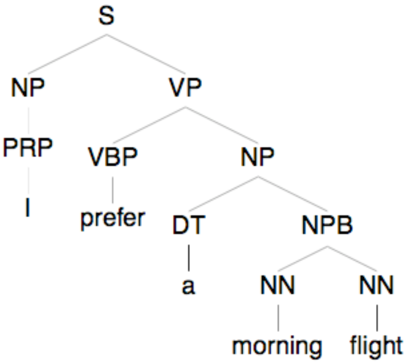


Parse Trees

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*

Inherently Ambiguous Languages

- Consider the following context-free grammar:
  - $S \rightarrow S1 \mid S2$
  - $S1 \rightarrow aXd \mid \epsilon$
  - $X \rightarrow bXc \mid \epsilon$
  - $S2 \rightarrow YZ \mid \epsilon$
  - $Y \rightarrow aYb \mid \epsilon$
  - $Z \rightarrow cZd \mid \epsilon$
- Now parse the input string *abcd* with this grammar
- Notice that we get two parse trees (one with the $S1$ sub-grammar and another with the $S2$ subgrammar).

Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

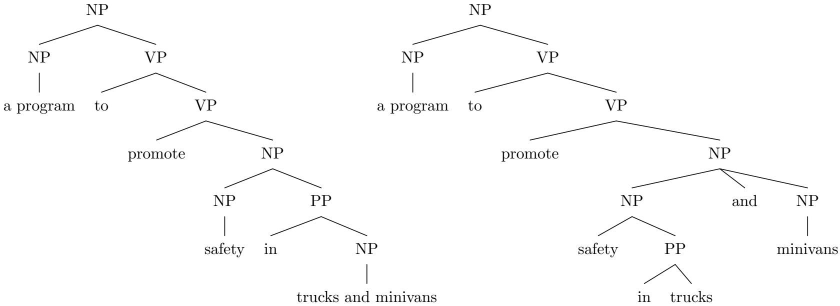
Simon Fraser University

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 ← 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:

$$VP \rightarrow V\ NP\ PP \tag{1}$$

$$NP \rightarrow NP\ PP \tag{2}$$

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

$$
\begin{array}{lcll}
S & \rightarrow & NP\ VP & c = 1 \\
NP & \rightarrow & Det\ NP & c = 3 \\
NP & \rightarrow & man & c = 1 \\
NP & \rightarrow & game & c = 1 \\
NP & \rightarrow & dog & c = 1 \\
VP & \rightarrow & VP\ PP & c = 1 \\
VP & \rightarrow & V\ NP & c = 1 \\
PP & \rightarrow & P\ NP & c = 1 \\
Det & \rightarrow & the & c = 2 \\
Det & \rightarrow & a & c = 1 \\
V & \rightarrow & played & c = 1 \\
P & \rightarrow & with & c = 1
\end{array}
$$

- 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)

$$
\begin{array}{lclc}
S & \rightarrow & NP\ VP & 1 \\
VP & \rightarrow & V\ NP & 0.9 \\
VP & \rightarrow & VP\ PP & 0.1 \\
PP & \rightarrow & P\ NP & 1 \\
NP & \rightarrow & NP\ PP & 0.25 \\
NP & \rightarrow & Calvin & 0.25 \\
NP & \rightarrow & monsters & 0.25 \\
NP & \rightarrow & school & 0.25 \\
V & \rightarrow & imagined & 1 \\
P & \rightarrow & in & 1
\end{array}
$$

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

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

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)))))
```

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

Probabilistic CFG (PCFG)

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

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

Probabilistic CFG (PCFG)

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

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

PCFG

- 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

$$
\begin{array}{cccccc}
S & \rightarrow & NP\ VP & c=1 & p=1/1 & =1.0 \\
NP & \rightarrow & Det\ NP & c=3 & p=3/6 & =0.5 \\
NP & \rightarrow & man & c=1 & p=1/6 & =0.1667 \\
NP & \rightarrow & game & c=1 & p=1/6 & =0.1667 \\
NP & \rightarrow & dog & c=1 & p=1/6 & =0.1667 \\
VP & \rightarrow & VP\ PP & c=1 & p=1/2 & =0.5 \\
VP & \rightarrow & V\ NP & c=1 & p=1/2 & =0.5 \\
PP & \rightarrow & P\ NP & c=1 & p=1/1 & =1.0 \\
Det & \rightarrow & the & c=2 & p=2/3 & =0.67 \\
Det & \rightarrow & a & c=1 & p=1/3 & =0.33 \\
V & \rightarrow & played & c=1 & p=1/1 & =1.0 \\
P & \rightarrow & with & c=1 & p=1/1 & =1.0
\end{array}
$$

- 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.