Ling/CSE 472:
Introduction to Computational Linguistics

5/2: PCFGs

Overview

- Syntax & parsing
- Context free grammar
- CKY
- PCFG & probabilistic CKY
- Evaluating parsing
- Reading questions

Parsing = making explicit structure that is inherent (implicit) in natural language strings

- What is that structure?
- Why would we need it?

Implicit structure

- What do these sentences have in common?
  - Kim gave the book to Sandy.
  - Kim gave Sandy the book.
  - The book was given to Sandy by Kim.
  - This is the book that Kim gave to Sandy.
  - Which book did Kim give to Sandy?
  - Kim will be expected to continue to try to give the book to Sandy.
  - This book everyone agrees Pat thinks Kim gave to Sandy.
  - This book is difficult for Kim to give to Sandy.

Implicit structure: Constituent structure & Dependency structure

- Kim gave the book to Sandy.
  - (S (NP Kim) (VP (V gave) (NP (D the) (N book)) (PP (P to) (NP Sandy))))
  - subj(gave, Kim)
  - dobj(gave, book)
  - iobj(gave, to)
  - dobj(to, Sandy)
  - spec(book, the)

What is parsing good for?

Total Results: 0

Why do we need it?

- When is constituent structure useful?
- When is dependency structure (or semantic structure) useful?

Why do we need it?

- When is constituent structure useful?
  - Structured language models (ASR, MT)
  - Translation models (MT)
  - Generation
  - TTS: assigning intonation information
- When is dependency structure (or semantic structure) useful?
  - Information extraction (... QA, machine reading)
  - Dialogue systems
  - Sentiment analysis
  - Transfer-based MT

CFG

- Context-Free Grammars generate Context-Free Languages
- CF languages fit into the Chomsky hierarchy between regular languages and context-sensitive languages
  - All regular languages are also context free languages
  - All sets of strings describable by FSAs can be described by a CFG
    - But not vice versa
    - Case in point: $a^n b^n$

$$\mathrm{S} \rightarrow a\ \mathrm{S}\ b$$
$$\mathrm{S} \rightarrow \epsilon$$

CFGs

- Represent *constituent structure*
    - Equivalence classes: Wherever *it* can appear, so can *the lazy brown dog that the quick red fox jumped over*
    - Structural ambiguity: *I saw the astronomer with the telescope*
- Encode a sharp notion of grammaticality

CFGs, formally

- A CFG is a 4-tuple: < C, Σ, P, S >:
  - C is the set of *categories* (aka *non-terminals*, e.g., { S, NP, VP, V, ...})
  - Σ is the *vocabulary* (aka *terminals*, e.g., { Kim, snow, adores, ... })
  - P is the set of *rewrite rules*, of the form: α -> β1, β2, ..., βn
  - S (in C) is the *start-symbol*
  - For each rule α -> β1, β2, ..., βn in P, α is drawn from C and each β is drawn from C or Σ

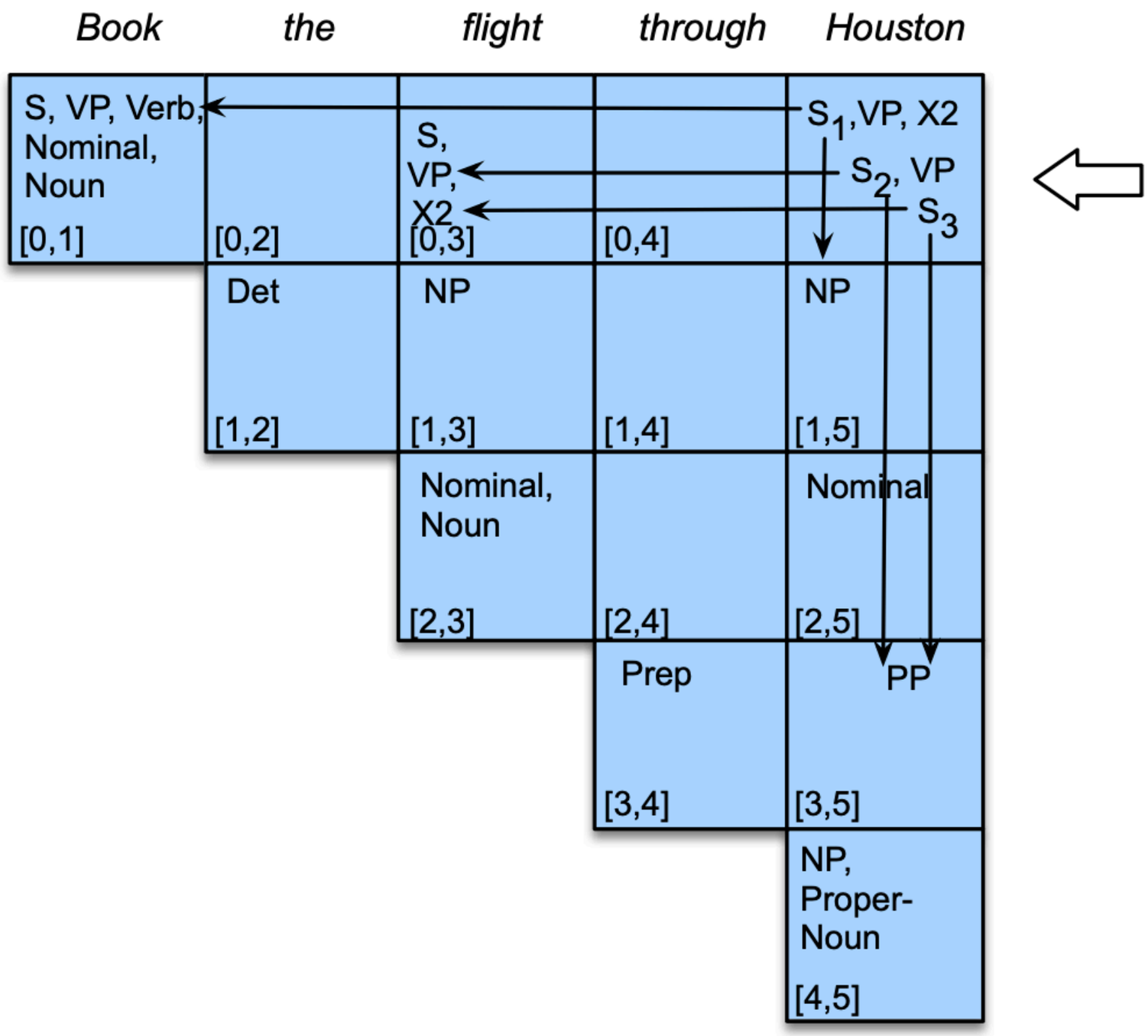
Parsing

- Given a CFG and a sentence, whether the CFG accepts it, with what and how many structures, is a mathematical fact

- Given a CFG and a sentence, determining whether the CFG accepts it, with what and how many structures, is a *search problem*

- Parsing algorithms can be:
  - Top-down or bottom-up
  - Breadth-first or depth-first
  - “Best”-first or exhaustive

CKY


Book
the
flight
through
Houston
S, VP, Verb, Nominal, Noun
[0,1]
[0,2]
[0,3]
[0,4]
S1, VP, X2
S2, VP
S3
S, VP, X2
Det
[1,2]
NP
[1,3]
[1,4]
NP
[1,5]
Nominal, Noun
[2,3]
[2,4]
Nominal
[2,5]
Prep
[3,4]
PP
[3,5]
NP, Proper-Noun
[4,5]

CKY

```
function CKY-PARSE(words, grammar) returns table

  for j ← from 1 to LENGTH(words) do
      for all {A | A → words[j] ∈ grammar}
              table[j − 1, j] ← table[j − 1, j] ∪ A
      for i ← from j − 2 downto 0 do
          for k ← i + 1 to j − 1 do
              for all {A | A → BC ∈ grammar and B ∈ table[i, k] and C ∈ table[k, j]}
                      table[i,j] ← table[i,j] ∪ A
```

Figure 13.5 The CKY algorithm.

CKY Parsing

- In which cell of the chart does one find the spanning edge(s)?
- Is the CKY algorithm top-down or bottom-up?
- Best-first or exhaustive?
- How does it handle ambiguity?
- How does it avoid inefficient re-parsing of subtrees?
- How would this algorithm return the trees?

If time: LKB demo

Why statistical parsing?

- Parsing = making explicit structure that is inherent (implicit) in natural language strings

- Useful for: language modeling + any app that needs access to the meaning of sentences

- Most application scenarios that use parser output want just one parse
    - Have to choose among all the possible analyses

- Most application scenarios need robust parsers
    - Need some output for every input, even if its not grammatical

CFGs, formally

- A CFG is a 4-tuple: < C, Σ, P, S >:
  - C is the set of *categories* (aka *non-terminals*, e.g., { S, NP, VP, V, ...})
  - Σ is the *vocabulary* (aka *terminals*, e.g., { Kim, snow, adores, ... })
  - P is the set of *rewrite rules*, of the form: α -> β1, β2, ..., βn each with a probability p of β1, β2, ..., βn given α
  - S (in C) is the *start-symbol*
  - For each rule α -> β1, β2, ..., βn in P, α is drawn from C and each β is drawn from C or Σ

PCFGs

- How does this differ from CFG?
- How do we use it to calculate the probability of a parse?
- The probability of a sentence?
- What assumptions does that require?

PCFGs

- How does this differ from CFG? -- added probability to each rule
- How do we use it to calculate the probability of a parse? -- multiply probability of each rule used (= P(T|S) = P(T))
- The probability of a sentence? -- sum of probability of all trees
- What assumptions does that require? -- expansion of a node does not depend on the context

PCFGs: Why

- When would you want to know the probability of a parse?
- When would you want to know the probability of a sentence?

How to estimate the rule probabilities

- Get a Treebank
- Gather all instances of each non-terminal
- For each expansion of the non-terminal (= rule), count how many times it occurs

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

Using the probabilities for best-first parsing

- Probabilistic CKY: in each cell, store just the most probable edge for each non-terminal

- Probabilities based on rule probability and daughter edge probabilities

Probabilistic-CKY

```
function PROBABILISTIC-CKY(words,grammar) returns most probable parse
                                                and its probability
  for j ← from 1 to LENGTH(words) do
    for all { A | A → words[j] ∈ grammar}
      table[j − 1, j,A] ← P(A → words[j])
    for i ← from j − 2 downto 0 do
      for k ← i + 1 to j − 1 do
        for all { A | A → BC ∈ grammar,
                    and table[i,k,B] > 0 and table[k, j,C] > 0 }
          if (table[i,j,A] < P(A → BC) × table[i,k,B] × table[k,j,C]) then
            table[i,j,A] ← P(A → BC) × table[i,k,B] × table[k,j,C]
            back[i,j,A] ← {k,B,C}
  return BUILD_TREE(back[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```

Figure 14.3 The probabilistic CKY algorithm for finding the maximum probability parse of a string of *num_words* words given a PCFG grammar with *num_rules* rules in Chomsky normal form. *back* is an array of backpointers used to recover the best parse. The *build_tree* function is left as an exercise to the reader.

Evaluating parsing

- How would you do extrinsic evaluation of a parsing system?
- How would you do intrinsic evaluation?
    - Gold standard data?
    - Metrics?

Gold-standard data

- There's no ground truth in trees
- Semantic dependencies might be easier to get cross-framework agreement on, but even there it's non-trivial
- The Penn Treebank (Marcus et al 1993) was originally conceived of as a target for cross-framework parser evaluation
- For project-internal/regression testing, grammar-based treebanking is effective for creating (g)old-standard data

Parseval measures

- Labeled precision:

$$\frac{\text{\# of correct constituents in candidate parse}}{\text{total \# of constituents in candidate parse}}$$

- Labeled recall:

$$\frac{\text{\# of correct constituents in candidate parse}}{\text{total \# of constituents in gold standard parse}}$$

  - Constituents defined by starting point, ending point, and non-terminal symbol of spanning node

- Cross brackets: average number of constituents where the phrase boundaries of the gold standard and the candidate parse overlap

  - Example overlap: ((A B) C) v. (A (B C))

Reading questions

- What is the difference between a recognizer and a parser?

- What is the time complexity of cky parsing, and how does it compare to other algorithms is it the best one to use?

Reading questions

- Can you have as many layers in CKY rules as you want? The textbook shows rules going two layers deep (A -> X1 Y, X1 -> BC), but, for example, could you add an X2 into the X1, and then an X3 into the X2, and ect.?

- I was so happy that people decided on a normal form of the syntax tree as it is easier to process by computer and easier to understand by human readers. I was almost screaming in my heart when I read it. Is there any more standardization of the input and the output format? I know the named entity recognition has that as I am working on that for final project, but is there more?

Reading questions

- Can these parsers and tree generators be expanded to include higher syntactic theories like X-bar theory or movement? Would that even be useful? I'm often frustrated in my syntax classes by things like X-bar theory which seem very flawed in their modelling of natural language, so would it be necessary or advantageous at all for computational linguists to include X-bar theory in their models?

- Do parsing trees implement theories in generative grammar such as X’ theory and transformations such as movement, for example? Or is it only concerned with basic levels of constituency? Is the two-branch method the main part of Chomskian syntactic theory that is implemented in CFG?

Reading questions

- If 'parameters' in syntactic theory are a framework for comparing different languages' grammars, are parameters considered when building parsing trees to handle different languages?

- When used with languages that have more complex or flexible word orderings, are we still able to generate a comprehensive CNF to use with this method? How well do CFGs perform in this context compared to other parsing techniques?

Reading questions

- Can PCFG's be trained to disambiguate sentences based on the context? Going back to the guy in the pajamas vs. the elephant in the pajamas example, could a parser be trained to have the elephant in the pajamas sentence be more probable than the guy in the pajamas sentence, maybe in the context of a piece of media or something where animals wear clothes?

- To what extent does a PCFG's inability to incorporate semantic information impact its usefulness? Is this mostly only relevant to the systems ability to parse sentences with ambiguous meanings?

Reading questions

- Is it possible to improve the reliability of PCFGs and mitigate their issues as probability estimators?

- Would modifying a word's embeddings possibly be able to help with PCFG's problems with lexical dependency?

Reading questions

- Are treebanks always hand-corrected by humans or can computers generate treebanks themselves entirely based on prior info about a given language? If always hand-corrected, how much treebank data is really out there for people to use given that it would take a long time to make?

- Around how much treebank data is out there, and what kinds/genres of text tend to be used to create treebanks? Does the genre of text of training treebank data matching the genre of new text significantly affect the quality of new parses by CKY algorithms?

Reading questions

- How does a well-developed parser deal with transcript of speaking language? In many languages, inverted order can occur often in coloquial situations while it is almost impossible to find in formal writtings. How will the parser do with these?

- Are there a significant drawbacks to modeling a natural language with a formal language or does it, for the most part, work very well?

NLP/compling in the news

- https://www.cureus.com/articles/154683-chatgpt-in-dentistry-a-comprehensive-review?score_article=true#!/

- https://www.bloomberg.com/news/articles/2023-05-01/ai-chatbots-have-been-used-to-create-dozens-of-news-content-farms