CS 679: Natural Language Processing

Lecture #21: PCFG Transformations

Thanks to Dan Klein of UC Berkeley for many of the materials used in this lecture.

Parsing Outline

1. Introduction to parsing natural language: Probabilistic Context-Free Grammars (PCFGs)
2. Independence Assumptions
3. Parsing Algorithm: Probabilistic CKY
4. PCFG Transformations
5. Markov grammars and other generative models
6. (Extra) Agenda-Based Parsing
7. Dependency Parsing

Quick quiz

- Is PCKY greedy?
- What is the running time of PCKY?
- What is the space requirement for PCKY?

Objectives

- Explore how to improve a Treebank PCFG by changing the independence assumptions while remaining in the PCFG formalism
- Introduce binarization and Markovization; evaluate!
- Further guidance for Project #4

Independence Assumptions

- How can we overcome the inappropriately strong independence assumptions in a Treebank PCFG?
- Constraint: stay in the PCFG framework!
- We can relax independence assumptions by encoding dependencies into the non-terminal symbols.

Encode Dependencies in Non-terminals

- Re-annotate the Treebank using information already in the Treebank!
- What are the most useful features to encode?
- Annotations split the non-terminals into sub-categories.
- Conditioning on history: \( P(NP^S \rightarrow PRP) \)
  - \( P(NP^S \rightarrow PRP) \)
  - \( P(PR \rightarrow PRP | S) \)
  - \( P(PR | \text{Grandparents} = \text{Parent} = NP) \)
- Should make you think “Markov order 2”
- Packing two labels into one state! – the same optimization we made in the tagger

[Johnson 98]
Another Example: Marking Possessive NPs

- Percolating important info. upwards: P(NP-POS → NNP POS)
  - Isn’t history conditioning
  - P(NP-POS → NNP POS) = P(NNP POS | NP-POS)
  - Making information from inside available to influence outside by pretending it came from outside.

- Feature grammars vs. annotation
  - Can think of a symbol like N^NP-POS as NP[parent:NP, +POS]
  - Many grammar formalisms do this: GPSG, HPSG, LFG, ...

Evaluation: Vertical Markovization

- Vertical Markov order: rewrites depend on past \( k \) ancestor nodes.
  (generalizes parent annotation)

Binarization of N-ary Rules

- Often we want to write non-binary grammar rules such as:
  \[ \text{VP} \rightarrow \text{VBD} \text{ NP PP PP} \]
  - We also observe these sorts of constituents in the Treebank.
  - But PDQY requires Chomsky Normal Form
  - We can still work with these rules by introducing new intermediate symbols into our grammar:

Binarization

- Vertical Markov Order
  - Order 1
  - Order 2

Horizontal Markovization

- "Merged states": in other words, collapses distinctions
Generalization

Binarized

H.M. Order 1

Which of these grammars can parse the following?

Evaluation: Horizontal Markovization

Order 1

Order \( \infty \)

Evaluation: Vertical and Horizontal

Examples:
- Raw treebank: \( v=1, h=1 \)
- Johnson 98: \( v=2, h=1 \)
- Collins 99: \( v=2, h=2 \)
- K&M Starting Pt.: \( v=3, h=2v \)
- Best F1: \( v=3, h=2v \)

Note: Evaluation

- After parsing with an annotated grammar, the annotations are then stripped for evaluation against the Treebank.
Train / Learn

Legend:
- You implement
- You invoke provided helper classes

Parse / Test

Legend:
- You implement
- You invoke provided helper classes

Plan B: Parse Tree Fitting

What if you run PCKY and don’t find a valid parse (nothing is in the top-left corner)?

Make something up to ensure you have a valid (binary) tree. That way you’ll get partial credit for the pieces you do have right. ?A and ?B represent non-terminals that you’ll need to specify.

Next

- Leaving the PCFG framework:
  - Markov Grammars