CS 679:
Natural Language Processing

Lecture #19: Independence Assumptions & CKY

Thanks to Dan Klein of UC Berkeley, Noah Smith of CMU, and Andrew Rosenberg of Columbia for many of the slides in this presentation.

Objectives

- Understand basic models of syntactic structure and their independence assumptions
- Understand the CKY parsing algorithm

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

Independence Assumptions in our (Generative) Models

PCFGs and Independence

- The rules and symbols in a PCFG define independence assumptions:
  - $S \rightarrow NP \ VP$
  - $NP \rightarrow DT \ NN$
- At any node, the material inside that node is conditionally independent of the material outside that node, given the label of that node.
- Any information that connects behavior inside and outside a node must flow through that node.
  - i.e., must be part of the value of the mediating node

Independence of inside and outside

$$G \rightarrow C \ L \ U \ | \ p \ ?$$

$$P(c|p,u) = \frac{P(c,p,u)}{P(p,u)}$$

$$= \frac{\sum_{u'} P(c,p,u')}{\sum_{u'} P(p,u')}$$

$$= \frac{P(c)P(g_1|g_2)P(g_2|g_3)P(g_3|g_4)P(g_4|g_5)}{P(p)P(g_1|g_2)P(g_2|g_3)P(g_3|g_4)P(g_4|g_5)}$$

$$= \frac{P(c)}{P(p)}$$

$$\therefore \ C \ L \ U \ | \ p$$
Non-Independence I

- PCFG independence assumptions are often too strong.
- Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects).

Non-Independence II

- Who cares?
  - NB, HMMs, all make false assumptions!
  - For generation, what would the consequences be?
  - For parsing, does it impact accuracy?
- Symptoms of overly strong assumptions:
  - Rules get used where they don’t belong.
  - Rules get used too often or too rarely.

Fangorn

- An open source tool for querying very large treebanks, built on top of Apache Lucene.
- Implements the LPath linguistic path language, which has an XPath-like syntax along with linguistically motivated extensions.
- Result trees are annotated with the query in order to show how the query matched the tree, and these annotations can themselves be modified and submitted as further queries.
- Demonstration site: http://nltk.ldc.upenn.edu:9090/
- Query language tutorial: https://code.google.com/p/fangorn/wiki/Query_Language
- Source code: http://code.google.com/p/fangorn/
- Steven Bird and Sumukh Ghodke

Quiz

1. (Y/N) In this tree, is the grandparent conditionally independent of the children, given the parent?
2. (Y/N) How about in this one?

General Problems

- Given a PCFG G
- For any given sentence, you might want to:
  1. Find the best parse (according to G)
  2. Find a bunch of reasonable parses
  3. Find the total probability of a sentence
- For problem #1:
  - The PCKY algorithm (today)
  - Agenda-based search
  - Beam search
- For problem #2:
  - N-best modification of PCKY (or the others)
- For problem #3:
  - Inside algorithm

Convention

- Sentence: \( w = (w_1, w_2, ..., w_n) \)
- Positions:
  - Word \( w_i \) occupies position \( i \)
  - To cover \( (w_1, w_{i+1}, ..., w_k) \)
    - We span positions \( i \) through \( k \)
    - We build “edges” or constituents \( X(\mathbf{I}, \mathbf{\bar{X}}) \)
Alternate Convention (not ours)

- Sentence: $w = (w_1, w_2, ..., w_n)$
- Positions:
  - Word $w_i$ spans positions $i-1$ through $i$
  - To cover $(w_k, w_{k+1}, ..., w_n)$
    - We span positions $i-1$ through $k$
    - We build “edges” or constituents $X(i-1,k)$

CKY Parser

- Assuming:
  - You’ve got a lot of memory
  - You’re willing to do exhaustive parsing
  - Your grammar is in CNF
- There’s an easy solution: CKY parsing
  - C = Cocke
  - K = Kasami
  - Y = Younger
  - Some call it CYK

CKY Parser: Visually

CKY Parser: Declaratively

- Input:
  - CFG $G = (\Sigma, \Gamma, S, R)$ in CNF
  - $w \in \Sigma^*$/|w| = n
- Start State(s): $C(X, i, j)$ for each $X \rightarrow w_i$
- Goal State: $C(S, 1, n)$
- C is a 3-D data structure called “the chart”
- Next State Rules:
  - $C(Y, i, j)$ and $C(Z, j + 1, k)$ together yield $C(X, i, k) \# X \rightarrow Y Z$
  - Note: keep back-pointers: $\pi(X, i, k) = (Y, Z, j)$
- Output: parse tree(s) for $w$, if they exist: $C(S, 1, n)$

To be continued ...