CS 378 Lecture 12
Constituency Parsing

Announcements

- Midterm: Weds-Sat of next week
- A3 due Tues
- A2 back soon

Recap

Viterbi: dynamic programming alg to compute $\max_y P(y, x)$ for an HMM
(similar alg today for parsing: CKY)

Beam search: approximates Viterbi by only preserving $K$ states at a timestep
Today Constituency intro
Probabilistic context-free grammars
CKY algorithm

Constituency see slides

Context-Free grammars

\[
\{ \text{N, T, S, R} \}
\]

nonterminals

S, NP, VP, PP

DT, NN, VPD, IN, NNS

terminals

\[
\text{words} \quad \downarrow \quad \text{rules} \quad \downarrow \quad \text{start symbol}
\]

\[
\text{the children ate cake with spoon}
\]

preterminals
binary

\[
S \rightarrow NP \ VP \ 1.0
\]

\[
VP \rightarrow VBD \ NP \ 1.0
\]

\[
NP \rightarrow DT \ NN \ 0.5
\]

\[
NP \rightarrow DT \ NNS \ 0.5
\]

unary

\[
DT \rightarrow the \ 1.0
\]

\[
nns \rightarrow children \ 1.0
\]

\[
NN \rightarrow cake \ 0.5
\]

\[
NN \rightarrow spoon \ 0.5
\]

\[
VBD \rightarrow ate \ 1.0
\]

CFG defines a set of trees

\[
P(tree) = \frac{1}{8}
\]

\[
\text{the cake ate}
\]

\[
\text{the children}
\]
Probabilistic CFGs

Each rule has a prob.
Probs normalize per parent

\[ P(\text{rule} \mid \text{NP}) = \left\{ \begin{array}{ll}
\text{NP} \rightarrow \text{DT} \text{NN} & 0.5 \\
\text{NP} \rightarrow \text{DT} \text{NNS} & 0.5 
\end{array} \right. \]

(\sim \text{HMM transitions})

\[ P(T) = \prod_{\text{rules}} P(\text{rule} \mid \text{parent} \ (\text{rule})) \]

Building a parser

Input: tree bank (sent w/ labeled trees)
Output: model + way to compute \( \max P(T \mid x) \)

① Grammar preprocessing (binarization) next time
② Computing grammar + getting probs
③ Parsing
1. Make every tree binary
2. Read off grammar + count rules

Treebank:

```
    NP     NP
    /\     /\  
   DT NN  DT NN
  'the' 'cake' 'the' 'dog' 'dogs'
```

This is the maximum likelihood estimate of params for this data:

\[
P(w|NN) = \begin{cases} 1/2 & \text{cake} \\ 1/2 & \text{dog} \end{cases}
\]

\[
P(w|NNS) = \begin{cases} 1.0 & \text{dogs} \end{cases}
\]

\[
P(r|NP) = \begin{cases} 2/3 & \text{DT NN} \\ 1/3 & \text{NNS} \end{cases}
\]
CKY algorithm

Input: PCFG sentence \( \overline{x} \)

Output: \( \arg \max \; P(T|\overline{x}) \)

most likely tree for \( \overline{x} \)

\[
\arg \max \; P(T|\overline{x})
\]

dynamic program: track the best score for each nonterminal over each span of the sentence

\[
T(i, j, x) = \text{score (\( \sim \log \text{prob} \)) of the best way to build symbol } x \text{ over the span } (i, j)
\]

sent len + sent len + number of symbols

\[\text{sent indices} \]
the Child raises it

Grammar:

DT → the
NN → child
NNS → raises
VBE → raises
PRP → it

S → NP VP
NP → DT NN
NP → NN NNS
VP → VBE PRP

\[ \log \left( \frac{1}{2} \right) = -1 \]
CKY

**base:** \( T(i_{i+1}, X) = \log P(w_1 | X) \)

**recursive:**

\[
T(i, j, X) = \max_{k: i < k < j} \max_{X \rightarrow X_1 X_2} \left[ \log P(X \rightarrow X_1 X_2) + T[i, j, X_1] + T[k, j, X_2] \right]
\]

\[\wedge \wedge \wedge \text{cell} (0, 3)\]
\[(0, 1) + (1, 3)\]
\[(0, 2) + (2, 3)\]