PCFGs: Parsing & Evaluation

Deep Processing Techniques for NLP
Ling 571
January 23, 2017

Roadmap

- PCFGs:
  - Review: Definitions and Disambiguation
  - PCKY parsing
    - Algorithm and Example
  - Evaluation
    - Methods & Issues
  - Issues with PCFGs

PCFGs

- Probabilistic Context-free Grammars
  - Augmentation of CFGs

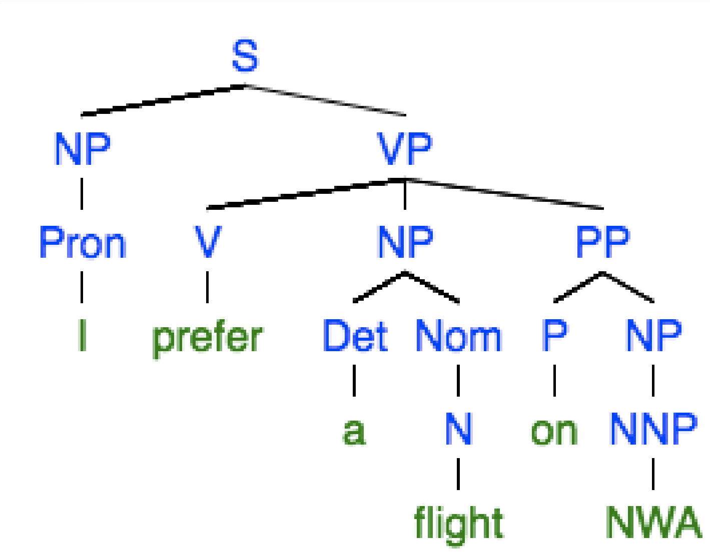
N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$ $\boxed{[p]}$, where A is a non-terminal, β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$, and p is a number between 0 and 1 expressing $P(\beta \mid A)$
S	a designated **start symbol**

Disambiguation

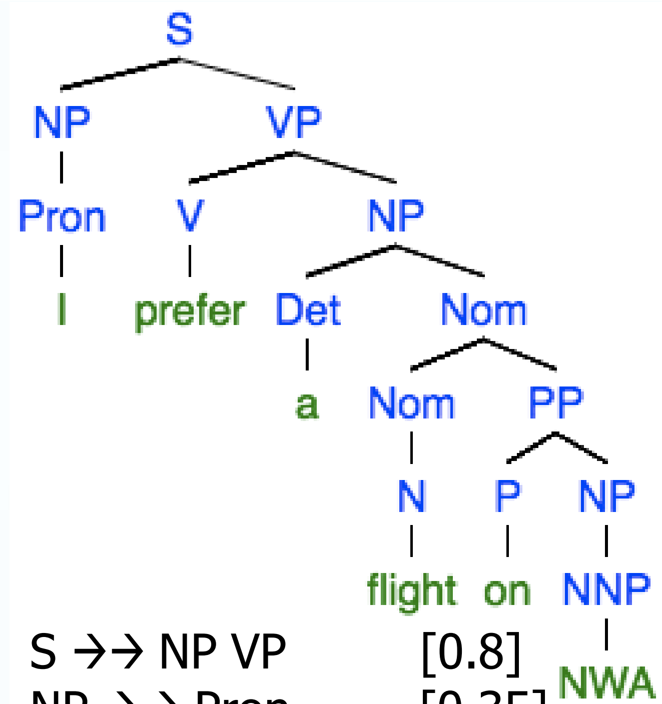
- A PCFG assigns probability to each parse tree T for input S.
  - Probability of T: product of all rules to derive T

$$P(T,S) = \prod_{i=1}^{n} P(RHS_i \mid LHS_i)$$

$$P(T,S) = P(T)P(S \mid T) = P(T)$$



S →→ NP VP	[0.8]
NP →→ Pron	[0.35]
Pron →→ I	[0.4]
VP →→ V NP PP	[0.1]
V →→ prefer	[0.4]
NP →→ Det Nom	[0.2]
Det →→ a	[0.3]
Nom →→ N	[0.75]
N →→ flight	[0.3]
PP →→ P NP	[1.0]
P →→ on	[0.2]
NP →→ NNP	[0.3]
NNP →→ NWA	[0.4]



S →→ NP VP	[0.8]
NP →→ Pron	[0.35]
Pron →→ I	[0.4]
VP →→ V NP	[0.2]
V →→ prefer	[0.4]
NP →→ Det Nom	[0.2]
Det →→ a	[0.3]
Nom →→ Nom PP	[0.05]
Nom →→ N	[0.75]
N →→ flight	[0.3]
PP →→ P NP	[1.0]
P →→ on	[0.2]
NP →→ NNP	[0.3]
NNP →→ NWA	[0.4]

Parsing Problem for PCFGs

- Select T such that:

$$\hat{T}(S) = \underset{T s.t, S = yield(T)}{\operatorname{argmax}} P(T)$$

  - String of words S is *yield* of parse tree over S
  - Select tree that maximizes probability of parse

- Extend existing algorithms: e.g., CKY
  - Most modern PCFG parsers based on CKY
    - Augmented with probabilities

Probabilistic CKY

- Like regular CKY
  - Assume grammar in Chomsky Normal Form (CNF)
    - Productions:
      - A →→ B C or A →→ w
  - Represent input with indices b/t words
    - E.g., $_0$ Book $_1$ that $_2$ flight $_3$ through $_4$ Houston $_5$

- For input string length *n* and non-terminals *V*
  - Cell[i,j,A] in (n+1)x(n+1)xV matrix contains
    - Probability that constituent A spans [i,j]

Probabilistic CKY Algorithm

tion PROBABILISTIC-CKY(wor:ds,g am,n ar) **re turns** rn ost probable paise
and its probability

· *j* +- **from 1 to** LE GTH(*words*) **do**

for all { *A* I *A* ----+ *words*[J] E *grammar*}

table [*j* - 1, *j* *A*]+-- *P*(*A* --+ *words*[}])

for *i* +- **from** *j* - 2 **do,vuto** O **do**

for *k* +-- *i* +1 to *j* - 1 **do**

for al.I { *A* I *A* -----+ *BC* E *grarnm,ar*,

and *table*[i k B] > 0 **and** *table*[k, *j* C] > 0 }

if (*table* [,iJ,A] < *P*(*A* -----+ *BC*) x *tab le*[*i*,*k* ,*B*] x *table*[*kJ* C]) **then**

table[ij;I] +-*P*(*A* ----+ B*C*) x *table* [*i*,*k*,*B*] x *table* [*k*,*j*,*C*]

back [*iJ*,*A*] +-- { *k* *B* C}

return BUILD_TREE(ba ck[1 LE *GTH*(*words*), S']), *table*[l LE *GTH*(*words*), S']

PCKY Grammar Segment

- S →→ NP VP [0.80]
- NP →→ Det N [0.30]
- VP →→ V NP [0.20]
- V →→ includes [0.05]
- Det →→ the [0.40]
- Det →→ a [0.40]
- N →→ meal [0.01]
- N →→ flight [0.02]

PCKY Matrix:
The flight includes a meal

Det: 0.4 **[0,1]**	**NP: 0.3*0.4*0.02 =.0024** **[0,2]**	**[0,3]**	**[0,4]**	**S: 0.8* 0.000012* 0.0024** **[0,5]**
	N: 0.02 [1,2]	[1,3]	[1,4]	[1,5]
		V: 0.05 [2,3]	[2,4]	VP: 0.2*0.05* 0.0012=0.0 00012 [2,5]
			Det: 0.4 [3,4]	NP: 0.3*0.4*0.01 =0.0012 [3,5]
				N: 0.01 [4,5]

Learning Probabilities

- Simplest way:
  - Treebank of parsed sentences
  - To compute probability of a rule, count:
    - Number of times non-terminal is expanded
    - Number of times non-terminal is expanded by given rule

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

  - Alternative: Learn probabilities by re-estimating
    - (Later)

Probabilistic Parser Development Paradigm

- Training:
  - (Large) Set of sentences with associated parses (Treebank)
    - E.g., Wall Street Journal section of Penn Treebank, sec 2-21
      - 39,830 sentences
    - Used to estimate rule probabilities
- Development (dev):
  - (Small) Set of sentences with associated parses (WSJ, 22)
    - Used to tune/verify parser; check for overfitting, etc.
- Test:
  - (Small-med) Set of sentences w/parses (WSJ, 23)
    - 2416 sentences
  - Held out, used for final evaluation

Parser Evaluation

- Assume a 'gold standard' set of parses for test set
- How can we tell how good the parser is?
- How can we tell how good a parse is?
  - Maximally strict: identical to 'gold standard'
  - Partial credit:
    - Constituents in output match those in reference
      - Same start point, end point, non-terminal symbol

Parseval

- How can we compute parse score from constituents?
- Multiple measures:
  - Labeled recall (LR):

    $$\frac{\text{\# of correct constituents in hyp. parse}}{\text{\# of constituents in reference parse}}$$

  - Labeled precision (LP):

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

Parseval (cont'd)

- F-measure:
  - Combines precision and recall

$$F_\beta = \frac{(\beta^2+1)PR}{\beta^2(P+R)}$$

  - F1-measure: $\beta = 1$ $$F_1 = \frac{2PR}{(P+R)}$$

- Crossing-brackets:
  - # of constituents where reference parse has bracketing ((A B) C) and hyp. has (A (B C))

Precision and Recall

- Gold standard
  - (S (NP (A a)) (VP (B b) (NP (C c)) (PP (D d))))
- Hypothesis
  - (S (NP (A a)) (VP (B b) (NP (C c) (PP (D d)))))
- G: S(0,4) NP(0,1) VP (1,4) NP (2,3) PP(3,4)
- H: S(0,4) NP(0,1) VP (1,4) NP (2,4) PP(3,4)
- LP: 4/5
- LR: 4/5
- F1: 4/5

State-of-the-Art Parsing

- Parsers trained/tested on *Wall Street Journal* PTB
  - LR: 90%+;
  - LP: 90%+;
  - Crossing brackets: 1%

- Standard implementation of Parseval: **evalb**

Evaluation Issues

- Constituents?
  - Other grammar formalisms
    - LFG, Dependency structure, ..
  - Require conversion to PTB format
  - Extrinsic evaluation
    - How well does this match semantics, etc?