Computational Linguistics 1

CMSC/LING 723, LBSC 744

Kristy Hollingshead Seitz
Institute for Advanced Computer Studies
University of Maryland

Lecture 15: 20 October 2011

Agenda

- Questions, comments, concerns?
- Context-Free Grammars
  - Treebanks
  - Inducing CFGs from trees
  - Probabilistic CFGs
- Next week: parsing algorithms

Treebanks

- Treebanks are corpora in which each sentence has been paired with a parse tree
  - Hopefully the right one!
- Encodes a particular grammatical framework
- These are generally created:
  - By first parsing the collection with an automatic parser
  - And then having human annotators correct each parse as necessary
- But…
  - Detailed annotation guidelines are needed
  - Explicit instructions for dealing with particular constructions
  - Difficult, but essential, to ensure consistency
- Starting point for a data-driven approach

Penn Treebank

- Penn TreeBank is a widely used treebank
  - 1 million words from the Wall Street Journal
  - "Least-common denominator" syntactic annotation, i.e.relatively theory-neutral
- Treebanks implicitly define a grammar for the language

Penn Treebank: Example

```
( (S (‘‘ ‘‘)
    (S-TPC-2
      (NP-SBJ-1 (PRP We) )
      (VP (MD would)
        (VP (VB have)
          (S
            (NP-SBJ (-NONE- *-1) )
            (VP (TO to)
              (VP (VB wait)
                (SBAR-TMP (IN until)
                  (S
                    (NP-SBJ (PRP we) )
                    (VP (VBP have)
                      (VP (VBN collected)
                        (PP-CLR (IN on)
                          (NP (DT those)(NNS assets)))))))))))))
    (, ,) (’’ ’’)
    (NP-SBJ (PRP he) )
    (VP (VBD said)
      (S (-NONE- *T*-2) ))
    (. .) ))
```


Treebank Grammars

- Such grammars tend to be very flat
  - Recursion avoided to ease annotators burden
  - Sometimes criticized for being so flat, e.g., (NP (NN system) (NN analyst) (NN arbitration) (NN chef))
- Penn Treebank has 4500 different rules for VPs, including…
  - VP → VBD PP
  - VP → VBD PP PP
  - VP → VBD PP PP PP
  - VP → VBD PP PP PP PP

Penn WSJ Non-Terminals (NTs)

- Basic non-terminal tagset (not including pre-terminals)

ADJP	Adjective Phrase	ADVP	Adverbial Phrase	CONJP	Conjunction Phrase
FRAG	Fragment	INTJ	Interjection	LST	List marker
NAC	Not a Constituent	NP	Noun Phrase	NX	Complex NP
PP	Prepositional Phrase	PRN	Parenthetical	PRT	Particle
QP	Quantifier Phrase	RRC	Reduced Relative Clause	S	Simple Clause
SBAR	Subordinate Clause	SBARQ	Subordinate Question Clause	SINV	Inverted Clause
SQ	Inverted Question	UCP	Unlike Coordinated Phrase	VP	Verb Phrase
WHADJP	Wh-adjective Phrase	WHAVP	Wh-adverb Phrase	WHNP	Wh-noun Phrase
WHPP	Wh-prepositional Phrase	X	Unknown		

- Other "function" tags may label constituents, e.g. PP-TMP means temporal PP
- Raw treebank contains empty categories

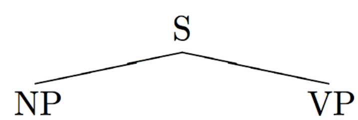
Why treebanks?

- Treebanks are critical to training statistical parsers
- Also valuable to linguist when investigating phenomena

Grammar Induction

- Extract context-free rules from trees in the treebank
- Context-free rules of the form:
  A → B C D E
  - where A is the (one and only) 'parent'
  - and B, C, D, and E are the 'children'
  - also refer to left-hand side (LHS): A and right-hand side (RHS): B C D E

CFG Induction



- Local tree: Parent (S), children (NP VP)
- Each local tree represents a context-free rule:
  S → NP VP

Interpretations of a CFG rule

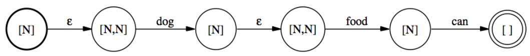
- For a rule such as S → NP VP, there are various interpretations of what this means
- Derivations:
  - An NP and a VP can combine (or compose) to produce an S
  - An S can be split into an NP followed by a VP
- Trees:
  - An S node can generate an NP and a VP node
  - An S node can be the parent of an NP and a VP node

Derivations

- If we have a rule A → α, then define a *derives* relation: $\beta A \gamma \Rightarrow \beta \alpha \gamma$.
- A string $w_1 \ldots w_n$ is in the language of a CFG *G* if S† ⇒* $w_1 \ldots w_n$
- For example, consider these noun compounding rules:
  (i) N → N N (ii) N → *dog* (iii) N → *food* (iv) N → *can*
- There are many possible derivations, s.t. N ⇒* *dog food can*
  1. N ⇒ N N ⇒ N *can* ⇒ N N *can* ⇒ N *food can* ⇒ *dog food can*
  2. N ⇒ N N ⇒ N N N ⇒ N N *can* ⇒ N *food can* ⇒ *dog food can*
  3. N ⇒ N N ⇒ N N N ⇒ *dog* N N ⇒ *dog food* N ⇒ *dog food can*
  4. N ⇒ N N ⇒ *dog* N ⇒ *dog* N N ⇒ *dog food* N ⇒ *dog food can*
  5. . . .
- Derivation 1. is the *rightmost* derivation, always expanding the rightmost non-terminal; derivation 4. is a *leftmost* derivation

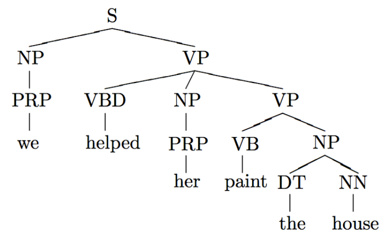
Pushdown automata

- Consider the leftmost derivation:
  N ⇒ N N ⇒ *dog* N ⇒ *dog* N N ⇒ *dog food* N ⇒ *dog food can*
- We can represent this as an automaton, with a *stack* at each state:



- Generally cannot be represented with finite-state automaton

Parse Tree, Derivation



leftmost derivation

S → NP VP
NP → PRP
PRP → we
VP → VBD NP VP
VBD → helped
NP → PRP
PRP → her
VP → VB NP
VB → paint
NP → DT NN
DT → the
NN → house

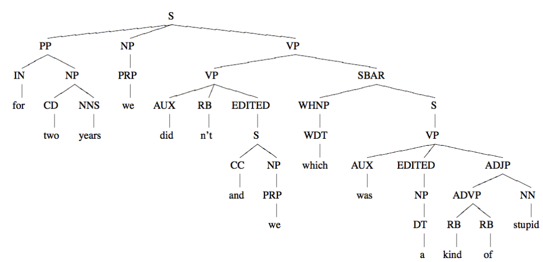
Labeled Bracketing

- Another representation of the same tree:

(S (NP (PRP we)) (VP (VBD helped) (NP (PRP her))
(VP (VB paint) (NP (DT the) (NN house)))))

- Some terminology (review):
  - Terminals are words.
  - Penn Treebank non-terminal set has 2 disjoint subsets:
    - Pre-terminal (POS) tags rewrite to exactly 1 word.
    - The rest never have terminals as children.

Parse Tree, of speech



from Switchboard Corpus

Probabilistic CFGs (PCFGs)

- A PCFG is a CFG with a probability assigned to each rule:

$$P(S \rightarrow NP\ VP) = P(rhs = (NP\ VP) \mid lhs = S) = P(NP\ VP \mid S)$$

- Joint probability of the right-hand side (RHS) can be decomposed using the chain rule:

$$P(S \rightarrow NP\ VP) = P(NP \mid S) * P(VP \mid S, NP) * P(</r> \mid S, NP\ VP)$$

where </r> is an "end-of-rule" symbol

- Standard PCFG induction approach
  - Count the number of times rules (local trees) occur
  - Use relative frequency estimation for conditional probabilities

CFG Equivalence

- Two CFGs G and G′ are *strongly* equivalent if they describe the same language, and they produce identical trees for strings, modulo node labels
- Two CFGs G and G′ are *weakly* equivalent if they describe the same language
- Sometimes a grammar G can be transformed to a weakly equivalent grammar G′ that has some beneficial computational properties

Normal Forms

- Chomsky Normal Form (CNF)
  - A grammar G = (V,T,P,S†) is in CNF if all productions in P are in one of two forms:
  - A → BC where A, B, C ∈ V or
  - A → a where A ∈ V and a ∈ T
- Greibach Normal Form (GNF)
  - A grammar G = (V,T,P,S†) is in GNF if all productions in P are of the following form:
  - A → a X where A ∈ V, a ∈ T and X ∈ V*
- Every CFG G has weakly equivalent CFGs in CNF or GNF
  - Chomsky Normal Form very useful for *chart parsing*

Grammar Factorization

- Take a rule from the grammar such as
  NP → DT JJ NN NNS
  and factor it into multiple rules
- Left factorization:
  - NP → DT NP-DT
  - NP-DT → JJ NP-DT,JJ
  - NP-DT,JJ → NN NNS
- Right factorization:
  - NP → DT-JJ-NN NNS
  - DT-JJ-NN → DT-JJ NN
  - DT-JJ → DT JJ

Penn Treebank CNF

- Disjoint pre-terminal set, so all POS → word productions already in CNF
- Left or right factorization removes productions with > 2 RHS categories
- Remaining issues:
  - Remove empty categories (0 categories on RHS)
  - Collapse unary productions (1 non-terminal on RHS)
    remove production A → B then do the following:

Productions of the form	Create new production
$C \rightarrow X\ A$	$C \rightarrow X\ A\vert B$
$C \rightarrow A\ X$	$C \rightarrow A\vert B\ X$
$C \rightarrow A\ A$	$C \rightarrow A\vert B\ A\vert B$
$B \rightarrow \alpha$	$A\vert B \rightarrow \alpha$

PCFG Induction *and* Factorization

- Original CFG rules:

$$\hat{\mathrm{P}}(A \rightarrow \alpha) = \frac{C(A \rightarrow \alpha)}{\sum_{A \rightarrow \beta \in P} C(A \rightarrow \beta)}$$

- Left factorization:

$$\hat{\mathrm{P}}(A \rightarrow B \ \ A{-}B) = \frac{\sum_{\substack{A \rightarrow B\alpha \in P \\ \alpha \in V^k \ \ k>1}} C(A \rightarrow B\alpha)}{\sum_{A \rightarrow \beta \in P} C(A \rightarrow \beta)}$$

$$\hat{\mathrm{P}}(A{-}X \rightarrow B \ \ A{-}X{-}B) = \frac{\sum_{\substack{A \rightarrow XB\alpha \in P \\ \alpha \in V^k \ \ k>1}} C(A \rightarrow XB\alpha)}{\sum_{A \rightarrow X\beta \in P} C(A \rightarrow X\beta)}$$

$$\hat{\mathrm{P}}(A{-}X \rightarrow B \ \ D) = \frac{C(A \rightarrow X \ \ B \ \ D)}{\sum_{A \rightarrow X\beta \in P} C(A \rightarrow X\beta)}$$

PCFG Induction *and* Factorization

- Right factorization

$$\hat{\mathrm{P}}(A \rightarrow X_1{-}\ldots{-}X_k \ B) = \hat{\mathrm{P}}(A \rightarrow X_1 \ldots X_k \ B)$$

$$\hat{\mathrm{P}}(X_1{-}\ldots{-}X_k \rightarrow X_1 - \ldots - X_{k-1}X_k) = 1$$

- Collapsed unary productions

$$\hat{\mathrm{P}}(C \rightarrow X \ \ A|B) = \hat{\mathrm{P}}(C \rightarrow X \ \ A) * \hat{\mathrm{P}}(A \rightarrow B)$$

$$\hat{\mathrm{P}}(C \rightarrow A|B \ \ X) = \hat{\mathrm{P}}(C \rightarrow A \ \ X) * \hat{\mathrm{P}}(A \rightarrow B)$$

$$\hat{\mathrm{P}}(A|B \rightarrow \alpha) = \hat{\mathrm{P}}(A \rightarrow B) * \hat{\mathrm{P}}(B \rightarrow \alpha)$$

Sparsity

- We may observe in our corpus the following rule:
  NP → DT JJ JJ NN NN NNS
- We may not observe:
  NP → DT JJ JJ JJ NN NN NNS
- Does this mean that the second rule should have zero probability?
- A “Markov” grammar is a factored grammar that provides probability mass to unobserved rules

Left Factorization & "Markov" Grammars

- Take a rule from the grammar such as NP → DT JJ NN NNS
- Left factorization:
  - NP → DT NP-DT
  - NP-DT → JJ NP-DT,JJ
  - NP-DT,JJ → NN NNS
- Markov grammar, order 1:
  - NP → DT NP-DT
  - NP-DT → JJ NP-JJ **"forget" that we saw a DT**
  - NP-JJ → NN NP-NN
  - NP-NN → NN

Agenda: Summary

- Questions, comments, concerns?
- Context-Free Grammars
  - Treebanks
  - Inducing CFGs from trees
  - Probabilistic CFGs
- Next week: parsing algorithms