

Ling/CSE 472: Introduction to Computational Linguistics

5/7/12

Syntax and parsing

Overview

- Why parsing?
- Target representations
- Evaluating parsing
- CFG
- Parsing algorithms
- Grammar engineering
- Treebanks

Parsing = making explicit structure that is inherent (implicit) in natural language strings

- What is that structure?
- Why would we need it?

Implicit structure

- What do these sentences have in common?
 - Kim gave the book to Sandy.
 - Kim gave Sandy the book.
 - The book was given to Sandy by Kim.
 - This is the book that Kim gave to Sandy.
 - Which book did Kim give to Sandy?
 - Kim will be expected to continue to try to give the book to Sandy.
 - This book everyone agrees Pat thinks Kim gave to Sandy.
 - This book is difficult for Kim to give to Sandy.

Implicit structure: Constituent structure & Dependency structure

- Kim gave the book to Sandy.
 - (S (NP Kim) (VP (V gave) (NP (D the) (N book)) (PP (P to) (NP Sandy))))
 - subj(gave, Kim)
 - dobj(gave, book)
 - iobj(gave, to)
 - dobj(to, Sandy)
 - spec(book, the)

Exercise: Constituent Structure & Dependency Structure

- How much wood would a woodchuck chuck if a woodchuck could chuck wood?

Sample answers

- Let's see what the ERG has to say: <http://erg.delph-in.net>

Why do we need it?

- When is constituent structure useful?
- When is dependency structure (or semantic structure) useful?

Why do we need it?

- When is constituent structure useful?
 - Structured language models (ASR, MT)
 - Translation models (MT)
 - Generation
 - TTS: assigning intonation information
- When is dependency structure (or semantic structure) useful?
 - Information extraction (... QA, machine reading)
 - Dialogue systems
 - Sentiment analysis
 - Transfer-based MT

Overview

- Why parsing?
- Target representations
- Evaluating parsing
- CFG
- Parsing algorithms
- Grammar engineering
- Treebanks

Evaluating parsing

- How would you do extrinsic evaluation of a parsing system?
- How would you do intrinsic evaluation?
 - Gold standard data?
 - Metrics?

Gold-standard data

- There's no ground truth in trees
- Semantic dependencies might be easier to get cross-framework agreement on, but even there it's non-trivial
- The Penn Treebank (Marcus et al 1993) was originally conceived of as a target for cross-framework parser evaluation
- For project-internal/regression testing, grammar-based treebanking is effective for creating (g)old-standard data

Parseval measures

- Labeled precision:

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

- Labeled recall:

$$\frac{\# \text{ of correct constituents in candidate parse}}{\text{total } \# \text{ of constituents in gold standard parse}}$$

- Constituents defined by starting point, ending point, and non-terminal symbol of spanning node
- Cross brackets: average number of constituents where the phrase boundaries of the gold standard and the candidate parse overlap
 - Example overlap: ((A B) C) v. (A (B C))

Overview

- Why parsing?
- Target representations
- Evaluating parsing
- CFG
- Parsing algorithms
- Grammar engineering
- Treebanks

CFG

- Context-Free Grammars generate Context-Free Languages
- CF languages fit into the Chomsky hierarchy between regular languages and context-sensitive languages
 - All regular languages are also context free languages
 - All sets of strings describable by FSAs can be described by a CFG
 - But not vice versa
 - Case in point: $a^n b^n$

$$S \rightarrow a S b$$

$$S \rightarrow \epsilon$$

CFGs

- Represent *constituent structure*
 - Equivalence classes: Wherever *it* can appear, so can *the lazy brown dog that the quick red fox jumped over*
 - Structural ambiguity: *I saw the astronomer with the telescope*
- Encode a sharp notion of grammaticality

CFGs, formally

- A CFG is a 4-tuple: $\langle C, \Sigma, P, S \rangle$:
 - C is the set of *categories* (aka *non-terminals*, e.g., $\{ S, NP, VP, V, \dots \}$)
 - Σ is the *vocabulary* (aka *terminals*, e.g., $\{ \text{Kim, snow, adores, } \dots \}$)
 - P is the set of *rewrite rules*, of the form: $\alpha \rightarrow \beta_1, \beta_2, \dots, \beta_n$
 - S (in C) is the *start-symbol*
 - For each rule $\alpha \rightarrow \beta_1, \beta_2, \dots, \beta_n$ in P , α is drawn from C and each β is drawn from C or Σ

CFG example

- *Book my flight. Do you know the number? He gave me the number.*
- Using the following lexicon, write rules that will generate (at least) these three sentences, and assign them plausible structures.
 - Aux = {*do*}
 - V = {*book, know, gave*}
 - N = {*flight, number, you, me, he*}
 - Det = {*my, the*}

Overview

- Why parsing?
- Target representations
- Evaluating parsing
- CFG
- Parsing algorithms
- Grammar engineering
- Treebanks

Parsing

- Given a CFG and a sentence, whether the CFG accepts it, with what and how many structures, is a mathematical fact
- Given a CFG and a sentence, determining whether the CFG accepts it, with what and how many structures, is a *search problem*
- Parsing algorithms can be:
 - Top-down or bottom-up
 - Breadth-first or depth-first
 - “Best”-first or exhaustive

The Earley Parser

- For a sentence of N words, the chart contains $N+1$ cells.
- Each cell contains a list of states.
- A state consists of: a local subtree ('edge'), information about the degree of completion of that subtree, and information about how much of the string corresponds to the subtree.
- For example (in dotted-rule notation):
 - $S \rightarrow \bullet V P, [0, 0]$
 - $NP \rightarrow Det \bullet Nominal, [1, 2]$
 - $V P \rightarrow V NP\bullet, [0, 3]$

The Earley Chart Parser, Outline

- Add the initial S ($\gamma \rightarrow \bullet S, [0,0]$) to the chart at position 0.
- Loop, for each of the rest of the cells in the chart:
 - If the state is incomplete, and the category to the right of the dot is not a POS, add (if not redundant) new states to the chart in the current position for each rule that expands that category. These new states all have the dot at the beginning of the rule, and a span that starts and ends at the current position. (PREDICTOR)

The Earley Chart Parser, Outline

- Loop (continued):
 - If the state is incomplete, and the category to the right of the dot ('B') is a POS, and B is a possible POS for the next word in the string, add the rule $B \rightarrow \text{word} \bullet$ (if not redundant) to the next cell in the chart. (SCANNER)
 - If the state is complete (dot all the way to the right), look in the cell corresponding to the beginning of the current state's span for states which are currently seeking a daughter of the same category as the mother of this state. For each one of those, add a state (if not redundant) to the current cell, with the dot moved over one, and the span increased to the end of the current word. (COMPLETER)

The Earley Chart Parser

- In which cell of the chart does one find the spanning edge(s)?
- Is the Earley algorithm top-down or bottom-up?
- Best-first or exhaustive?
- Breadth-first or depth-first?
- How does it handle ambiguity?
- How does it avoid inefficient re-parsing of subtrees?
- Returning the trees is still potentially exponential. How would this algorithm return the trees?

Other parsing algorithms

- CKY: bottom-up
- Chart parsing: more flexible determination of the order in which chart entries are processed

Overview

- Why parsing?
- Target representations
- Evaluating parsing
- CFG
- Parsing algorithms
- Grammar engineering
- Treebanks

Parsers need grammars to function

- The same parsing algorithm can handle different languages if given different grammars
- Grammars can be build by hand (grammar engineering), learned from a Treebank (supervised machine learning), or learned from raw text (unsupervised machine learning)
- How would these different kinds of grammars differ?

CFGs don't really scale

- What problems does one run into when building CFGs by hand?
- What information might we want to represent that CFG doesn't make explicit?
- Are natural languages actually context-free?

Grammar Engineering requirements

- Stable grammar formalism
- Parsing and generation algorithms
- Grammar development tools
- Regression testing system

DELPH-IN: Deep Linguistic Processing in HPSG Initiative: www.delph-in.net

- Open-source software for grammar development and deployment
- Open-source resource grammars for several languages, starting with English (erg.delph-in.net)
- Grammar Matrix starter-kit (www.delph-in.net/matrix)
- Joint reference formalism (tdl)
- Standardized semantic representations (in Minimal Recursion Semantics, Copestake et al 2005)

Treebanks

- The Penn Treebank (Marcus et al 1993): 1 million words of English text, annotated by hand with phrase structure trees
- The Redwoods Treebank (Oepen et al 2002): Smaller collection of text, annotated by selecting among analyses provided by the English Resource Grammar
 - More consistent analyses
 - More detailed analyses
 - Automatically updated to keep in sync with the grammar
 - Doesn't necessarily have complete coverage over source texts

Overview

- Why parsing?
- Target representations
- Evaluating parsing
- CFG
- Parsing algorithms
- Grammar engineering
- Treebanks

Summary questions

- How is natural language syntax and semantics different from that of programming languages?
- How does this affect the design of parsing algorithms and more generally systems?
- How does this affect the application of parsing algorithms?
- How does computational syntax differ from theoretical syntax?