Modern Approaches to Parsing: Symbolic and Statistical
Part 1: Symbolic methods

CSE 590AI, University of Washington
February 27, 2009
Emily M. Bender
Department of Linguistics
University of Washington
Background: Parsing and grammars

- Parsing: The process of assigning syntactic (and semantic) structures to input strings, according to a grammar

- Grammar: A representation of linguistic knowledge
  - Part of speech assignments for words
  - Rules for combining words into phrases
  - Rules for calculating semantic representations from syntactic structures
  - Rules for deriving inflected word forms from stems
Background: Sources of linguistic knowledge

• Grammars can be created by:
  
  • Supervised machine learning --- knowledge source is hand annotation of example sentences
  
  • Unsupervised machine learning --- knowledge source is inherent patterns in data and biases incorporated in learning algorithm
  
  • Linguists (grammar engineers) --- hand built rule systems
Background: Why symbolic parsing?

• Cons:
  • Less robust
  • More expensive, though perhaps becoming competitive with Treebanks

• Pros:
  • Richer structure: Semantic representations
  • Reversible: Parsing and generation
  • Linguistic interest
Overview

- Rule-based systems: Frameworks and examples
- Parse/realization selection
- Obstacles
- Broad-coverage grammars and maintainability
- Scaling: resource grammars, lexical acquisition, multilingual grammar engineering
Rule-base systems: Many frameworks

- Tree-Adjoining Grammar (TAG, Joshi et al 1975)
- Categorial Grammar (CG, Wood 1993)
- Lexical-Functional Grammar (LFG, Kaplan & Bresnan 1982)
- Head-driven Phrase Structure Grammar (HPSG, Pollad & Sag 1994)
- Stanford parser: Stochastic parser demo for comparison (nlp.stanford.edu)
Detailed example: HPSG

• Start with CFG, change atomic node labels to (recursive) feature structures

• Allow rules and lexical entries to be partially underspecified feature structures.

• Allow constraints identifying values of features

• Organize lexical entries, lexical rules, and phrase structure rules into a type hierarchy (cf OOP)
Example lexical entry (simplified)

\[
\langle \text{bake} , \begin{bmatrix}
\text{SYNTAX} \\
\text{SUBJ} \\
\text{COMPS}
\end{bmatrix}
\begin{bmatrix}
\text{POS} & \text{verb} \\
\{ \text{NP}:x \} \\
\{ \text{NP}:y \}
\end{bmatrix}
\begin{bmatrix}
\text{SEMANTICS} \\
\{ \text{PRED} \quad '\text{bake}' \} \\
\{ \text{ARG1} \quad x \} \\
\{ \text{ARG2} \quad y \}
\end{bmatrix}\rangle
\]
Example phrase structure rule (simplified)

\[
\begin{align*}
\text{MOTHER} & : \left[ \begin{array}{c}
\text{SYNTAX} \\
\text{SEMANTICS}
\end{array} \right] \\
\text{SYNTAX} & : \left[ \begin{array}{c}
\text{POS} \\
\text{SUBJ} \\
\text{COMPS} \langle \rangle
\end{array} \right] \\
\text{SEMANTICS} & : \left[ \begin{array}{c}
3 \\
\oplus \\
4
\end{array} \right]
\end{align*}
\]

\[
\begin{align*}
\text{DAUGHTERS} & : \left\langle \left[ \begin{array}{c}
\text{SYNTAX} \\
\text{SEMANTICS}
\end{array} \right] \right\rangle \\
\text{SYNTAX} & : \left[ \begin{array}{c}
\text{POS} \\
\text{SUBJ} \\
\text{COMPS} \langle 5 \rangle
\end{array} \right] \\
\text{SEMANTICS} & : \left[ \begin{array}{c}
3
\end{array} \right]
\end{align*}
\]

\[
\begin{align*}
\text{SEMANTICS} & : \left[ \begin{array}{c}
\left[ \begin{array}{c}
3
\end{array} \right] \oplus \left[ \begin{array}{c}
4
\end{array} \right]
\right]
\end{align*}
\]
Result

\[
\begin{align*}
\text{VP} \\
\left[ \text{SYNTAX.SUBJ} \langle \text{NP:x} \rangle \right] \\
\left[ \text{SEMANTICS} \langle \text{bake(x,y), cake(y)} \rangle \right]
\end{align*}
\]
English Resource Grammar (Flickinger 2000)

- http://erg.delph-in.net

- 17+ person years of effort

- July 2008 version:
  - 3549 types
  - 187 phrase structure rules
  - 65 lexical rules
  - 35889 lexical entries

- ~80%+ coverage on unseen text from new domains (Ytrestøl et al 2009)
Parse selection

• Natural language is ambiguous
  • “Have that report on my desk by Friday” (32 parses)

• People rarely notice the ambiguity

• Stochastic parsers achieve robustness by being underconstrained and relying on statistical models to select the best parses

• Deep grammars find less ambiguity, but still some

• Practical applications only want one analysis per input
Treebanks

- Redwoods project introduced grammar-based treebanks (Oepen et al 2002)
  - Parse test corpus with grammar, store parse forests
  - Calculate minimal discriminants between parses
  - Present these to annotator to select trees
  - Store decisions on discriminants as well as resulting tree
  - When grammar changes, reparse test corpus and rerun decisions to minimize additional annotation effort
- Result: Highly consistent treebank, kept in sync with grammar
Treebanks for parse selection

• Given set of trees returned by the grammar, select the preferred one

• Not amenable to hand-built rule systems, apply machine learning instead (here, MaxEnt with fairly shallow features of derivations)

• State of the art (Zhang et al, 2007), English translations of Norwegian hiking guides, 8000 training sentences:

<table>
<thead>
<tr>
<th></th>
<th>Exact match</th>
<th>Top 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>11.34</td>
<td>43.06</td>
</tr>
<tr>
<td>Best system</td>
<td>56.28</td>
<td>84.51</td>
</tr>
</tbody>
</table>
Obstacles

• Parsing efficiency: much less of an issue now (e.g., Oepen et al 2002)

• Robustness: when the grammar can’t find a parse, how can we leverage partial analyses? (Zhang 2007)

• Scaling to knew genres/languages: resource grammars, lexical acquisition, multilingual grammar engineering; see later slides
Maintaining a broad-coverage grammar

- Strive to capture linguistic generalizations

- Document analyses (comment code!)

- Regression testing: curate test corpora and test suites, test regularly for changes in number of parses or semantic representations

- Treebanking: Annotate parse forests for preferred parse

  - Software assistance calculates minimal discriminants and updates treebank to new grammar version (Oepen et al 2002)
Scaling

• Toy grammars are fun to build, but how do we scale them to real-world applications?

• Has been done in at least a few cases (e.g., ERG, ParGram English grammar, others)

  • “Resource grammar” approach

  • Lexical acquisition

  • Multilingual grammar engineering
Resource grammars

• Different domains feature different linguistic structures

• ... yet also reuse many “core” language structures.

• The “resource grammar” approach feature sustained development of one grammar across different application domains.

• Develop against particular target corpora, but with an eye towards linguistic generalizations (and input from the linguistics literature)

• Each new domain requires less development work than the last.
Lexical acquisition

- Deep grammars rely on detailed information for lexical entries
  - Valence (number and type of complements)
  - Semantic features (mass/count nouns, raising/control verbs)
- Treebanks can be used as training data for machine learning approaches to lexical acquisition, on-line or off-line
- Supertagging: Use lexical types from grammar as part of speech tags, and do sequence labeling (Blunsom & Baldwin 2006)
Multilingual grammar engineering

- Languages differ in: Vocabulary, phonology, morphological complexity, syntactic structures

- But this variation is not unbounded!

- Deep grammars for one language can be leveraged to speed up development of grammars for other languages

- Grammars for different languages can even share code

- Added bonus: Grammars developed in multilingual projects shared standardized output formats (e.g., systems of semantic representations) and so can be used (somewhat) interoperably in applications
Multilingual Grammar Engineering Projects

- Many projects in different frameworks:
  - ParGram (LFG; Butt et al 2002, King et al 2005)
  - MetaGrammar (TAG, LFG; Kinyon et al 2006)
  - KPML (SFG; Bateman et al 2005)
  - Grammix (HPSG; Müller 2007)
  - OpenCCG (CCG; Baldridge et al 2007)
  - MedSLT (Regulus; Santaholma 2008)
The Grammar Matrix (www.delph-in.net/matrix)

• Derive core grammar (Bender et al 2002) from ERG (Flickinger 2000), with reference to JACY grammar of Japanese (Siegel and Bender 2002)

• Emphasis on construction of semantic representations

• But: there are crosslinguistically common structures which are not universal

• Solution: A series of “libraries” implementing variations on different phenomena, accessible through a web-based questionnaire (Bender & Flickinger 2005, Drellishak forthcoming)

• Output: “Starter grammars” covering small fragments of the languages modeled, but ready for sustained development
Overview

• Rule-based systems: Frameworks and examples

• Parse/realization selection

• Obstacles

• Broad-coverage grammars and maintainability

• Scaling: resource grammars, lexical acquisition, multilingual grammar engineering
References

Baldridge, Jason, Sudipta Chatterjee, Alexis Palmer, and Ben Wing. 2007. DotCCG and VisCCG: Wiki and programming paradigms for improved grammar engineering with OpenCCG. In T. H. King and E. M. Bender (Eds.), Proceedings of the GEAF 2007 Workshop, Stanford, CA. CSLI.


