Introduction, organization
LKB formalism

Ling 567
January 4, 2010
Overview

- The BIG picture
- Goals (of grammar engineering, of this course)
- The LinGO Grammar Matrix
- Other approaches
- Course requirements/workflow
- Pick a language, (almost) any language
- Components
- LKB formalism
What is grammar engineering?

- The implementation of natural language grammars in software.
- Grammars can be used for parsing and/or generation.
  - Relate surface strings to semantic representations
- Grammars can be practically focused or theoretically focused.
- Knowledge-engineering approach to parsing.
  - “Precision” grammars can give deeper representations
  - ... but tend to be less robust.
How is grammar engineering different from other approaches to syntax?

- Implementation requires fully explicit analyses
- Implementation allows automated verification of analyses
  - Parse test suites
  - Parse test corpora
- Generate from stored semantic representations
- Implementations allows/requires incremental development
  - Interrelatedness of analyses becomes more apparent
Pen and paper syntax work-flow

- Identify phenomena to analyze
- Identify key examples
- Develop analysis
- Identify cases of interesting predictions
- Refine analysis
- Test acceptability of new key examples
Grammar engineering work flow

1. Develop initial test suite
2. Identify phenomena to analyze
3. Parse sample sentences
4. Debug implementation
5. Compile grammar
6. Parse full test suite
7. Implement analysis
8. Extend test suite with examples documenting analysis
Applications

• Language documentation

• Linguistic hypothesis testing

• MT

• IR (“semantic search” --- PowerSet)

• Automated email response

• Augmentative and assistive communication

• Computer assisted language learning (CALL)

• ...
Challenges

- efficient processing (Oepen et al 2002)
- ambiguity resolution (Toutanova et al 2005)
- domain portability
- lexical acquisition (Baldwin 2005)
- extragrammatical/ungrammatical input
- scaling to many languages
Hybrid approaches

- Naturally occurring language is noisy
  - typos
  - “mark up”
  - addresses and other non-linguistic strings
  - false starts
  - hesitations

- Allowing for noise within the grammar would reduce precision

- And then there’s ambiguity, unknown words, ...
Hybrid approaches

- Combine knowledge engineering and machine learning approaches:
  
  - Statistical parse selection
  
  - (Statistical) named-entity recognition and POS tagging in a pre-processing step (for unknown word handling)
  
  - Tiered systems with shallow parser as fallback for precision grammar
  
- Other direction:
  
  - Deep grammars providing richer linguistic resources or seed information to train machine learners
Overview

• The BIG picture

• Goals (of grammar engineering, of this course)

• The LinGO Grammar Matrix

• Other approaches

• Course requirements/workflow

• Pick a language, (almost) any language

• Components

• LKB formalism
Goals: Of Grammar Engineering

• Build useful, usable resources

• Test linguistic hypotheses

• Represent grammaticality/minimize ambiguity

• Build modular systems: maintenance, reuse
Goals: Of this course

• Mastery of tfs formalism

• Hands-on experience with grammar engineering

• A different perspective on natural language syntax

• Practice building (and debugging!) extensible system

• Contribute to on-going research in multilingual grammar engineering
Goals: Of this course

• Understand a range of grammatical facts about a language, plus how to get them from descriptive materials

• Learn more about using HPSG to model grammatical facts

• Deeper understanding of relationship between syntax and semantics

• Learn how to use the computational tools of grammar engineering to test and develop formalizations
Testing and developing formalizations

- Tools: LKB, [incr tsdb()]

- Steps:
  - Identify intended analysis (primarily semantic)
  - Hypothesize new rules/lexical entries or new constraints on existing rules/lexical entries that will produce intended analyses
  - Implement constraints (and debug until grammar compiles)
  - Test and examine results: Overconstrained? Underconstrained?
Relationship between syntax and semantics

• What does syntax do?
  • Constrain ambiguity
  • Provide scaffolding for building semantic representations
  • Handle grammaticality (agreement, word order, case, ...)

• What do semantic representations do?
  • Make explicit who did what to whom
  • Serve as input for tactical generation
  • Relate multiple surface forms to each other
  • Differentiate multiple analyses of same surface form
Overview

• The BIG picture

• Goals (of grammar engineering, of this course)

• The LinGO Grammar Matrix

• Other approaches

• Course requirements/workflow

• Pick a language, (almost) any language

• Components

• LKB formalism
The LinGO Grammar Matrix

• Addresses the scalability challenge by reducing the cost of creating grammars

• Starter-kit which allows for quick initial development while supporting long-term expansion

• Represents a set of hypotheses about cross-linguistic universals and cross-linguistic variation

• Includes typologically grounded “libraries” exploring the range of variation in certain phenomena
A sampling of hypotheses

• Words and phrases combine to make larger phrases.

• The semantics of a phrase is determined by the words in the phrase and how they are put together.

• Some rules for phrases add semantics (but some don’t).

• Most phrases have an identifiable head daughter.

• Heads determine which arguments they require and how they combine semantically with those arguments.

• Modifiers determine which kinds of heads they can modify, and how they combine semantically with those heads.

• No lexical or syntactic rule can remove semantic information.
Multilingual grammar engineering: Other approaches

• The DELPH-IN consortium specializes in large HPSG grammars

• Other broad-coverage precision grammars have been built by/in/with

  • LFG (ParGram: Butt et al 1999)

  • F/XTAG (Doran et al 1994)

  • ALE/Controll (Götz & Meurers 1997)

  • SFG (Bateman 1997)

• Proprietary formalisms and Microsoft and Boeing
Overview

• The BIG picture

• Goals (of grammar engineering, of this course)

• The LinGO Grammar Matrix

• Other approaches

• Course requirements/workflow

• Pick a language, (almost) any language

• Components

• LKB formalism
Course requirements/workflow

- Mondays lecture, Wednesdays discussion
- Office/lab hours on (most) Fridays
- Weekly lab assignments, posted one week ahead, due on Friday
- Be sure to start the lab by class on Wednesday, so you can bring useful questions
- At least half of each lab grade will be on the documentation
- Labs 2-9 as partner projects, taking turns doing the write-up
- No exams; front-loaded course schedule
- “Uncheatable”
Course requirements/workflow

• Week 1: Getting to know the LKB (English exercise); pick your language

• Weeks 2-4: Test suite construction, iteratively customize starter grammar

• Weeks 5-9: Build out your grammar

• Week 10: MT extravaganza
Surviving the course

• Communication is key: Please ask questions!
  • Get started early, to have time for collaboration and question turn-around

• Use GoPost (link on course page)
  • Subscribe to the GoPost

• Read (and contribute to!) FAQs, glossary (→ demo)

• EB’s lab hours

• 10 minute rule
Pick a language, any language

• And pick a partner. (Ideally each team should have at least one linguist.)

• Each team must pick a different language.

• Previous languages are on the wiki, only languages from 2004 are available for re-treatment.

• No English, non-Indo European preferred.

• Consider using an ascii transliteration.

• Languages with complex morphophonology require abstraction (assume a morphophonological preprocessor).

• Pick a language with a good descriptive grammar available.
Overview

• The BIG picture

• Goals (of grammar engineering, of this course)

• The LinGO Grammar Matrix

• Other approaches

• Course requirements/workflow

• Pick a language, (almost) any language

• Components

• LKB formalism
Components

- HPSG: Theoretical foundations
- LKB
- Grammar (Matrix-provided, plus extensions)
- Emacs: editor, interaction with LKB
- [incr tsdb()]
LKB

- tdl reader/compiler
- parser
- generator
- grammar exploration tools
  - parse chart
  - interactive unification
  - type and hierarchy exploration
Grammar

• A set of tdl files:
  • Grammar Matrix core
  • Additions from the customization system
  • Your additions

• Actually separated into:
  • Type definitions
  • Instances of grammar rules, lexical rules, lexical entries
  • Root symbols
  • Node label abbreviations

• Also includes: Lisp code for LKB interaction
Pronounced “tee ess dee bee plus plus”

Loading in test suites

Running test suites (batch processing)

Comparing multiple test suite runs:
  - Changes in which examples parse
  - Changes in number of analyses per item
  - Changes in representations per item
Overview

• The BIG picture

• Goals (of grammar engineering, of this course)

• The LinGO Grammar Matrix

• Other approaches

• Course requirements/workflow

• Pick a language, (almost) any language

• Components

• LKB formalism