LING 520: Computational Analysis of English

Semester: FALL '16

Instructor: Sowmya Vajjala

Iowa State University, USA

1 November 2016

Class Outline

- Exercise from Thursday
- Assignment 4 Discussion
- Overview of constituency parsing methods
- very briefly about Dependency Parsing
- Assignment 5 Description

Exercise from Thursday

Source: Chapter 8 in NLTK Book

1. Follow the groucho grammar example in Section 1.2 and simple grammar example in Section 3.1 that uses recursive descent parser.
2. After that, use the grammar in Section 3.3 instead of groucho grammar, and try to parse examples 10 (a) and (b) in the textbook with this grammar.
3. Finally: Figure out how to make Example 3.2 work on your computer, with your own custom created grammar.

- Did everyone finish this?.

Exercise from Thursday

Source: Chapter 8 in NLTK Book

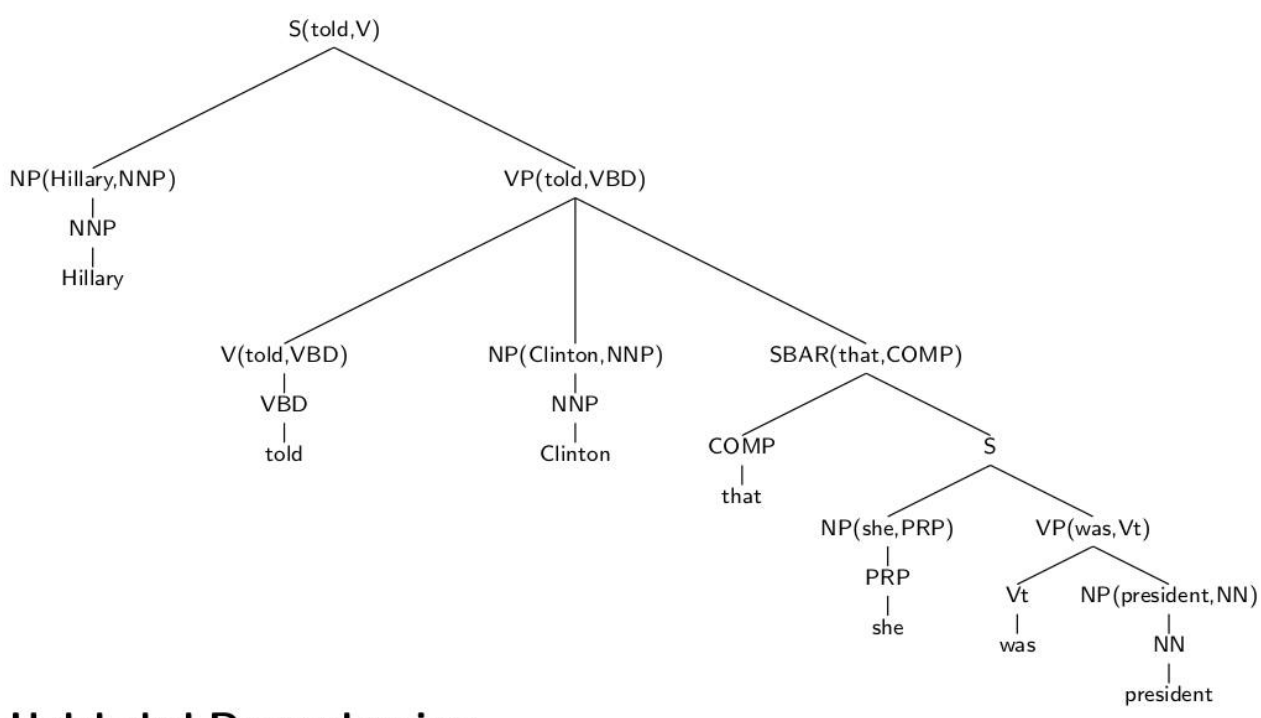
1. Follow the groucho grammar example in Section 1.2 and simple grammar example in Section 3.1 that uses recursive descent parser.
2. After that, use the grammar in Section 3.3 instead of groucho grammar, and try to parse examples 10 (a) and (b) in the textbook with this grammar.
3. Finally: Figure out how to make Example 3.2 work on your computer, with your own custom created grammar.

- Did everyone finish this?. Sonca is going to come and talk about this exercise now.

Assignment 4 Discussion

- Q1 of A3 and A4: I will have a inclass discussion about them when we discuss NLP4CALL soon.
- A4-Q2: Stephanie agreed to talk about her writeup.

Parsing Fun



Unlabeled Dependencies:

(0,2) (for root → told)
(2,1) (for told → Hillary)
(2,3) (for told → Clinton)
(2,4) (for told → that)
(4,6) (for that → was)
(6,5) (for was → she)
(6,7) (for was → president)

Source: Coursera course by Michael Collins, from 2013 or early 2014.

Constituent Parsing Methods

Constituent Parsing Methods

Broadly speaking, there are two ways of doing constituent parsing:

- Top-down parsing: Start from the root of the parse tree, and go towards the end when you reach the words of the sentence.
- Bottom-up parsing: Start from the words in the sentence, and go upwards building the parse tree.

All phrase structure parsers use one of these strategies in their parsing algorithms (or a combination of both)

Top-Down parsing

- Start with the rule that has the Sentence (S) or ROOT (depending on how your grammar is written) as parent.
- Look at all grammar rules that has a S on LHS. Mark all of them as a possibility.
- Explore each rule, recursively keep going further and further down until you see a leaf node.
- If the rule appears incompatible anywhere, backtrack to previous step and keep doing this until you reach the end of a sentence.
- Top-down parsing uses grammar to predict the input sentence structure, but without inspecting the input first!!

Recursive Descent parsing

- Recursive Descent parser is a form of top-down parser.
- nltk.app.rdparser() has a good demo of how this works.

Recursive Descent parsing

- Recursive Descent parser is a form of top-down parser.
- nltk.app.rdparser() has a good demo of how this works.
- Advantage: Finds all possible correct parses.
- Disadvantage: This kind of approach to parsing has 3 major short comings.
  1. Left-recursive rules (e.g., NP $->$ NP PP) will make this parser fall in an infinite loop.
     (Example: Edit the parser app to add this rule and see what happens).
  2. This parser wastes a lot of processor resources trying to explore paths that are unrelated to the sentence being parsed.
  3. While back-tracking, it starts rebuilding all discarded constituents again.
- One alternative: Do Bottom-up Parsing

Bottom-up parsing

- Idea: Start from the sentence, build the tree bottom up, finally reaching the root node.
- Example: Shift-Reduce parser in NLTK - nltk.app.srparser()

Bottom-up parsing

- Idea: Start from the sentence, build the tree bottom up, finally reaching the root node.
- Example: Shift-Reduce parser in NLTK - nltk.app.srparser()
- Advantage: This works only with rules that match actual words in the sentence. So, does not explore irrelevant options.
- Disadvantage: May not find a right parse even if there is one.
- What to do?: Combine both approaches (LeftCornerParser in NLTK)

Dynamic Programming for Parsing

- For ambiguous, and long sentences, both top-down and bottom-up approaches become very inefficient because of the number of possible parse paths to explore
- Dynamic programming helps solve this problem by storing all partial parses generated during the parsing process in a "chart".
- This avoids the re-parsing problem of seen parses, and the partially solves ambiguity issues as well.
- Three commonly used parsers of this kind: chart parser, earley parser, CKY parser
- Note: Chart parsers can be top-down or bottom-up.

Dynamic Programming for Parsing

- For ambiguous, and long sentences, both top-down and bottom-up approaches become very inefficient because of the number of possible parse paths to explore
- Dynamic programming helps solve this problem by storing all partial parses generated during the parsing process in a "chart".
- This avoids the re-parsing problem of seen parses, and the partially solves ambiguity issues as well.
- Three commonly used parsers of this kind: chart parser, earley parser, CKY parser
- Note: Chart parsers can be top-down or bottom-up.
- Real word parsing: Stanford parser uses an implementation of CKY (bottom-up) parsing with probabilistic grammar.
- More on the exact algorithms: Chapter 13.1–13.4 in J&M.

Constituency Parsing in NLTK

- NLTK has several parsing algorithms implemented in Python. Some Top-down, some bottom-up, some hybrid (http://www.nltk.org/howto/parse.html)

Constituency Parsing in NLTK

- NLTK has several parsing algorithms implemented in Python. Some Top-down, some bottom-up, some hybrid (http://www.nltk.org/howto/parse.html)
- If you are really curious about the differences between different parsers in NLTK, visit this: https://goo.gl/9dgGlq
- All these examples expect you to provide a grammar. You can also create a grammar out of treebank data in NLTK (Section 6 in Chapter 8 of NLTK book).
- My suggestion: NLTK has interface code written to interact with external parsers such as Stanford parser - better use these for real-world use.
- Note: Parsing is a very active are of R&D, and there are full courses focusing on Parsing algorithms alone, around the world.

Dependency Parsing

Dependency Parsing: Methods

- What you need: a set of dependency relations, lexicon of the language.
- You can again do top-down, bottom-up, a combo, use Dynamic programming etc.
- Not as much explored as constituency parsing in NLP.
- Malt parser: a state-of-the-art dependency parser that uses dynamic programming with bottom-up parsing.
- Stanford dependency parser: This is not primarily a dependency parser, it converts constituency tree into dependency relations.
- Dependency relations encode relations between words. So useful for information extraction.

Dependency Parsers in NLTK

- NLTK does not have a real dependency parser. But it has interface code to existing dependency parsers such as MALT parser and Stanford Dependency parser.
- spacy.io has support for Dependency parsing in Python. If you want, figure out its installation and use that!
- MATE parser is another popular dependency parser (works for English and German).

Dependency Parsing and CALL

- Dependency parsing is relatively tolerant to word-order changes. So, it is used by NLP-CALL researchers to analyse the syntactic structure of learner language more commonly than constituency parsing.
- Parsing learner language is an active area of research in NLP researchers who work in CALL topics.
- Will discuss briefly about this in a few weeks

Assignment 5 Description

Next Class

1. Topics:
   - Partial Parsing, incremental parsing and other such methods
   - Parsing: Conclusion
   - practice exercises with using parsers in Python
2. Readings: Chapter 8 in NLTK (Mandatory). Chapter 12–14 in J&M (Optional)
3. Video lectures (optional): Week 5 Lectures in Jurafsky and Manning's course or Weeks 4 and 5 lectures in Radev's course.

If there is time: another Exercise

Figure out how to use Stanford parser in Python (with or without NLTK). I will ask about this in Thursday's class.