Lecture 17
More constituency parsing and dependency parsing

Intro to NLP, CS585, Fall 2014
http://people.cs.umass.edu/~brenocon/inlp2014/
Brendan O’Connor
• Projects
  • No exercise this week

• Constituent parsing continued
  • Better PCFG’s
  • Whole-tree features
  • Shift-reduce parsers

• Start dependency parsing

• Thursday: examples of what you do with parsing
Modern statistical parsers

- PCFG assumptions are too strong. How to improve?
  - Transform the training data
    - splitting/“annotating” non-terminals
  - Automatically learn better splits with EM
    \((\text{Berkeley parser})\)
  - Discriminative whole-tree features -- need to use re-ranking \((\text{BLIPP parser})\)

- Inference tricks at runtime
Non-terminal splits

- Annotate a nonterminal symbol its parent/grandparent/sibling
- Relaxes PCFG independence assumptions
Latent-variable PCFG

- Automatically *learn* useful splits.
- Latent-variable PCFG: augment training data with latent states. Learn with EM. Use “split-merge” training to vary number of latent states.
  - NP_1, NP_2, NP_3....
- [Petrov (2009), used today in open-source Berkeley parser]
  - The software is pretty easy to use
Discriminative re-ranking

• This is the most accurate method known so far (for English constituent parses on WSJ PTB...)
  • BLIPP parser, open-source
• No more PCFG: Why not use a log-linear model with whole-tree features?
  • Does this NP contain 15-20 words?
  • Right-branching tendencies?
  • PP object is “telescope” and VP verb is “see” (bilexical features: sparse but very useful!)
  • Now CKY is no longer possible. Why?
• Make it fast with **re-ranking**:
  • Take top-K trees from a PCFG. (Variant of CKY can do this)
  • Extract features for each, and re-rank them.
• Re-ranking is a very powerful general technique in NLP
  • Simple, fast model generates candidates
  • Slow, more accurate model decides the best one
Shift-reduce parsing

• Totally different paradigm.
• Incrementally build up the parse tree left-to-right.
• Eat words as you go along: for each word, decide what to do with it ... add to current subtree, or pop the stack and attach to a higher one.
• Make these decisions according to a classifier.
• Think of the parser as an automaton.
• No dynamic programming! O(n) runtime!
• Potentially related to cognitive processing?

• Open-source implementations: Yue Zhang’s zpar is extremely fast, and quite accurate. (Also see recent Stanford NLP releases...)
Example from a similar incremental parser (slightly different than current work)

Ratnaparkhi (1998)

Build:
- START NP
- START VP
- START S
- ...

[Slides: Noah Smith]
Ratnaparkhi (1998)

Build:
- START NP
- START VP
- START S
- ...

[labeled diagram with tokens: NP, PN, VB, DT, NNS, PRP, NNS, PRP, NNS]

Thursday, November 6, 14

[Slides: Noah Smith]
Ratnaparkhi (1998)

Check:
- yes
- no

[Slides: Noah Smith]
Ratnaparkhi (1998)

Check:
- yes (REDUCE)
- no
Ratnaparkhi (1998)

Build:
- START NP
- START VP
- START S
- ...

I hit the cats on mats with bats

[Slides: Noah Smith]
Ratnaparkhi (1998)

Build:
- START NP
- START VP
- START S
- ...

I hit the cats on mats with bats

[Slides: Noah Smith]
Ratnaparkhi (1998)

I hit the cats on mats with bats

Check:
☐ yes
☐ no
Ratnaparkhi (1998)

Check:
- yes
- no (SHIFT)

NP → S

[Slides: Noah Smith]

Thursday, November 6, 14
Ratnaparkhi (1998)

Build:
- START NP
- START VP
- START S
- ...
- JOIN S

I hit the cats on mats with bats

[Slides: Noah Smith]
Ratnaparkhi (1998)

Build:
- START NP
- START VP
- START S
- ...
- JOIN S

I hit the cats on mats with bats
Ratnaparkhi (1998)

I hit the cats on mats with bats

Check:
- yes
- no

[Slides: Noah Smith]

Thursday, November 6, 14
Ratnaparkhi (1998)

Check:
☑ yes
☑ no (SHIFT)
Ratnaparkhi (1998)

Build:
- START NP
- START VP
- START S
- ...
- JOIN VP

I hit the cats on mats with bats
Ratnaparkhi (1998)

S

NP

VP

NP

PN

VB

DT

NNS

PRP

NNS

PRP

NNS

I

hit

the
cats

on

mats

with

bats

Check:
☑ yes
☑ no (SHIFT)

[Slides: Noah Smith]
Ratnaparkhi (1998)

Build:
- START NP
- START VP
- START S
- ...
- JOIN NP

I hit the cats on mats with bats
Ratnaparkhi (1998)

Check:
☑ yes
☑ no (SHIFT)
I hit the cats on mats with bats

Ratnaparkhi (1998)
• Shift-reduce / incremental parsers have powerful features: can look at currently built-up structure so far, and current automaton state.
• Also called a “history-based model”
• Contrast to global inference approaches. Have to choose between
  • local features + fast dynamic prog. inference (PCFG)
  • global features + slow inference (whole-tree loglinear)
How accurate are these things?

- These constit. parsers all get low 90’s% labeled span F-score
- Dependency parses get low 90’s% arc accuracy.
- Which ambiguities or errors matter for what types of tasks?
How accurate are these things?

(Numbers taken from Collins (2003))

- Subject-verb pairs: over 95% recall and precision
- Object-verb pairs: over 92% recall and precision
- Other arguments to verbs: ≈ 93% recall and precision
- Non-recursive NP boundaries: ≈ 93% recall and precision
- PP attachments: ≈ 82% recall and precision
- Coordination ambiguities: ≈ 61% recall and precision
Dependencies
The basic idea:

- Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies.

In the words of Lucien Tesnière [Tesnière 1959]:

- The sentence is an organized whole, the constituent elements of which are words. [1.2] Every word that belongs to a sentence ceases by itself to be isolated as in the dictionary. Between the word and its neighbors, the mind perceives connections, the totality of which forms the structure of the sentence. [1.3] The structural connections establish dependency relations between the words. Each connection in principle unites a superior term and an inferior term. [2.1] The superior term receives the name governor. The inferior term receives the name subordinate. Thus, in the sentence Alfred parle [. . .], parle is the governor and Alfred the subordinate. [2.2]
Economic news had little effect on financial markets.

Diagram:

```
Economic
  adj
  nsubj
  amod

news
  noun
  dobj
  amod

had
  verb
  prep
  amod

little
  adj
  pobj
  amod

effect
  noun
  prep
  amod

on
  prep

financial
  adj
  noun

markets
  noun
```
Connectedness, Acyclicity and Single-Head

Intuitions:
- Syntactic structure is complete (Connectedness).
- Syntactic structure is hierarchical (Acyclicity).
- Every word has at most one syntactic head (Single-Head).

Connectedness can be enforced by adding a special root node.

ROOT Economic news had little effect on financial markets.

ROOT adj noun verb adj noun prep prep adj noun .
Constits -> Deps

- Every phrase has a head word. It dominates all other words of that phrase in the dep. graph.
- Head rules: for every nonterminal in tree, choose one of its children to be its “head”. This will define head words.
- Every nonterminal type has a different head rule; e.g. from Collins (1997):

  - If parent is NP,
    - Search from right-to-left for first child that’s NN, NNP, NNPS, NNS, NX, JJR
    - Else: search left-to-right for first child which is NP
Adding Headwords to Trees

NP

DT the

NN lawyer

VP

Vt questioned

NP

DT the

NN witness

S

NP

DT the

NN lawyer

VP

Vt questioned

NP

DT the

NN witness

S(questioned)

NP(lawyer)

DT the

NN lawyer

VP(questioned)

Vt questioned

NP(witness)

DT the

NN witness

Thursday, November 6, 14
Constitits -> Deps

- Head rules were first used to add words into PCFG nonterminals ("lexicalized PCFGs")
- But you can also extract dependency graphs from them!
- Two ways to parse to dependencies:
  - Run a constit parser, then run your favorite constit->deps converter
  - Direct dependency parsing
- Dependencies useful for many applications
- Dependency annotations are available for more languages ... perhaps easier to annotate