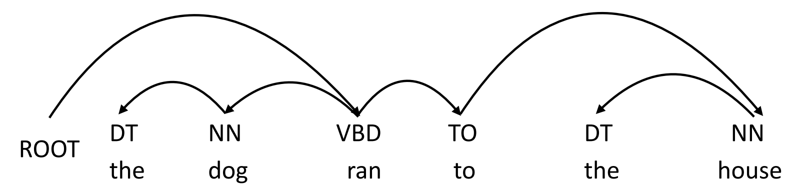
CS388: Natural Language Processing

Lecture 13: Dependency II

Greg Durrett
TEXAS
The University of Texas at Austin

Recall: Dependencies

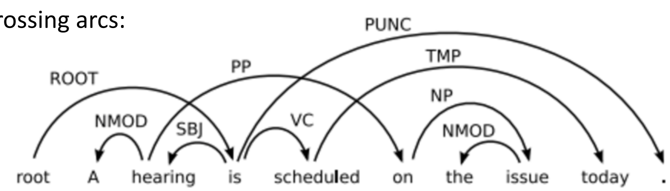
- Dependency syntax: syntactic structure is defined by dependencies
  - Head (parent, governor) connected to dependent (child, modifier)
  - Each word has exactly one parent except for the ROOT symbol
  - Dependencies must form a directed acyclic graph



Recall: Projectivity

- Projective <-> no "crossing" arcs

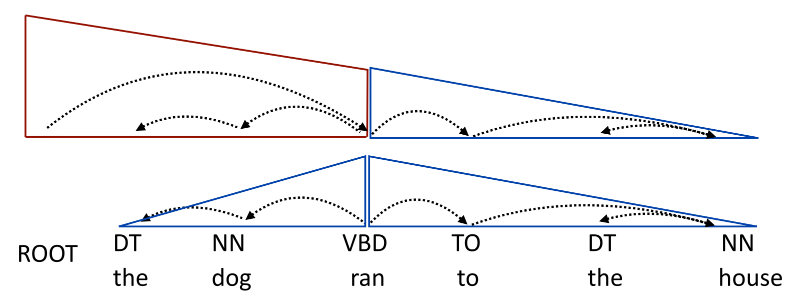
- Crossing arcs:



credit: Language Log

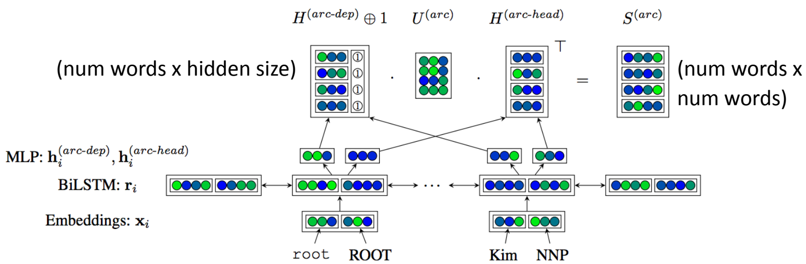
Recall: Eisner's Algorithm

- Left and right children are built independently, heads are edges of spans
- Complete item: all children are attached, head is at the "tall end"
- Incomplete item: arc from "tall end" to "short end", may still expect children



Recall: Biaffine Neural Parsing

- Neural CRFs for dependency parsing: let *c* = LSTM embedding of *i*, *p* = LSTM embedding of parent(*i*). *score*(*i*, parent(*i*), **x**) = $p^{\mathsf{T}}Uc$
  score(tree) = sum of edge scores



LSTM looks at words and POS

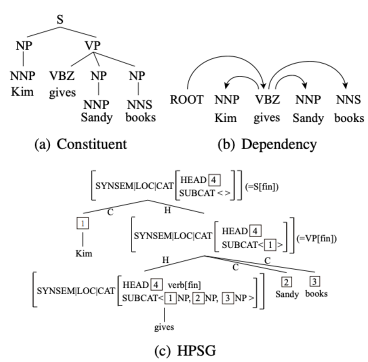
Dozat and Manning (2017)

Evaluating Dependency Parsing

- UAS: unlabeled attachment score. Accuracy of choosing each word's parent (*n* decisions per sentence)
- LAS: additionally consider label for each edge
- Log-linear CRF parser, decoding with Eisner algorithm: 91 UAS
- Higher-order features from Koo parser: 93 UAS
- Best English results with neural CRFs (Dozat and Manning): 95-96 UAS

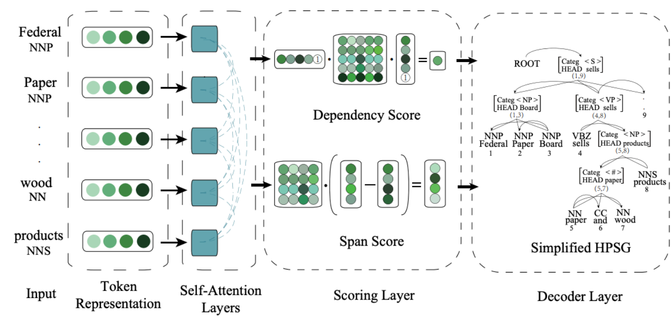
HPSG

- Head-driven phrase structure grammar (HPSG): very complex grammar formalism which annotates large feature structures over tree
- Very little work on HPSG in NLP since no real treebank exists



Pollard and Sag (1994), Zhou and Zhao (2019)

Parsing with "HPSG"



- Joint model of constituency and dependency combining ideas from Dozat + Manning and Stern et al.

Zhou and Zhao (2019)

Parsing with "HPSG"

- Slightly stronger results than Dozat + Manning, significantly better results on Chinese

Model	English		Chinese	
	UAS	LAS	UAS	LAS
Chen and Manning (2014)	91.8	89.6	83.9	82.4
Andor et al. (2016)	94.61	92.79	-	-
Zhang et al. (2016)	93.42	91.29	87.65	86.17
Cheng et al. (2016)	94.10	91.49	88.1	85.7
Kuncoro et al. (2016)	94.26	92.06	88.87	87.30
Ma and Hovy (2017)	94.88	92.98	89.05	87.74
Dozat and Manning (2017)	95.74	94.08	89.30	88.23
Li et al. (2018a)	94.11	92.08	88.78	86.23
Ma et al. (2018)	95.87	94.19	90.59	**89.29**
Our (Division)	94.32	93.09	89.14	87.31
Our (Joint)	**96.09**	**94.68**	**91.21**	89.15
Our (Division*)	-	-	91.69	90.54
Our (Joint*)	-	-	93.24	91.95

Zhou and Zhao (2019)

This Lecture

- Transition-based (shift-reduce) dependency parsing
  - Approximate, greedy inference — fast, but a little bit weird!

Shift-Reduce Parsing

Shift-Reduce Parsing

- Similar to deterministic parsers for compilers
  - Also called transition-based parsing
- A tree is built from a sequence of incremental decisions moving left to right through the sentence
- Stack containing partially-built tree, buffer containing rest of sentence
- Shifts consume the buffer, reduces build a tree on the stack

Shift-Reduce Parsing

ROOT

I ate some spaghetti bolognese

- Initial state: Stack: [ROOT] Buffer: [I ate some spaghetti bolognese]
- Shift: top of buffer -> top of stack
  - Shift 1: Stack: [ROOT I] Buffer: [ate some spaghetti bolognese]
  - Shift 2: Stack: [ROOT I ate] Buffer: [some spaghetti bolognese]

Shift-Reduce Parsing

ROOT

I ate some spaghetti bolognese

- State: Stack: [ROOT I ate] Buffer: [some spaghetti bolognese]
- Left-arc (reduce): Let σ denote the stack, $\sigma|w_{-1}$ = stack ending in w_{-1}
  - "Pop two elements, add an arc, put them back on the stack"

    $\boxed{\sigma|w_{-2}, w_{-1}} \rightarrow \boxed{\sigma|w_{-1}}$ w_{-2} is now a child of w_{-1}
- State: Stack: [ROOT ate] Buffer: [some spaghetti bolognese]

  I

Arc-Standard Parsing

ROOT

I ate some spaghetti bolognese

- Start: stack contains [ROOT], buffer contains [I ate some spaghetti bolognese]
- Arc-standard system: three operations
  - Shift: top of buffer -> top of stack
  - Left-Arc: $\boxed{\sigma|w_{-2}, w_{-1}} \rightarrow \boxed{\sigma|w_{-1}}$, w_{-2} is now a child of w_{-1}
  - Right-Arc $\boxed{\sigma|w_{-2}, w_{-1}} \rightarrow \boxed{\sigma|w_{-2}}$, w_{-1} is now a child of w_{-2}
- End: stack contains [ROOT], buffer is empty []
- How many transitions do we need if we have n words in a sentence?

Arc-Standard Parsing

ROOT

I ate some spaghetti bolognese

S	top of buffer -> top of stack
LA	pop two, left arc between them
RA	pop two, right arc between them

[ROOT]		[I ate some spaghetti bolognese]
	S	
[ROOT I]		[ate some spaghetti bolognese]
	S	
[ROOT I ate]		[some spaghetti bolognese]
	L	
[ROOT ate]		[some spaghetti bolognese]

I

- Could do the left arc later! But no reason to wait
- Can't attach ROOT <- ate yet even though this is a correct dependency!

Arc-Standard Parsing

ROOT

I ate some spaghetti bolognese

S	top of buffer -> top of stack
LA	pop two, left arc between them
RA	pop two, right arc between them

[ROOT ate] [some spaghetti bolognese]

I

S

S

[ROOT ate some spaghetti] [bolognese]

I

L

[ROOT ate spaghetti] [bolognese]

I some

S

Arc-Standard Parsing

ROOT

I ate some spaghetti bolognese

S	top of buffer -> top of stack
LA	pop two, left arc between them
RA	pop two, right arc between them

[ROOT ate spaghetti bolognese] []

I some

R

[ROOT ate spaghetti] []

I some bolognese

R

[ROOT ate] []

spaghetti

I some bolognese

- Stack consists of all words that are still waiting for right children, end with a bunch of right-arc ops

Final state:

[ROOT] []

ate

spaghetti

I some bolognese

Other Systems

- Arc-eager (Nivre, 2004): lets you add right arcs sooner and keeps items on stack, separate reduce action that clears out the stack
- Arc-swift (Qi and Manning, 2017): explicitly choose a parent from what's on the stack
- Many ways to decompose these, which one works best depends on the language and features (nonprojective variants too!)

Building Shift-Reduce Parsers

[ROOT] [I ate some spaghetti bolognese]

- How do we make the right decision in this case?
- Only one legal move (shift)

[ROOT ate some spaghetti] [bolognese]

I

- How do we make the right decision in this case? (all three actions legal)
- Multi-way classification problem: shift, left-arc, or right-arc?

$$\text{argmax}_{a \in \{\text{S,LA,RA}\}} w^\top f(\text{stack}, \text{buffer}, a)$$

Features for Shift-Reduce Parsing

[ROOT ate some spaghetti]   [bolognese]

I

- Features to know this should left-arc?
- One of the harder feature design tasks!
- In this case: the stack tag sequence VBD - DT - NN is pretty informative — looks like a verb taking a direct object which has a determiner in it
- Things to look at: top words/POS of buffer, top words/POS of stack, leftmost and rightmost children of top items on the stack

Training a Greedy Model

[ROOT ate some spaghetti]   [bolognese]

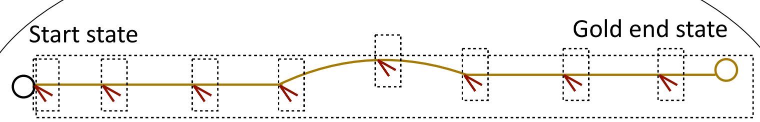
I

$$\text{argmax}_{a \in \{\text{S,LA,RA}\}} w^\top f(\text{stack}, \text{buffer}, a)$$

- Can turn a tree into a decision sequence ***a*** by building an *oracle*
- Train a classifier to predict the right decision using these as training data
- Training data assumes you made correct decisions up to this point and teaches you to make the correct decision, but what if you screwed up…

Greedy training

State space



- Greedy: 2n local training examples
- Non-gold states unobserved during training: consider making bad decisions but don't *condition* on bad decisions

Speed Tradeoffs

	Parser	Dev UAS	Dev LAS	Test UAS	Test LAS	Speed (sent/s)
Unoptimized S-R	standard	89.9	88.7	89.7	88.3	51
	eager	90.3	89.2	89.9	88.6	63
Optimized S-R	Malt:sp	90.0	88.8	89.9	88.5	560
	Malt:eager	90.1	88.9	90.1	88.7	535
Graph-based	MSTParser	92.1	90.8	**92.0**	90.5	12
Neural S-R	Our parser	**92.2**	**91.0**	**92.0**	**90.7**	**1013**

- Many early-2000s constituency parsers were ~5 sentences/sec
- Using S-R used to mean taking a performance hit compared to graph-based, that's no longer (quite as) true

Chen and Manning (2014)

Global Decoding

Global Decoding

ROOT

I gave him dinner

[ROOT gave him] [dinner]

I

- Is it a problem that we make decisions greedily?
- Correct: Right-arc, Shift, Right-arc, Right-arc

[ROOT gave] [dinner]

I him

[ROOT gave dinner] []

I him

[ROOT gave] []

I him dinner

Global Decoding: A Cartoon

ROOT

I gave him dinner

[ROOT gave him] [dinner]

I

S

LA

RA

[ROOT gave him dinner] []

I

LA

RA

- Both wrong! Also both probably low scoring!

[ROOT gave] [dinner]

I him

S

- Correct, high scoring option

Global Decoding: A Cartoon

ROOT

I gave him dinner

[ROOT gave him] [dinner]

I

- Lookahead can help us avoid getting stuck in bad spots
- Global model: maximize sum of scores over all decisions
- Similar to how Viterbi works: we maintain uncertainty over the current state so that if another one looks more optimal going forward, we can use that one

Global Shift-Reduce Parsing

- Greedy: repeatedly execute

$$a_{\text{best}} \leftarrow \text{argmax}_a w^\top f(s,a)$$
$$s \leftarrow a_{\text{best}}(s)$$

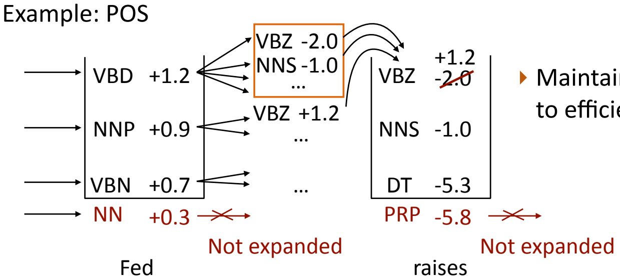
- Global:

$$\text{argmax}_{\mathbf{s},\mathbf{a}} w^\top f(\mathbf{s},\mathbf{a}) = \sum_{i=1}^{2n} w^\top f(s_i, a_i)$$
$$s_{i+1} = a_i(s_i)$$

- Can we do search exactly? How many states *s* are there?
- No! Use beam search

Beam Search

- Maintain a beam of *k* plausible states at the current timestep, expand each and only keep top *k* best new ones
- Example: POS

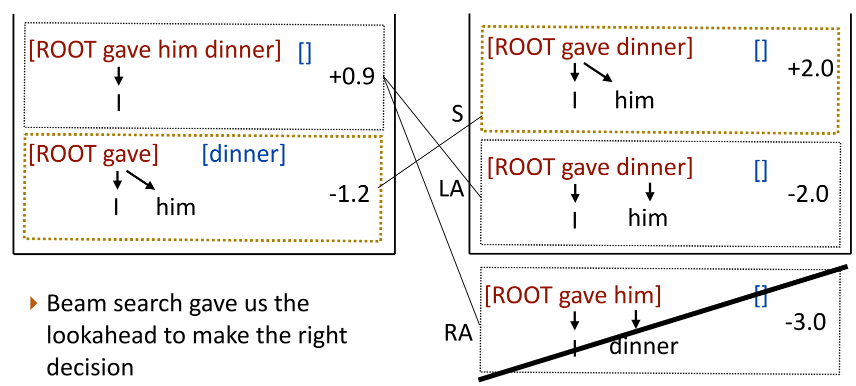


- Maintain priority queue to efficiently add things
- Beam size of k, n words, s states, time complexity O(nks log(k))

How good is beam search?

- *k*=1: greedy search
- Choosing beam size:
  - 2 is usually better than 1
  - Usually don't use larger than 50
  - Depends on problem structure

Global Shift-Reduce Parsing



- Beam search gave us the lookahead to make the right decision

Global Training

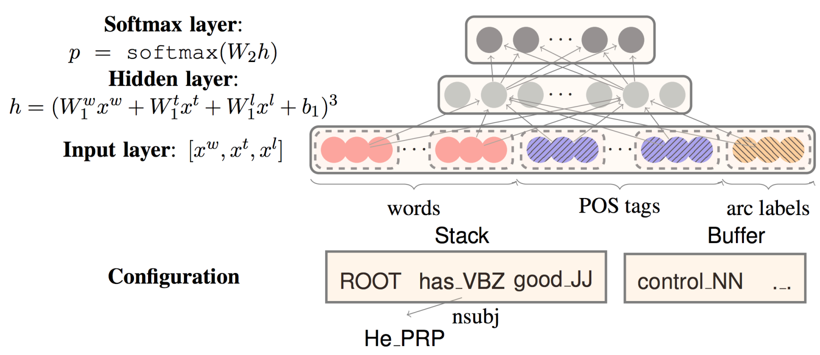
- If using global inference, should train the parser in a global fashion as well: use structured perceptron / structured SVM
- Model treats an entire derivation as something to featurize
- No algorithm like Viterbi for doing efficient parsing, so use beam search

State-of-the-art Transition-Based Parsers

Dependency Parsers

- 2005: Eisner algorithm graph-based parser was SOTA (~91 UAS)
- 2010: Koo's 3rd-order parser was SOTA for graph-based (~93 UAS)
- 2012: Maltparser was SOTA was for transition-based (~90 UAS)
- 2014: Chen and Manning got 92 UAS with transition-based neural model
- 2016: Improvements to Chen and Manning

State-of-the-art Parsers



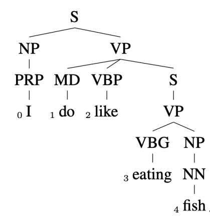
Chen and Manning (2014)

Parsey McParseFace (a.k.a. SyntaxNet)

- Close to state-of-the-art, released by Google publicly
- 94.61 UAS on the Penn Treebank using a global transition-based system with early updating (compared to 95.8 for Dozat, 93.7 for Koo in 2009)
  - Additional data harvested via “tri-training”, form of self-training
- Feedforward neural nets looking at words and POS associated with words in the stack / those words’ children / words in the buffer
- Feature set pioneered by Chen and Manning (2014), Google fine-tuned it
- Shift-reduce parsers are often playing “catch-up”, hard to really push the SOTA with shift-reduce because it’s harder to design models

Andor et al. (2016)

Shift-Reduce Constituency



(a) gold parse tree

steps	structural action	label action	stack after	bracket
1–2	sh(I/PRP)	label-NP	$_0\triangle_1$	$_0$NP$_1$
3–4	sh(do/MD)	nolabel	$_0\triangle_1\triangle_2$	
5–6	sh(like/VBP)	nolabel	$_0\triangle_1\triangle_2\triangle_3$	
7–8	comb	nolabel	$_0\triangle_1\triangle_3$	
9–10	sh(eating/VBG)	nolabel	$_0\triangle_1\triangle_3\triangle_4$	
11–12	sh(fish/NN)	label-NP	$_0\triangle_1\triangle_3\triangle_4\triangle_5$	$_4$NP$_5$
13–14	comb	label-S-VP	$_0\triangle_1\triangle_3\triangle_5$	$_3$S$_5$, $_3$VP$_5$
15–16	comb	label-VP	$_0\triangle_1\triangle_5$	$_1$VP$_5$
17–18	comb	label-S	$_0\triangle_5$	$_0$S$_5$

(b) static oracle actions

combine with no label for ternary rules

- Can do shift-reduce for constituency as well, reduce operation builds constituents

Cross and Huang (2016)

Recap

- Shift-reduce parsing can work nearly as well as graph-based
- Arc-standard system for transition-based parsing
- Purely greedy or more “global” approaches
- Next time: semantic parsing