COS 584

Advanced Natural Language Processing

P8: Dependency Parsing

Spring 2021

TACL 2016

Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations

Eliyahu Kiperwasser
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
elikip@gmail.com

Yoav Goldberg
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
yoav.goldberg@gmail.com

Main take-aways:

- Use bidirectional LSTMs to “build” features for dependency parsing
- It is applied to both **transition-based** dependency parsing and **graph-based** dependency parsing
- End-to-end training of a structured prediction model with **neural feature extractors**

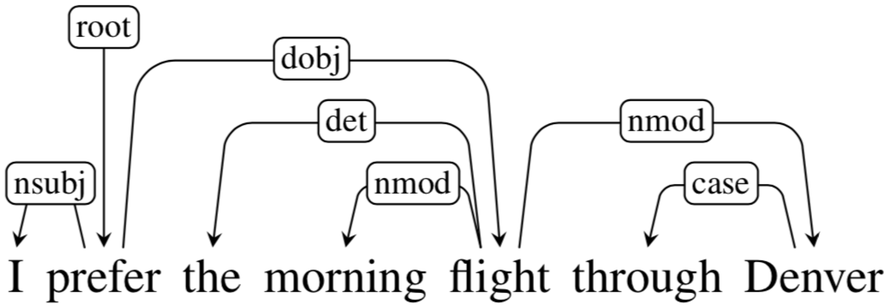
Dependency parsing

Dependency parsing is the task of recognizing a sentence and assigning a **dependency** structure to it.

Input

I prefer the morning flight through Denver

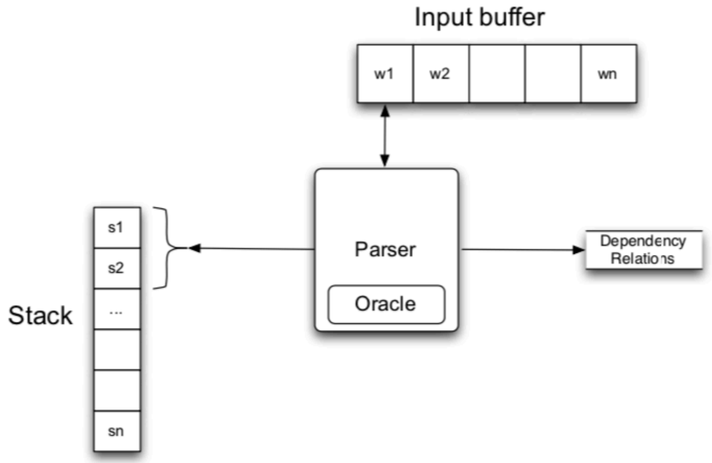
Output



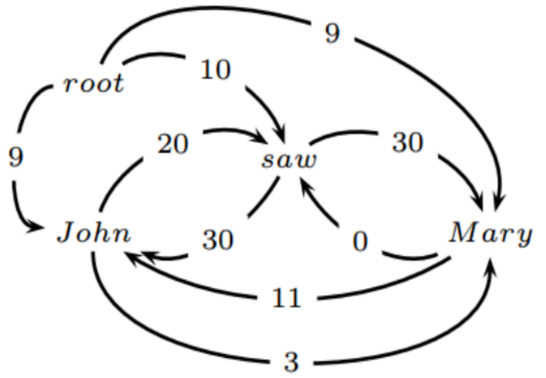
Two families of algorithms

Transition-based dependency parsing

- Also called “shift-reduce parsing”



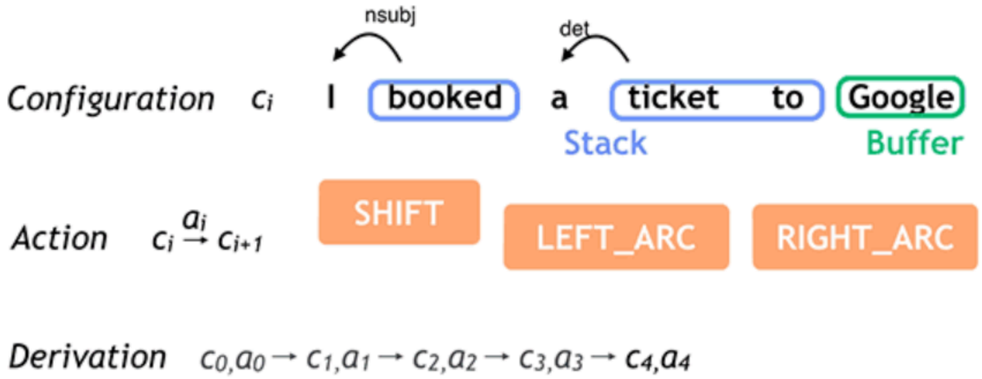
Graph-based dependency parsing



Transition-based Dependency Parsing

- A configuration consists of a stack s, a buffer b and a set of dependency arcs A: $c = (s, b, A)$
- Initially, $s = [\text{ROOT}]$, $b = [w_1, w_2, \ldots, w_n]$, $A = \emptyset$
- Three types of transitions:

  LEFT-ARC (r), RIGHT-ARC (r), SHIFT



- A configuration is terminal if $s = [\text{ROOT}]$ and $b = \emptyset$
- **Inference**: let the classifier predict the next transition repeatedly until we reach a terminal configuration

Greedy transition-based dependency parsing

Graph-based Dependency Parsing

- **Basic idea**: let's predict the dependency tree directly

  X: sentence, Y: any possible dependency tree

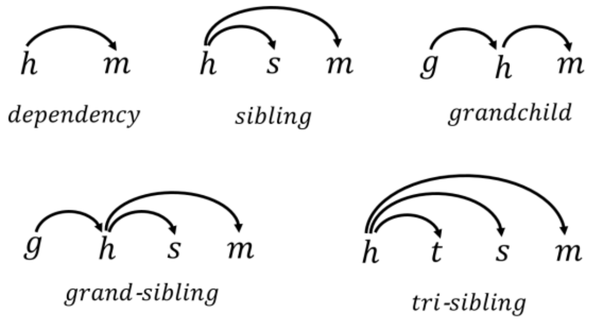
$$Y^* = argmax_{Y \in \Phi(X)} \text{score}(X, Y)$$

$$\max(0, 1 + \max_{y' \neq y} \text{score}(x, y') - \text{score}(x, y))$$

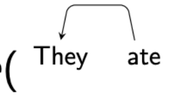
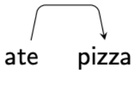
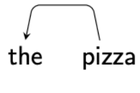
- **Factorization**:

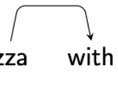
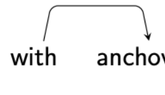
$$\text{score}(X, Y) = \sum_{e \in Y} \text{score}(e) = \sum_{e \in Y} w^\intercal f(e)$$

e: $h \rightarrow m$, first-order



Score() =

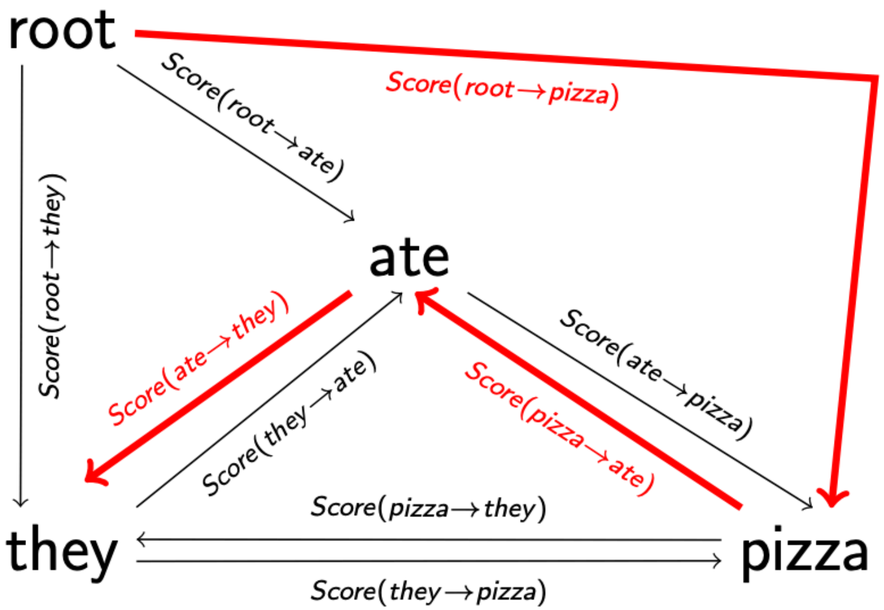
Score() + Score() + Score() +

Score() + Score()

Graph-based Dependency Parsing

- **Inference**: finding maximum spanning tree (MST) for weighted, directed graph

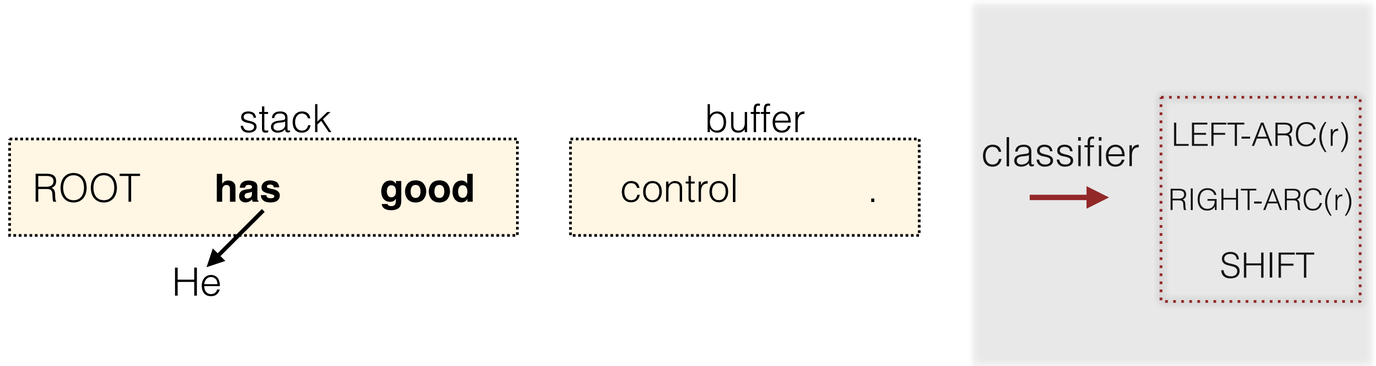
The Chu-Liu Edmonds algorithm $O(n^3)$



Spanning tree with maximal score

Feature Functions

- Transition-based dependency parsing
  - Extract features from the configuration $c = (s, b, A)$



- Graph-based dependency parsing
  - Extract features for each edge (h, m)

Example: The word and POS of the head and modifier items, as well as POS tags of the items around the head and modifier, POS tags of items between the head and modifier, and the distance and direction between the head and modifier.

A History of Dependency Parsing

- Transition-based dependency parsing started from ~2004

Incrementality in Deterministic Dependency Parsing

Joakim Nivre
School of Mathematics and Systems Engineering
Växjö University
SE-35195 Växjö
Sweden
joakim.nivre@msi.vxu.se

- Graph-based dependency parsing started from ~2005

Online Large-Margin Training of Dependency Parsers

Ryan McDonald **Koby Crammer** **Fernando Pereira**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA
{ryantm,crammer,pereira}@cis.upenn.edu

All these methods are based on millions of sparse indicator features

A History of Dependency Parsing

- 2004-2013: a lot of improvements focus on
  - Constructing better features for both families of algorithms
  - Solutions to handle non-projective parse trees
  - Transition-based:
    - Arc-standard, Arc-eager, **Arc-hybrid**, Easy-first, …
    - Better search strategies: beam search, **dynamic oracle**
  - Graph-based:
    - From first-order to second-order/third-order
    - Better inference algorithms

- (Chen and Manning, 2014) introduced neural networks in dependency parsing
  - The features are built based on word/part-of-speech tag/label embeddings of 18 different elements and let FFNNs learn the composition of these elements

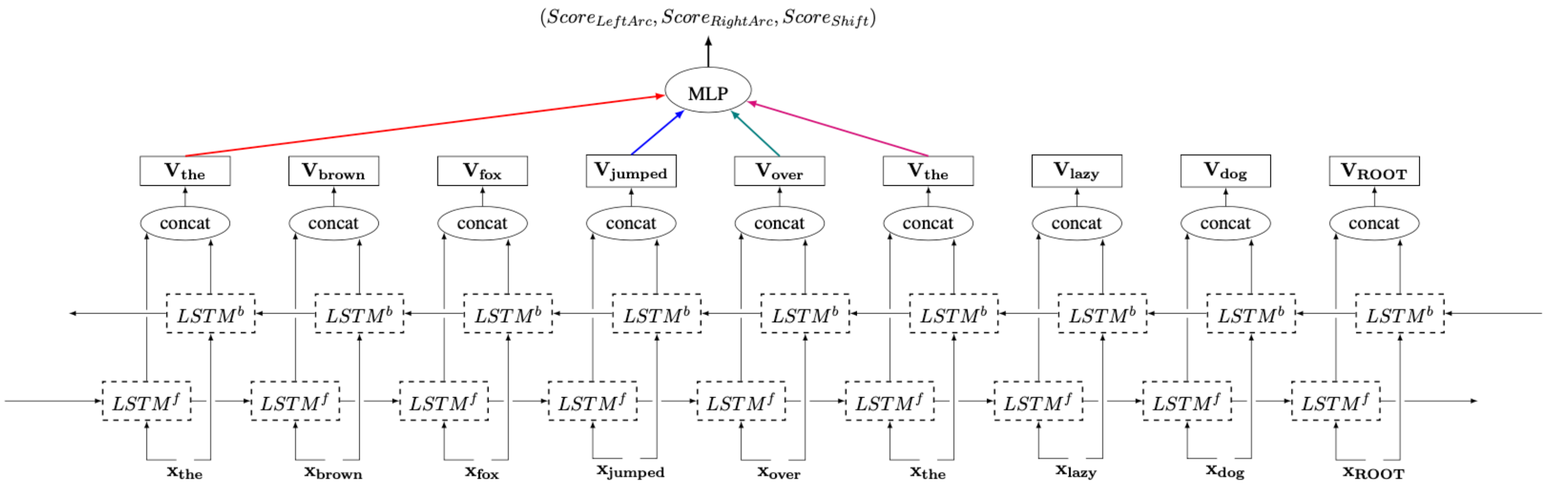
This paper: BiLSTM vectors as minimal features

$$x_i = e(w_i) \circ e(p_i) \qquad v_i = \text{BiLSTM}(x_{1:n}, i)$$

word embedding POS embedding

Configuration:

Scoring:



- Transition-based DP

$$\phi(c) = v_{s_2} \circ v_{s_1} \circ v_{s_0} \circ v_{b_0}$$

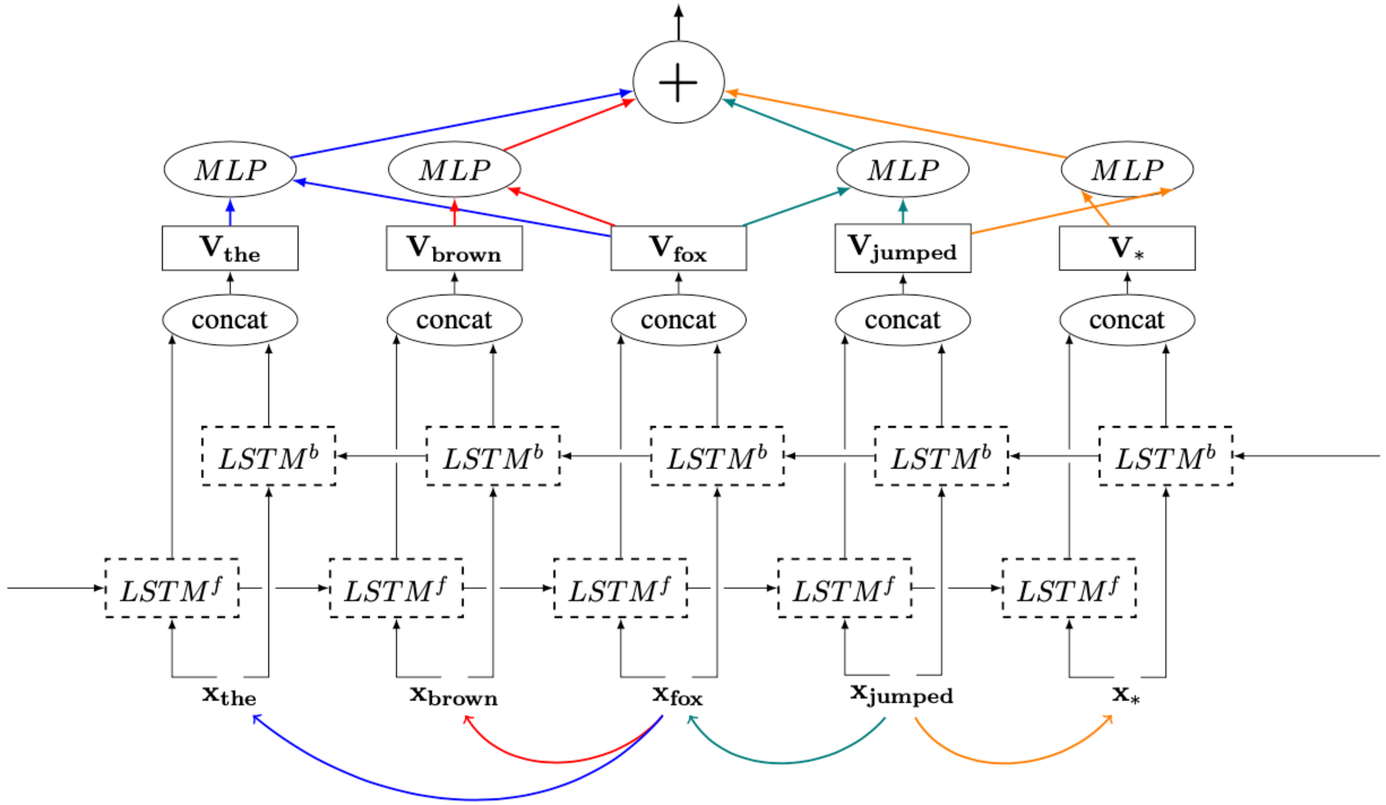
This paper: BiLSTM vectors as minimal features

$$x_i = e(w_i) \circ e(p_i) \qquad v_i = \text{BiLSTM}(x_{1:n}, i)$$

word embedding POS embedding

- Graph-based DP

$$\text{score}(h, m) = \text{MLP}(v_h \circ v_m)$$



Experimental results

System	Method	Representation	Emb	PTB-YM	PTB-SD		CTB	
				UAS	UAS	LAS	UAS	LAS
This work	graph, 1st order	2 BiLSTM vectors	–	–	93.1	91.0	**86.6**	**85.1**
This work	transition (greedy, dyn-oracle)	4 BiLSTM vectors	–	–	93.1	91.0	86.2	85.0
This work	transition (greedy, dyn-oracle)	11 BiLSTM vectors	–	–	**93.2**	**91.2**	86.5	84.9
ZhangNivre11	transition (beam)	large feature set (sparse)	–	92.9	–	–	86.0	84.4
Martins13 (TurboParser)	graph, 3rd order+	large feature set (sparse)	–	92.8	93.1	–	–	–
Pei15	graph, 2nd order	large feature set (dense)	–	93.0	–	–	–	–
Dyer15	transition (greedy)	Stack-LSTM + composition	–	–	92.4	90.0	85.7	84.1
Ballesteros16	transition (greedy, dyn-oracle)	Stack-LSTM + composition	–	–	92.7	90.6	86.1	84.5
This work	graph, 1st order	2 BiLSTM vectors	YES	–	93.0	90.9	86.5	84.9
This work	transition (greedy, dyn-oracle)	4 BiLSTM vectors	YES	–	93.6	91.5	87.4	85.9
This work	transition (greedy, dyn-oracle)	11 BiLSTM vectors	YES	–	93.9	91.9	**87.6**	86.1
Weiss15	transition (greedy)	large feature set (dense)	YES	–	93.2	91.2	–	–
Weiss15	transition (beam)	large feature set (dense)	YES	–	**94.0**	**92.0**	–	–
Pei15	graph, 2nd order	large feature set (dense)	YES	93.3	–	–	–	–
Dyer15	transition (greedy)	Stack-LSTM + composition	YES	–	93.1	90.9	87.1	85.5
Ballesteros16	transition (greedy, dyn-oracle)	Stack-LSTM + composition	YES	–	93.6	91.4	**87.6**	**86.2**
LeZuidema14	reranking /blend	inside-outside recursive net	YES	93.1	93.8	91.5	–	–
Zhu15	reranking /blend	recursive conv-net	YES	93.8	–	–	85.7	–

Breakout discussion

- Group 1 (Danqi)
  - How can we further improve the models presented in this paper?
- Group 2 (Zexuan)
  - Compare the pros and cons of transition-based and graph-based dependency parsing
- Group 3 (Shunyu)
  - Why do you think bidirectional LSTMs build good features for dependency parsing? What features are important for making parsing decisions?
- Group 4 (Kaiyu)
  - Is there anything else interesting in this paper (that we haven't covered yet)?

Use the remaining time for free-form discussion!!!