

Natural Language Processing with Deep Learning

CS224N/Ling284



Christopher Manning
Lecture 5: Dependency Parsing



Lecture Plan

Linguistic Structure: Dependency parsing

1. Syntactic Structure: Consistency and Dependency (25 mins)
2. Dependency Grammar and Treebanks (15 mins)
3. Transition-based dependency parsing (15 mins)
4. Neural dependency parsing (15 mins)

Reminders/comments:

Assignment 2 was due just before class ☺

Assignment 3 (dep parsing) is out today ☹

Start installing and learning PyTorch (Ass 3 has scaffolding)

Final project discussions – **come meet with us**; focus of week 5

Chris make-up office hour this week: Wed 1:00–2:20pm



1. Two views of linguistic structure:

Constituency = phrase structure grammar = context-free grammars (CFGs)

human language structure
from Two views

Phrase structure organizes words into nested constituents

Starting unit: words

the, cat, cuddly, by, door

Words combine into phrases

the cuddly cat, by the door

Phrases can combine into bigger phrases

the cuddly cat by the door



1. Two views of linguistic structure:

Constituency = phrase structure grammar = context-free grammars (CFGs)

computation linguistic
dominant grammer .

Phrase structure organizes words into nested constituents

Can represent the grammar with CFG rules

Starting unit: words are given a category (part of speech = pos)

the, cat, cuddly, by, door
Det N Adj P N

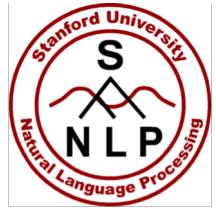
Words combine into phrases with categories

the cuddly cat, by the door
 $\text{NP} \rightarrow \text{Det Adj N}$ $\text{PP} \rightarrow \text{P NP}$

CFGs ?
free grammar rule context - . (ex.
 $\text{NP} \rightarrow \text{Det N}$)

Phrases can combine into bigger phrases recursively

the cuddly cat by the door
 $\text{NP} \rightarrow \text{NP PP}$



Two views of linguistic structure:
Constituency = phrase structure grammar
= context-free grammars (CFGs)

Phrase structure organizes words into nested constituents.

the	cat
a	dog
large	in a crate
barking	on the table
cuddly	by the door
large	barking

talk to
walked behind

Context - free
nonterminal
nonterminal

Context - free Grammar Rule

Recursively
(ex. NP NP가 , NP
PP가 , PP NP가 .)
(
)

NP -> Det N
NP -> Det (Adj) N

NP -> Det (Adj) N PP
PP -> Prep NP

Context - free Grammar Rule

Language Structure

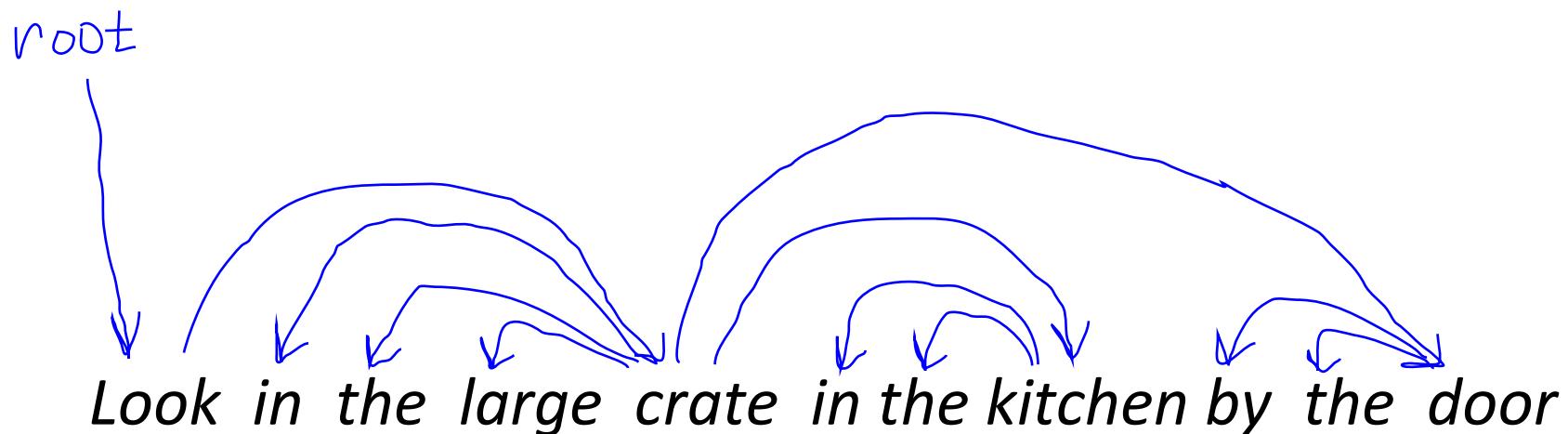
Linguistics



Two views of linguistic structure: Dependency structure

- Dependency structure shows which words depend on (modify or are arguments of) which other words.

phrase category
dependency structure
how words are arguments or modifiers of other words in a recursive fashion.





Why do we need sentence structure?

We need to understand sentence structure in order to be able to interpret language correctly

가
sentence structure .

what words are arguments and modifiers of other words
, sentence . ()

Humans communicate complex ideas by composing words together into bigger units to convey complex meanings

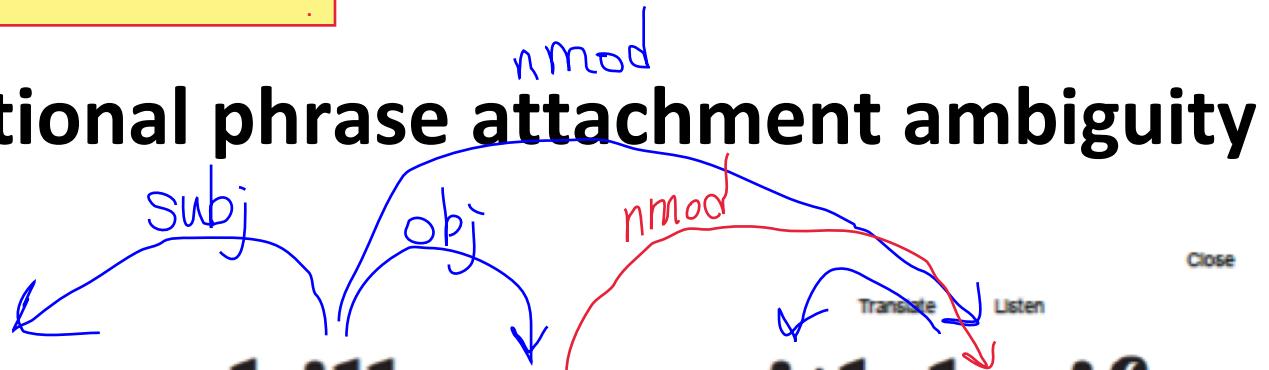
complex idea or meaning
(recursively) .

We need to know what is connected to what



(1)
(2)
가

Prepositional phrase attachment ambiguity



San Jose cops kill man with knife

BBC



Sign in

News

Sport

Weather

Shop

Reel

Travel

NEWS

Home

Video

World

US & Canada

UK

Business

Tech

Science

Stories

Science & Environment

Scientists count whales from space

By Jonathan Amos

BBC Science Correspondent



Prepositional phrase attachment ambiguity

nmod 가 가 .

human language
programming language . ()
langauge understandning .

subj obj nmod

Scientists count whales from space



subj obj nMod

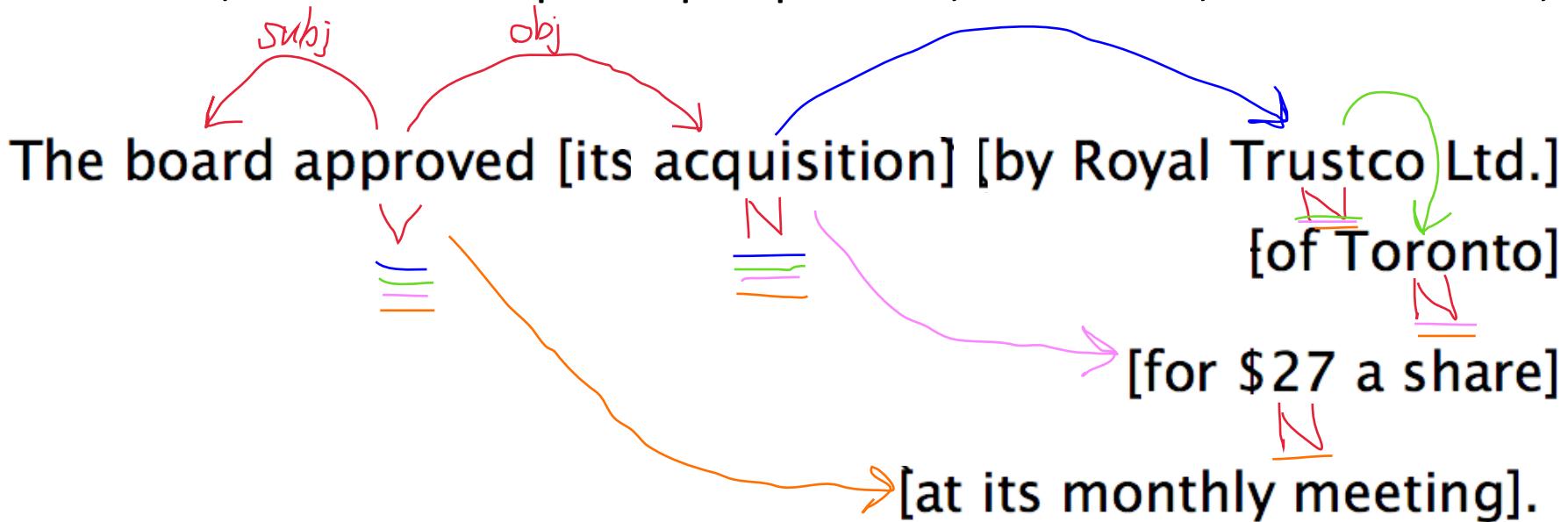
Scientists count whales from space



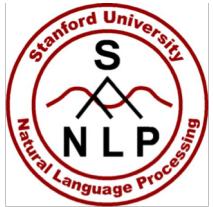


PP attachment ambiguities multiply

- A key parsing decision is how we ‘attach’ various constituents
 - PPs, adverbial or participial phrases, infinitives, coordinations,



- **Catalan numbers:** $C_n = (2n)!/[(n+1)!n!]$
- An exponentially growing series, which arises in many tree-like contexts:
 - E.g., the number of possible triangulations of a polygon with $n+2$ sides
 - Turns up in triangulation of probabilistic graphical models (CS228)....



Coordination scope ambiguity

[Shuttle veteran and longtime NASA executive] Fred Gregory] appointed to board

[Shuttle veteran] and [longtime NASA executive Fred Gregory] appointed to board



Coordination scope ambiguity

The image shows a newspaper clipping from "THE NEWS-GAZETTE" with the section header "NATION / WO". The main headline reads "PRESIDENT'S FIRST PHYSICAL Doctor: No heart, cognitive issues". A yellow box highlights two alternative interpretations of the doctor's statement: "[No heart and/or cognitive issues] ()" and "[No heart] and/or [cognitive issues]".

PRESIDENT'S FIRST PHYSICAL

Doctor: No heart, cognitive issues

But Trump needs to reduce his cholesterol, lose weight

By JILL COLVIN

6-foot-3 president weighed in at 239 pounds — three pounds heavier than he was in September 2016, the last time Trump revealed his weight to the public.

Trump's blood pressure was 122 over 74, and his

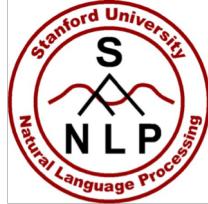
[No heart and/or cognitive issues] ()

[No heart] and/or [cognitive issues]

with no medical issues." Trump has no heart disease and no family history of it.

The 71-year-old president performed "exceedingly well" on cognitive screening, which is not standard but was requested by

White House reporter



Adjectival Modifier Ambiguity

numbers, including some that featured a bucket and bells brigade or people in buckets and trash cans with drums sticks and hammer mallets. PHOTO BY JENNIFER STULTZ

MENTORING DAY

Students get first hand job experience

By Gale Rose

grose@pratttribune.com

Eager students invaded businesses all over Pratt Tuesday, October 24 as they looked for future job opportunities on Disability Mentoring Day.

The 97 students from schools fanned out across Pratt and got first

experience what it would be like to work at those 40 businesses. They asked questions and got some hands on experience with various operations.

Paola Luna of Pratt High School, Gina Pat-

imal Veterinary Clinic for their business. Students got a tour of the facility, learned what happens in an examination, got to handle various animals and watched a snake eat a mouse.

imais. Patton likes all kinds of animals and said she learned a lot from the experience. Watching the snake eat the mouse impressed her the most.

Fernandez wants to become a veterinarian and

MENTORING, 6

...?
Students get [first hand job experience]

- Hospital Pharmacist for 41 years
- 4 years Commissioner for Pratt Planning and Zoning Board of Appeals
- 3 years Pratt City Commission
- Graduate of Pratt High School and KU School of Pharmacy
- Past Member and President of Civic Groups and Organizations
- Experience and Knowledge of Financial Responsibility and Budgeting
- Supports Family Values, Education, and Business Growth
- Common Sense Approach for the Sustained Progress of Pratt

ng Meyer
ty Commissioner

Meyer: Jim Meyer, Treasurer



Verb Phrase (VP) attachment ambiguity

The screenshot shows a news article from theguardian.com. The navigation bar includes icons for user profile, search, and more, followed by the site's name. Below the bar, the breadcrumb navigation shows 'home > world > americas'. To the right are links for 'asia' and '≡ all'. The main title 'Rio de Janeiro' is displayed. A yellow box labeled 'dependency JT' is overlaid on the text. The headline reads: 'Mutilated body washes up on Rio beach to be used for Olympics beach volleyball'. Handwritten-style annotations include blue brackets around 'body washes up', red arrows pointing from 'body' to 'washes' and from 'beach' to 'to be used', and a red bracket under 'beach volleyball'.

the guardian

home > world > americas asia ≡ all

Rio de Janeiro

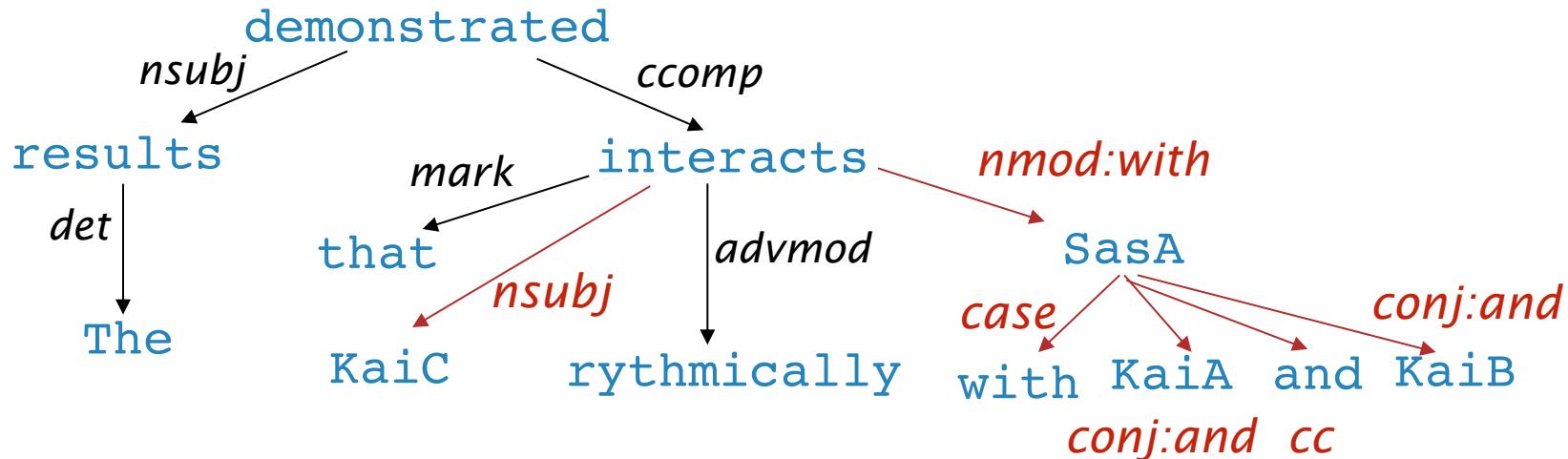
dependency JT

Mutilated body washes up
on Rio beach to be used for
Olympics beach volleyball



Dependency paths identify semantic relations – e.g., for protein interaction

[Erkan et al. EMNLP 07, Fundel et al. 2007, etc.]



KaiC ← nsubj interacts nmod:with → SasA

KaiC ← nsubj interacts nmod:with → SasA conj:and → KaiA

KaiC ← nsubj interacts ~~prep with~~ ^{nmod:with} → SasA conj:and → KaiB

dependency analysis

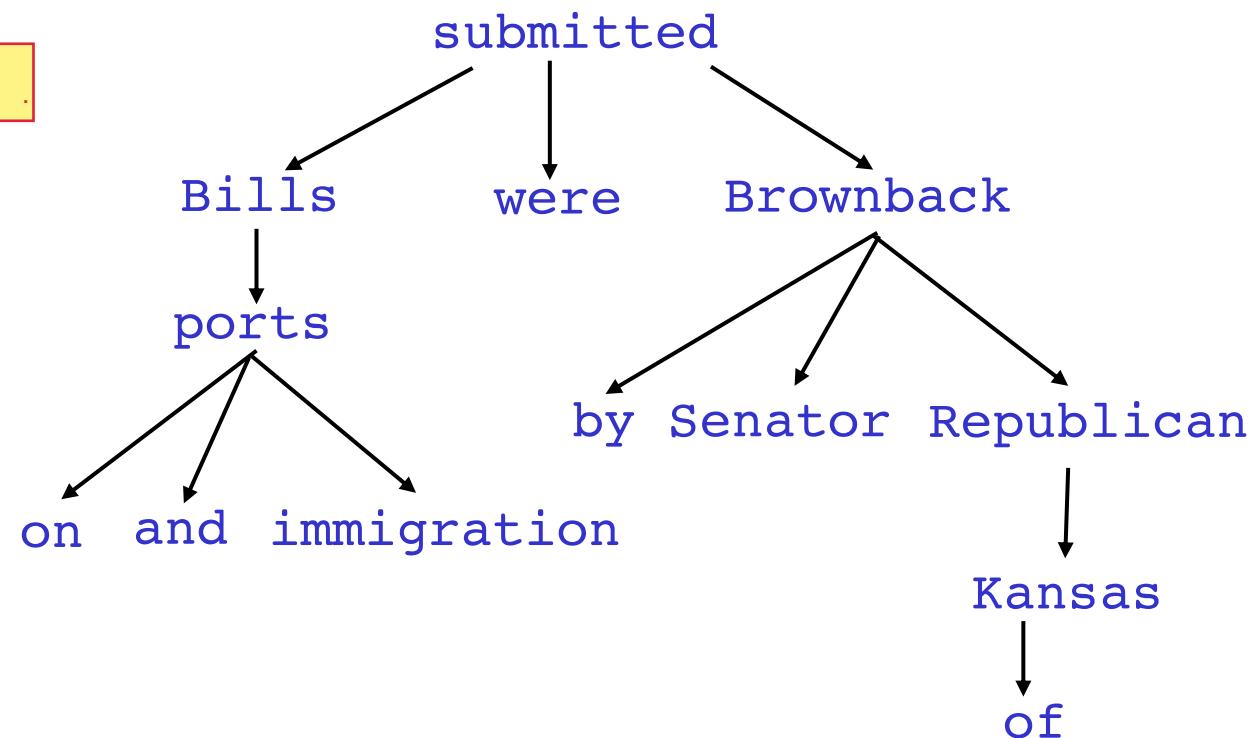
i.e. () interacts ()



2. Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of relations between lexical items, normally binary asymmetric relations (“arrows”) called **dependencies**

Dependency structure ...
tree .

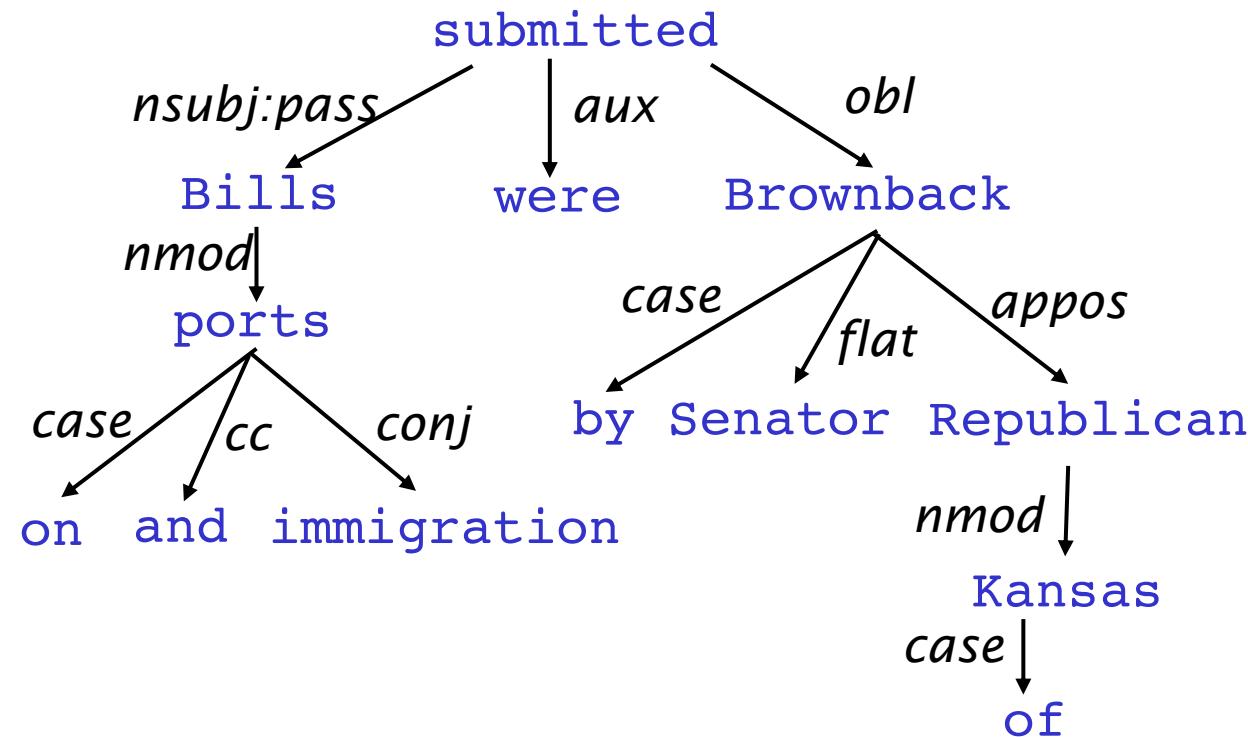




Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of relations between lexical items, normally binary asymmetric relations (“arrows”) called **dependencies**

The arrows are commonly **typed** with the name of grammatical relations (subject, prepositional object, apposition, etc.)



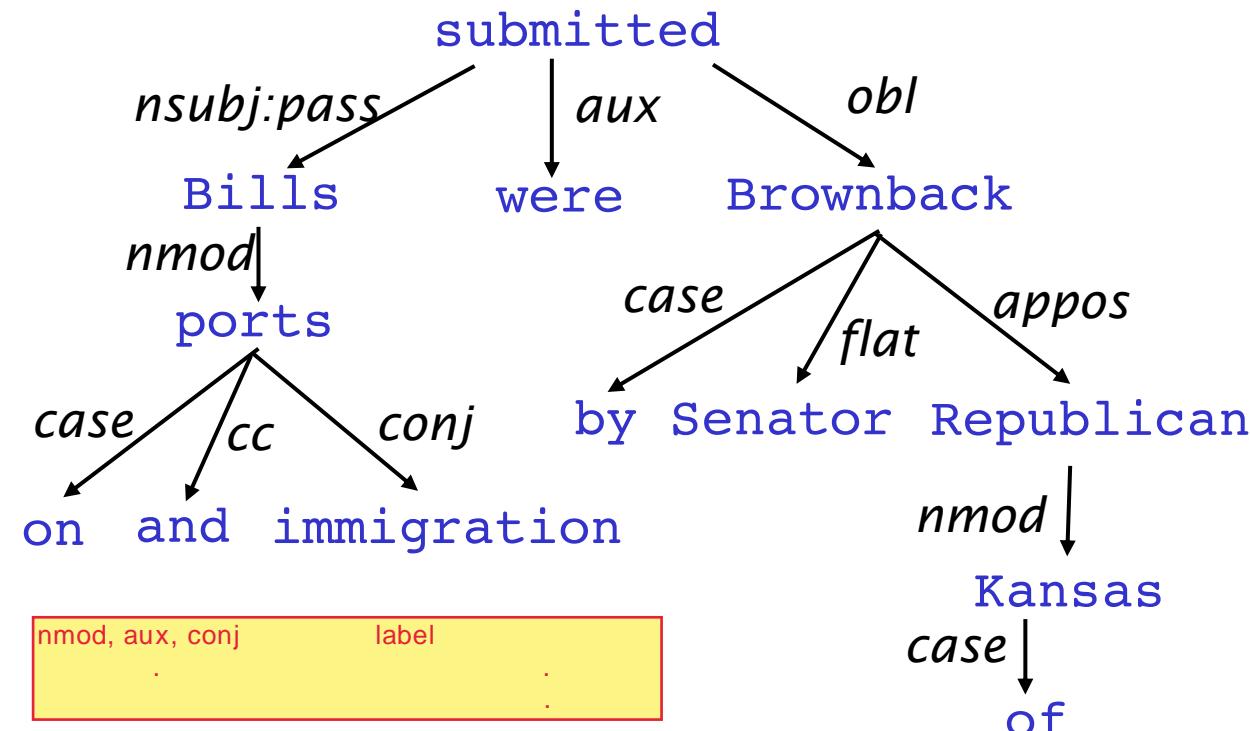


Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of relations between lexical items, normally binary asymmetric relations (“arrows”) called **dependencies**

The arrow connects a **head** (governor, superior, regent) with a **dependent** (modifier, inferior, subordinate)

Usually, dependencies form a **tree** (connected, acyclic, single-head)





Pāṇini's grammar (c. 5th century BCE)

, dependency grammar
가
Pannini's grammar dependency
grammar grammar
Sanskrit structure



Gallery: <http://wellcomeimages.org/indexplus/image/L0032691.html>

CC BY 4.0 File:Birch bark MS from Kashmir of the Rupavatra Wellcome L0032691.jpg



Dependency Grammar/Parsing History

- The idea of dependency structure goes back a long way
 - To Pāṇini's grammar (c. 5th century BCE)
 - Basic approach of 1st millennium Arabic grammarians
- Constituency/context-free grammars is a new-fangled invention
 - 20th century invention (R.S. Wells, 1947; then Chomsky)
- Modern dependency work often sourced to L. Tesnière (1959)
 - Was dominant approach in “East” in 20th Century (Russia, China, ...)
 - Good for free-er word order languages
- Among the earliest kinds of parsers in NLP, even in the US:
 - David Hays, one of the founders of U.S. computational linguistics, built early (first?) dependency parser (Hays 1962)

chomsky ↗ linguistic computer science .



Dependency Grammar and Dependency Structure



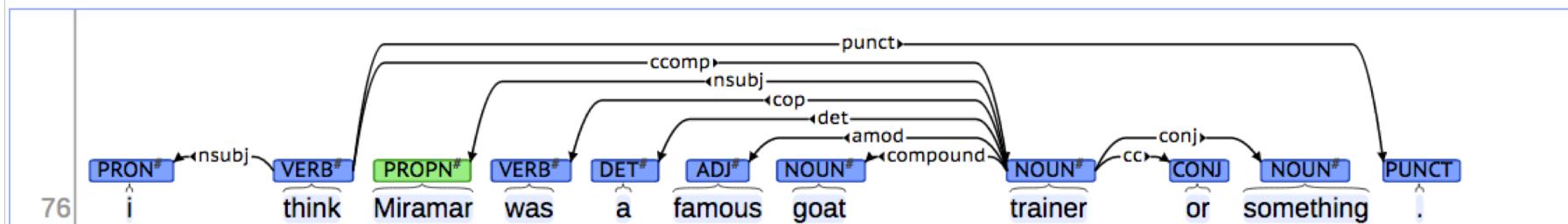
- Some people draw the arrows one way; some the other way!
 - Tesnière had them point from head to dependent...
- Usually add a fake ROOT so every word is a dependent of precisely 1 other node



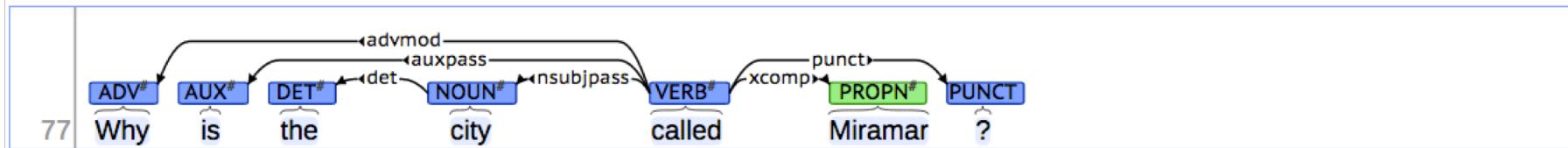
The rise of annotated data: Universal Dependencies treebanks

[Universal Dependencies: <http://universaldependencies.org/> ;
cf. Marcus et al. 1993, The Penn Treebank, *Computational Linguistics*]

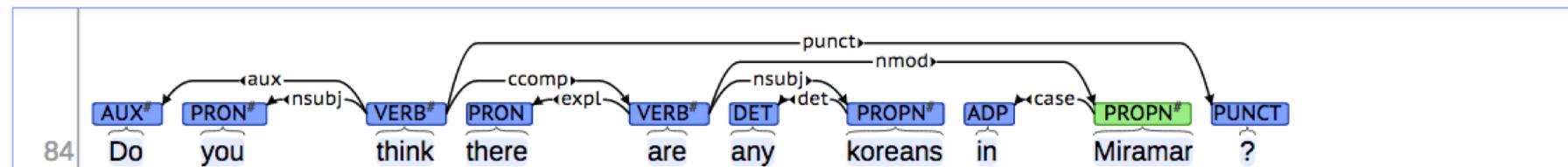
[context] [conllu]



[context] [conllu]



[context] [conllu]



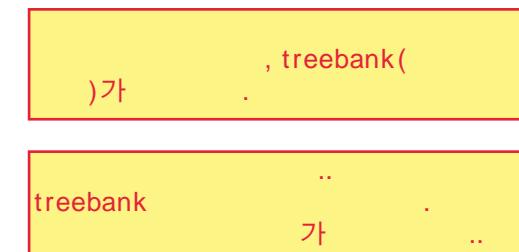


The rise of annotated data

Starting off, building a treebank seems a lot slower and less useful than building a grammar

But a treebank gives us many things

- Reusability of the labor
 - Many parsers, part-of-speech taggers, etc. can be built on it
 - Valuable resource for linguistics
- Broad coverage, not just a few intuitions
- Frequencies and distributional information
- A way to evaluate systems





Dependency Conditioning Preferences

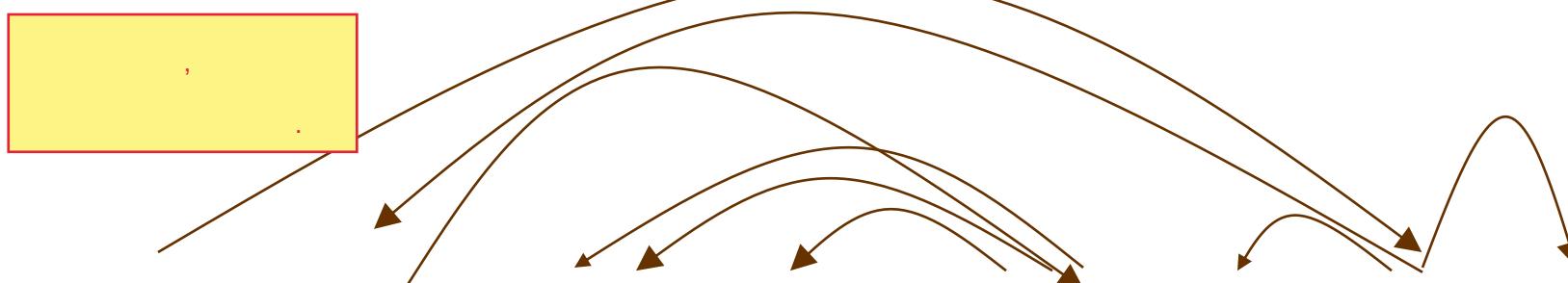
What are the sources of information for dependency parsing?

1. Bilexical affinities [discussion → issues] is plausible
2. Dependency distance mostly with nearby words
3. Intervening material

Dependencies rarely span intervening verbs or punctuation

4. Valency of heads

How many dependents on which side are usual for a head?

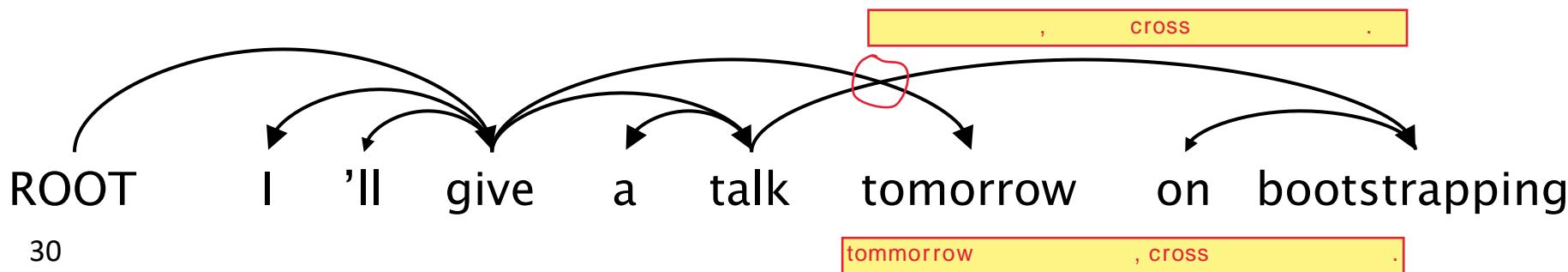


ROOT Discussion of the outstanding issues was completed .



Dependency Parsing

- A sentence is parsed by choosing for each word what other word (including ROOT) is it a dependent of
- Usually some constraints:
 - Only one word is a dependent of ROOT
 - Don't want cycles $A \rightarrow B, B \rightarrow A$
- This makes the dependencies a tree
- Final issue is whether arrows can cross (**non-projective**) or not





Projectivity

dependency structure
constituency structure
non - projective

- Defn: There are no crossing dependency arcs when the words are laid out in their linear order, with all arcs above the words
- Dependencies parallel to a CFG tree must be **projective**
 - Forming dependencies by taking 1 child of each category as head
- But dependency theory normally does **allow** non-projective structures to account for displaced constituents
 - You can't easily get the semantics of certain constructions right without these nonprojective dependencies





Methods of Dependency Parsing

1. Dynamic programming

Eisner (1996) gives a clever algorithm with complexity $O(n^3)$, by producing parse items with heads at the ends rather than in the middle

2. Graph algorithms

You create a Minimum Spanning Tree for a sentence

McDonald et al.'s (2005) MSTParser scores dependencies independently using an ML classifier (he uses MIRA, for online learning, but it can be something else)

3. Constraint Satisfaction

Edges are eliminated that don't satisfy hard constraints. Karlsson (1990), etc.

“Transition-based parsing” or “deterministic dependency parsing”

Greedy choice of attachments guided by good machine learning classifiers

MaltParser (Nivre et al. 2008). Has proven highly effective.



3. Greedy transition-based parsing

[Nivre 2003]



- A simple form of greedy discriminative dependency parser
- The parser does a sequence of bottom up actions
 - Roughly like “shift” or “reduce” in a shift-reduce parser, but the “reduce” actions are specialized to create dependencies with head on left or right
- The parser has:
 - a stack σ , written with top to the right
 - which starts with the ROOT symbol
 - a buffer β , written with top to the left
 - which starts with the input sentence
 - a set of dependency arcs A
 - which starts off empty
 - a set of actions



Basic transition-based dependency parser

Start: $\sigma = [\text{ROOT}], \beta = w_1, \dots, w_n, A = \emptyset$

1. Shift $\sigma, w_i | \beta, A \rightarrow \sigma | w_i, \beta, A$
2. Left-Arc_r $\sigma | w_i | w_j, \beta, A \rightarrow \sigma | w_j, \beta, A \cup \{r(w_j, w_i)\}$
3. Right-Arc_r $\sigma | w_i | w_j, \beta, A \rightarrow \sigma | w_i, \beta, A \cup \{r(w_i, w_j)\}$

Finish: $\sigma = [w], \beta = \emptyset$



Arc-standard transition-based parser

(there are other transition schemes ...)

Analysis of “I ate fish”

Stack

Start



Shift



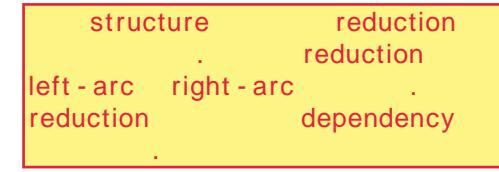
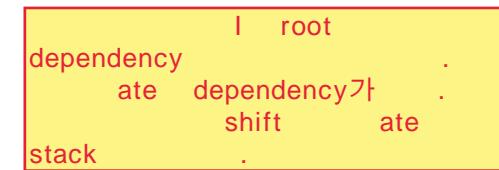
Shift



Start: $\sigma = [\text{ROOT}], \beta = w_1, \dots, w_n, A = \emptyset$

1. Shift $\sigma, w_i|\beta, A \rightarrow \sigma|w_i, \beta, A$
2. Left-Arc_r $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_j, \beta, A \cup \{r(w_j, w_i)\}$
3. Right-Arc_r $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_i, \beta, A \cup \{r(w_i, w_j)\}$

Finish: $\beta = \emptyset$



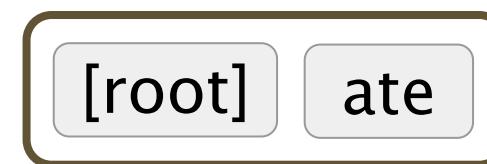


Arc-standard transition-based parser

Analysis of “I ate fish”

exponentially
dynamic programming

Left Arc

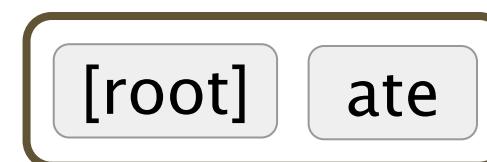


$A +=$
 $nsubj(ate \rightarrow I)$

Shift

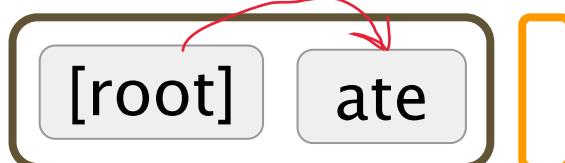


Right Arc



$A +=$
 $obj(ate \rightarrow fish)$

Right Arc



$A +=$
 $root([root] \rightarrow ate)$
Finish



MaltParser

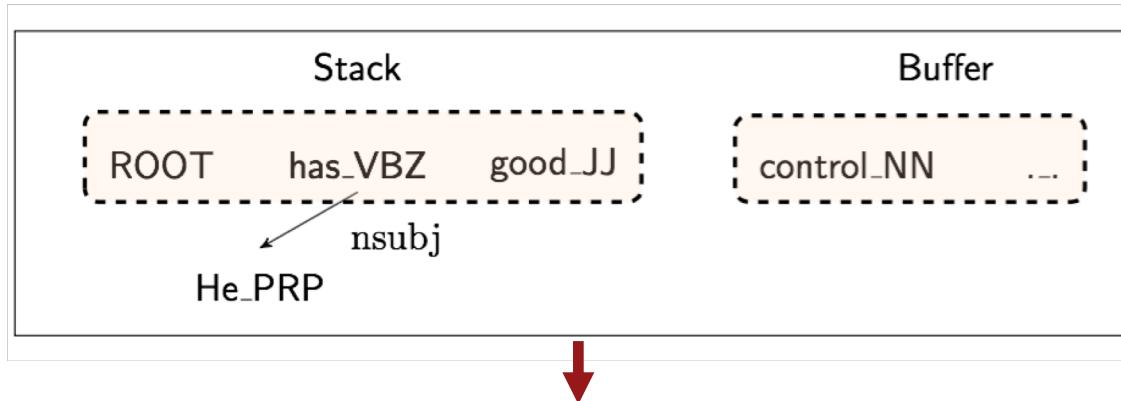
[Nivre and Hall 2005]

shift , left - arc right - arc
거 dependency

- We have left to explain how we choose the next action
 - Answer: Stand back, I know machine learning!
- Each action is predicted by a discriminative classifier (e.g., softmax classifier) over each legal move
 - Max of 3 untyped choices; max of $|R| \times 2 + 1$ when typed
 - Features: top of stack word, POS; first in buffer word, POS; etc.
- There is NO search (in the simplest form)
 - But you can profitably do a beam search if you wish (slower but better): You keep k good parse prefixes at each time step
- The model's accuracy is *fractionally* below the state of the art in dependency parsing, but
 - dynamic programming
cubic
- It provides very fast linear time parsing, with great performance



Conventional Feature Representation



binary, sparse
dim = $10^6 \sim 10^7$

0 0 0 1 0 0 1 0 ... 0 0 1 0

Feature templates: usually a combination of 1 ~ 3 elements from the configuration.

- $s1.w = \text{good} \wedge s1.t = \text{JJ}$
- $s2.w = \text{has} \wedge s2.t = \text{VBZ} \wedge s1.w = \text{good}$
- $lc(s_2).t = \text{PRP} \wedge s_2.t = \text{VBZ} \wedge s_1.t = \text{JJ}$
- $lc(s_2).w = \text{He} \wedge lc(s_2).l = \text{nsubj} \wedge s_2.w = \text{has}$

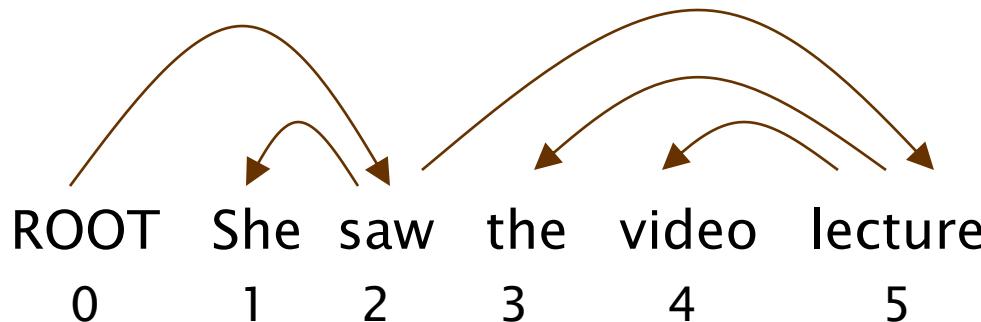
Indicator features

very complex binary indicator features

very complex hand engineering



Evaluation of Dependency Parsing: (labeled) dependency accuracy



UAS	dependency label	,
arc	가.	
LAS	arc label	가.

$$\text{Acc} = \frac{\# \text{ correct deps}}{\# \text{ of deps}}$$

$$\text{UAS} = 4 / 5 = 80\%$$

$$\text{LAS} = 2 / 5 = 40\%$$

Gold

1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	obj

Parsed

1	2	She	nsubj
2	0	saw	root
3	4	the	det
4	5	video	nsubj
5	2	lecture	ccomp

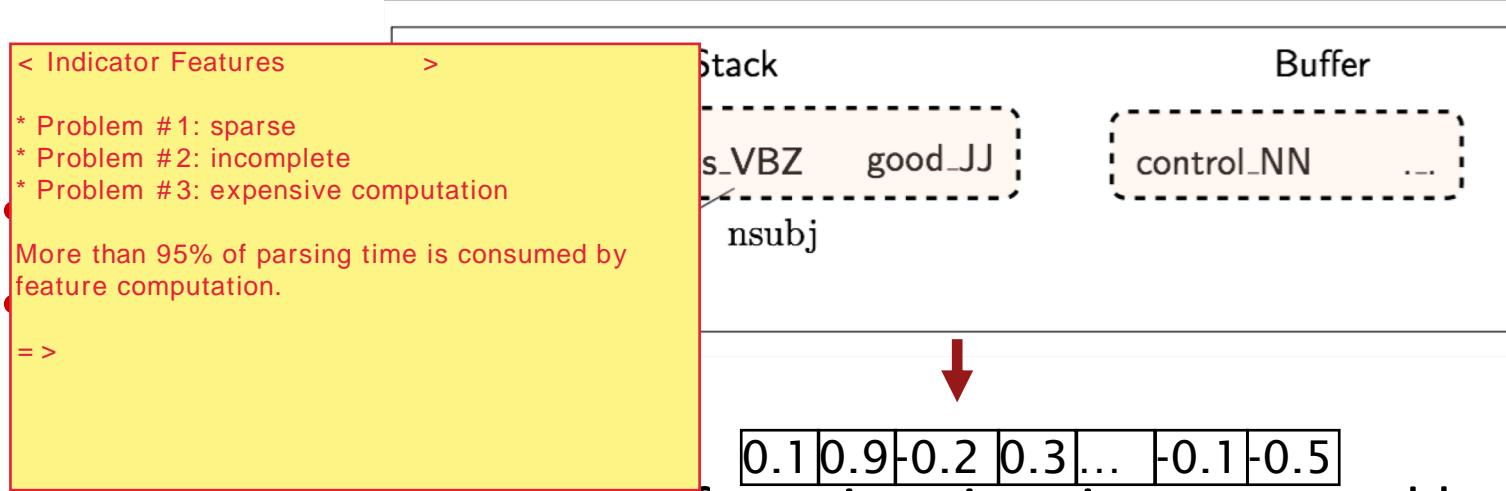


Handling non-projectivity

- The arc-standard algorithm we presented only builds projective dependency trees
- Possible directions to head:
 1. Just declare defeat on nonprojective arcs
 2. Use dependency formalism which only has projective representations
 - CFG only allows projective structures; you promote head of violations
 3. Use a postprocessor to a projective dependency parsing algorithm to identify and resolve nonprojective links
 4. Add extra transitions that can model at least most non-projective structures (e.g., add an extra SWAP transition, cf. bubble sort)
 5. Move to a parsing mechanism that does not use or require any constraints on projectivity (e.g., the graph-based MSTParser)



4. Why train a neural dependency parser? Indicator Features Revisited



More than 95% of parsing time is consumed by feature computation.

- $s1.w = \text{good} \wedge s1.t = \text{JJ}$
- $s2.w = \text{has} \wedge s2.t = \text{VBZ} \wedge s1.w = \text{good}$
- $lc(s_2).t = \text{PRP} \wedge s_2.t = \text{VBZ} \wedge s_1.t = \text{JJ}$
- $lc(s_2).w = \text{He} \wedge lc(s_2).l = \text{nsubj} \wedge s_2.w = \text{has}$



A neural dependency parser

[Chen and Manning 2014]



- English parsing to Stanford Dependencies:
 - Unlabeled attachment score (UAS) = head
 - Labeled attachment score (LAS) = head and label

Parser	UAS	LAS	sent. / s
MaltParser	89.8	87.2	469
MSTParser	91.4	88.1	10
TurboParser	92.3	89.6	8
C & M 2014	92.0	89.7	654

feature
computation
가 .



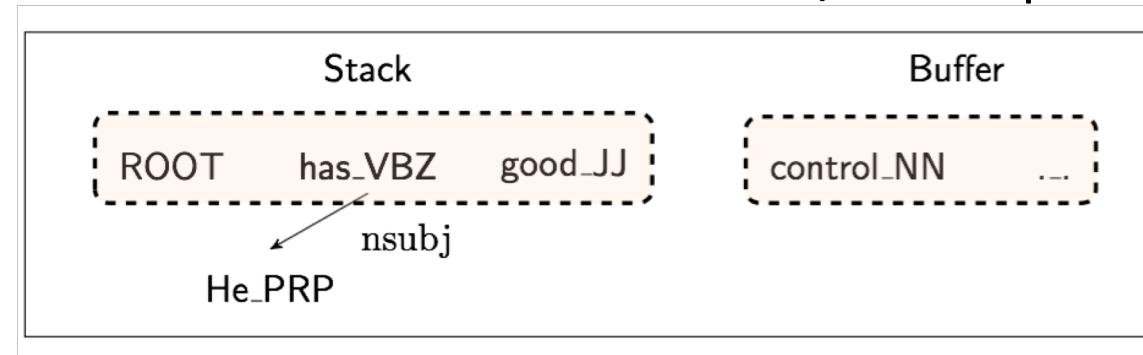
Distributed Representations

- We represent each word as a d -dimensional dense vector (i.e., word embedding)
 - Similar words are expected to have close vectors.
 - Meanwhile, **part-of-speech tags** (POS) and **dependency labels** are also represented as d -dimensional vectors.
 - The smaller discrete sets also exhibit many semantical similarities.
- POS dependency
-
- NNS (plural noun) should be close to NN (singular noun).
- num (numerical modifier) should be close to amod (adjective modifier).



Extracting Tokens and then vector representations from configuration

- We extract a set of tokens based on the stack / buffer positions:



	word	POS	dep.
s_1	good	JJ	\emptyset
s_2	has	VBZ	\emptyset
b_1	control	NN	\emptyset
$lc(s_1)$	\emptyset	\emptyset	\emptyset
$rc(s_1)$	\emptyset	\emptyset	\emptyset
$lc(s_2)$	He	PRP	nsubj
$rc(s_2)$	\emptyset	\emptyset	\emptyset

단계 단계

- We convert them to vector embeddings and concatenate them



Model Architecture

Softmax probabilities

Output layer y

$$y = \text{softmax}(Uh + b_2)$$

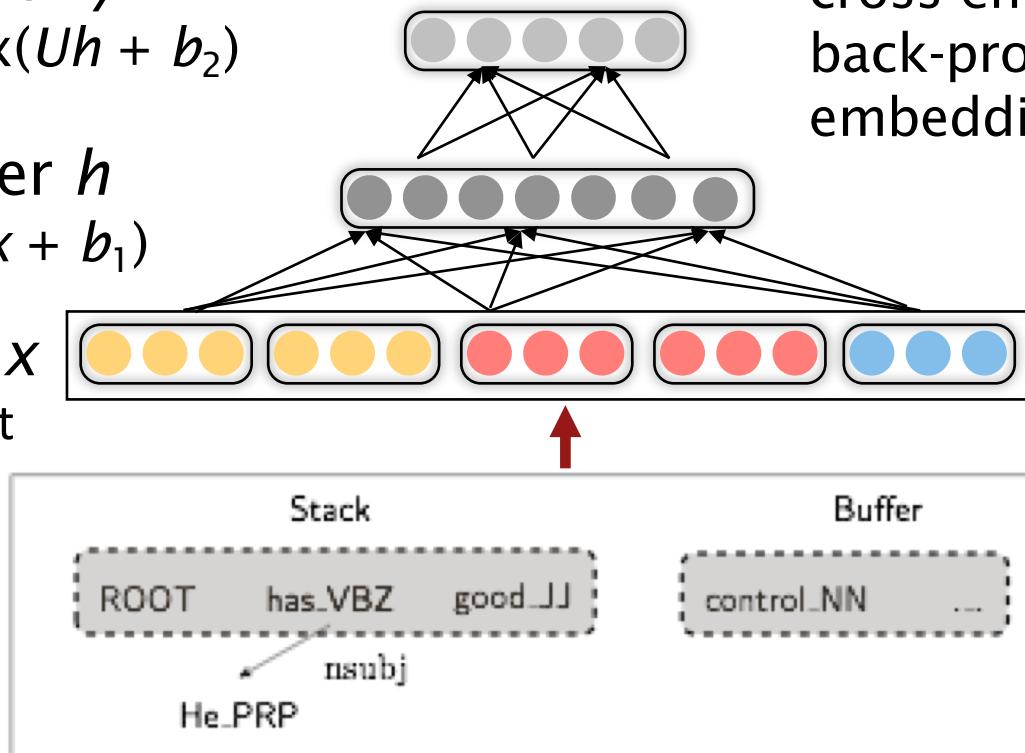
Hidden layer h

$$h = \text{ReLU}(Wx + b_1)$$

Input layer x

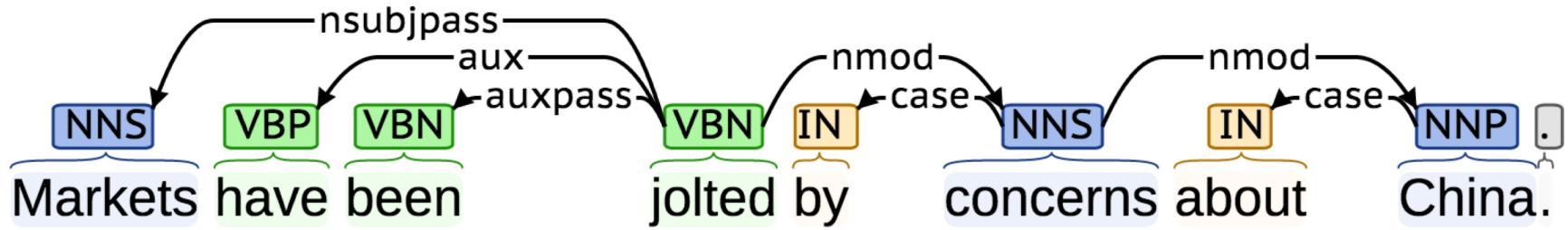
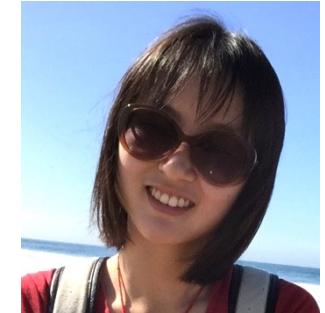
lookup + concat

cross-entropy error will be back-propagated to the embeddings.



Dependency parsing for sentence structure

Neural networks can accurately determine the structure of sentences, supporting interpretation



Chen and Manning (2014) was the first simple, successful neural dependency parser

The dense representations let it outperform other greedy parsers in both accuracy and speed

Further developments in transition-based neural dependency parsing

This work was further developed and improved by others, including in particular at Google

- Bigger, deeper networks with better tuned hyperparameters
- Beam search
- Global, conditional random field (CRF)-style inference over the decision sequence

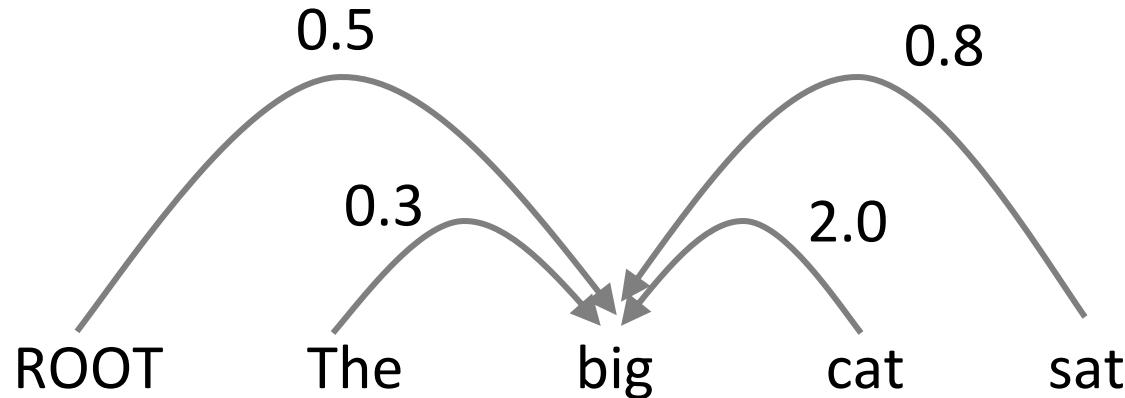
Leading to SyntaxNet and the Parsey McParseFace model

<https://research.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html>

Method	UAS	LAS (PTB WSJ SD 3.3)
Chen & Manning 2014	92.0	89.7
Weiss et al. 2015	93.99	92.05
Andor et al. 2016	94.61	92.79

Graph-based dependency parsers

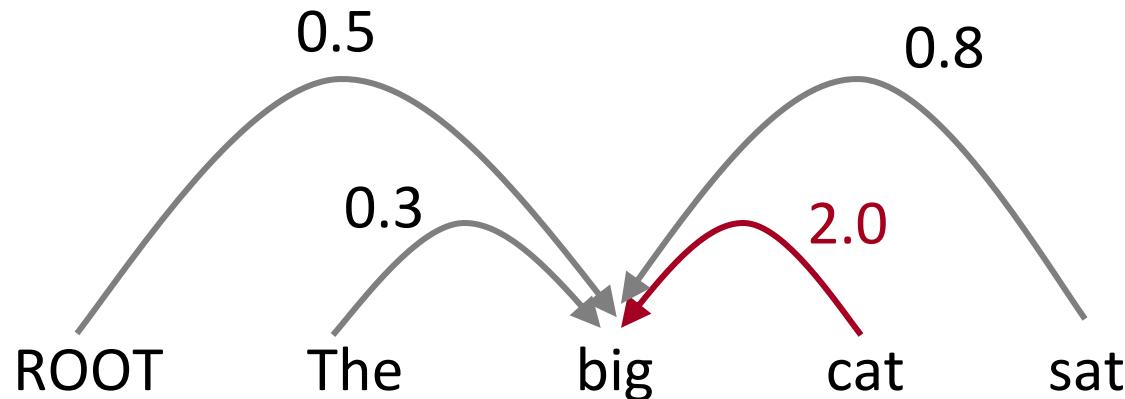
- Compute a score for every possible dependency for each edge



e.g., picking the head for “big”

Graph-based dependency parsers

- Compute a score for every possible dependency for each edge
 - Then add an edge from each word to its highest-scoring candidate head
 - And repeat the same process for each other word



e.g., picking the head for “big”

A Neural graph-based dependency parser

[Dozat and Manning 2017; Dozat, Qi, and Manning 2017]

- Revived graph-based dependency parsing in a neural world
 - Design a biaffine scoring model for neural dependency parsing
 - Also using a neural sequence model, as we discuss next week
- Really great results!
 - But slower than simple neural transition-based parsers
 - There are n^2 possible dependencies in a sentence of length n

Method	UAS	LAS (PTB WSJ SD 3.3)
Chen & Manning 2014	92.0	89.7
Weiss et al. 2015	93.99	92.05
Andor et al. 2016	94.61	92.79
Dozat & Manning 2017	95.74	94.08