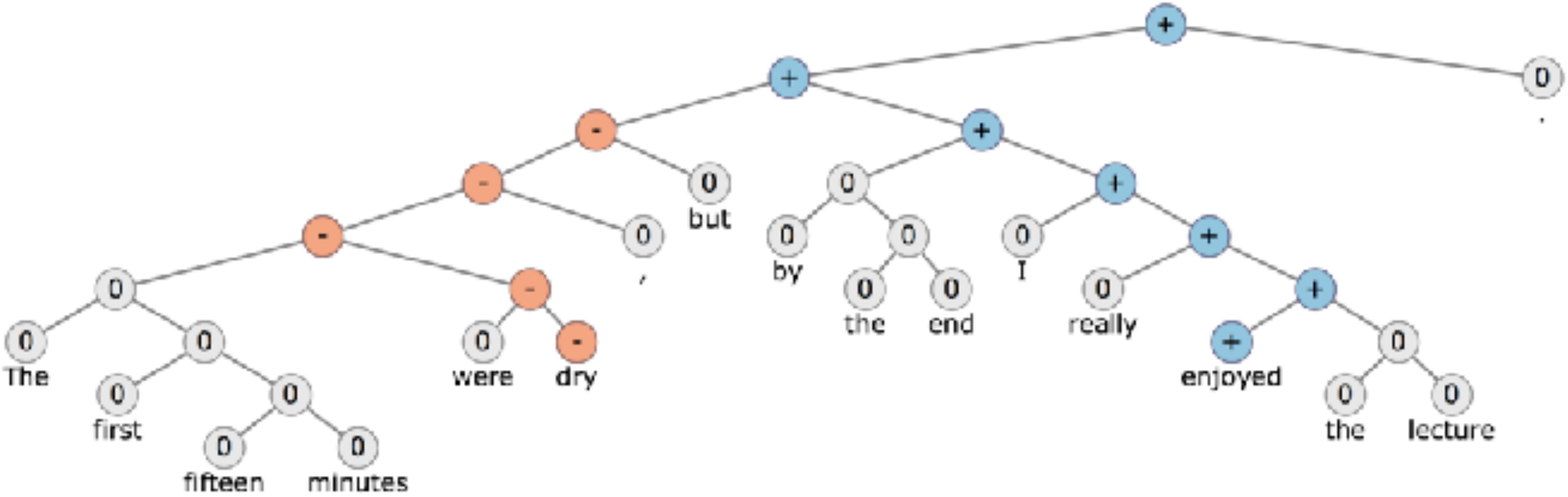
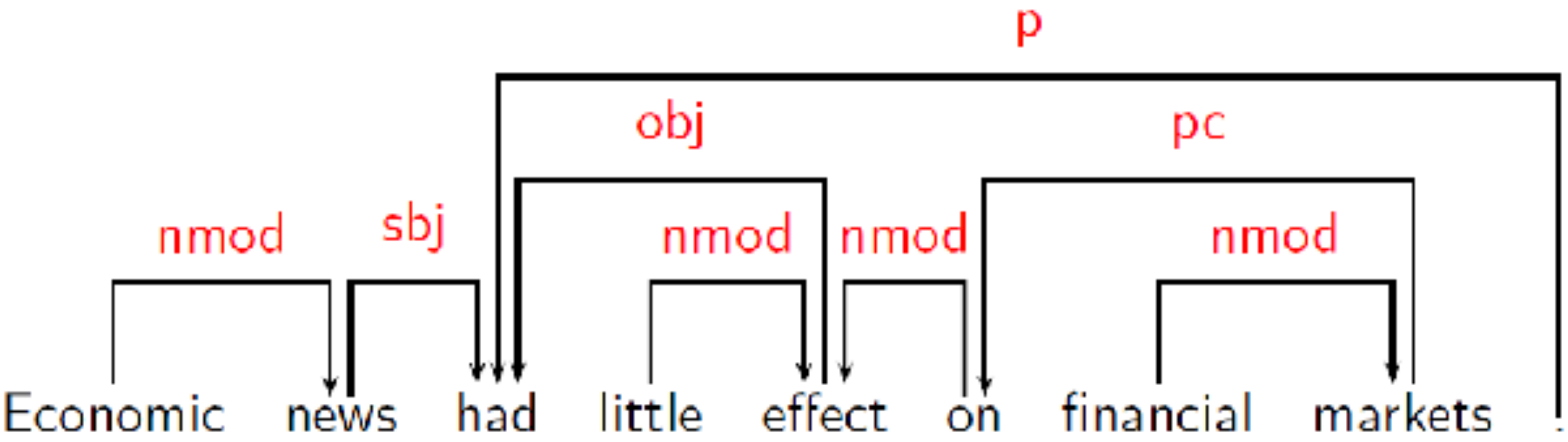
# Dependency Parsing



# Parsing

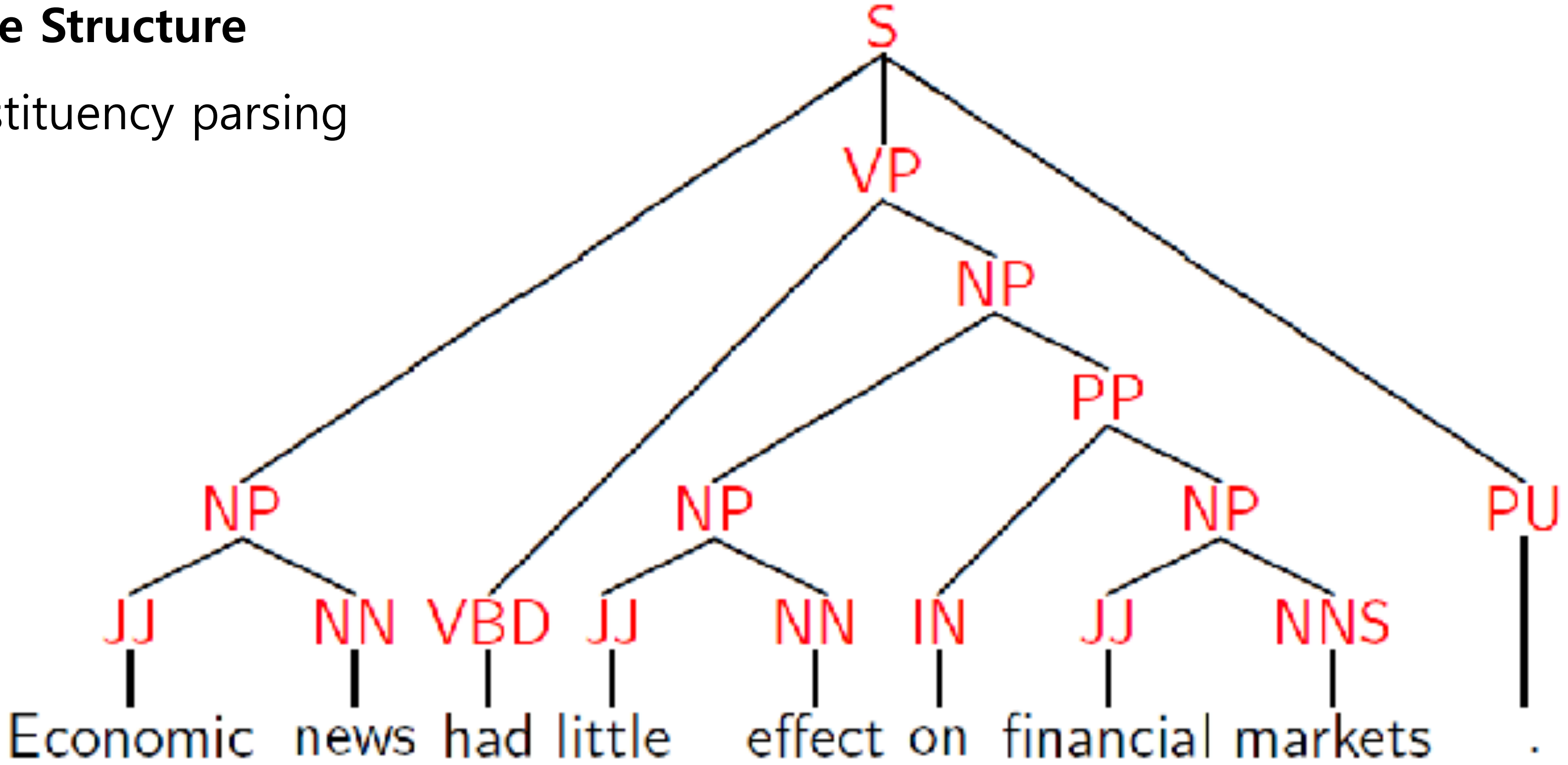
- **Parsing syntax analysis** or **syntactic analysis** is the process of analysing a string of symbols, either in natural language, computer languages or data structures, conforming to the rules of a formal grammar. The term parsing comes from Latin pars (orationis), meaning part (of speech).

## Dependency Structure



## Phrase Structure

constituency parsing



## Lecture Plan

1. Syntactic Structure: Constituency and Dependency
2. Dependency Grammar
3. Transition-based dependency parsing
4. Neural dependency parsing

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

Phrase structure organizes words into nested constituents.

**Basic 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

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

Phrase structure organizes words into nested constituents.

Can represent the grammar with CFG rules

**Basic unit: words**

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

**Words combine into phrases**

the cuddly cat, by the door

NP -> Det Adj N PP -> P NP

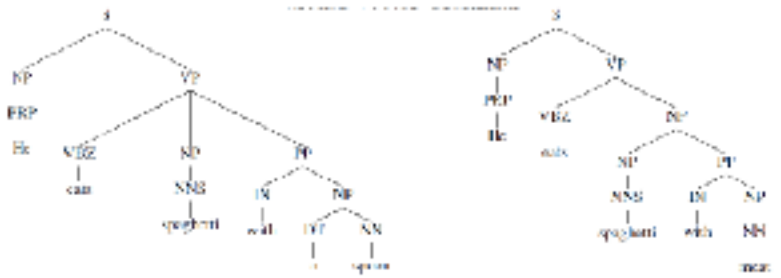
**Phrases can combine into bigger phrases**

the cuddly cat by the door

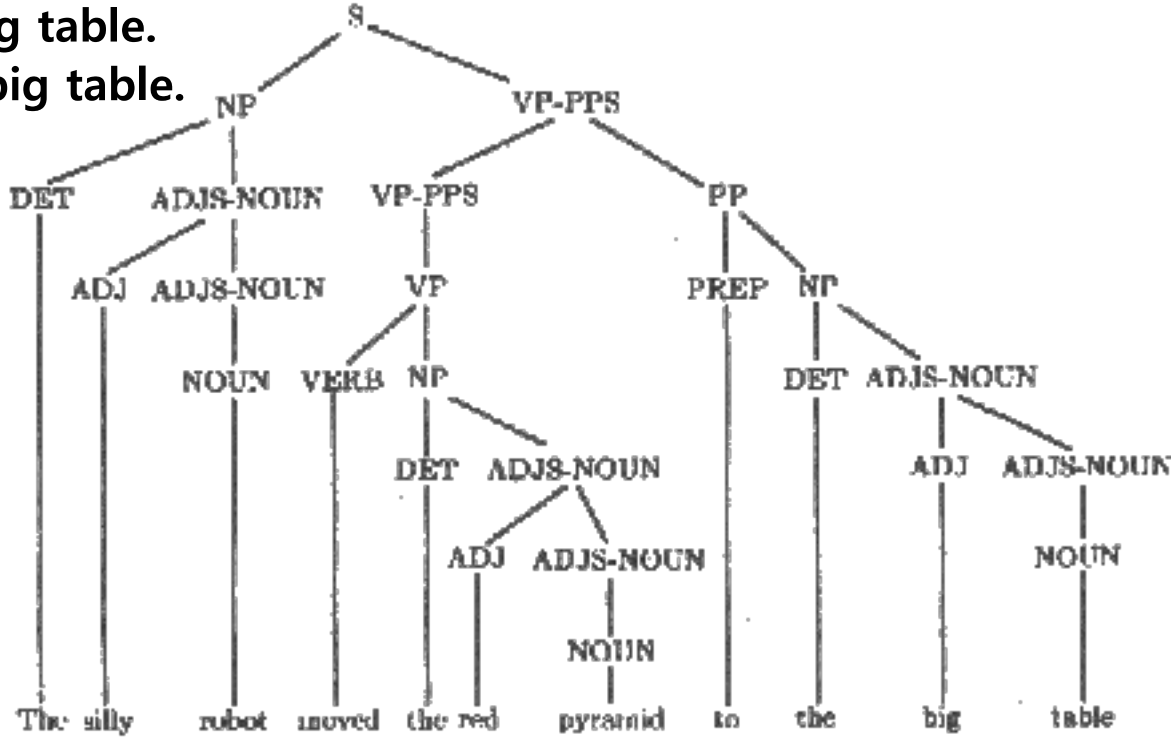
NP -> NP PP

## Example Constituency Trees

- PP attachment ambiguities in constituency structure



The silly robot moved the red pyramid to the big table.
DET silly robot moved the red pyramid to the big table.
DET ADJ robot moved the red pyramid to the big table.
DET ADJ NOUN moved the red pyramid to the big table.
DET ADJS-NOUN moved the red pyramid to the big table.
NP moved the red pyramid to the big table.
NP VERB the red pyramid to the big table.
NP VERB DET red pyramid to the big table.
NP VERB DET ADJ pyramid to the big table.
NP VERB DET ADJ NOUN to the big table.
NP VERB DET ADJS-NOUN to the big table.
NP VERB NP to the big table.
NP VP to the big table.
NP VP-PPS to the big table.
NP VP-PPS PREP the big table.
NP VP-PPS PREP DET big table.
NP VP-PPS PREP DET ADJ table.
NP VP-PPS PREP DET ADJS-NOUN
NP VP-PPS PREP NP.
NP VP-PPS PP.
NP VP-PPS.
S.



http://www.aistudy.com/ai/language_winston.htm

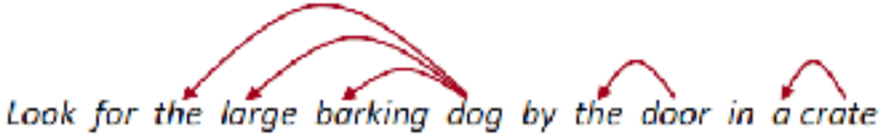
## Two views of linguistic structure: Dependency structure

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

*Look for the large barking dog by the door in a crate*

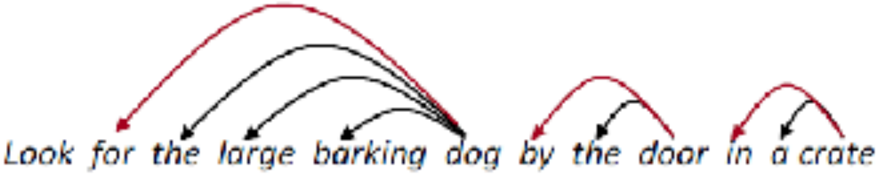
## Two views of linguistic structure: Dependency structure

- Dependency structure shows which words depend on (modify or are arguments of) which other words.
  - Determiners, adjectives, and (sometimes) verbs modify nouns


*Look for the large barking dog by the door in a crate*

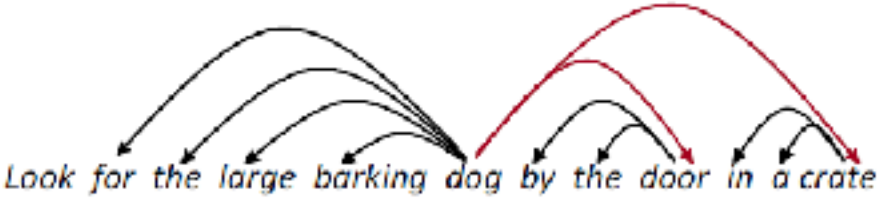
## Two views of linguistic structure: Dependency structure

- Dependency structure shows which words depend on (modify or are arguments of) which other words.
  - Determiners, adjectives, and (sometimes) verbs modify nouns
  - We will also treat prepositions as modifying nouns


*Look for the large barking dog by the door in a crate*

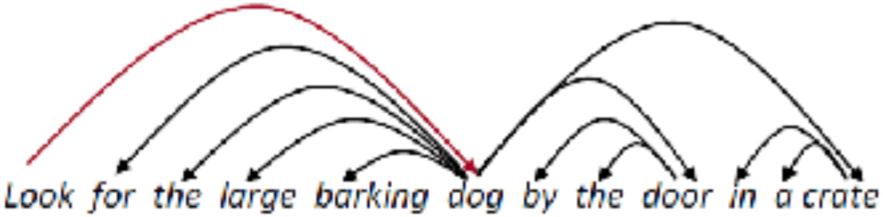
## Two views of linguistic structure: Dependency structure

- Dependency structure shows which words depend on (modify or are arguments of) which other words.
  - Determiners, adjectives, and (sometimes) verbs modify nouns
  - We will also treat prepositions as modifying nouns
  - The prepositional phrases are modifying the main noun phrase


*Look for the large barking dog by the door in a crate*

## Two views of linguistic structure: Dependency structure

- Dependency structure shows which words depend on (modify or are arguments of) which other words.
  - Determiners, adjectives, and (sometimes) verbs modify nouns
  - We will also treat prepositions as modifying nouns
  - The prepositional phrases are modifying the main noun phrase
  - The main noun phrase is an argument of "look"

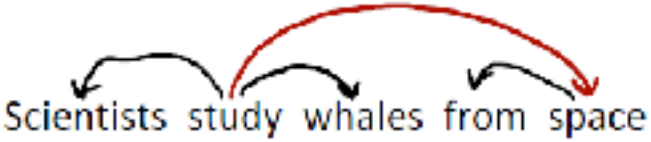


*Look for the large barking dog by the door in a crate*

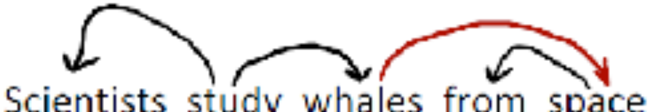
## Ambiguity: PP attachments

Scientists study whales from space

## PP attachment ambiguities in dependency structure



Scientists study whales from space



Scientists study whales from space

## Attachment ambiguities

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

The board approved [its acquisition] [by Royal Trustco Ltd.]

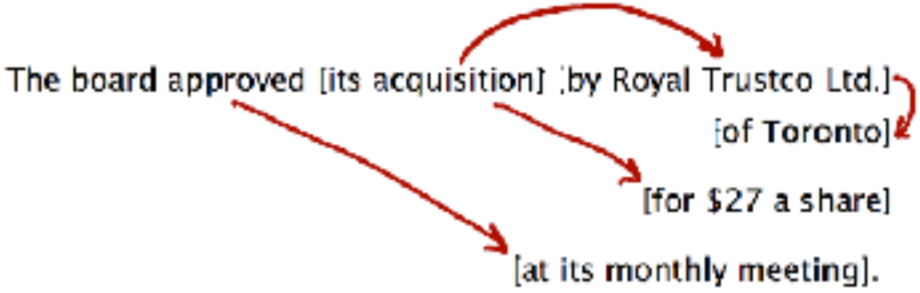
[of Toronto]

[for $27 a share]

[at its monthly meeting].

## Attachment ambiguities

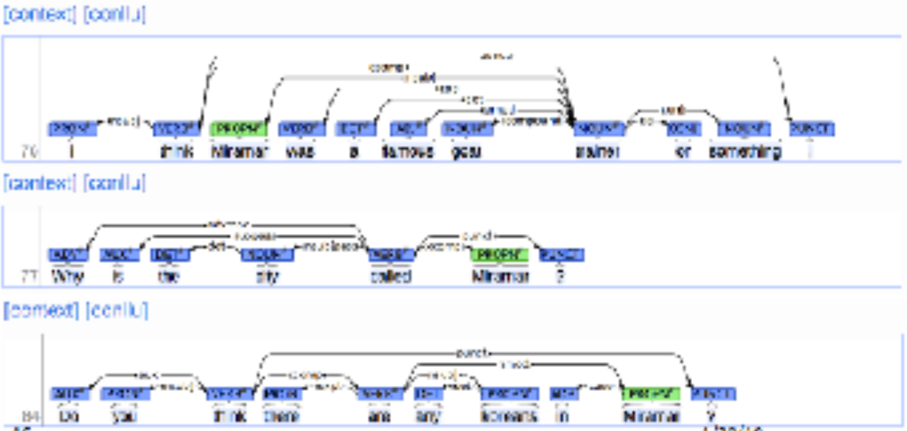
- A key parsing decision is how we 'attach' various constituents
  - PPs, adverbial or participial phrases, infinitives, coordinations,


The board approved [its acquisition] [by Royal Trustco Ltd.]
[of Toronto]
[for $27 a share]
[at its monthly meeting].

- Catalan numbers: $C_n = (2n)!/[(n+1)!n!]$
- An exponentially growing series, which arises in many tree-like contexts
- But normally, we assume nesting

## 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]
I think Miramar was a famous goat trainer or something
[context] [conllu]
Why is the city called Miramar ?
[context] [conllu]
Do you think there are any koreans in Miramar ?

## The rise of annotated data

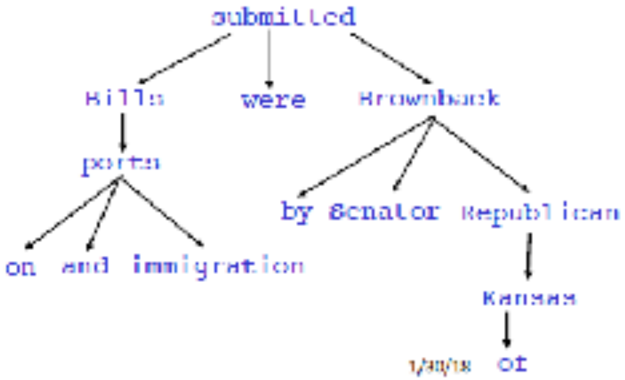
Starting off, building a treebank seems a lot slower and less useful than building a grammar

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 Grammar and Dependency Structure

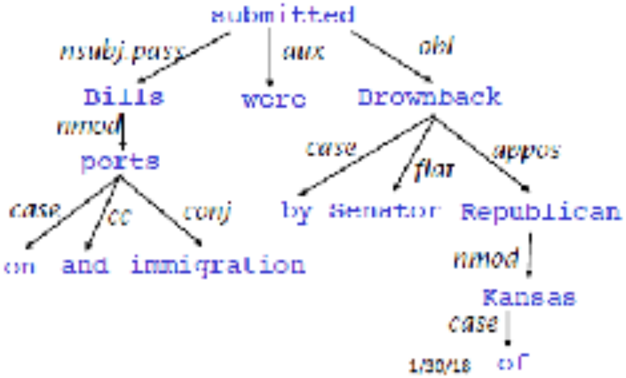
Dependency syntax postulates that syntactic structure consists of relations between lexical items, normally binary asymmetric relations ("arrows") called dependencies



# 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.)

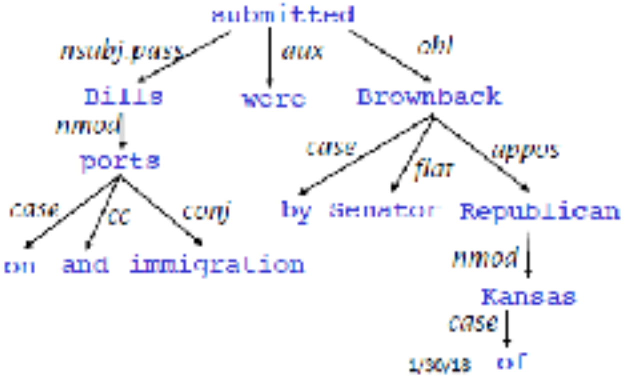


# Dependency Grammar and Dependency Structure

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)



# Dependency Relations

| Clausal Argument Relations | Description |
|---|---|
| NSUBJ | Nominal subject |
| DOBJ | Direct object |
| IOBJ | Indirect object |
| CCOMP | Clausal complement |
| XCOMP | Open clausal complement |
| **Nominal Modifier Relations** | **Description** |
| NMOD | Nominal modifier |
| AMOD | Adjectival modifier |
| NUMMOD | Numeric modifier |
| APPOS | Appositional modifier |
| DET | Determiner |
| CASE | Prepositions, postpositions and other case markers |
| **Other Notable Relations** | **Description** |
| CONJ | Conjunct |
| CC | Coordinating conjunction |

Selected dependency relations from the Universal Dependency set. (de Marneffe et al., 2014)
https://web.stanford.edu/~jurafsky/slp3/14.pdf

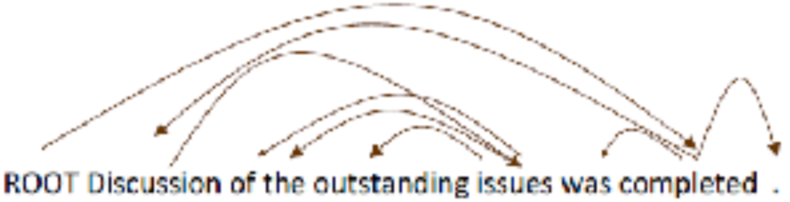
# Pāṇini's grammar (c. 5th century BCE)

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 more recent invention
  - 20th century (R.S. Wells, 1947)
- Modern dependency work often linked to work of L. Tesnière (1959)
  - Was dominant approach in "East" (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)

## Dependency Grammar and Dependency Structure



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

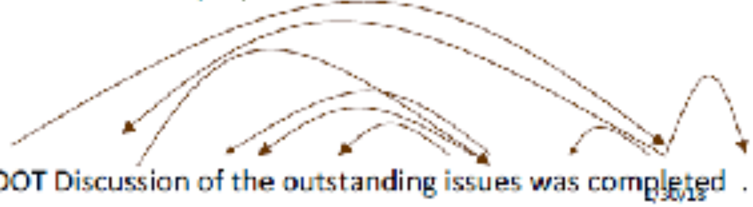
## Dependency Conditioning Preferences

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

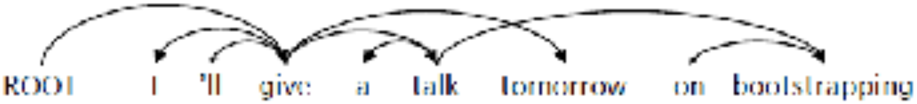
   Dependencies rarely span intervening verbs or punctuation
4. Valency of heads

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

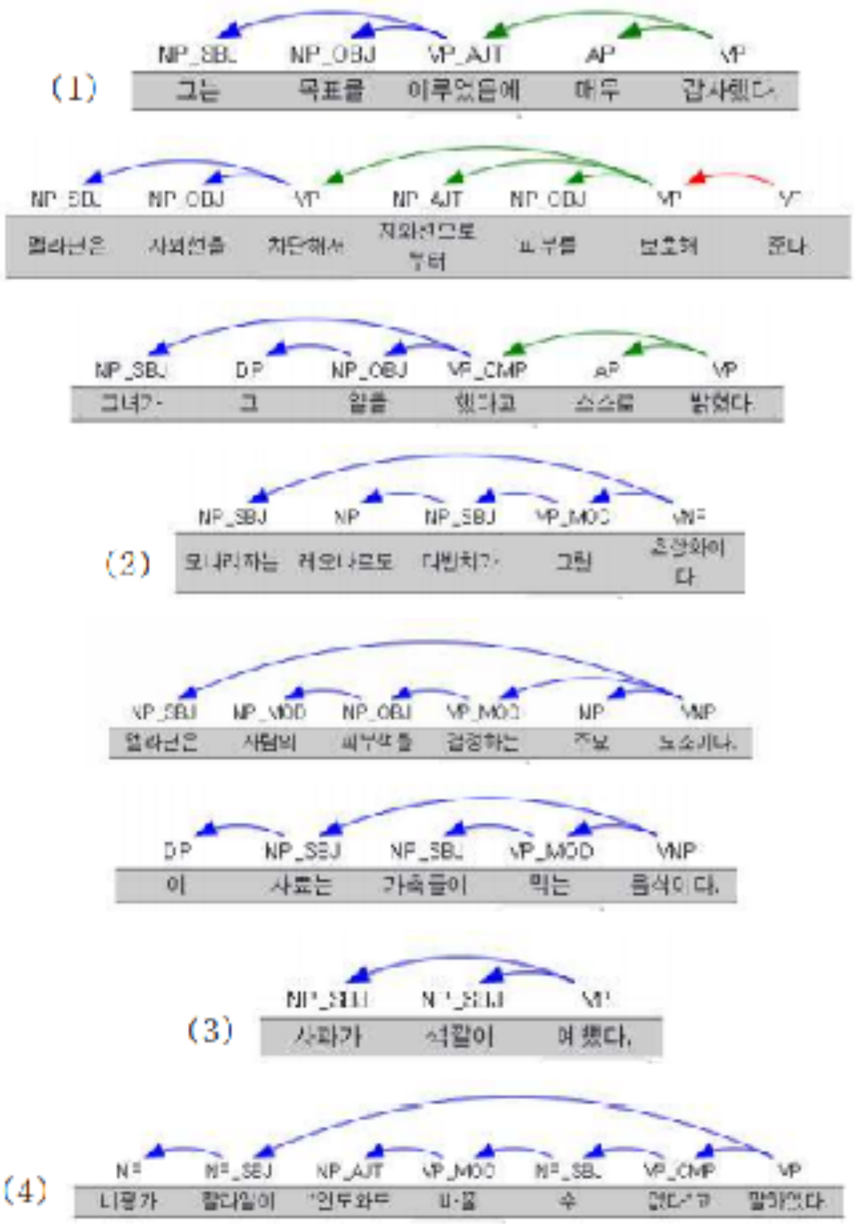


## Dependency Parsing

- A sentence is parsed by choosing for each word what other word (including ROOT) it is a dependent of
  - i.e., find the right outgoing arrow from each word
- Usually some constraints:
  - Only one word is a dependent of ROOT
  - Don't want cycles A → B, B → A
- This makes the dependencies a tree
- Final issue is whether arrows can cross (non-projective) or not



# 한글 적용 사례


(1)
NP_SBJ NP_OBJ VP_AJT AP VP
그는 목표를 이루었음에 매우 감사했다.
NP_SBJ NP_OBJ VP NP_AJT NP_OBJ VP VP
멜라닌은 자외선을 차단해서 자외선으로부터 피부를 보호해 준다.
NP_SBJ DP NP_OBJ VP_CMP AP VP
그녀가 그 일을 했다고 스스로 밝혔다
(2)
NP_SBJ NP NP_SBJ VP_MOD VNP
모나리자는 레오나르도 다빈치가 그린 초상화이다
NP_SBJ NP_MOD NP_OBJ VP_MOD NP VNP
멜라닌은 사람의 피부색을 결정하는 주요 요소이다.
DP NP_SBJ NP_SBJ VP_MOD VNP
이 사료는 가축들이 먹는 음식이다.
(3)
NP_SBJ NP_SBJ VP
사과가 색깔이 예뻤다.
(4)
NP NP_SBJ NP_AJT VP_MOD NP_SBJ VP_CMP VP
수 말하였다

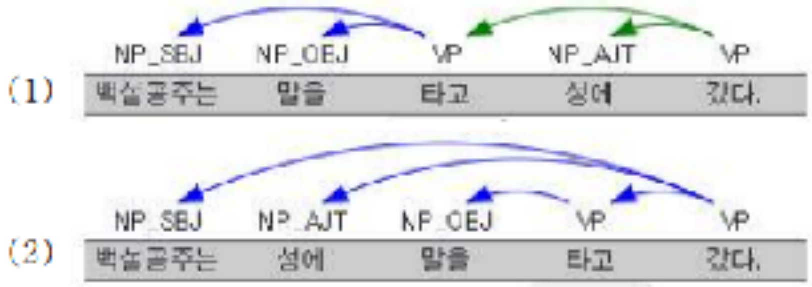

(1)
NP_SBJ NP_OBJ VP NP_AJT VP
백설공주는 말을 타고 섬에 갔다.
(2)
NP_SBJ NP_AJT NP_OBJ VP VP
백설공주는 섬에 말을 타고 갔다.

그림 1. 한국어 다중 지배소 현상(1- '백설공주는' 의 지배소는 '타고' 와 '갔다.' 이다.)과 교차 발생 현상 (2- '섬에' 가 '갔다' 에 의존하여 '백설공주는' 이 '타고' 에 의존하지 못한다.)

(1) 자연언어처리를 위한 일관성 유지와 효율성 제고에 초점을 두되, 일반 언어학적 관점에서도 크게 벗어나지 않도록 한다.
(2) 문장의 표층 구조를 중시하여 분석한다.
(3) 의존관계 분석의 기본 단위로 어절을 사용한다.
(4) 지배소 후위 원칙에 따라 각 어절의 지배소는 자신보다 뒤에 위치하도록 분석한다.
(5) 각 어절은 1개의 지배소를 가진다. (Single-Head Constraint)
(6) 각 어절 및 지배소 쌍은 서로 교차하지 않는다. (Projective Constraint)
(7) 보어와 부가어를 구분하되 보어의 범위를 엄격히 제한한다.
(8) 원칙적으로 접속과 내포를 구별하지 않으며, 접속절은 모두 부사절로 분석한다. (다만, 명사구 접속은 인정한다.)
(9) 하나의 주어가 모문과 내포문 모두에 관련되어 있으면 모문과 내포문의 관계에 따라 해당 주어의 지배소를 결정한다.

## Methods of Dependency Parsing

1. Dynamic programming
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.
4. "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.

## 4. 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

Nivre PDF (56쪽)참조

http://demo.clab.cs.cmu.edu/fa2015-11711/images/b/b1/TbparsingSmallCorrection.pdf

## Basic transition-based dependency parser

Start: $\sigma$ = [ROOT], $\beta = w_1, \ldots, w_n$ , A = ∅

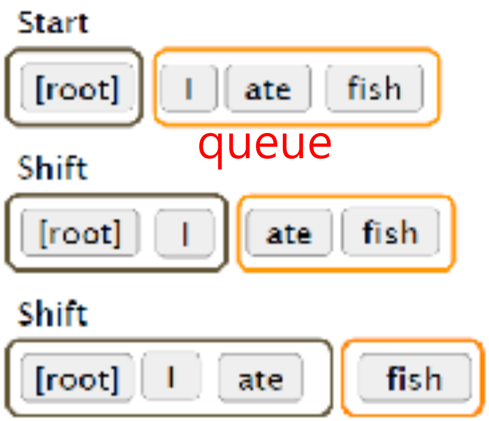
1. Shift $\sigma, w_i|\beta, A \rightarrow \sigma|w_i, \beta, A$
2. Left-Arc$_r$ $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_j, \beta, A \cup \{r(w_j,w_i)\}$
3. Right-Arc$_r$ $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_i, \beta, A \cup \{r(w_i,w_j)\}$

Finish: $\sigma$ = [w], $\beta$ = ∅

## Arc-standard transition-based parser

(there are other transition schemes ...)

Analysis of "I ate fish"



Start

[root] | I ate fish

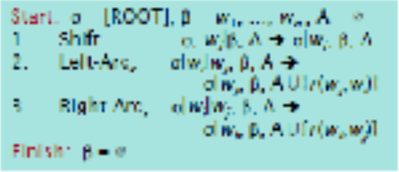
queue

Shift

[root] I | ate fish

Shift

[root] I ate | fish



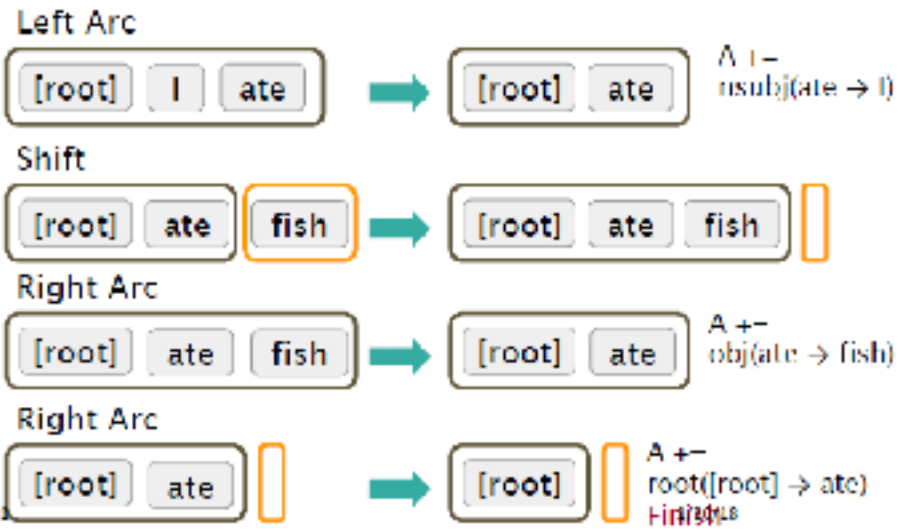
Start: σ = [ROOT], β = $w_1, \ldots, w_n$ , A = ∅

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: β = ∅

## Arc-standard transition-based parser

Analysis of "I ate fish"



Left Arc

[root] I ate → [root] ate

A +− nsubj(ate → I)

Shift

[root] ate fish → [root] ate fish

Right Arc

[root] ate fish → [root] ate

A +− obj(ate → fish)

Right Arc

[root] ate → [root]

A +− root([root] → ate)

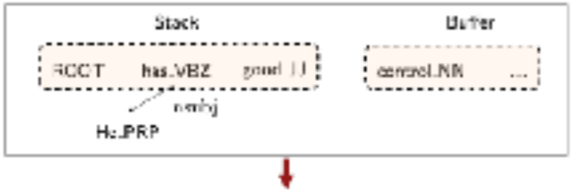
Finish

## MaltParser

[Nivre and Hall 2005]

- How could we choose the next action?
- Each action is predicted by a discriminative classifier (eg. SVM or logistic regression classifier) over each legal move
  - 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)
- It provides VERY fast linear time parsing
- The model's accuracy is only *slightly* below the best dependency parsers

## Feature Representation



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

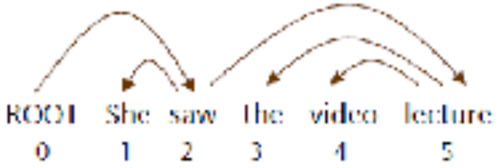
|0 0 0 |1 0 0 |1 0 |... 0 |0 |1 0 |

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

Indicator features

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

## Evaluation of Dependency Parsing: (labeled) dependency accuracy



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

UAS = 4 / 5 = 80%

LAS = 2 / 5 = 40%

Gold

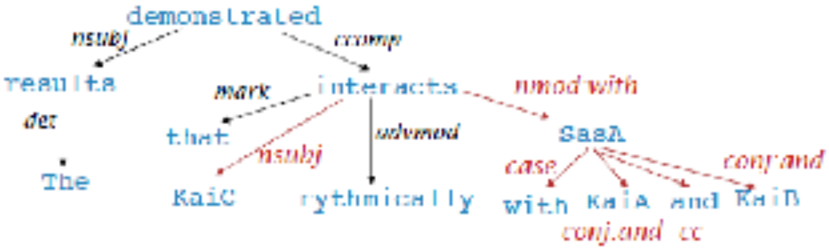
| | | | |
|---|---|---|---|
| 1 | 2 | She | nsubj |
| 2 | 0 | saw | root |
| 3 | 5 | the | det |
| 4 | 5 | video | nn |
| 5 | 2 | lecture | obj |

Parsed

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

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

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



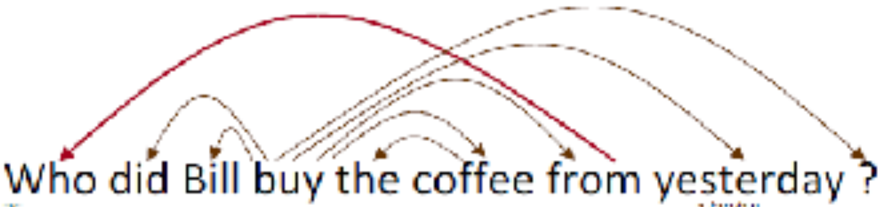
KaiC ←nsubj interacts nmod:with → SasA

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

KaiC ←nsubj interacts prep_with→ SasA conj:and→ KaiB

# Projectivity

- Dependencies parallel to a CFG tree must be projective
  - There must not be any crossing dependency arcs when the words are laid out in their linear order, with all arcs above the words.
- 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



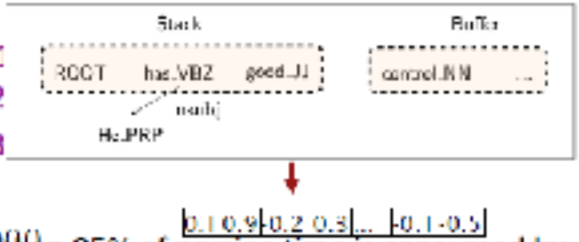
Who did Bill buy the coffee from yesterday ?

# Handling non-projectivity

- The arc-standard algorithm we presented only builds projective dependency trees
- Possible directions:
  1. Just declare defeat on nonprojective arcs
  2. Use a dependency formalism which only admits projective representations (a CFG doesn't represent such structures...)
  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)

# Why train a neural dependency parser? Indicator Features Revisited

- Problem #1
- Problem #2
- Problem #3



dense

dim ≈ 1000

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

Our Approach:

learn a dense and compact feature representation

$s1.w = good \wedge s1.t = JJ$

$s2.w = has \wedge s2.t = VBZ \wedge s1.w = good$

$lc(s_2).t = PRP \wedge s_2.t = VBZ \wedge s_1.t = JJ$

$lc(s_2).w = He \wedge lc(s_2).l = nsubj \wedge s_2.w = has$

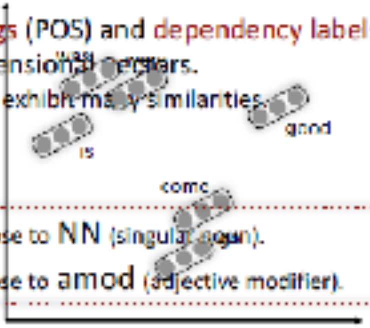
## A neural dependency parser
[Chen and Manning 2014]

- 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** |

## 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 similarities.

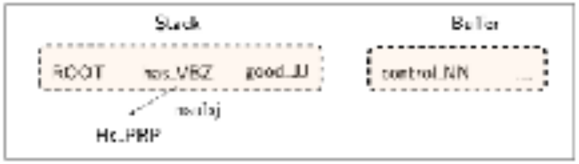


NNS (plural noun) should be close to NN (singular noun).

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:

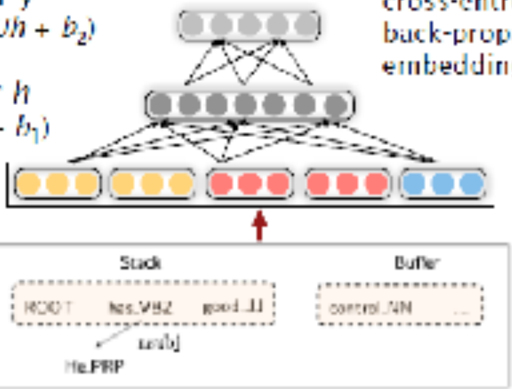


| | word | POS | dep. |
|---|---|---|---|
| $s_1$ | good | JJ | ∅ |
| $s_2$ | has | VBZ | ∅ |
| $b_1$ | control | NN | ∅ |
| $lc(s_1)$ | ∅ | ∅ | ∅ |
| $rc(s_1)$ | ∅ | ∅ | ∅ |
| $lc(s_2)$ | He | PRP | nsubj |
| $rc(s_2)$ | ∅ | ∅ | ∅ |

- We convert them to vector embeddings and concatenate them

## Model Architecture

Softmax probabilities



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

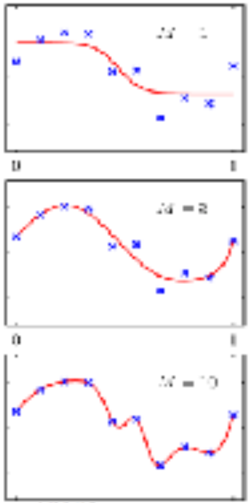
Hidden layer $h$
$h = \text{ReLU}(Wx + b_1)$

Input layer $x$
lookup + concat

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

# Non-linearities between layers: Why they're needed

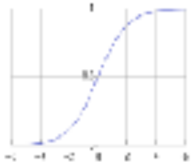
- For logistic regression: map to probabilities
- Here: function approximation, e.g., for regression or classification
  - Without non-linearities, deep neural networks can't do anything more than a linear transform
    - Extra layers could just be compiled down into a single linear transform
- People use various non-linearities



# Non-linearities: sigmoid and tanh

logistic ("sigmoid")

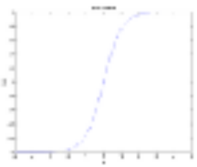
$$f(z) = \frac{1}{1+\exp(-z)}.$$



$$f'(z) = f(z)(1-f(z))$$

tanh

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}},$$



$$f'(z) = 1 - f(z)^2$$

tanh is just a rescaled and shifted sigmoid $\tanh(z) = 2\text{logistic}(2z) - 1$

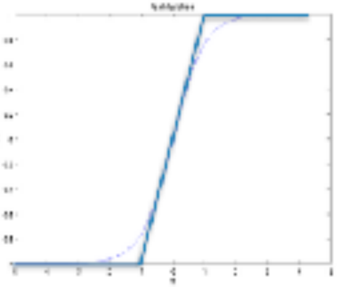
tanh is often used and often performs better for deep nets

- It's output is symmetric around 0

# Non-linearities: hard tanh

- Faster to compute than tanh (no exps or division)
- But suffers from "dead neurons"
  - If our model is initialized such that a neuron is always 1, it will never change!
  - "Saturated neurons" can also be a problem for regular tanh – initializing NNs right is really important!

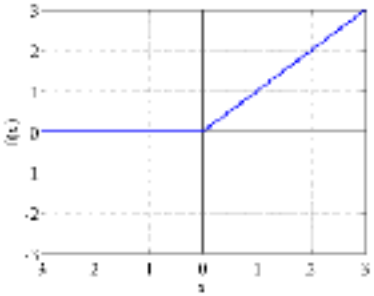
$$\text{HardTanh}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 <= x <= 1 \\ 1 & \text{if } x > 1 \end{cases}$$



# Non-linearities: ReLU

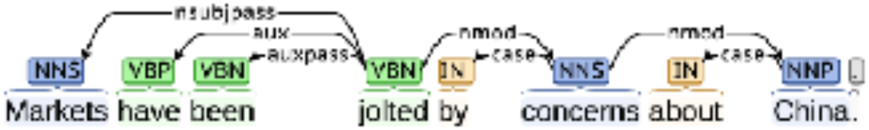
- Also fast to compute, but also can cause dead neurons
- Mega common: "go-to" activation function
- Transfers a linear activation when active
- Lots of variants: LReLU, SELU, ELU, PReLU...

$$\text{rect}(z) = \max(z, 0)$$



## 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

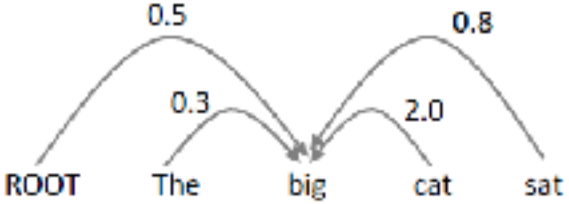
Leading to SyntaxNet and the Parsey McParseFace model

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

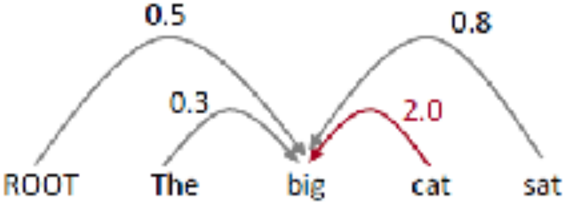
- Compute a score for every possible dependency
  - Then add an edge from each word to its highest-scoring candidate head



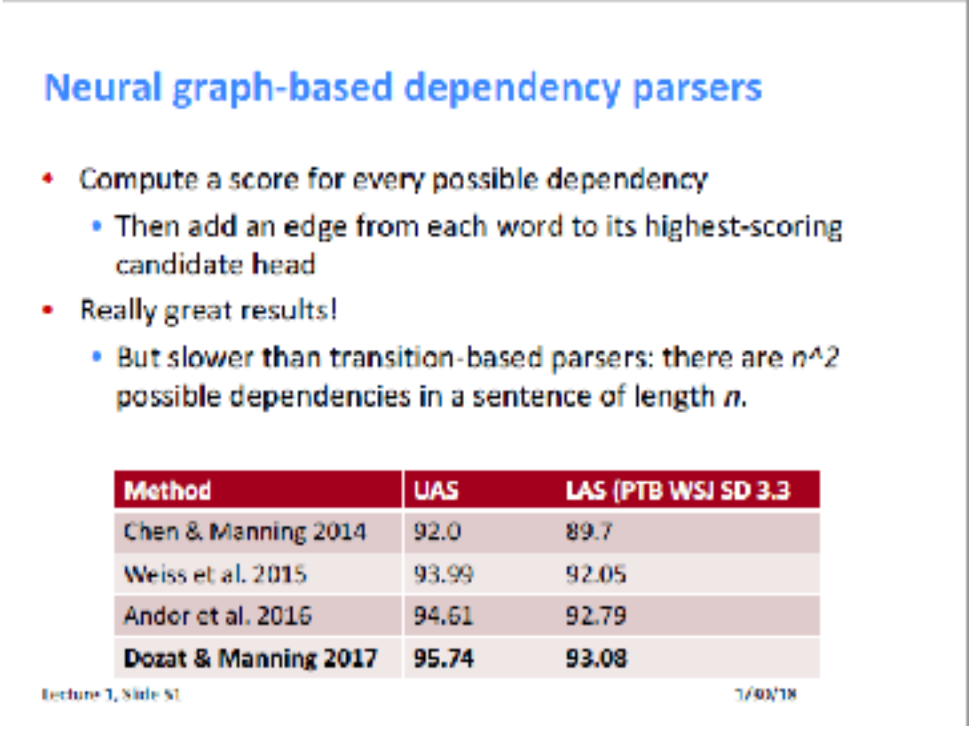
e.g., picking the head for “big”

## Graph-based dependency parsers

- Compute a score for every possible dependency
  - Then add an edge from each word to its highest-scoring candidate head



e.g., picking the head for “big”

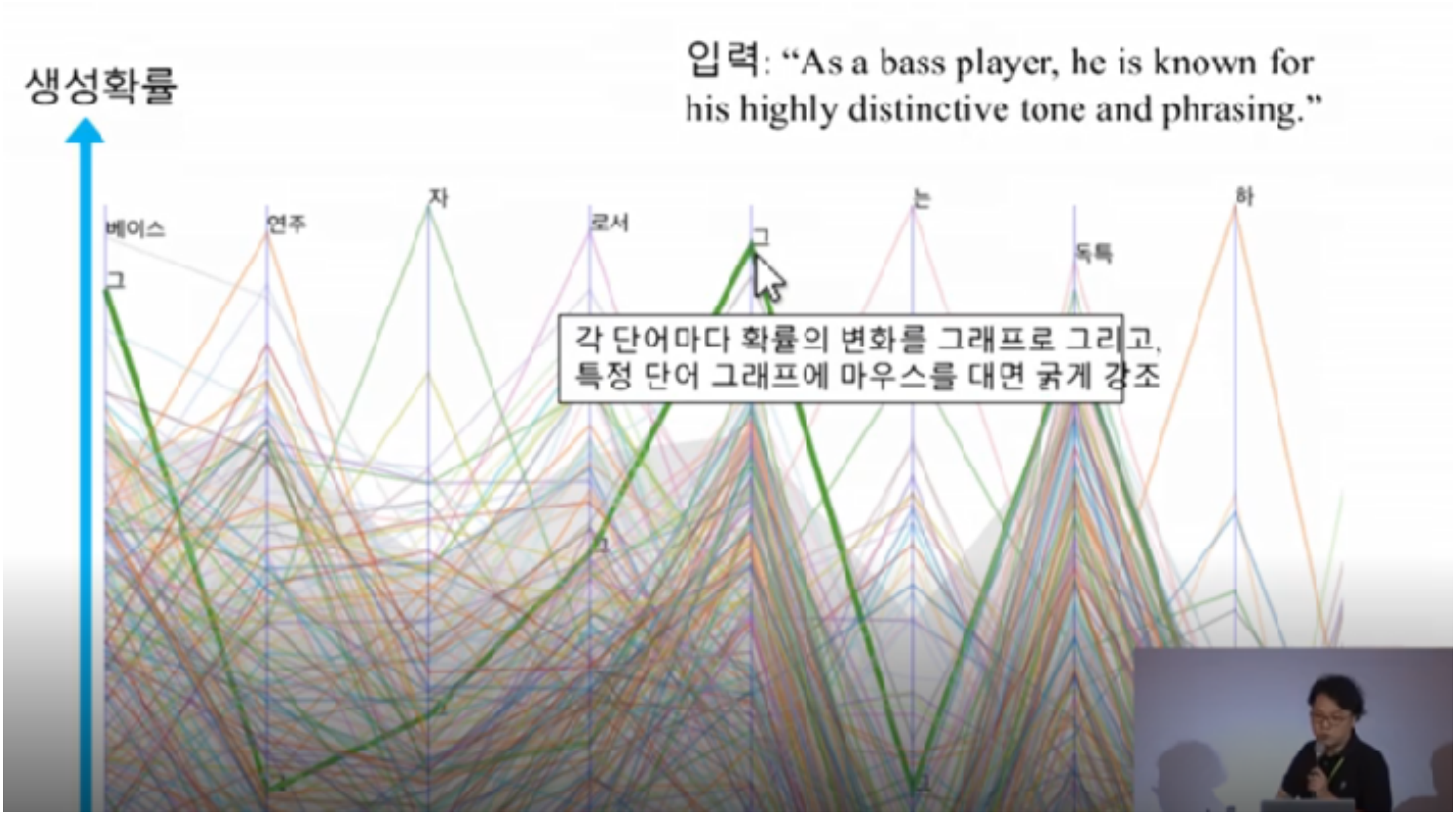


## Neural graph-based dependency parsers

- Compute a score for every possible dependency
  - Then add an edge from each word to its highest-scoring candidate head
- Really great results!
  - But slower than 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** | **93.08** |

# Day2, 2-2. Neural Machine Translation (NMT) 동작의 시각화 및 분석 방법



입력: "As a bass player, he is known for his highly distinctive tone and phrasing."

각 단어마다 확률의 변화를 그래프로 그리고,
특정 단어 그래프에 마우스를 대면 굵게 강조

https://tv.naver.com/v/2302940