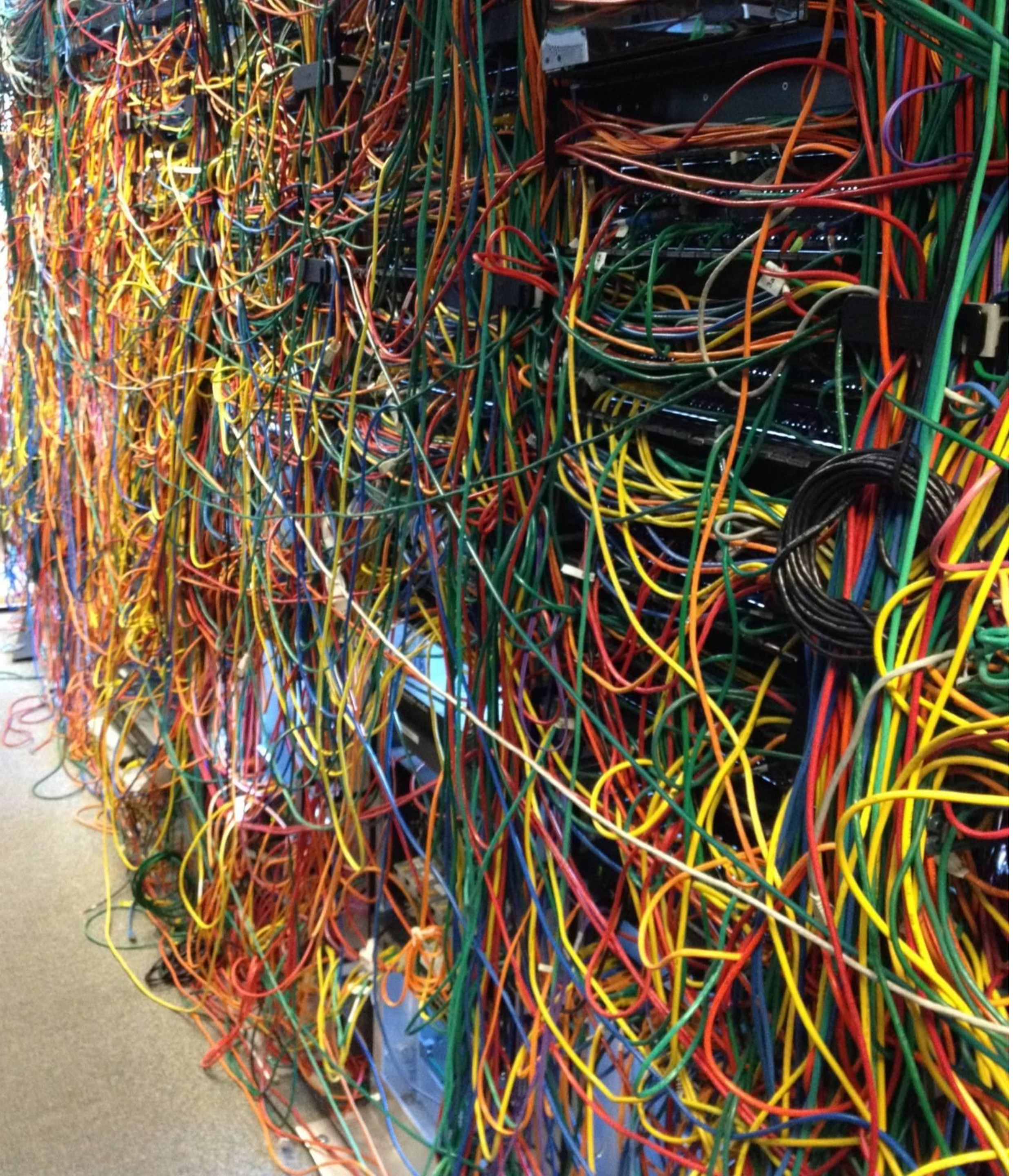


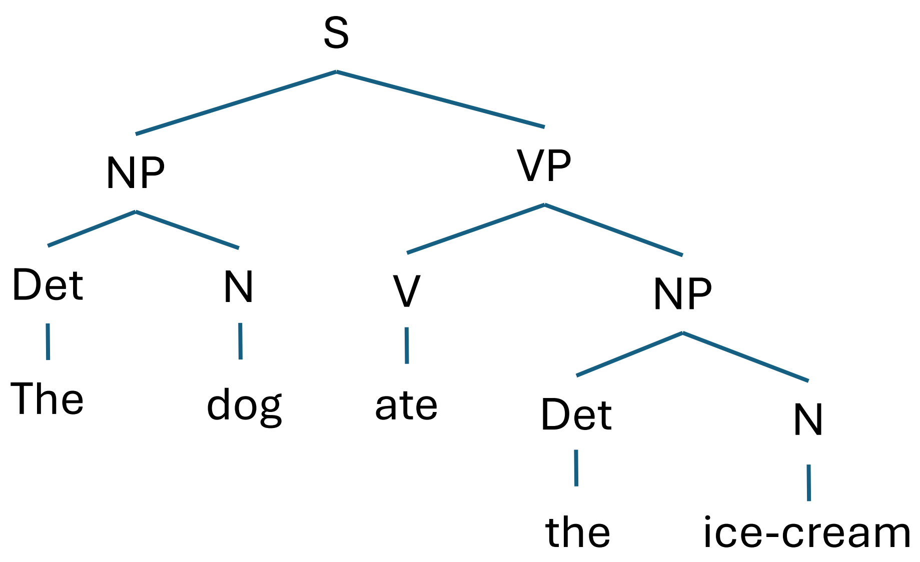
Dependency Parsing

CSC485/2501

Lecture 4

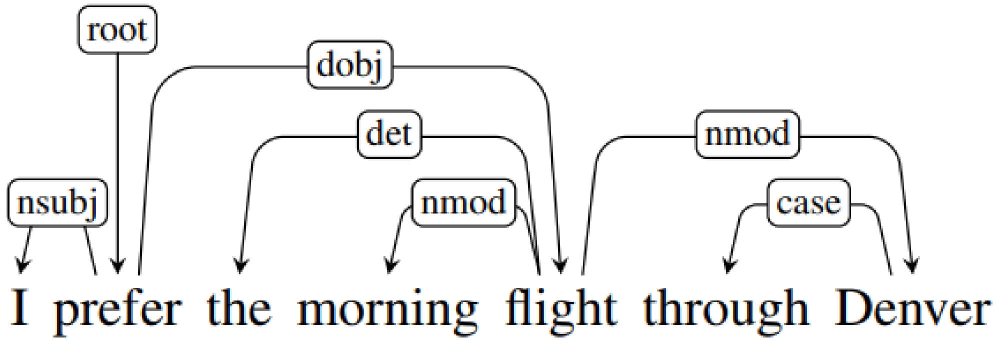
Based on slides by Roger Levy, Yuji Matsumoto, Dragomir Radev, Dan Roth, David Smith and Jason Eisner

Last lecture



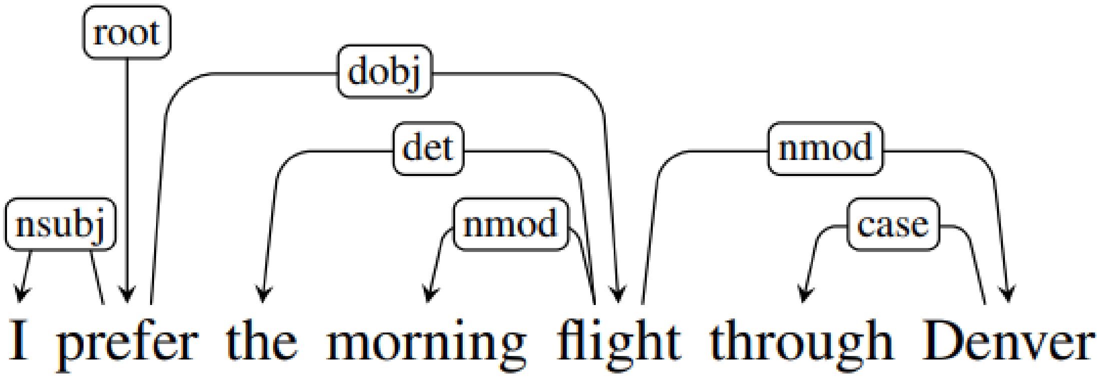
Constituent parsing:
- How words group together.
- Trees.

Today



Dependency parsing:
- Binary grammatical relations between the words.
- Graphs.

Dependency Grammar



- Arc: from **head** to **dependent**.
- Label on arc: **grammatical function**.
  - [Universal Dependencies (UD)]
- Dependency vs constituent:
  - More focus on semantics
  - Free word order

Relation	Description	Examples: head -> dep
nsubj	Nominal subject	United canceled the flight.
obj	Direct object	United diverted the flight to Reno.
iobj	Indirect object	We booked her the flight to Miami.
ccomp	Clausal complement	We took the morning flight.
nmod	Nominal modifier	flight to Houston.
amod	Adjectival modifier	Book the cheapest flight
appos	Appositional modifier	United, a unit of UAL, matched the fares.
det	Determiner	The flight was canceled.
…	…	…

Word Dependency Parsing

Raw sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.

Part-of-speech tagging

POS-tagged sentence

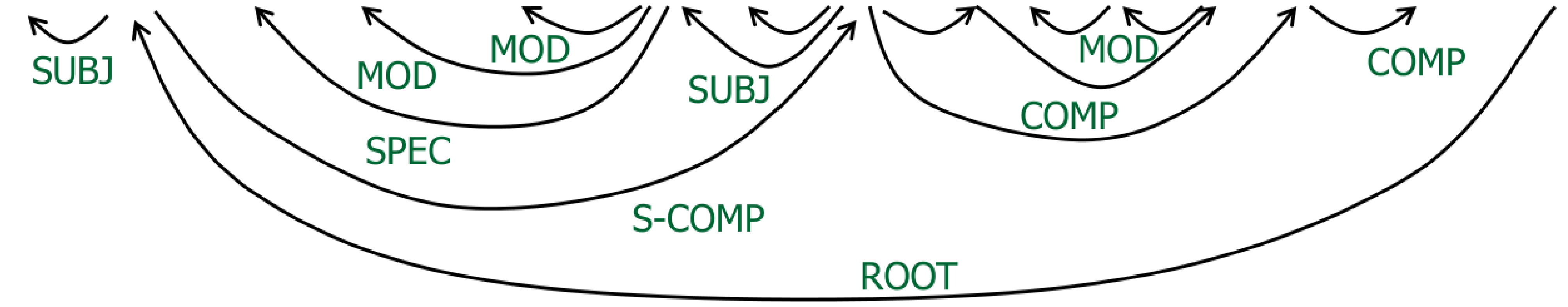
He reckons the current account deficit will narrow to only 1.8 billion in September.

PRP VBZ DT JJ NN NN MD VB TO RB CD CD IN NNP .

Word dependency parsing

Word dependency parsed sentence

He reckons the current account deficit will narrow to only 1.8 billion in September .



Dependency Graphs

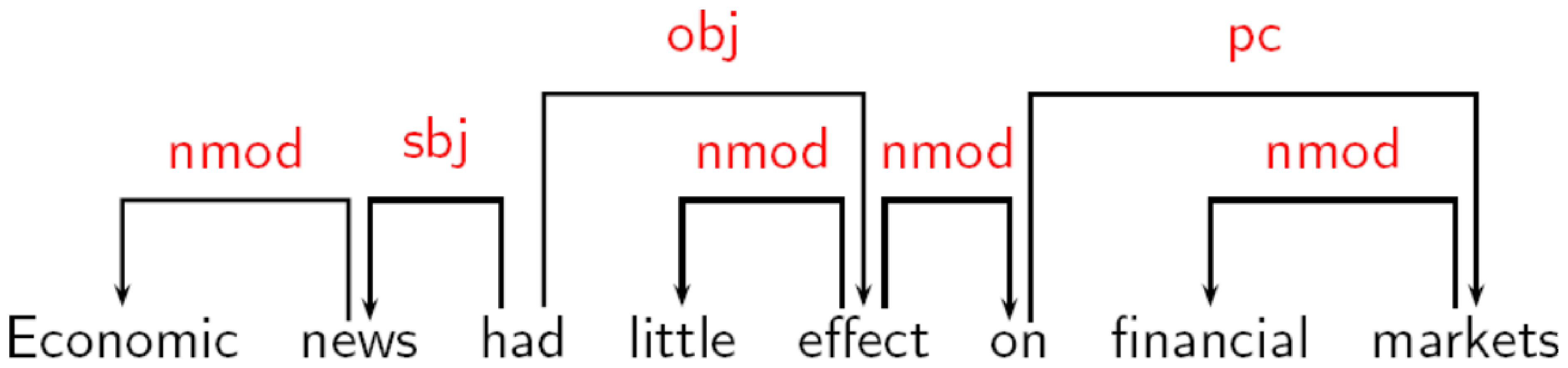
- A dependency structure can be defined as a directed graph G:
  - A set V of nodes,
  - A set E of arcs (edges),
  - A linear precedence order < on V.
- Labelled graphs:
  - Nodes in V are labelled with word forms (and annotation).
  - Arcs in E are labelled with dependency types.
- Notational conventions (i, j ∈ V):
  - $i \rightarrow j \equiv (i, j) \in E$
  - $i \rightarrow^* j \equiv i = j \vee \exists k : i \rightarrow k, k \rightarrow^* j$

Formal Conditions on Dependency Graphs

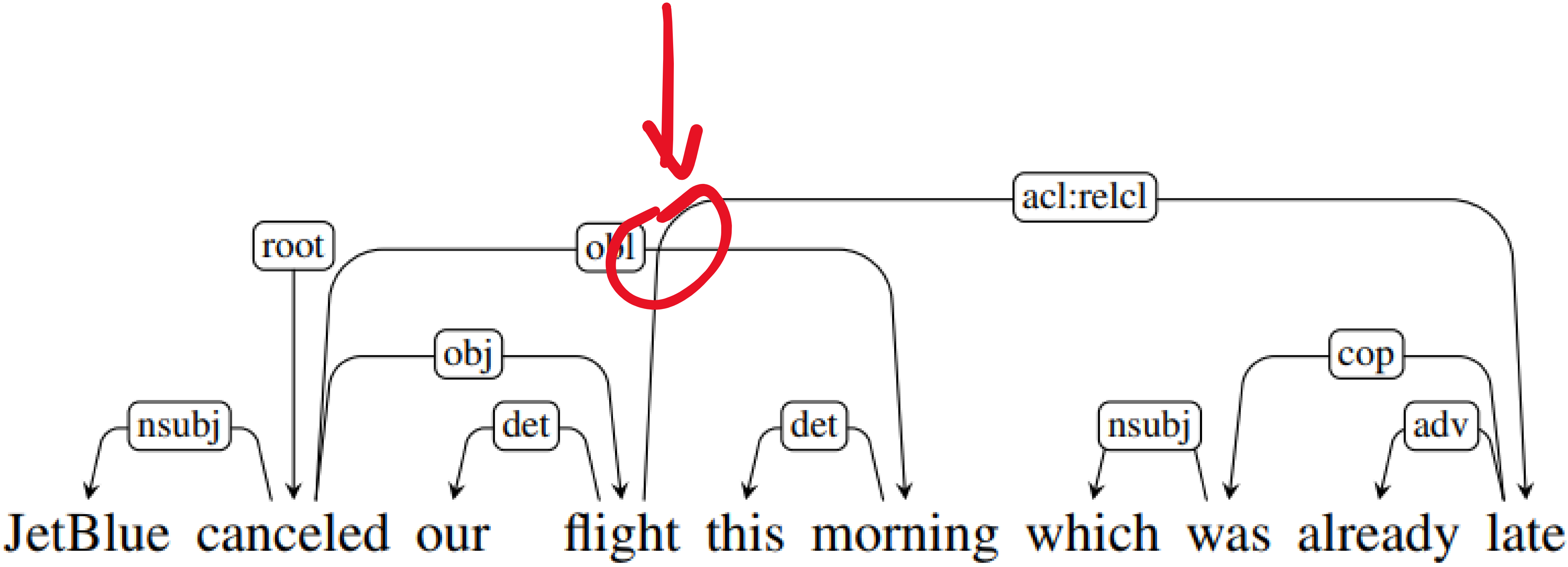
- G is (weakly) connected:
  - For every node i, there is a node j such that $i \to j$ or $j \to i$.
- G is acyclic:
  - If $i \to j$ then not $j \to^* i$.
- G obeys the single-head constraint:
  - If $i \to j$, then not $k \to j$, for any $k \neq i$.
- G is projective:
  - If $i \to j$ then $i \to^* k$, for any k such that $i < k < j$ or $j < k < i$.
  - *No crossing edges*.

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.

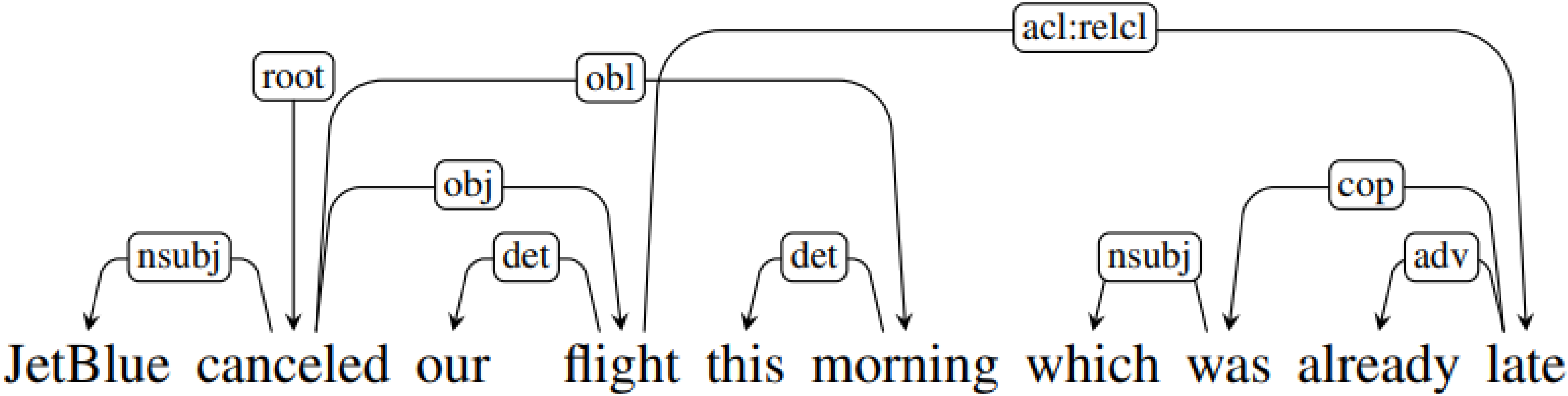


Projectivity



Projectivity

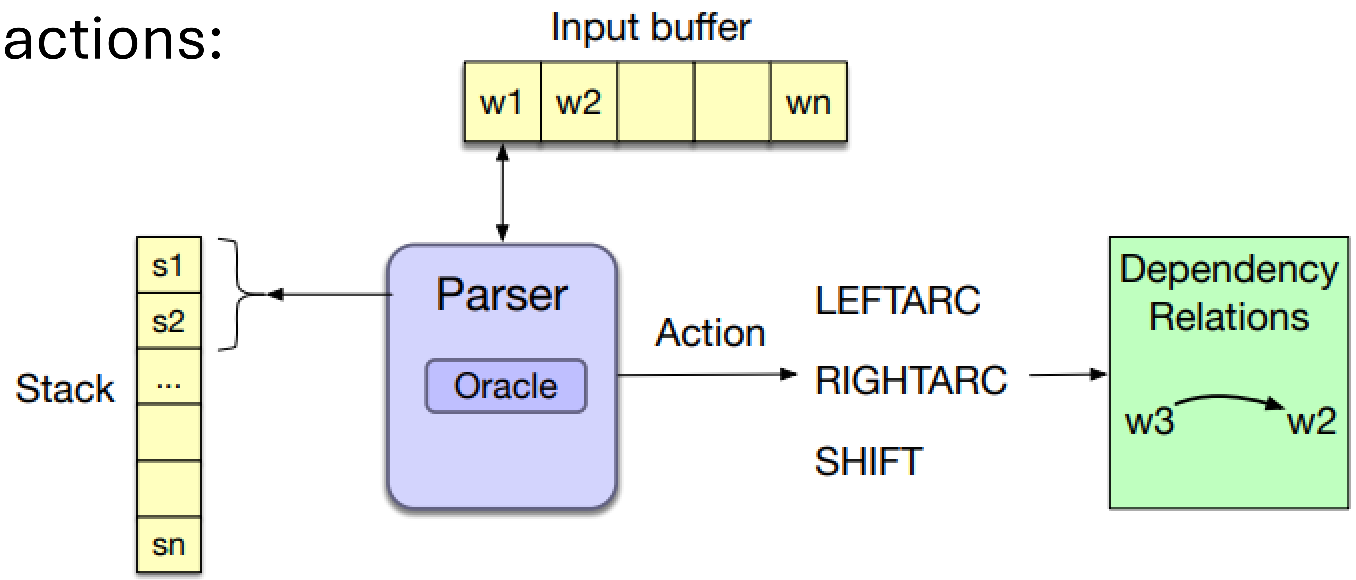
- Most theoretical frameworks do not assume projectivity.
- Non-projective structures are needed to account for:
  - Long-distance dependencies,
  - Free word order.



Transition-Based Dependency Parsing

Shift-Reduce Parsing:

- Data structures:
  - Stack: $[\ldots, w_i]_S$ of partially processed tokens
  - Queue: $[w_j, \ldots]_Q$ of remaining input tokens.
- Parsing actions built from atomic actions:
  - Adding arcs: $(w_i \rightarrow w_j,\ w_i \leftarrow w_j)$.
  - Stack and queue operations.
- Left-to-right parsing in O(n) time.
- Restricted to projective dependency graphs.
  - Non-projective: next week.



Yamada's Algorithm

- Tree parsing actions:

$$\text{Shift} \quad \frac{[\ldots]_S \quad [w_i, \ldots]_Q}{[\ldots, w_i]_S \quad [\ldots]_Q}$$

$$\text{Right} \quad \frac{[\ldots, w_i, w_j]_S \quad [\ldots]_Q}{[\ldots, w_i]_S \quad [\ldots]_Q \quad w_i \to w_j}$$

$$\text{Left} \quad \frac{[\ldots, w_i, w_j]_S \quad [\ldots]_Q}{[\ldots, w_j]_S \quad [\ldots]_Q \quad w_i \leftarrow w_j}$$

- Algorithm variants:
  - Originally developed for Japanese (strictly head-final) with only the Shift and Left actions [Kudo and Matsumoto 2002].
  - Adapted for English (with mixed headedness) by adding the Left action [Yamada and Matsumoto 2003].

Example

Stack: [root]
Queue:[Book, me, the, morning, flight]

$[\texttt{root}]_S$ $[\texttt{Book me the morning flight}]_Q$

Example

Stack: [root, Book]
Queue:[me, the, morning, flight]

$[\text{root Book}]_S\ [\text{me the morning flight}]_Q$

Shift

Example

Stack: [root, Book, me]
Queue:[the, morning, flight]

$[\text{root Book me}]_S\ [\text{the morning flight}]_Q$

Shift

Example

Stack: [root, Book]
Queue:[the, morning, flight]

iobj

[root Book]$_S$ me [the morning flight]$_Q$

Right

Example

Stack: [root, Book, the]
Queue:[morning, flight]

iobj

[root Book me the]$_S$ [morning flight]$_Q$

Shift

Example

Stack: [root, Book, the, morning]
Queue:[flight]

iobj

[root Book me the morning]$_S$ [flight]$_Q$

Shift

Example

Stack: [root, Book, the, morning, flight]
Queue:[]

iobj

[root Book me the morning flight]$_S$ []$_Q$

Shift

Example

Stack: [root, Book, the, flight]
Queue:[]

iobj

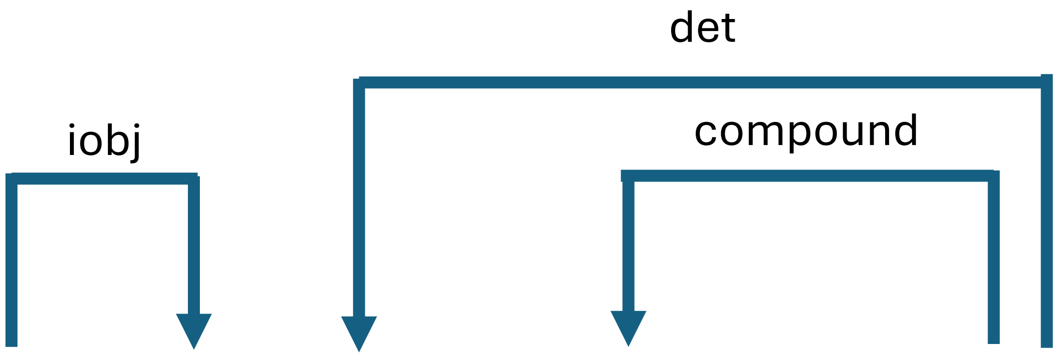
compound

[root Book me the morning flight]$_S$ []$_Q$

Left

Example

Stack: [root, Book, flight]
Queue:[]

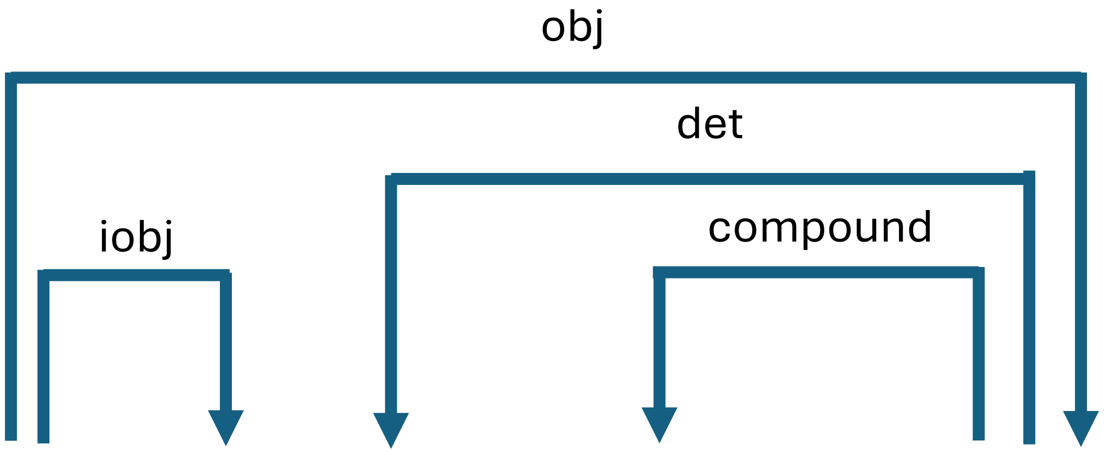


[root Book me the morning flight]$_S$ []$_Q$

Left

Example

Stack: [root, Book]
Queue:[]

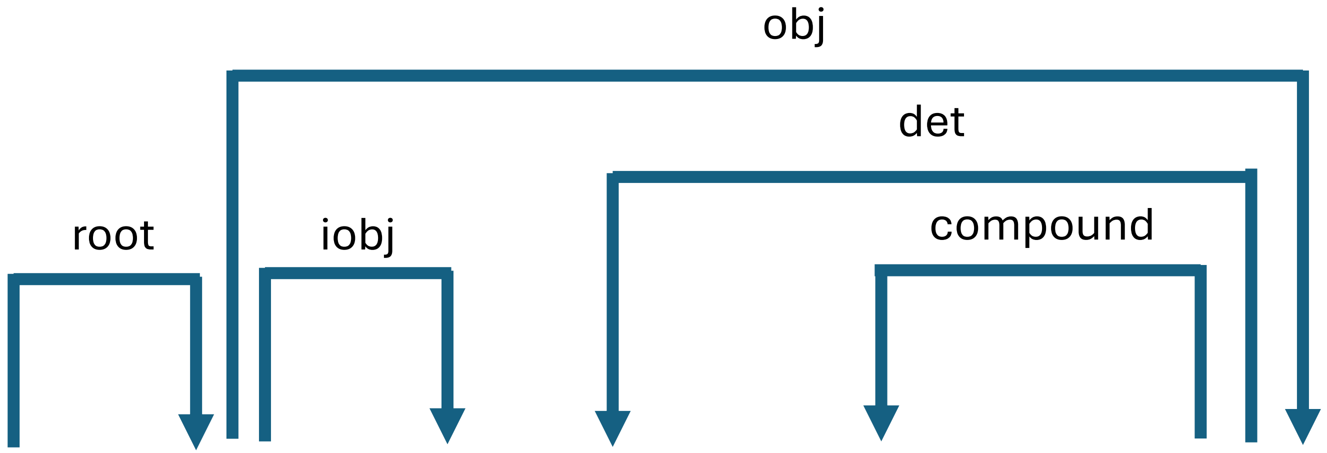


[root Book me the morning flight]$_S$ []$_Q$

Right

Example

Stack: [root]
Queue:[]

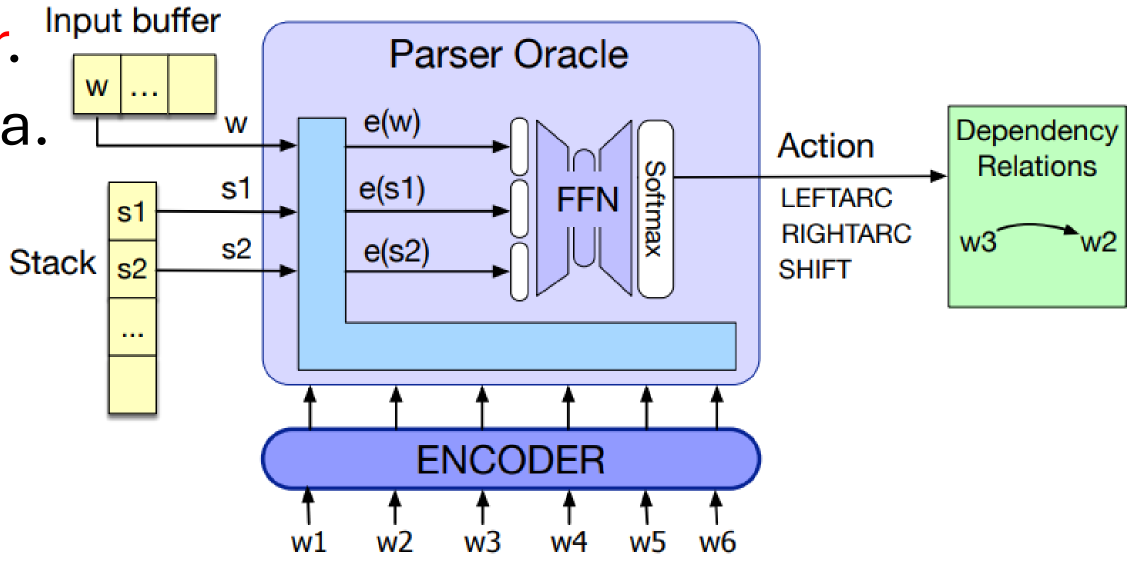


$[$root Book me the morning flight$]_S$ $[\,]_Q$

Done!

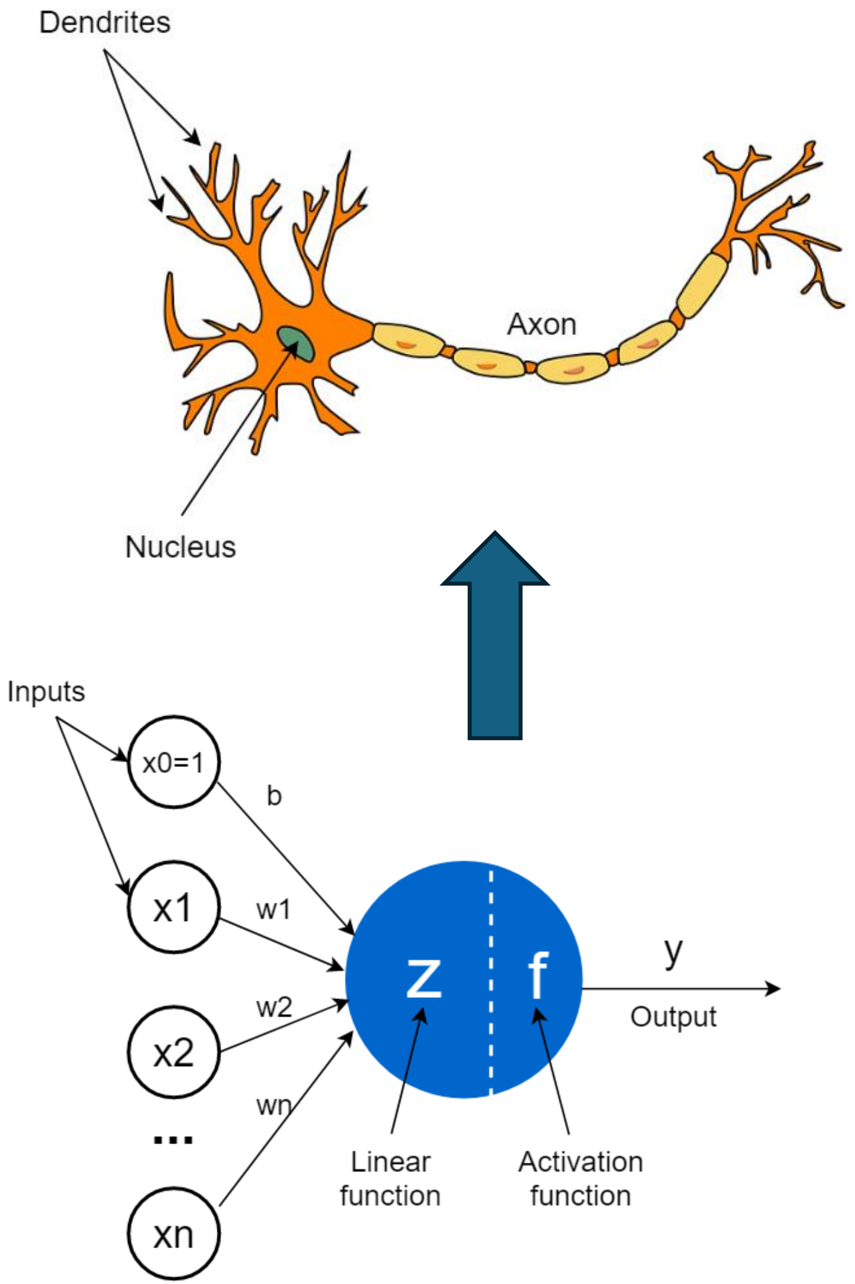
Classifier-Based Parsing

- Data-driven deterministic parsing:
  - Deterministic parsing requires an oracle.
  - An oracle can be approximated by a classifier.
  - A classifier can be trained using treebank data.
- Learning methods:
  - Support vector machines (SVM)
    [Kudo and Matsumoto 2002, Yamada and Matsumoto 2003, Isozaki et al. 2004, Cheng et al. 2004, Nivre et al. 2006]
  - Memory-based learning (MBL)
    [Nivre et al. 2004, Nivre and Scholz 2004]
  - Maximum entropy modelling (MaxEnt)
    [Cheng et al. 2005]
  - Neural networks
    [You! 2024] A1



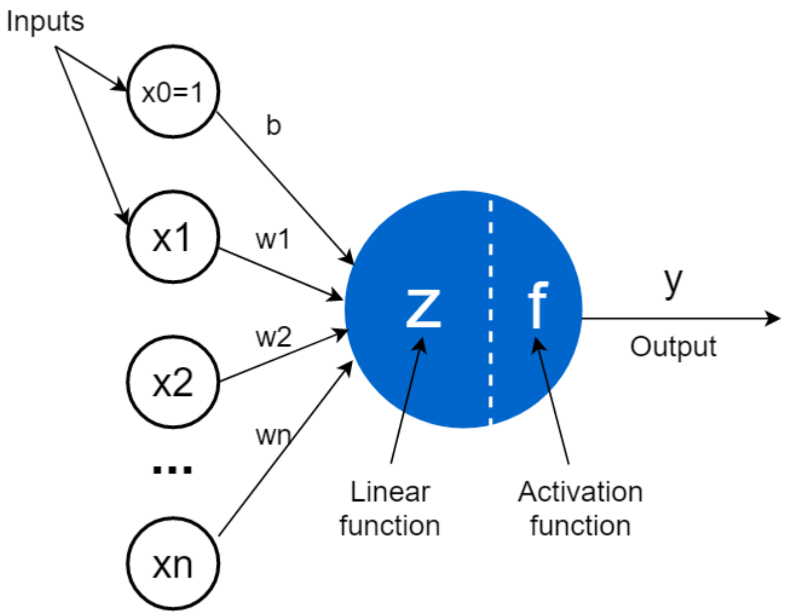
Neural Network

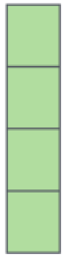
- Input can be:
  - Scalar number
  - Vector of Real numbers
  - Vector of Binary
- Outputs can be
  - Linear, single output (Linear)
  - Linear, multiple outputs (Linear)
  - Single output binary (Logistics)
  - Multi output binary (Logitics)
  - 1 of k Multinomial output (Softmax)
    (categorical)



Neural Network

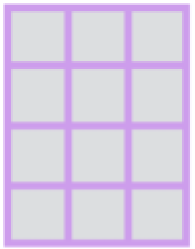
$$f(b + \sum_{i=1}^{n} x_i w_i) = f(\mathbf{x} \cdot \mathbf{W}^\top) + b$$





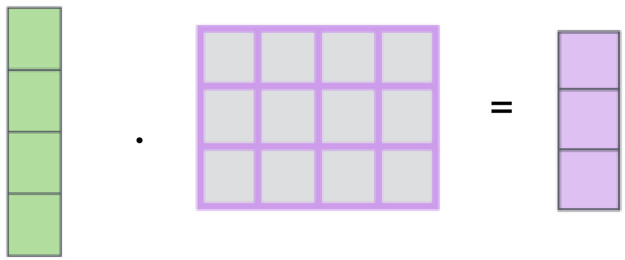
x

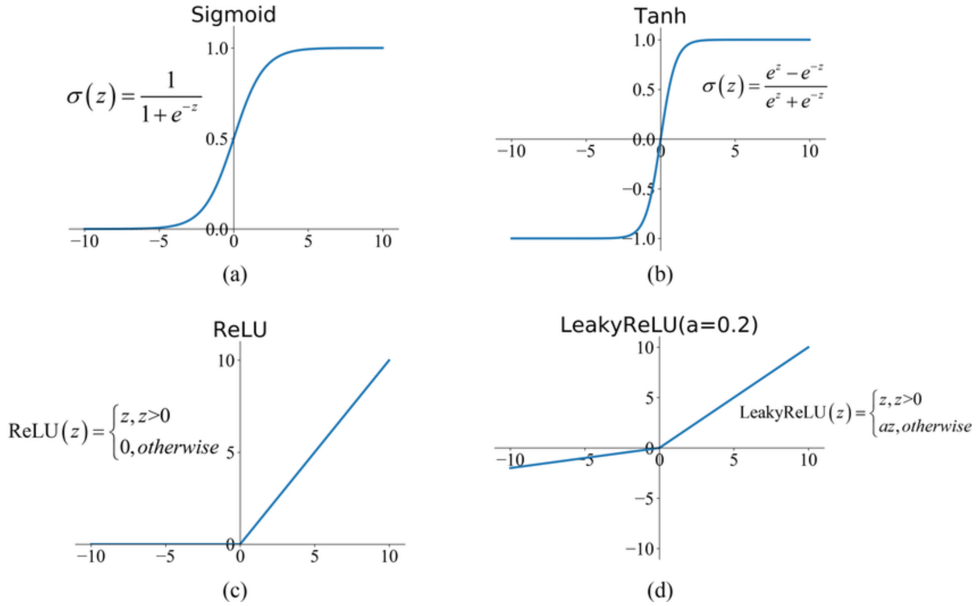
input vector



W

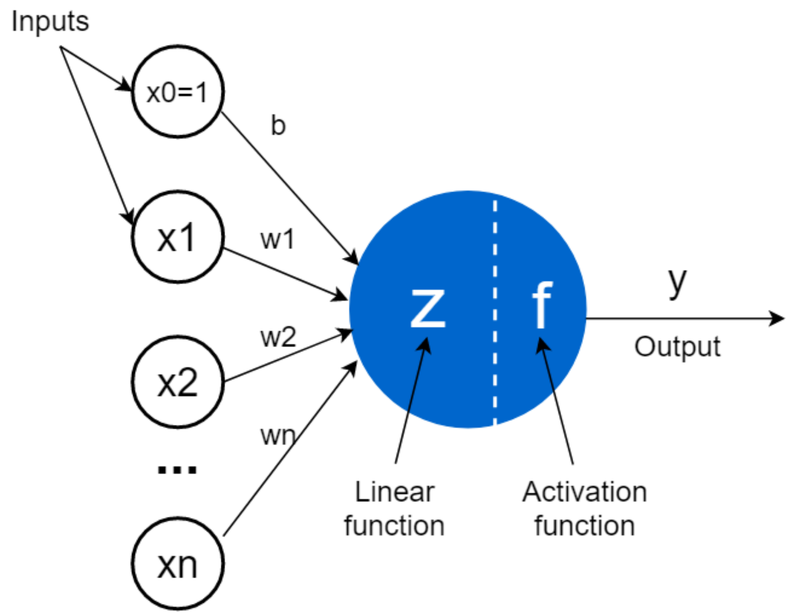
Model weights

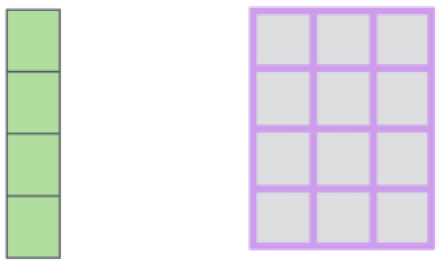




Neural Network

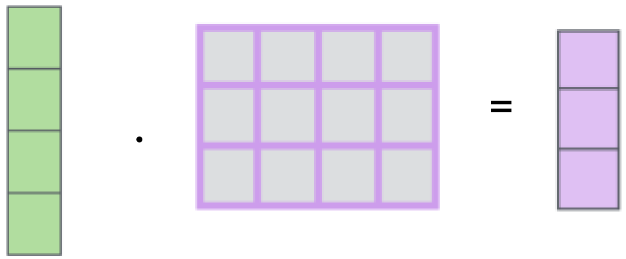
$$f(b + \sum_{i=1}^{n} x_i w_i) = f(\mathbf{x} \cdot \mathbf{W}^\top) + b$$

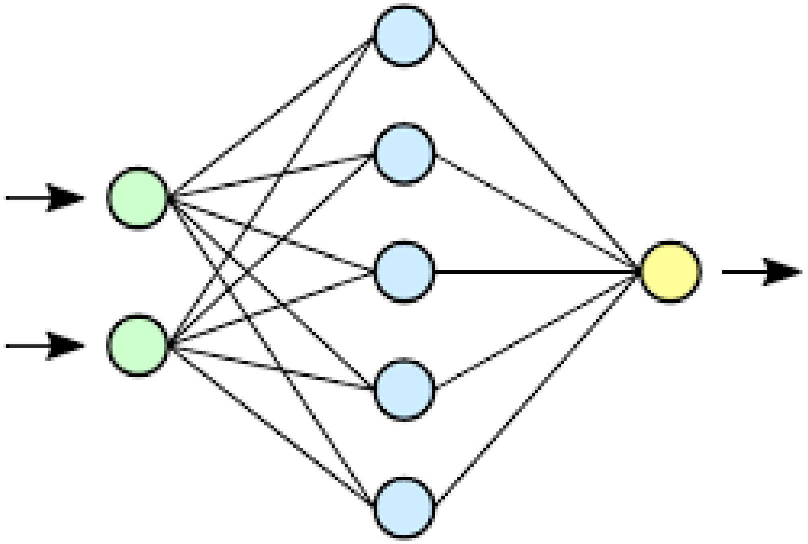




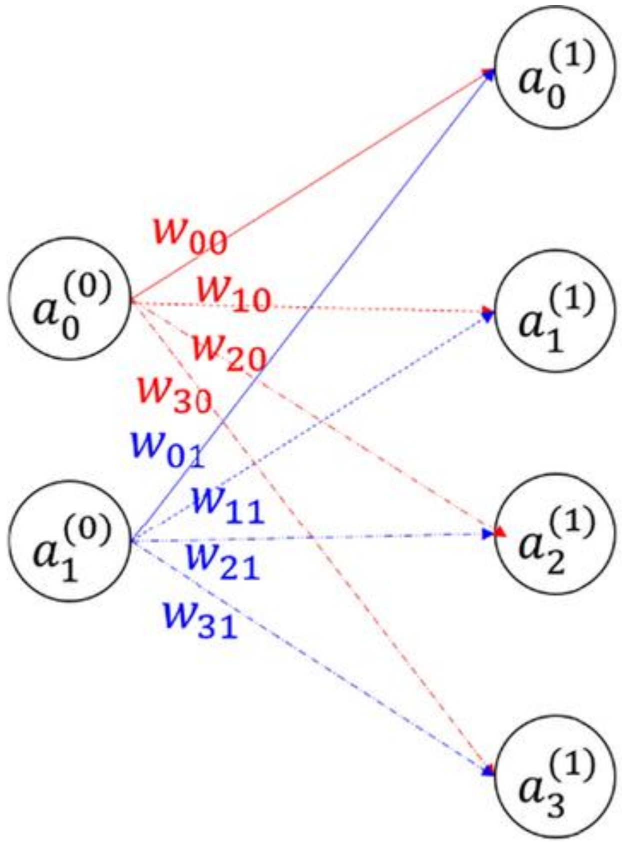
x
input
vector

W
Model
weights





Neural Network



$$a_0^{(1)} = \sigma(w_{00}\, a_0^{(0)} + w_{01}\, a_1^{(0)} + b_0)$$

$$a_1^{(1)} = \sigma(w_{10}\, a_0^{(0)} + w_{11}\, a_1^{(0)} + b_1)$$

$$a_2^{(1)} = \sigma(w_{20}\, a_0^{(0)} + w_{21}\, a_1^{(0)} + b_2)$$

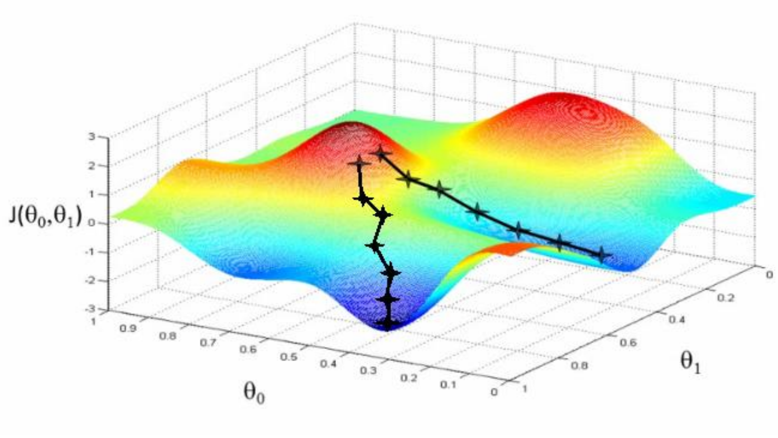
$$a_3^{(1)} = \sigma(w_{30}\, a_0^{(0)} + w_{31}\, a_1^{(0)} + b_3)$$

$$a_j^{(l)} = \sigma(\sum_{i=1}^{N_{l-1}} w_{ji}\, a_i^{(l-1)} + b_j)$$

Gradient Descent

- Each neuron:
  - Weight matrix.
  - Bias term.
  - How to determine the model parameters?

$$f(b + \sum_{i=1}^{n} x_i w_i) = f(\mathbf{x} \cdot \mathbf{W}^\top) + b$$



- Strategy:
  - Compute the error at the output.
  - Determine the contribution of each parameter to the error by taking the differential of error w.r.t. the parameter.
  - Update the parameter commensurate with the error it contributed.
  - Mountain analogy:
    - Error of every param. combination: contour map.
    - Slope: gradient of error.
    - Blindly going down hill → you will reach a lower place (local minimum of error).

Assignment 1

- Q1: transition-based parsing
  - Wednesday (L3) and today’s (L4) lecture!
- Q2: graph-based parsing
  - Next Monday (L5).