CS626: Speech, NLP and the Web

Deeper into Projectivity and Neural Dependency Parsing

Pushpak Bhattacharyya
Computer Science and Engineering Department
IIT Bombay

Week of 2nd November, 2020

Clarifying and finalizing projectivity

Makin approaches

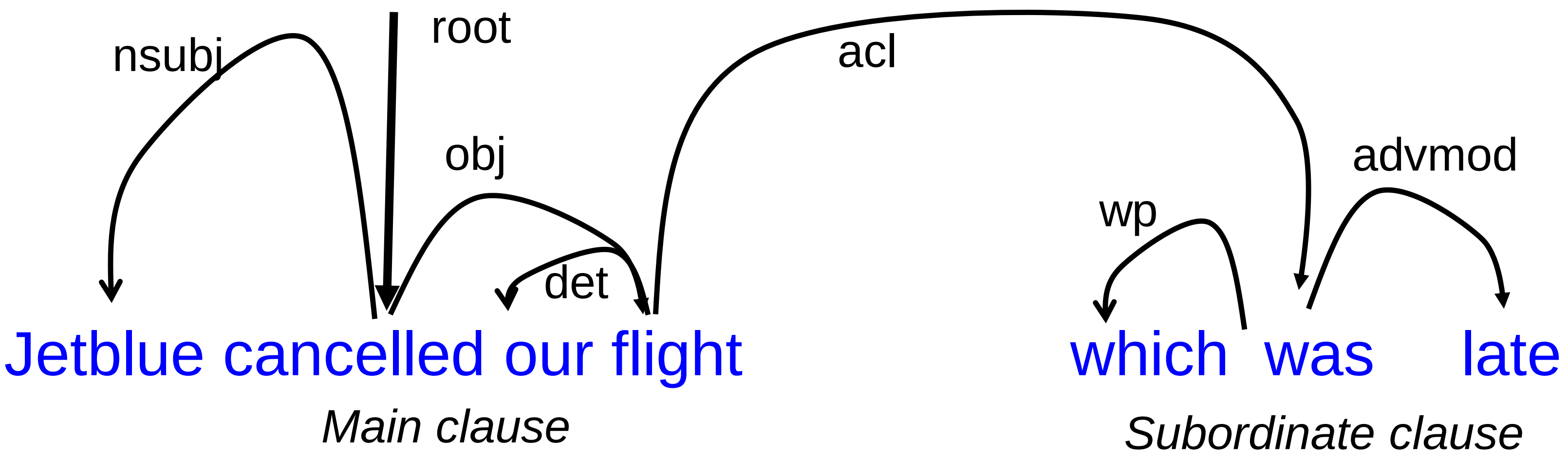
Conditions for projective dependency tree

1. All arcs are on ONE side (above or below) of the sentence.

2. There is NO crossing of arcs.

Equivalent: for EVERY Head-Modifier pair in the sentence, there is a path from the said Head to EVERY word in between the said Head and the said modifier.

Example (from J & M, 2019)



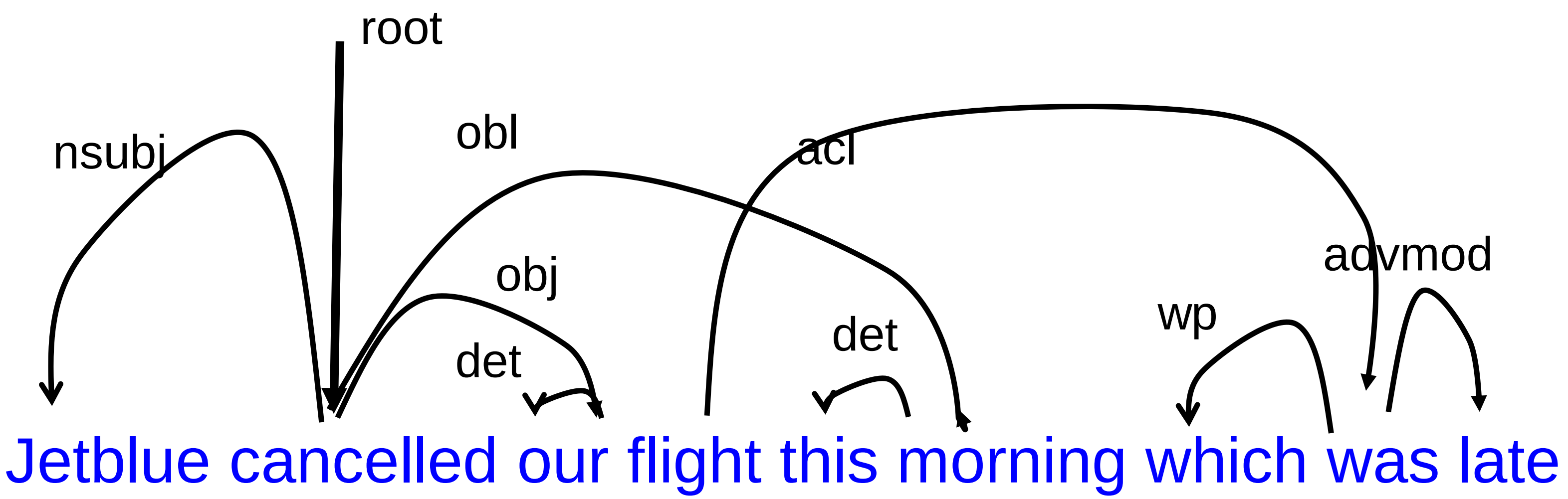
Uses Universal Dependency

wp- relative pronoun

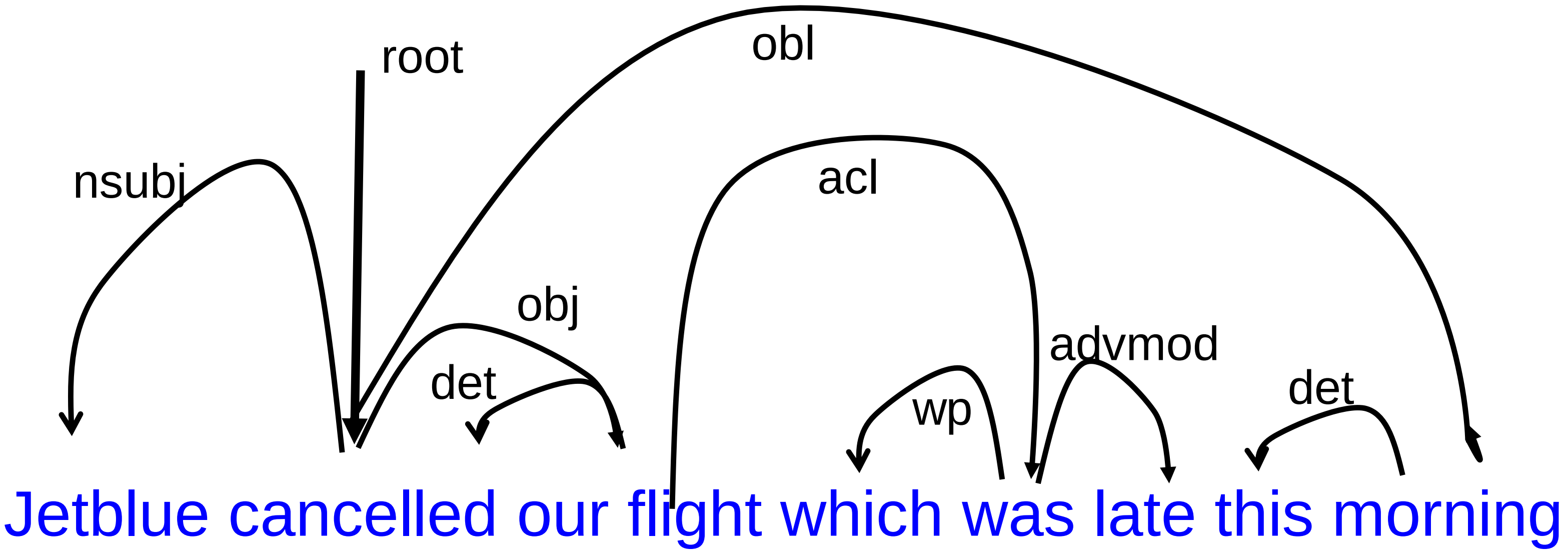
acl- clausal modifier of noun

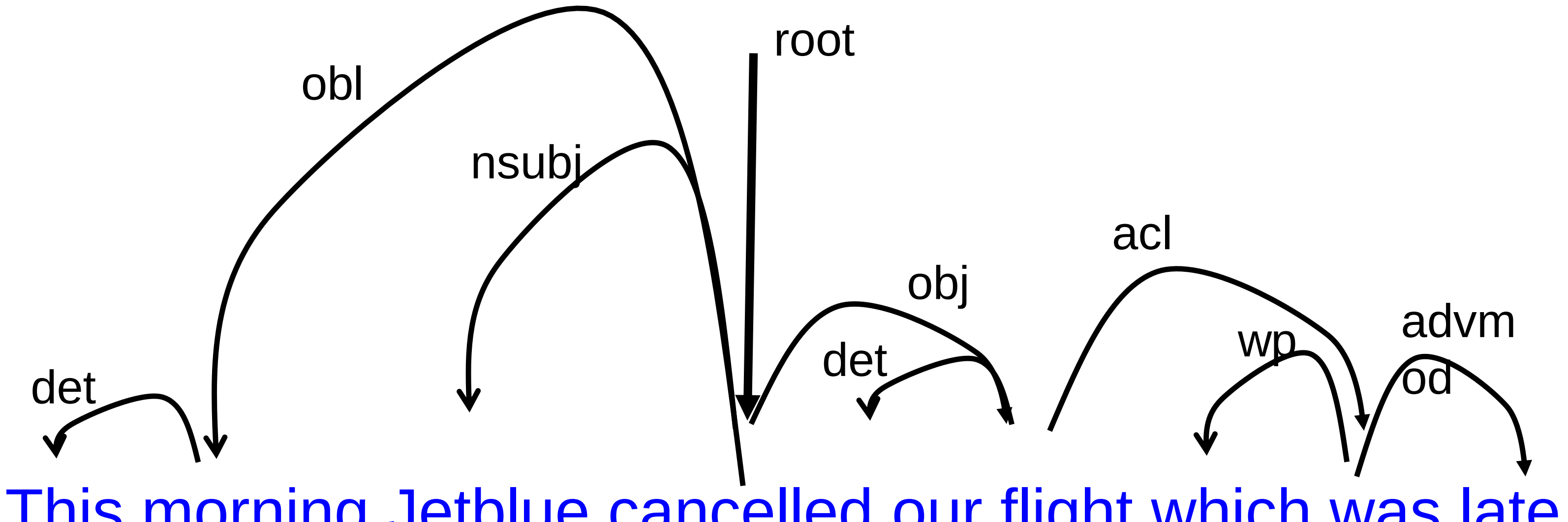
The head of the acl relation is the noun that is modified, and the dependent is the head of the clause that modifies the noun

Example cntd. (from J & M, 2019):
insert “this morning”

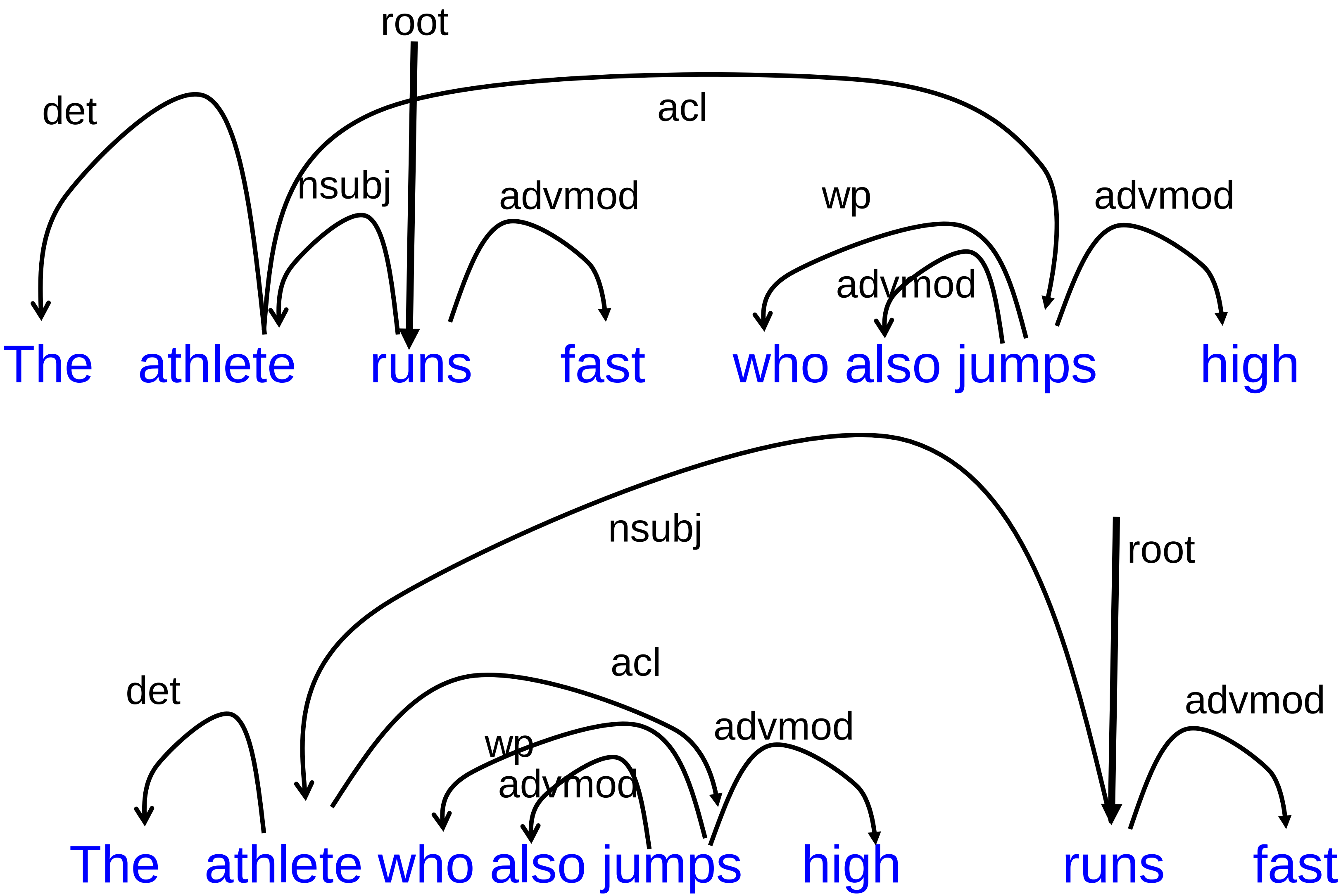


Move around “this morning”

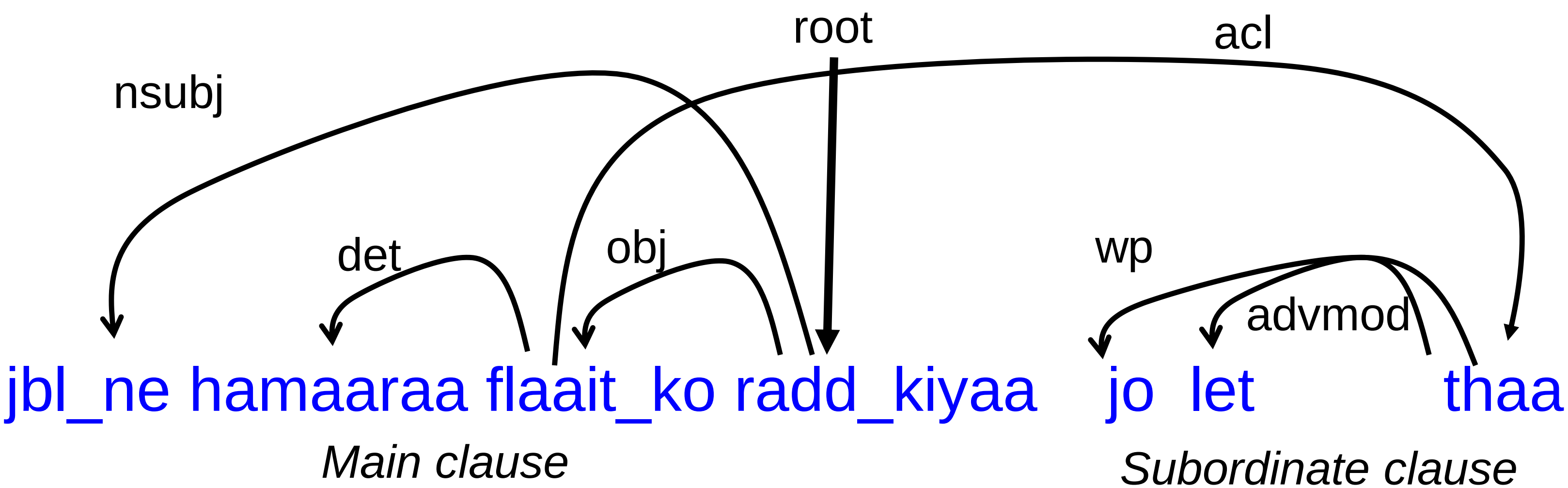




Another Example

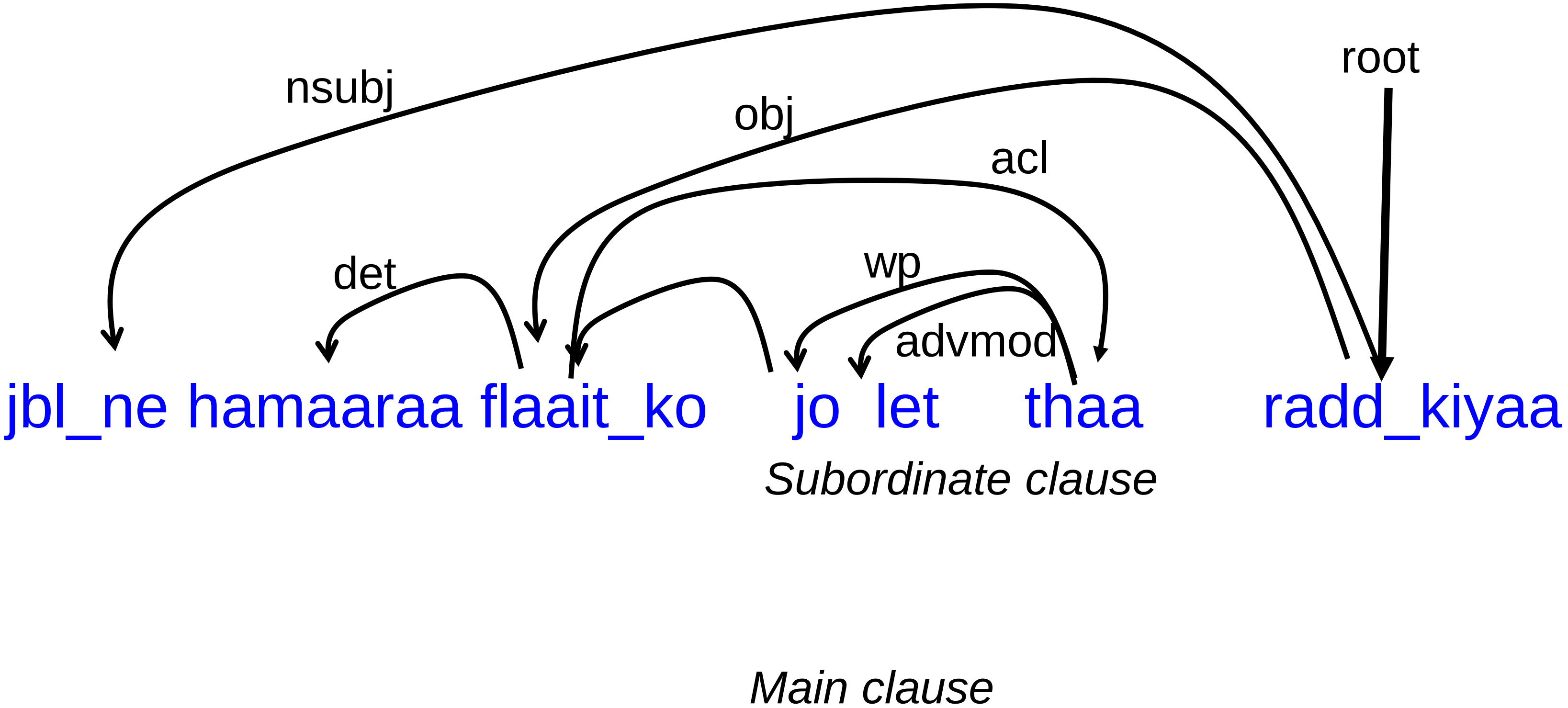


DP for other languages (Indian languages)



Uses Universal Dependency

DP for other languages (Indian languages)



Uses Universal Dependency

Pragmatic considerations

- The most sensitive parts of a sentence are the beginning and end parts
- Example: “Jetblue cancelled our flight which was late **this morning**” → “**This morning** Jetblue cancelled our flight which was late”
    - Emphasis on “this morning”
    - Restores projectivity

- **Pragmatic difference** is coming because of
    - Speech act
    - Topicalization
    - Emphasis
    - Focus

Vulnarability

The **greedy transition based dependency parsing** algorithm is constrained to make **projective tree**

Hence it might give wrong (semantically odd and/or wrong) parses because of projectivity constraint

Exercise: run the example “This morning…” by moving around the adjunct “this morning” on stanford online parser and observe results

Exercise

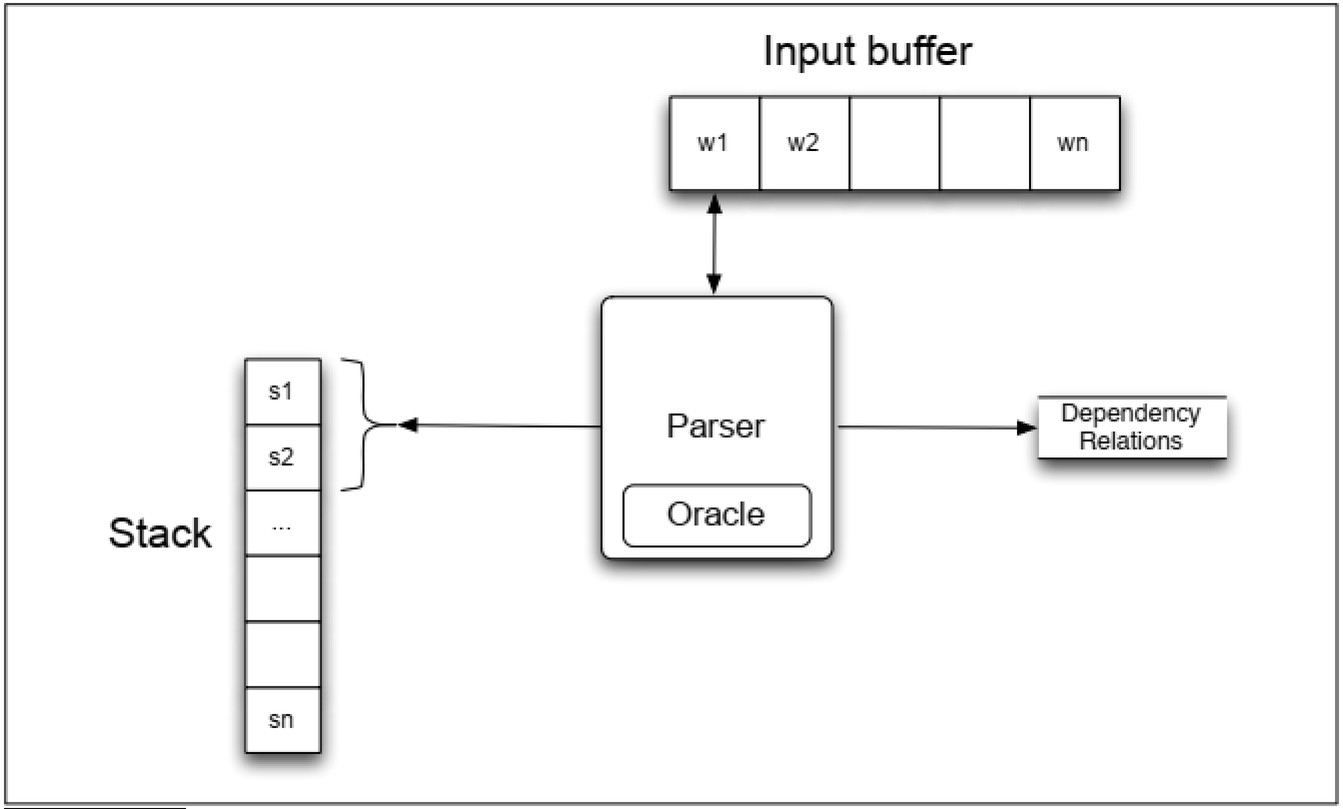
- Draw DP for
  - “*hamaara jo flaait let thaa usko jbl_ne radd_kiya*”
  - “*hamaara us flaait_ko jo let thaa jbl_ne radd_kiya*”
  - “*hamaara us flaait_ko jbl_ne radd_kiya jo let thaa*”
- Take any other language you know and repeat the above

Data Driven Algorithms for Dependency Tree Construction

Two Data Driven Approaches

- Transition-based
  - State machine for mapping a sentence to its dependency graph
  - Inducing a model for predicting the next transition, given the current state and the transition history so far.
- Graph-based
  - Induce a model for assigning scores to the candidate dependency graphs for a sentence
  - Find the maximum-scoring dependency Tree
  - Maximum spanning tree (MST) parsing

Basic Transition Based DP



Examines top two elements of the stack and selects an action based on consulting an oracle that examines the current configuration.

Speech and Language Processing, Jurafksy & Martin, Ch-15, 2019

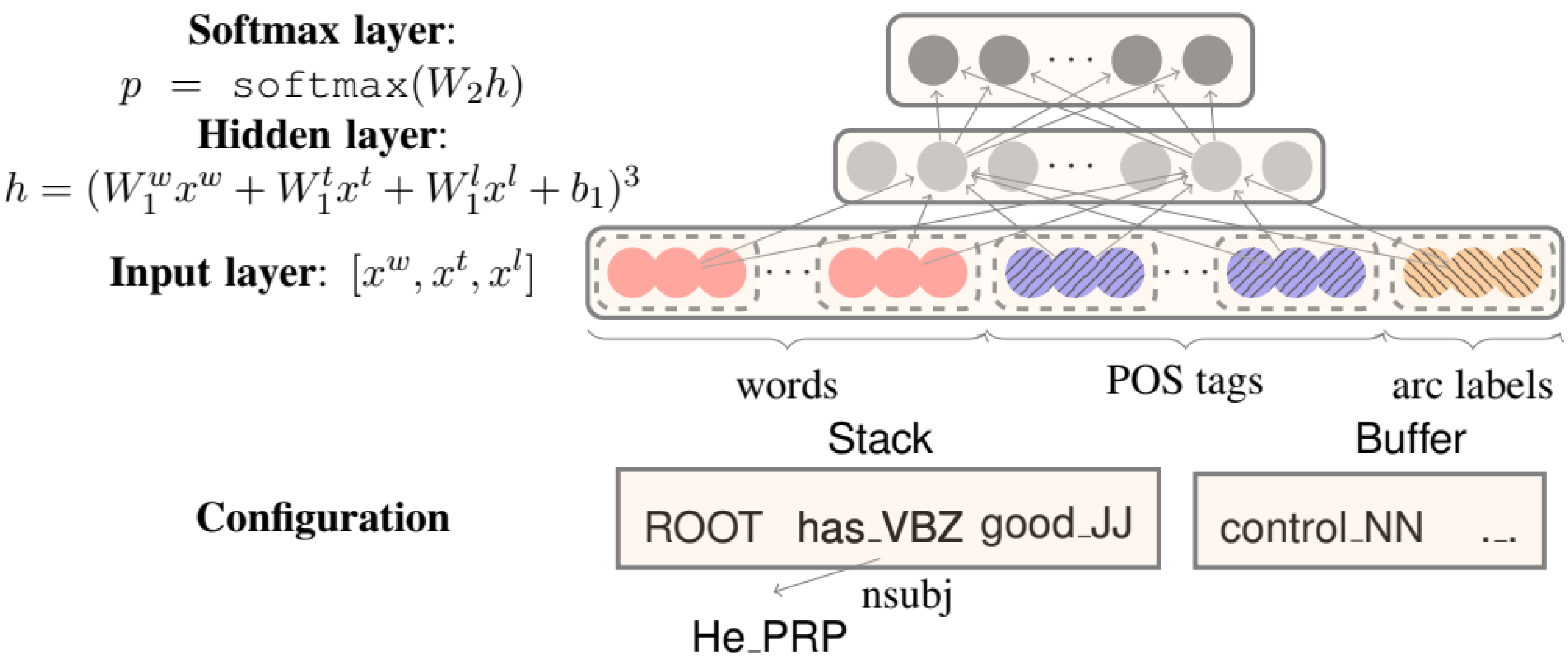
Example: transition based

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book $\rightarrow$ me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning $\leftarrow$ flight)
7	[root, book, the, flight]	[]	LEFTARC	(the $\leftarrow$ flight)
8	[root, book, flight]	[]	RIGHTARC	(book $\rightarrow$ flight)
9	[root, book]	[]	RIGHTARC	(root $\rightarrow$ book)
10	[root]	[]	Done	

Trace of a transition-based parse

Speech and Language Processing, Jurafksy & Martin, Ch-15, 2019

A neural transition based parser
(chen and Manning 2014)



Learning of transitions

Speech and NLP, J & M, Ch 15, 2019.

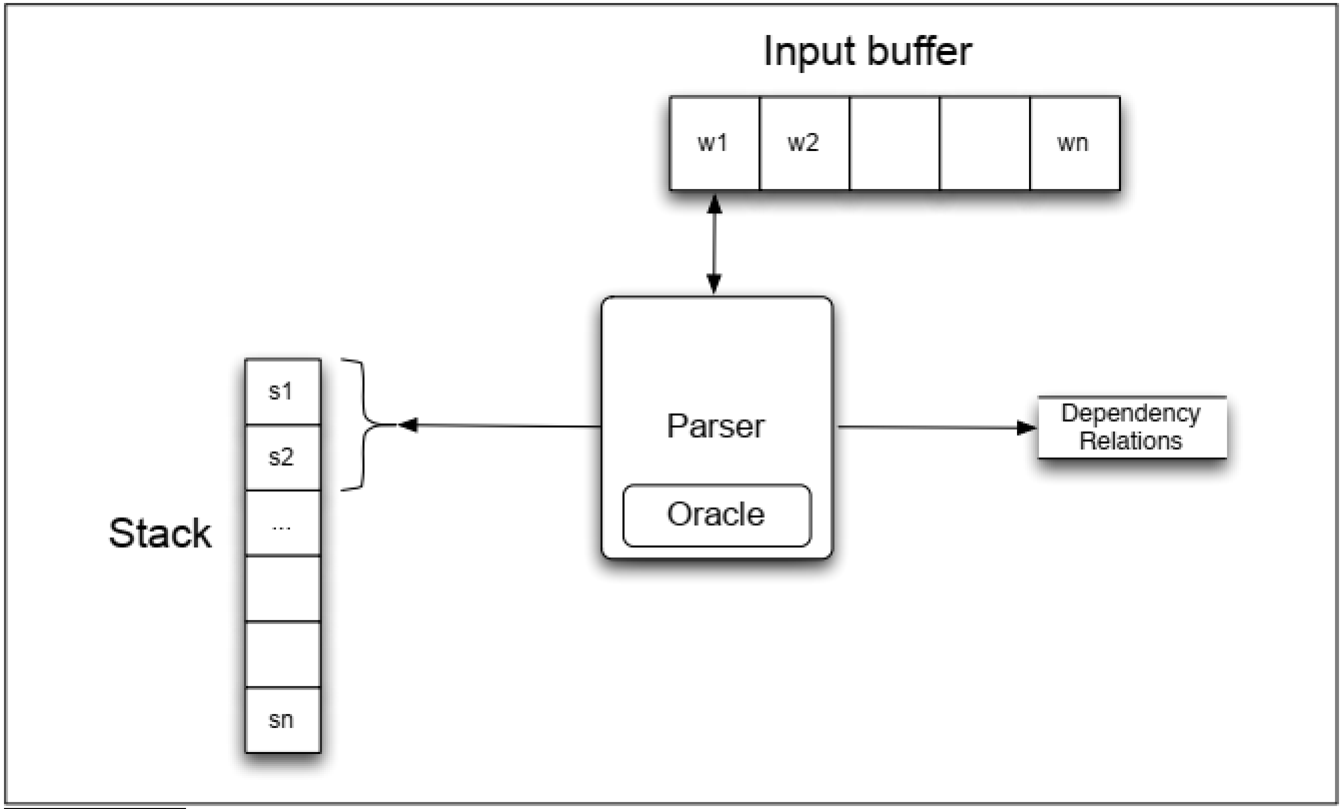
Recall: transition based DP

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book $\rightarrow$ me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning $\leftarrow$ flight)
7	[root, book, the, flight]	[]	LEFTARC	(the $\leftarrow$ flight)
8	[root, book, flight]	[]	RIGHTARC	(book $\rightarrow$ flight)
9	[root, book]	[]	RIGHTARC	(root $\rightarrow$ book)
10	[root]	[]	Done	

Trace of a transition-based parse

Speech and Language Processing, Jurafksy & Martin, Ch-15, 2019

Basic Transition Based DP



Examines top two elements of the stack and selects an action based on consulting an oracle that examines the current configuration.

Speech and Language Processing, Jurafksy & Martin, Ch-15, 2019

Operators: shift, leftarc, rightarc

```
function DEPENDENCYPARSE(words) returns dependency tree

  state ← {[root], [words], [] }  ; initial configuration
  while state not final
    t ← ORACLE(state)        ; choose a transition operator to apply
    state ← APPLY(t, state)  ; apply it, creating a new state
  return state
```

Generation of Training Data

Step	Stack	Word List	Predicted Action
0	[root]	[book, the, flight, through, houston]	SHIFT
1	[root, book]	[the, flight, through, houston]	SHIFT
2	[root, book, the]	[flight, through, houston]	SHIFT
3	[root, book, the, flight]	[through, houston]	LEFTARC
4	[root, book, flight]	[through, houston]	SHIFT
5	[root, book, flight, through]	[houston]	SHIFT
6	[root, book, flight, through, houston]	[]	LEFTARC
7	[root, book, flight, houston]	[]	RIGHTARC
8	[root, book, flight]	[]	RIGHTARC
9	[root, book]	[]	RIGHTARC
10	[root]	[]	Done

Training data

How are operators generated

LEFTARC(r): **if** $\left(S_1\ r\ S_2\right) \in R_p$

RIGHTARC(r): **if** $\left(S_2\ r\ S_1\right) \in R_p$ **and** $\forall r', w\ s.t. \left(S_1\ r'\ w\right) \in R_p$ **then** $\left(S_1\ r'\ w\right) \in R_c$

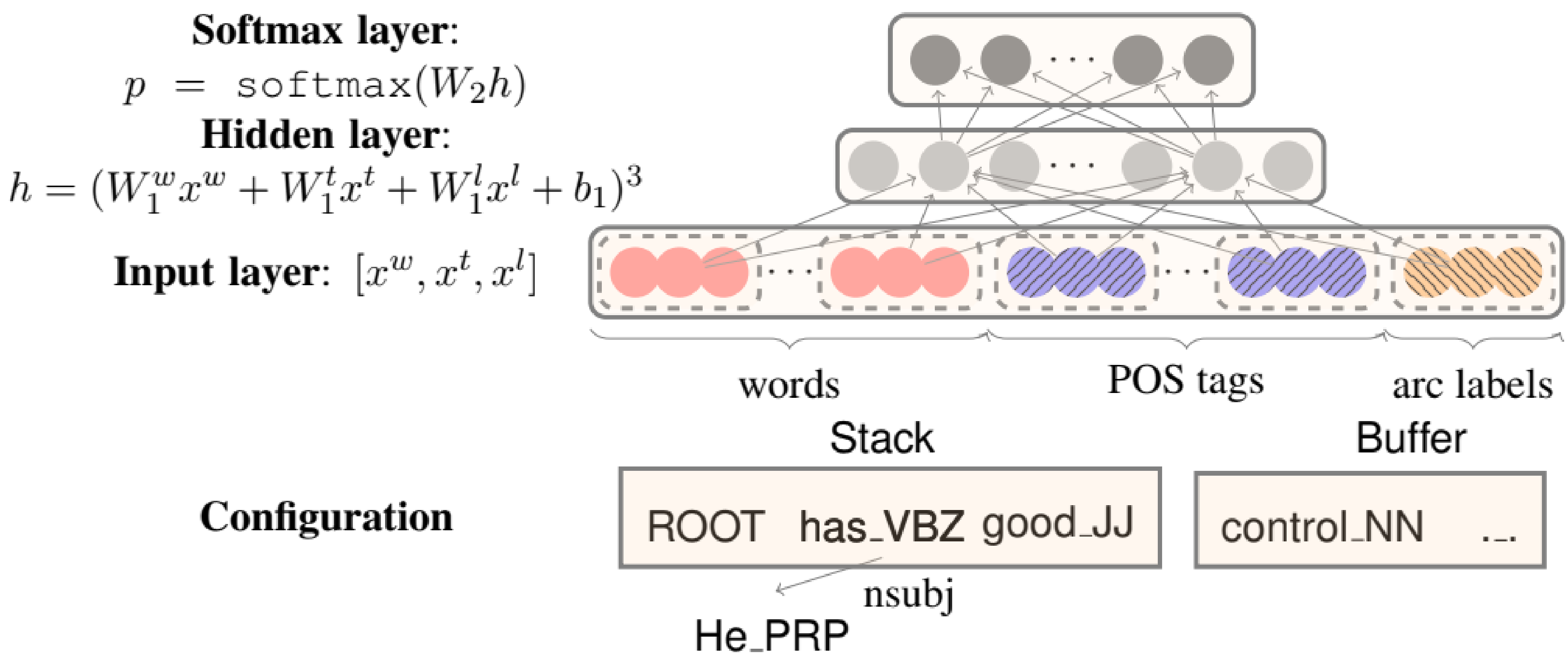
SHIFT: **otherwise**

Generation of Training Data

Step	Stack	Word List	Predicted Action
0	[root]	[book, the, flight, through, houston]	SHIFT
1	[root, book]	[the, flight, through, houston]	SHIFT
2	[root, book, the]	[flight, through, houston]	SHIFT
3	[root, book, the, flight]	[through, houston]	LEFTARC
4	[root, book, flight]	[through, houston]	SHIFT
5	[root, book, flight, through]	[houston]	SHIFT
6	[root, book, flight, through, houston]	[]	LEFTARC
7	[root, book, flight, houston]	[]	RIGHTARC
8	[root, book, flight]	[]	RIGHTARC
9	[root, book]	[]	RIGHTARC
10	[root]	[]	Done

Training data

A neural transition based parser
(chen and Manning 2014)



Input to Neural Net

Single-word features (9)
$s_1.w$; $s_1.t$; $s_1.wt$; $s_2.w$; $s_2.t$;
$s_2.wt$; $b_1.w$; $b_1.t$; $b_1.wt$
Word-pair features (8)
$s_1.wt \circ s_2.wt$; $s_1.wt \circ s_2.w$; $s_1.wts_2.t$;
$s_1.w \circ s_2.wt$; $s_1.t \circ s_2.wt$; $s_1.w \circ s_2.w$
$s_1.t \circ s_2.t$; $s_1.t \circ b_1.t$
Three-word feaures (8)
$s_2.t \circ s_1.t \circ b_1.t$; $s_2.t \circ s_1.t \circ lc_1(s_1).t$;
$s_2.t \circ s_1.t \circ rc_1(s_1).t$; $s_2.t \circ s_1.t \circ lc_1(s_2).t$;
$s_2.t \circ s_1.t \circ rc_1(s_2).t$; $s_2.t \circ s_1.w \circ rc_1(s_2).t$;
$s_2.t \circ s_1.w \circ lc_1(s_1).t$; $s_2.t \circ s_1.w \circ b_1.t$

- The feature templates used for analysis
  - *lc*1(*si*) and *rc*1(*si*) denote the leftmost and rightmost children of s_i
  - *w* denotes word
  - *t* denotes POS tag

Features: example sentence “*cancelled flights to Houston*”

$$\langle s_1.w = \mathit{flights}, op = \mathit{shift} \rangle$$

$$\langle s_2.w = \mathit{canceled}, op = \mathit{shift} \rangle$$

$$\langle s_1.t = NNS, op = \mathit{shift} \rangle$$

$$\langle s_2.t = VBD, op = \mathit{shift} \rangle$$

$$\langle b_1.w = \mathit{to}, op = \mathit{shift} \rangle$$

$$\langle b_1.t = TO, op = \mathit{shift} \rangle$$

$$\langle s_1.wt = \mathit{flightsNNS}, op = \mathit{shift} \rangle$$

DP across languages

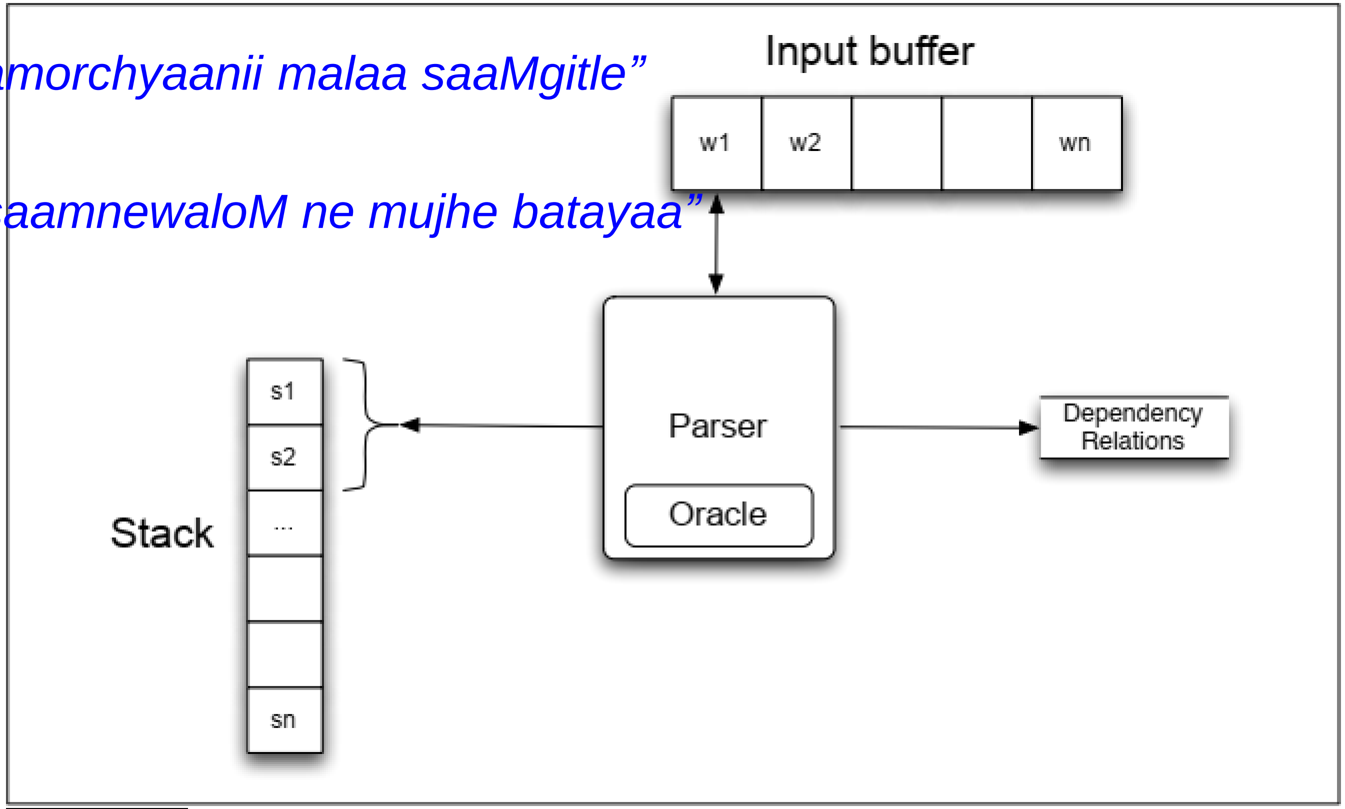
- *“people in front of the house told me”*

- *“gharaasamorchyaanii malaa saaMgitle”*

- *“ghar ke saamnewaloM ne mujhe batayaa”*

Multilingual DP

- *“people in front of the house told me”*
- *“gharaasamorchyaanii malaa saaMgitle”*
- *“ghar ke saamnewaloM ne mujhe batayaa”*



Examines top two elements of the stack and selects an action based on consulting an oracle that examines the current configuration.

Speech and Language Processing, Jurafksy & Martin, Ch-15, 2019

Essence of DP

- Cannot pop a *head* out of the stack if any of its dependents remains on the stack

- The above works if the sentence's semantics is consistent with projectivity

Graph Based DP Algorithm

Maximum Spanning Tree Algorithm (MST)

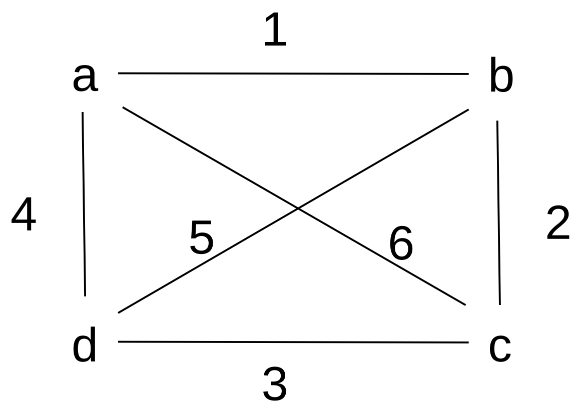
Edge Factored Scoring

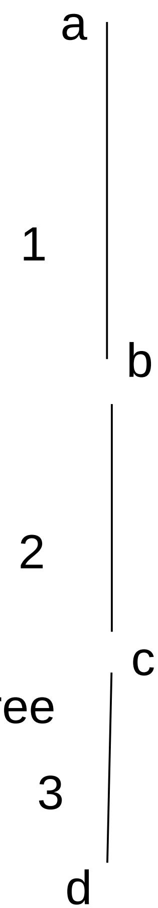
$$\overline{T} = \arg\max_{t \varepsilon D_S} score(S,t);\ D_S = Dependency\_Trees\_of\_S$$

$$score(S,t) = \sum_{e \varepsilon t} score(e)$$

$$score(e) = \sum_{f_i \varepsilon feature\_set(e)} w_i f_i$$

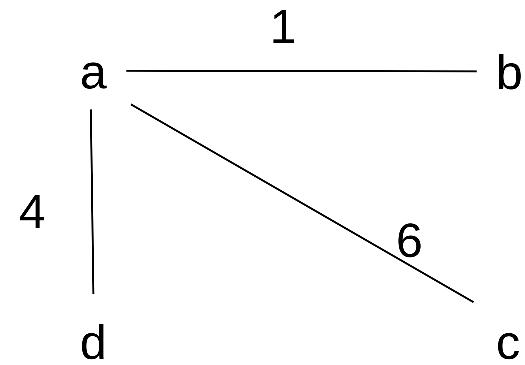
Spanning Tree Examples

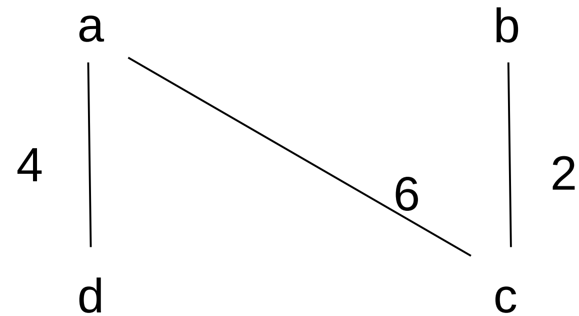




ST1:Minimum Spanning Tree
WT: 6

ST2: An ST
WT: 11





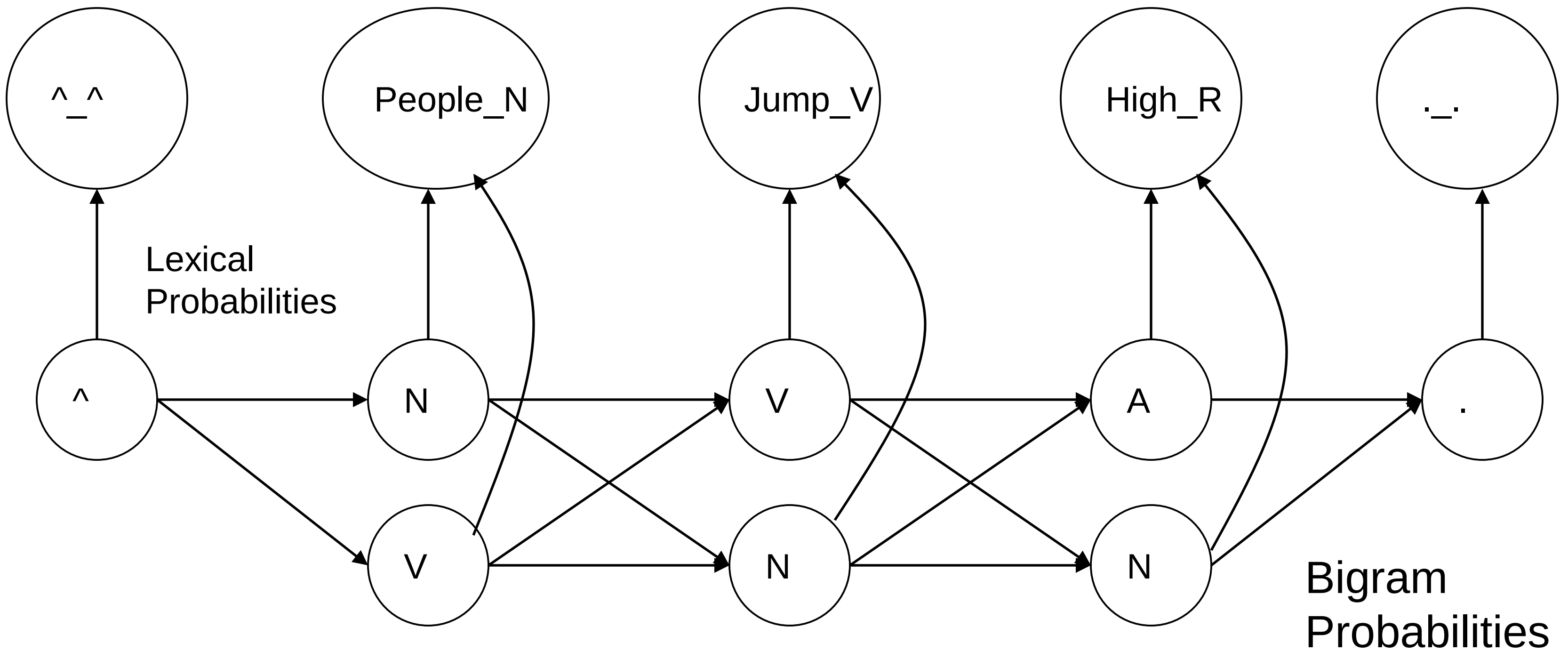
ST3: Maximum ST, WT: 12

What to score

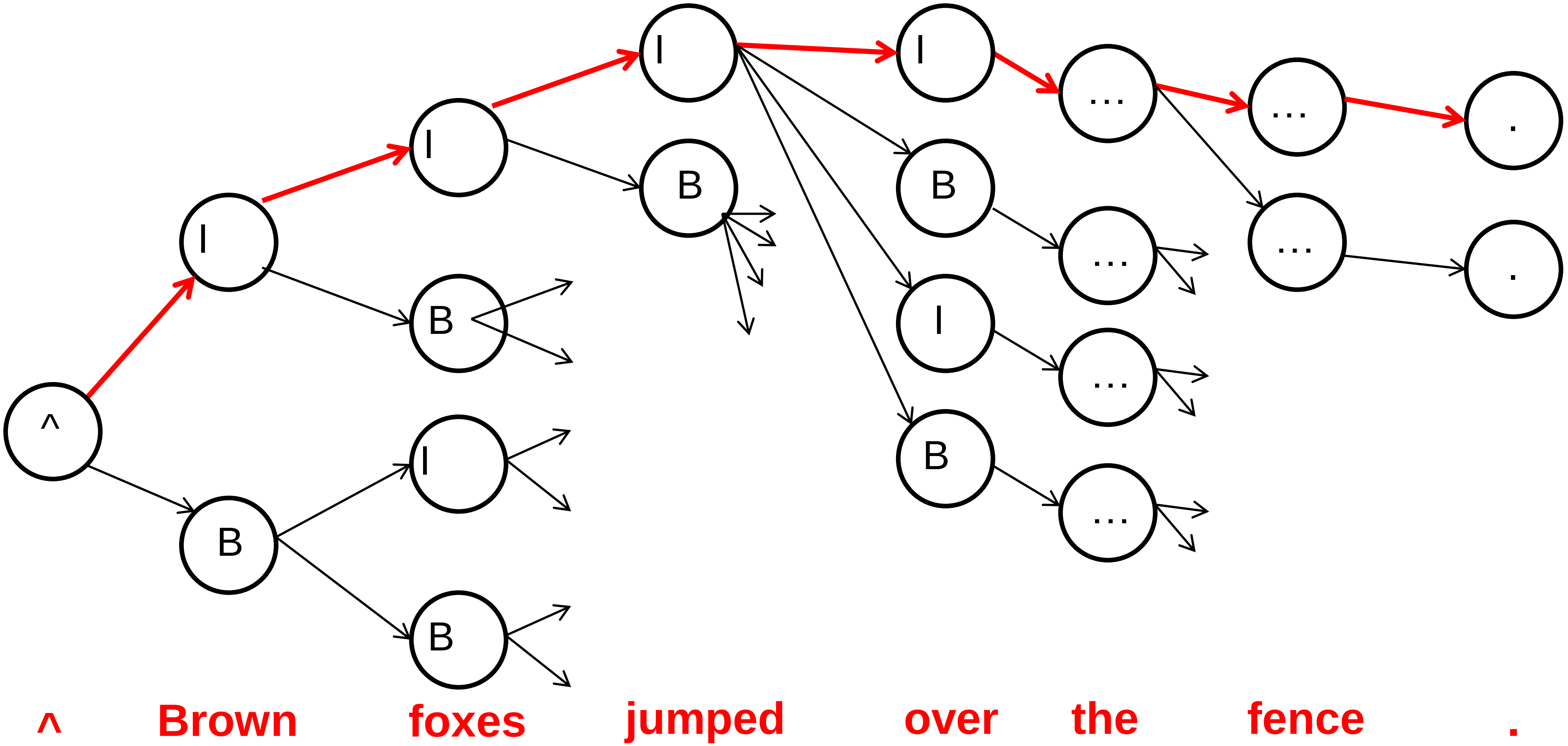
Recall

- In POS tagging, we score paths on the trellis
  - Trellis generation: trivial algorithm, erect column of states on each word
- In probabilistic constituent parsing, we score the parse trees
  - CYK algorithm

Trellis



This model is called Generative model.
Here words are observed from tags as states.
This is similar to HMM.



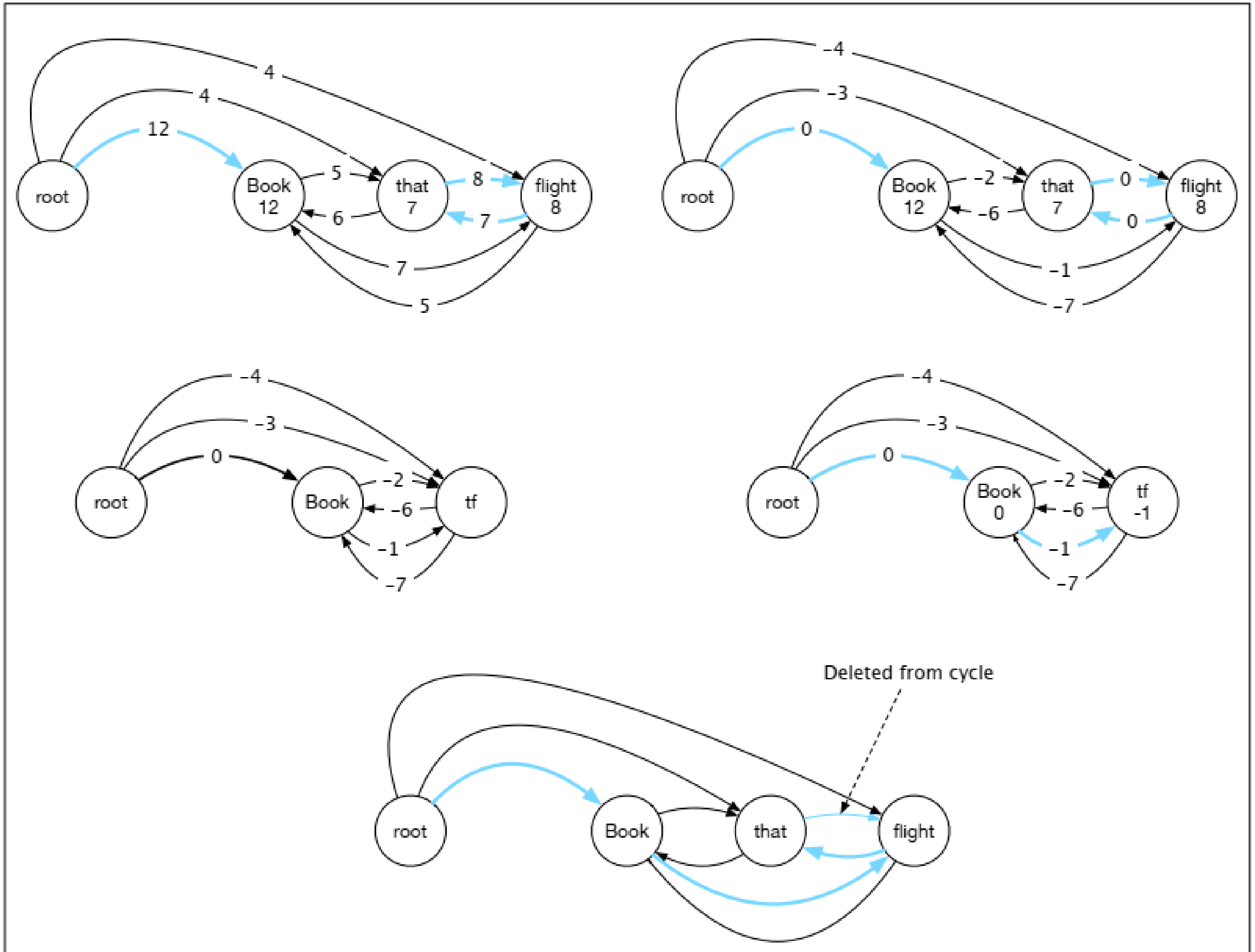
Probability of a path (e.g. Top most path) = *Product of $P(Y_i|Y_{i-1}, X)$*

Joint generation and scoring

	The 1	gunman 2	Sprayed 3	the 4	Building 5	with 6	Bullets 7
0	β_{DT} (0-1) =1.0	β_{NP} (0-2) =0.25					β_{S}(0-7) =0.006
1		β_{NN} (1-2) =0.5					
2			β_{VBD}(2-3) =1.0		β_{VP} (2-5) =0.1		β_{VP}(2-7) =0.024
3				β_{DT}(3-4) =1.0	β_{NP} (3-5) =0.25		β_{NP}(3-7) =0.015
4					β_{NN} (4-5) =0.5		
5						β_{P}(5-6) =1.0	β_{PP}(5-7) =0.3
6							$\beta_{NP/NNS}$(6-7) =1.0

CYK algo for generation and scoring

Chu-Liu-Edmonds graph-based example for *Book that flight*



Time Complexity of MST Algo

- $O(|E| log|V|)$

- *E: edge set*
- *V: vertex set*

Features for MST Algo for DP (J & M, 2019)

- Wordforms, lemmas, and parts of speech of the headword and its dependent
- Corresponding features derived from the contexts before, after and between the words
- Word embeddings
- The dependency relation itself
- The direction of the relation (to the right or left)
- The distance from the head to the dependent

Neural Graph Based DP

- State-of-the-art algorithms in multilingual parsing are based on recurrent neural networks (RNNs)

- Zeman et al. 2017, Dozat et al. 2017

Recurrent Neural Network

Acknowledgement:

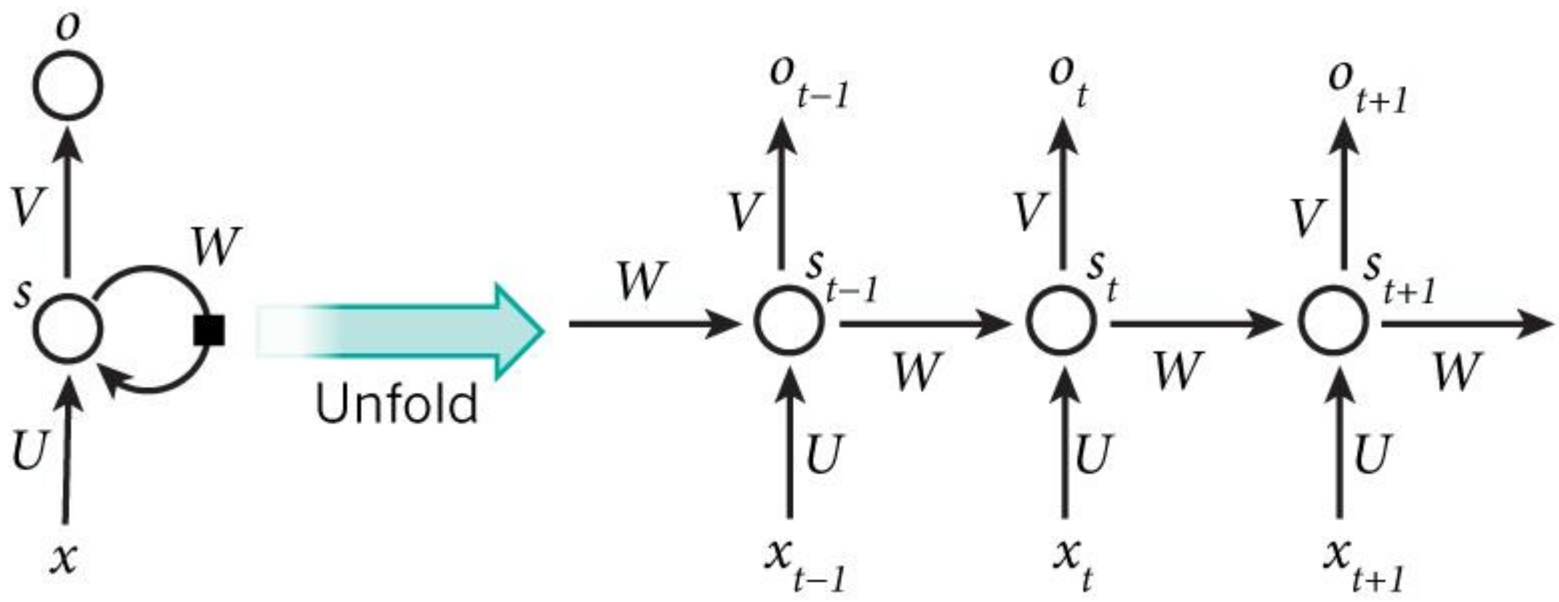
1. http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

By Denny Britz

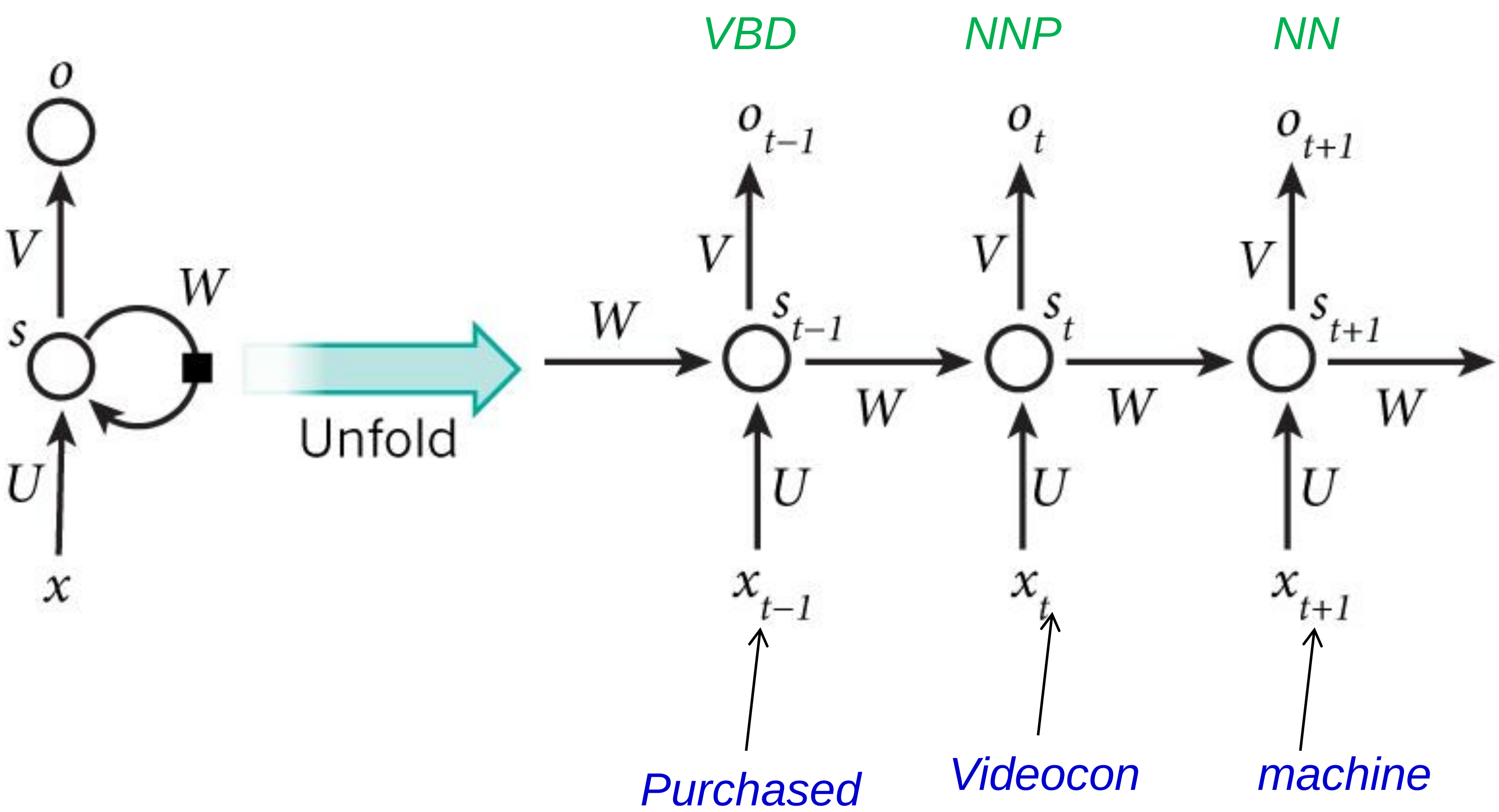
2. Introduction to RNN by Jeffrey Hinton

http://www.cs.toronto.edu/~hinton/csc2535/lectures.html

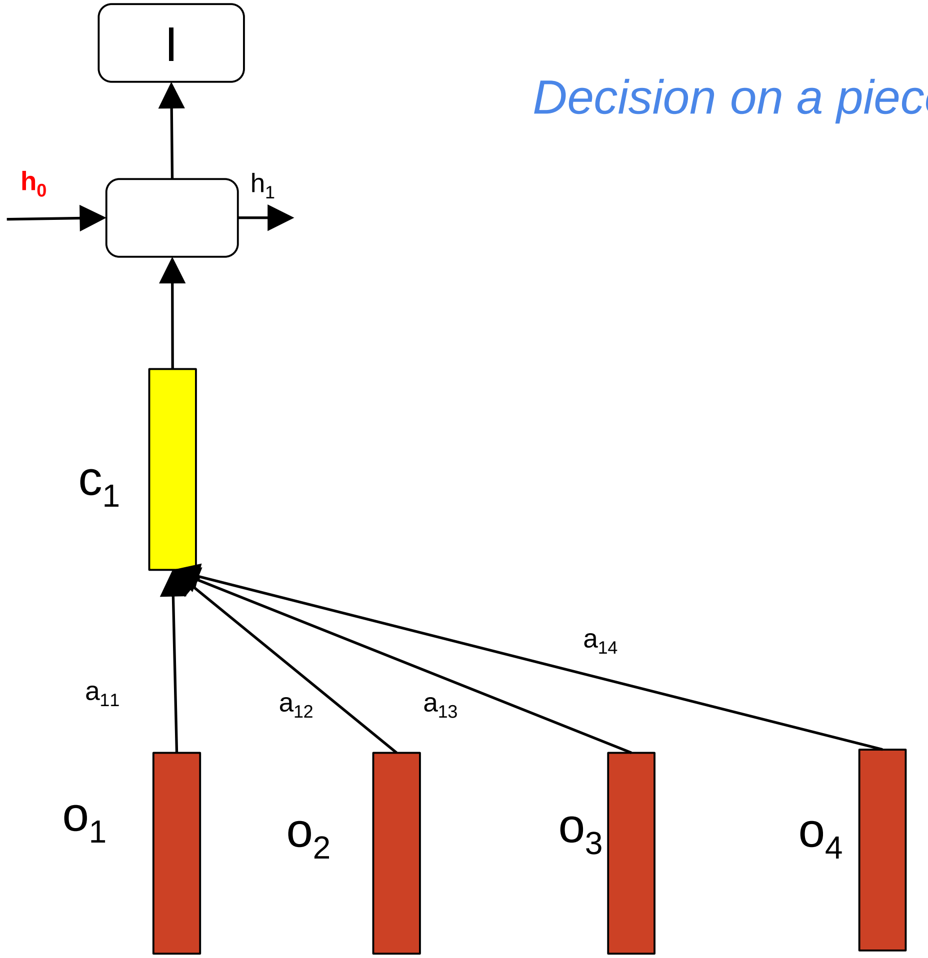
Sequence processing m/c



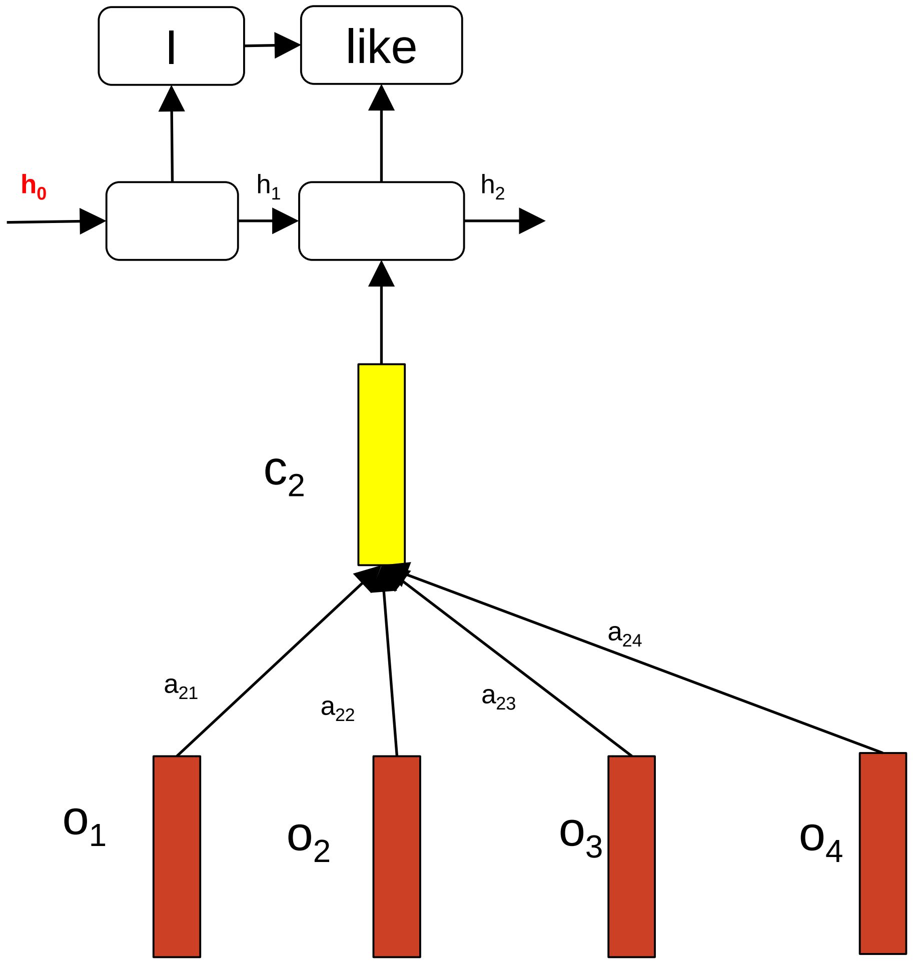
E.g. POS Tagging

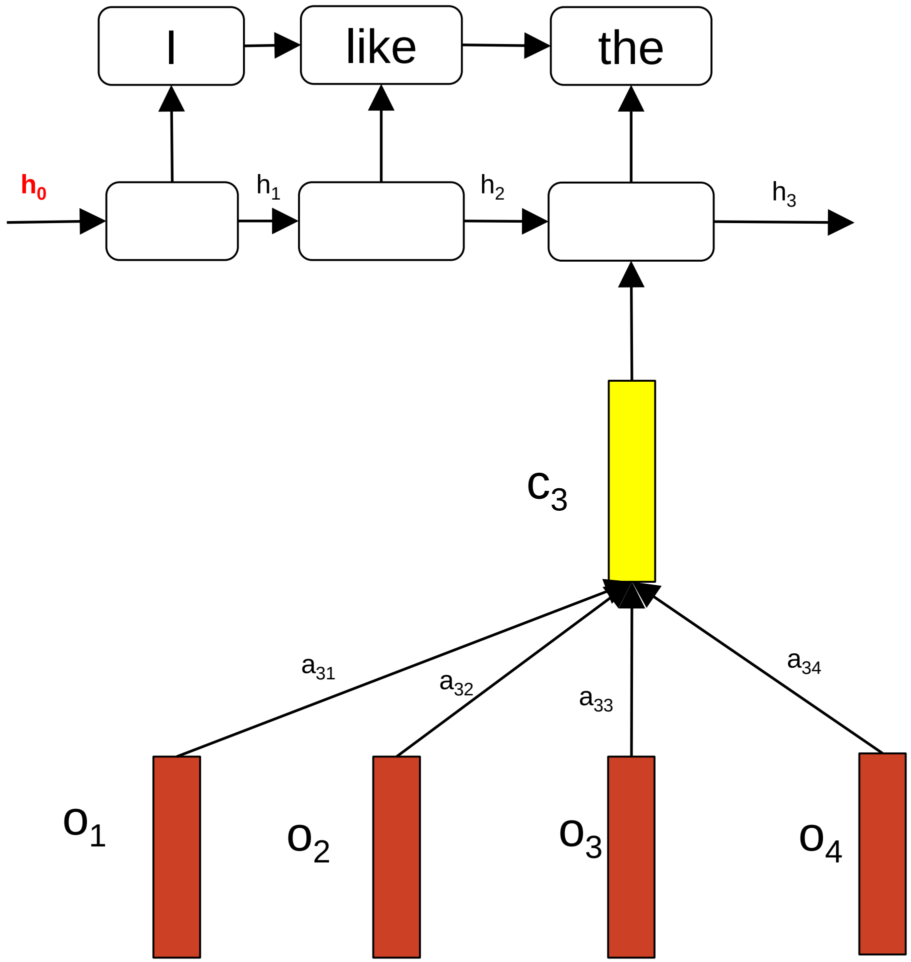


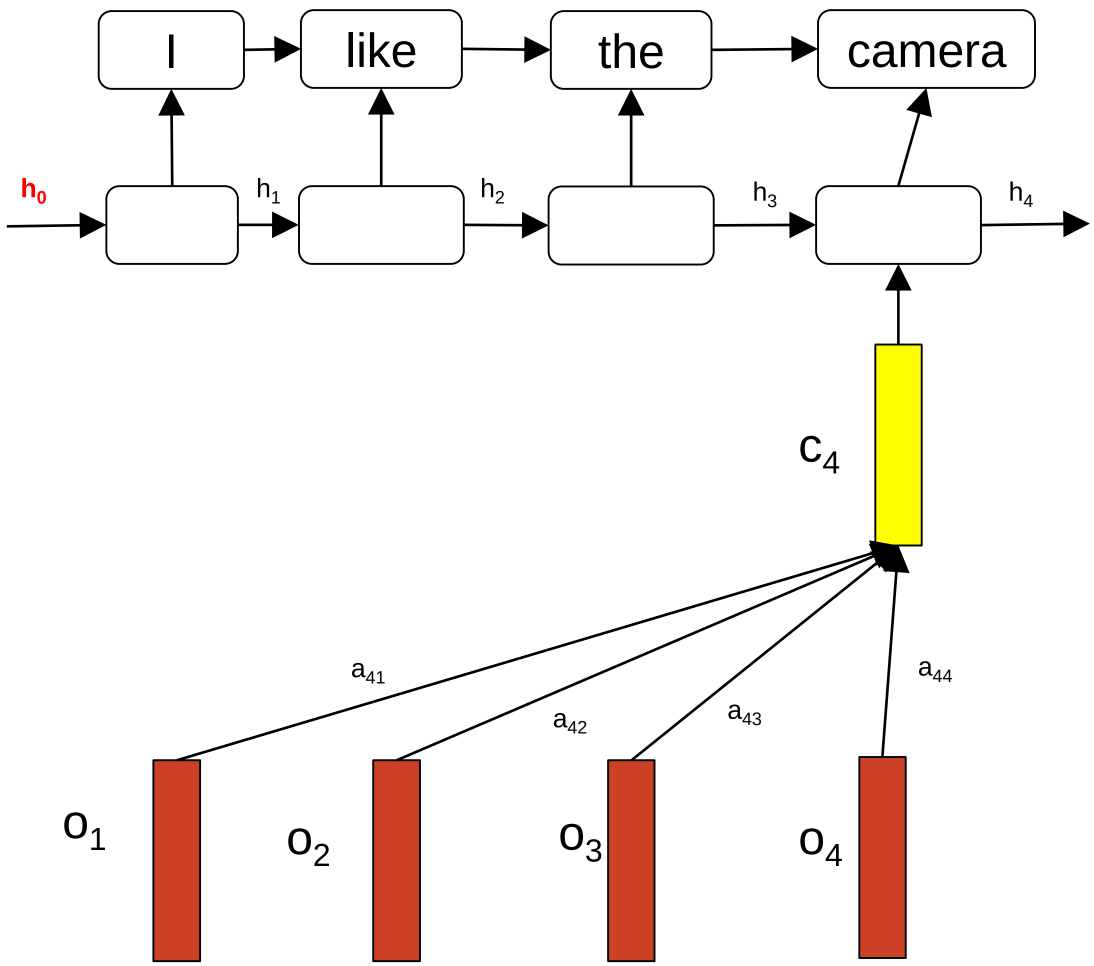
E.g. Sentiment Analysis

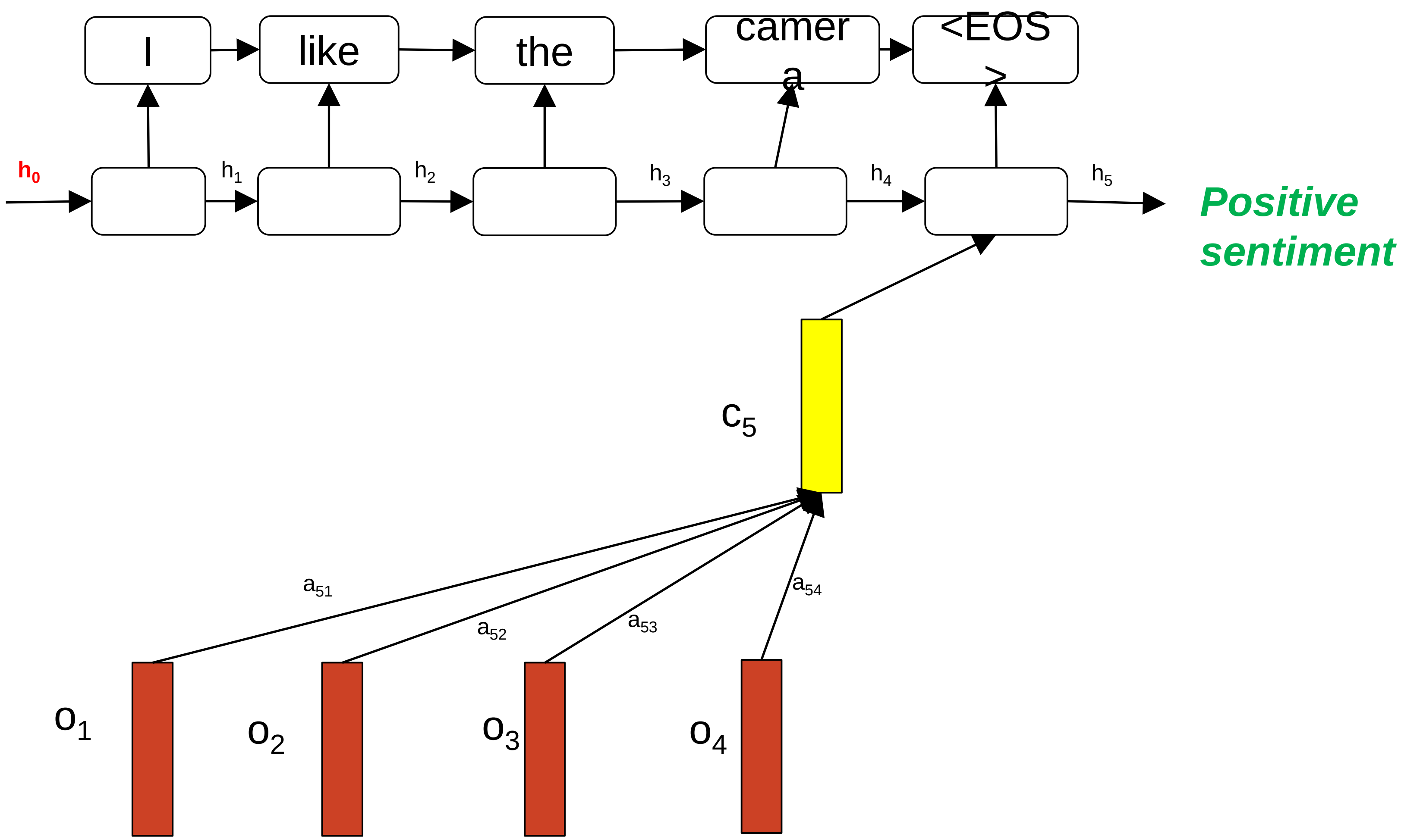


Decision on a piece of text

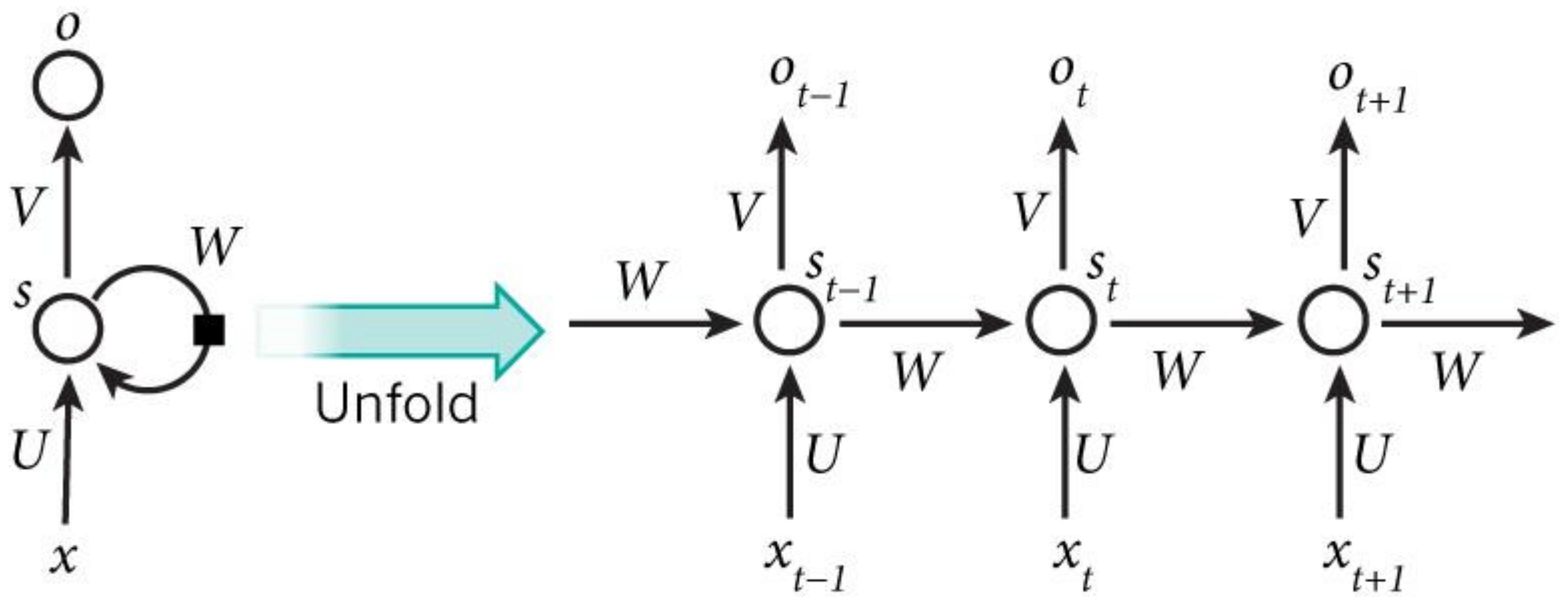

I
like
h_0
h_1
h_2
c_2
a_{21}
a_{22}
a_{23}
a_{24}
o_1
o_2
o_3
o_4


I
like
the
h_0
h_1
h_2
h_3
c_3
a_{31}
a_{32}
a_{33}
a_{34}
o_1
o_2
o_3
o_4


I
like
the
camera
h_0
h_1
h_2
h_3
h_4
c_4
a_{41}
a_{42}
a_{43}
a_{44}
o_1
o_2
o_3
o_4


I
like
the
camer a
<EOS >
h0
h1
h2
h3
h4
h5
Positive sentiment
c5
a51
a52
a53
a54
o1
o2
o3
o4

Back to RNN model



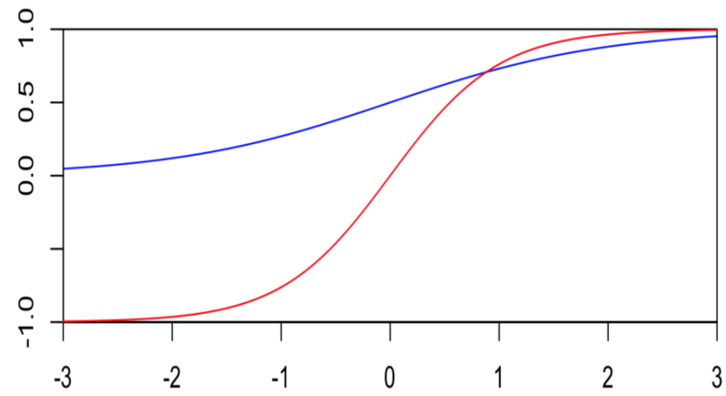
Notation: input and state

- x_t is the input at time step t. For example, could be a one-hot vector corresponding to the second word of a sentence.
- s_t is the hidden state at time step t. It is the "memory" of the network.
- $s_t = f(U.x_t + Ws_{t-1})$ ***U* and *W* matrices are learnt**
- *f* is a function of the input and the previous state
- Usually *tanh* or *ReLU* (approximated by *softplus)*

Tanh, *ReLU* (rectifier linear unit) and Softplus

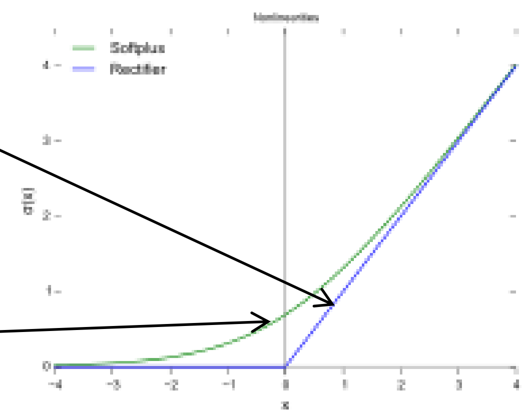
$$\tanh = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

tanh =



$$f(x) = \max(0, x)$$

$$g(x) = \ln(1 + e^{x})$$



Notation: output

- o_t is the output at step t

- For example, if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary

- $o_t = softmax(V.s_t)$

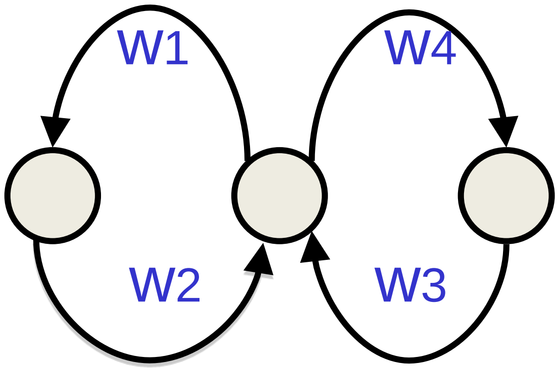
Operation of RNN

- RNN shares the same parameters (U, V, W) across all steps

- Only the input changes

- Sometimes the output at each time step is not needed: e.g., in sentiment analysis
- Main point: the hidden states !!

The equivalence between feedforward nets and recurrent nets



Assume that there is a time delay of 1 in using each connection.

The recurrent net is just a layered net that keeps reusing the same weights.