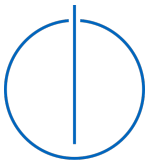


Faculty for Informatics

Technical
University
of Munich

Natural Language Processing
IN2361

PD Dr. Georg Groh

Social Computing
Research Group

Deep NLP
Part B: Simple Examples

- content is based on [2](lectures 7,8,9)
- certain elements (e.g. figures, equations or tables) were taken over or taken over in a modified form from [2]
- citations of [2] are omitted for legibility; citations of original sources cited in [2] are omitted for legibility
- errors on these slides are fully in the responsibility of Georg Groh
- BIG thanks to Richard Socher and his colleagues at Stanford for publishing materials [2] of a great Deep NLP lecture

Example 1: Neural Dependency Parsing

Repetition: Constituency Parsing vs Dependency Parsing

- Repetition from parsing chapters: different flavours of parsing: constituency parsing and dependency parsing:

- constituency parsing: phrase structure organizes words into nested constituents ←→ phrase structure grammar = CF-grammar

Basic unit: words

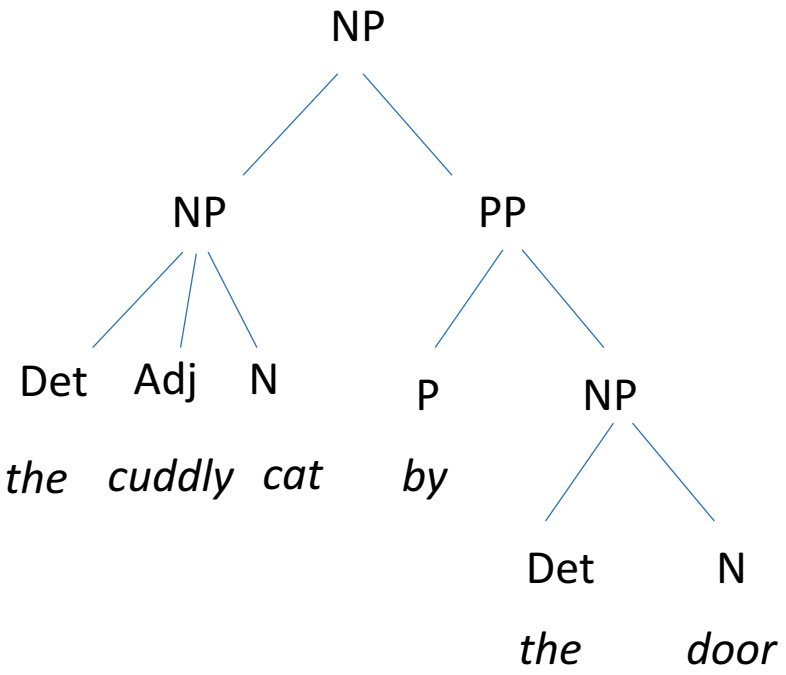
the, cat, cuddly, by, door
Det N Adj P N

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

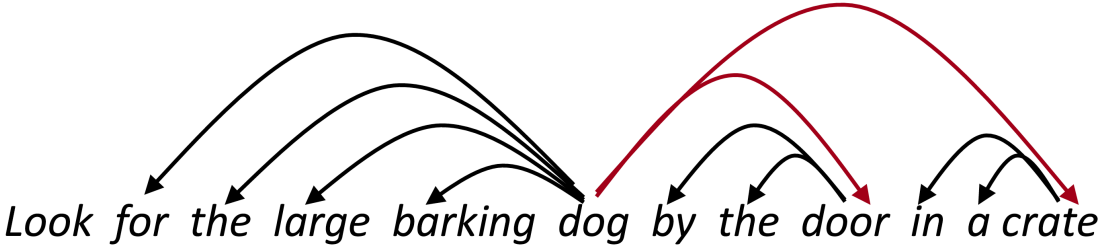


Repetition: Constituency Parsing vs Dependency Parsing

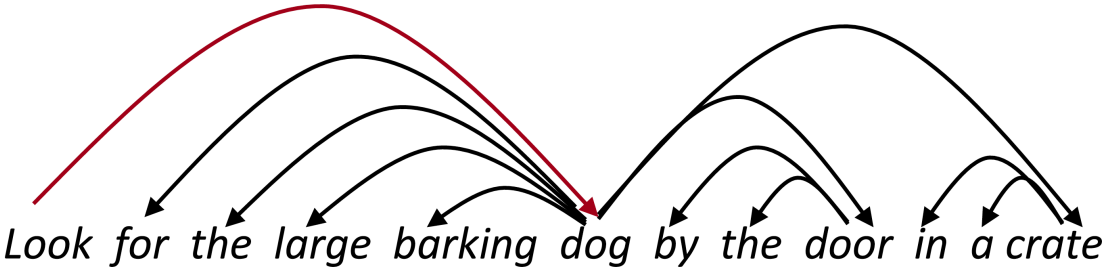
- dependency parsing: determine which words depend on (modify or are arguments of) which other words

Determiners, adjectives, and (sometimes) verbs modify nouns

prepositions modify nouns



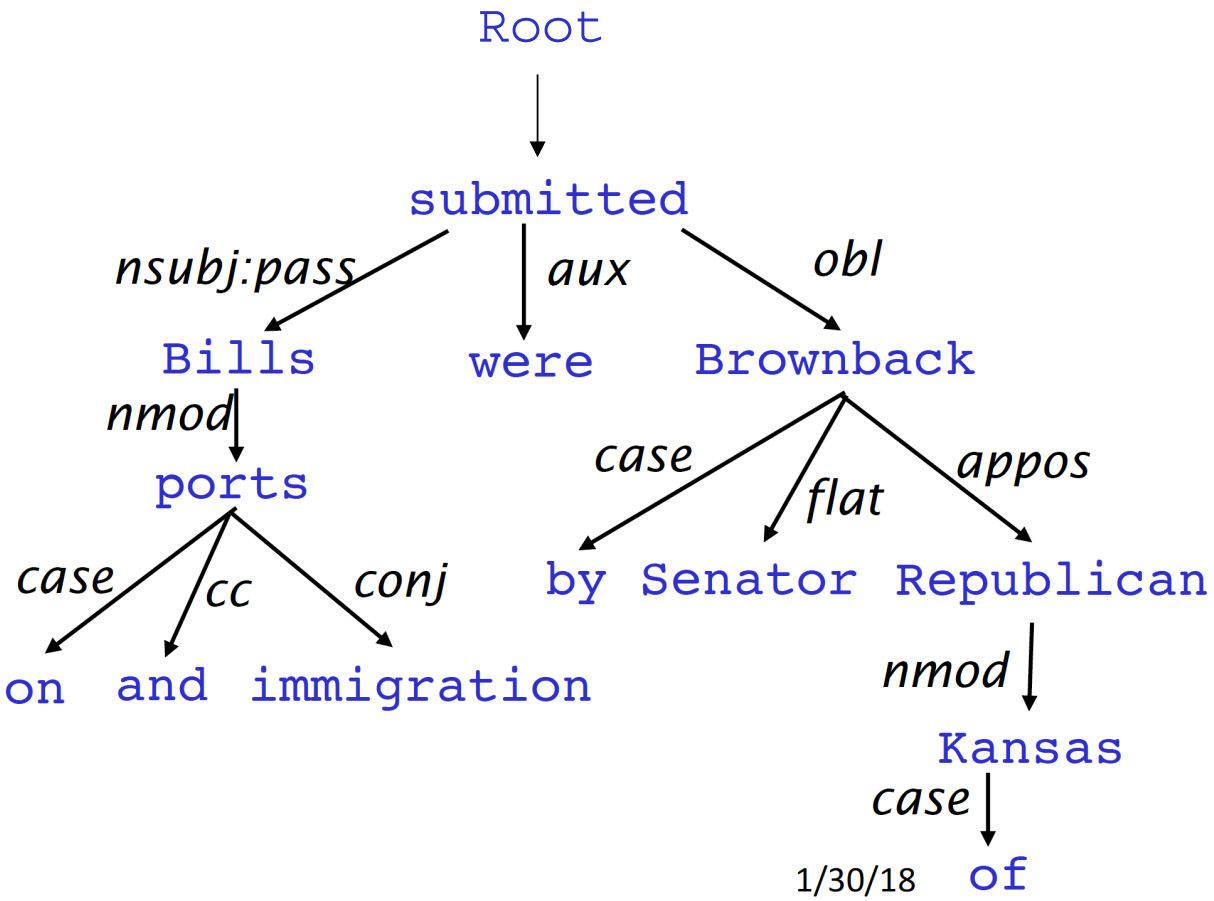
prepositional phrases modify the main noun phrase



main noun phrase is an argument of “look”

Dependency Parsing

- binary, asymmetric relations ("dependencies") (usually labelled with with name of grammatical relations (subject, prepositional object, apposition, etc.)) between head and dependent (modifier, subordinate)



Greedy Transition-Based Dependency Parsing

repetition from dependency parsing chapter:
components of parser

- stack σ, (top to the right)
- buffer β, (top to the left)
- set of dependencies A

- start state: (σ = [Root], β = [w_1, … , w_n] , A = ∅)
- state: (σ, β, A)
- end state: (σ = [Root], β = [] , A = final set of dependencies)

- set of actions (leftArc, rightArc: for each possible label):
  - sh shift: ([σ], [w_i,β], A) → ([σ w_i], [β], A)
  - leftArc.label: ([σ w_i w_j], [β], A) → ([σ w_j], [β], A ∪ {(w_j, w_i).label})
  - rightArc.label: ([σ w_i w_j], [β], A) → ([σ w_i], [β], A ∪ {(w_i, w_j).label})

Greedy Transition-Based Dependency Parsing

greedy transition-based **parsing process:** (no search) (Arc Standard)

- based on binary (sparse, dim: $\approx 10^7$) encoded features of parser-state (σ, β, A) (e.g. top of stack word, POS; first word in buffer, POS of top of stack, etc.):
  let discriminative classifier (SVM, logistic regression etc.) decide on next action:

s1.w=*good* ∧ s1.t=JJ ∧ s2.w=*has* ∧ s2.t=VBZ ∧ lc(s2).t=PRP ∧ lc(s2).w=*He*

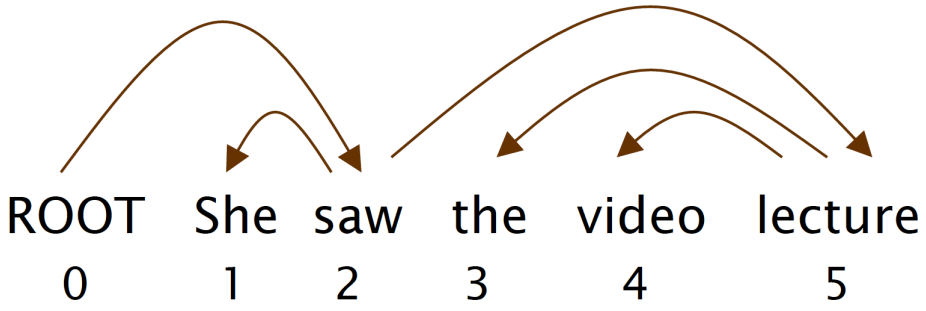
⇩

(0,0,1,0,...,0,0,1,0,0,0,0,...,0,0,0,1,0,0,...,0,1,0,...,0,0,0,1,0,...,0,1,0,0,...,0)

⇩ classifier

(P(sh),..., P(le.label$_1$), P(le.label$_2$),..., P(ri.label$_1$), P(ri.label$_2$),...)

Dependency Parsing: Evaluation



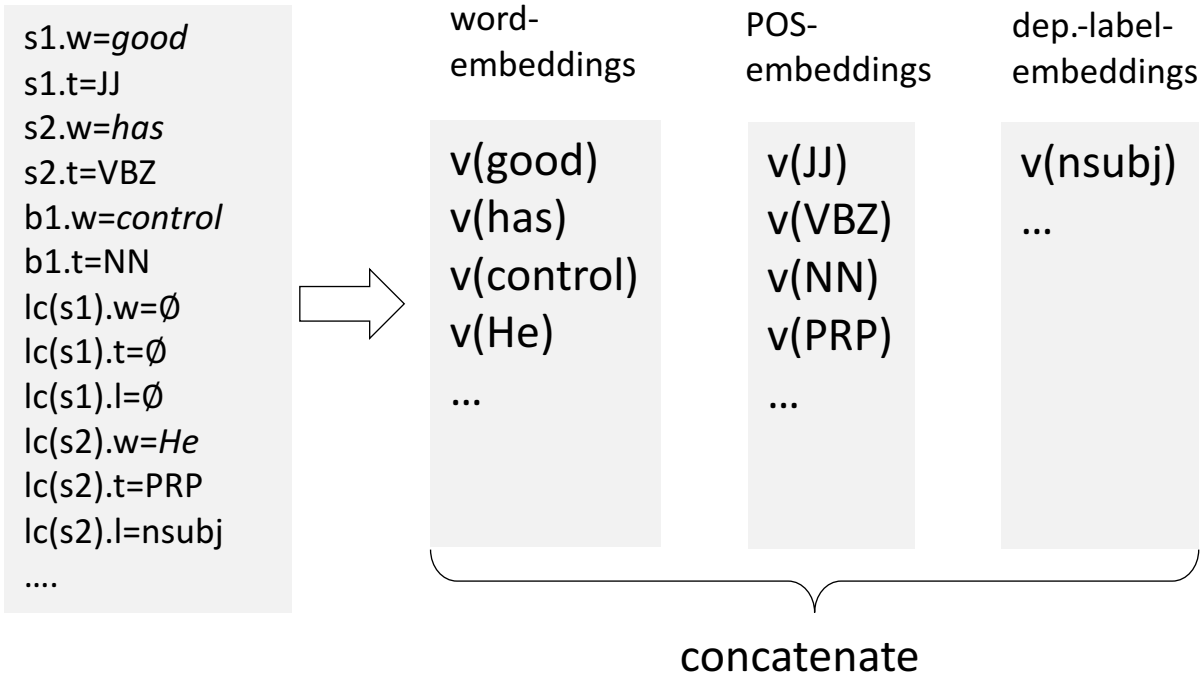
Ground Truth		
(1,2)	She	nsubj
(2,0)	saw	root
(3,5)	the	det
(4,5)	video	nn
(5,2)	lecture	obj

Parse Result				
			Unlabeled attachment score (UAS): = also head correct?	Labeled attachment score (LAS): = also head & label correct?
(1,2)	She	nsubj	yes	yes
(2,0)	saw	root	yes	yes
(3,4)	the	det	no	no
(4,5)	video	nsubj	yes	no
(5,2)	lecture	ccomp	yes	no
			UAS accuracy: 4/5=0.8	LAS accuracy: 2/5=0.2

NN-Based Dep. Parsers: Features

- already discussed: advantages: replace sparse binary features + discriminative classifier → embeddings for features (words, POS, dependency labels) + NN

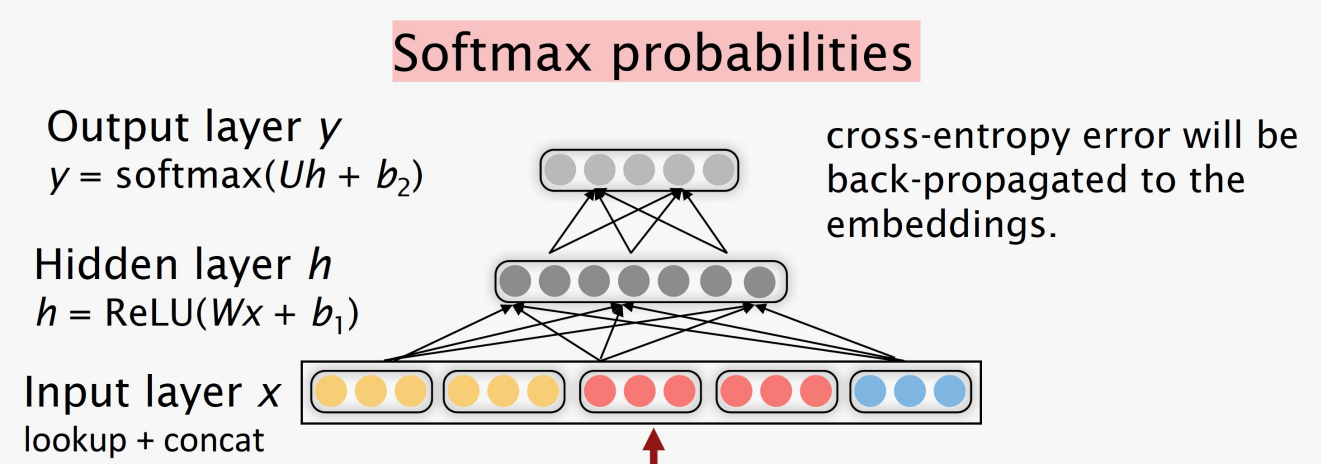
state: σ = [Root, has.VBZ, good.JJ], β = [control.NN,…] , A ={(has.VBZ, HE.PRP).nsubj,…}



NN-Based Dep. Parsers: Features

- POS embeddings and dependency label embeddings:
  desired: (→ train accordingly ☺): examples
  - POS: NNS (plural noun) should be close to NN (singular noun)
  - dependency label: num (numerical modifier should be close to amod (adjective modifier)

- simple NN architecture:



NN-Based Dep. Parsers: Performance

- performance of first NN dep.parser [Chen & Manning, 2014, in [2]] vs traditional methods

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**

- more advanced NN architectures (more layers, beam search, hyper-parameter tuning, CRF layer as final classification layer etc.):

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

NN-Based Dep. Parsers: Projectivity

- projective (crossing) arcs necessary, but greedy transition based approach does not deliver them

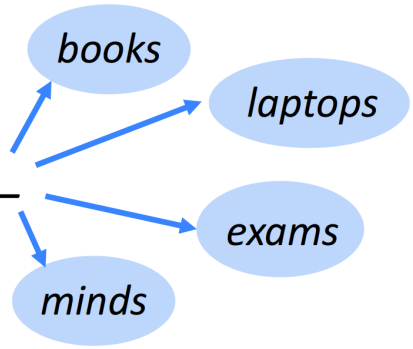
solution: → use graph-based NN-parser with search (see [1] (previous slide-set))

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**

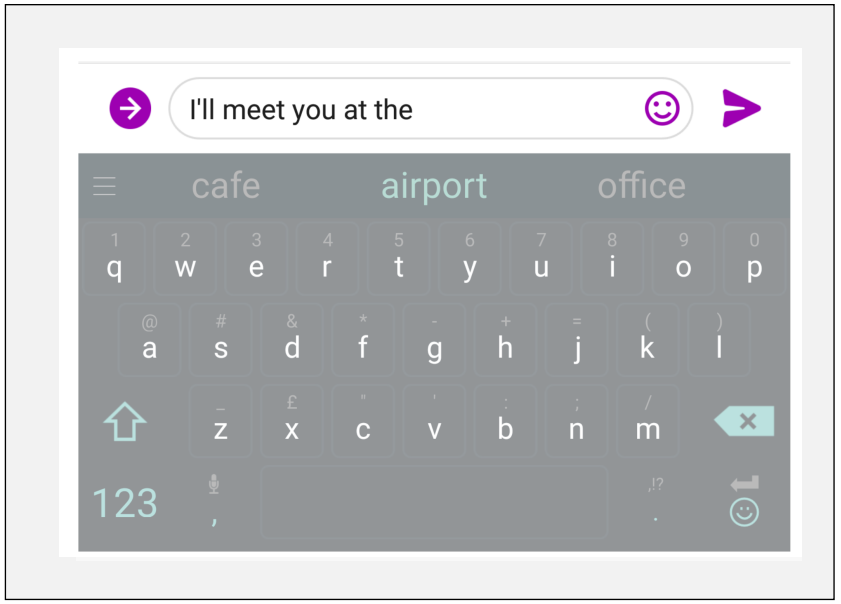
Example 2: Neural Language Modelling

Language Modelling

the students opened their _____



$$P(\boldsymbol{x}^{(t+1)} = \boldsymbol{w}_j \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$



Language Modelling: N-Gram Models

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})$$

$$= \frac{P(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{P(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$

$$\approx \frac{\text{count}(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{\text{count}(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$

Language Modelling: N-Gram Models: Problems

Sparsity Problem 1

Problem: What if *"students opened their* $\boldsymbol{w}_j$*"* never occurred in data? Then $\boldsymbol{w}_j$ has probability 0!

(Partial) Solution: Add small δ to count for every $\boldsymbol{w}_j \in V$. This is called *smoothing*.

$$P(\boldsymbol{w}_j|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w}_j)}{\text{count}(\text{students opened their})}$$

Sparsity Problem 2

Problem: What if *"students opened their"* never occurred in data? Then we can't calculate probability for *any* $\boldsymbol{w}_j$!

(Partial) Solution: Just condition on *"opened their"* instead. This is called *backoff*.

Note: Increasing *n* makes sparsity problems *worse*. Typically we can't have *n* bigger than 5.

Fixed Window NN Language Models

~~as the proctor started the clock~~ discard

the students opened their _____

fixed window

output distribution

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

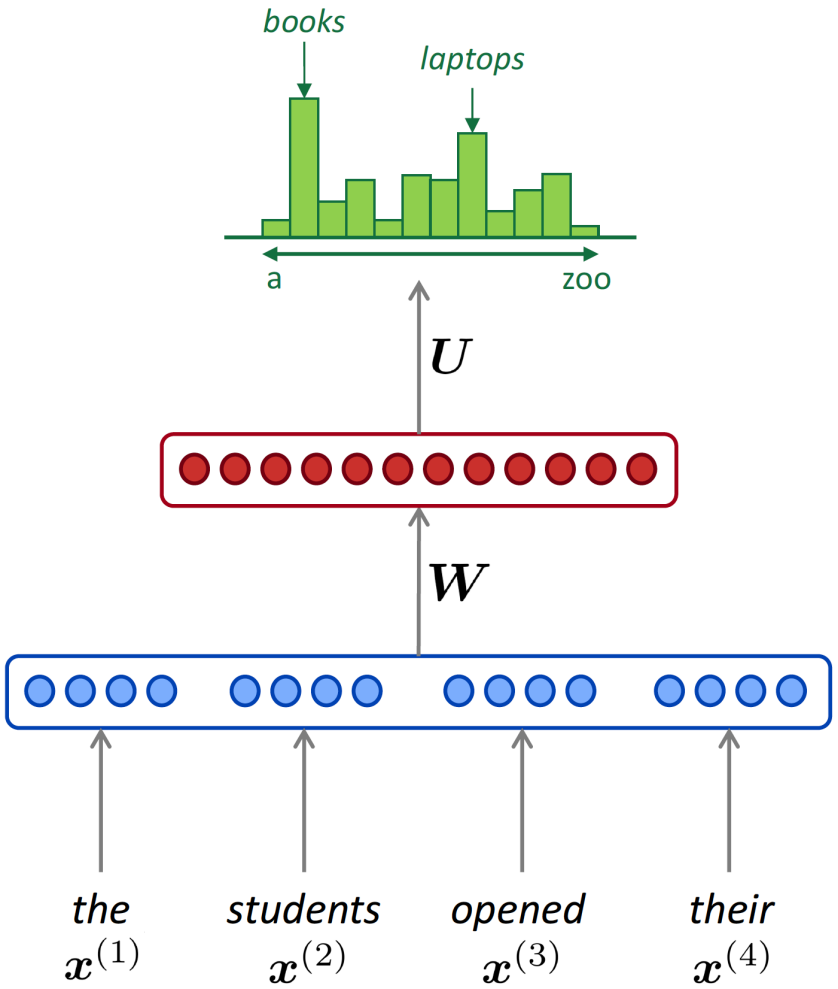
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



Fixed Window NN Language Models

~~as the proctor started the clock~~ discard | the students opened their _____ fixed window

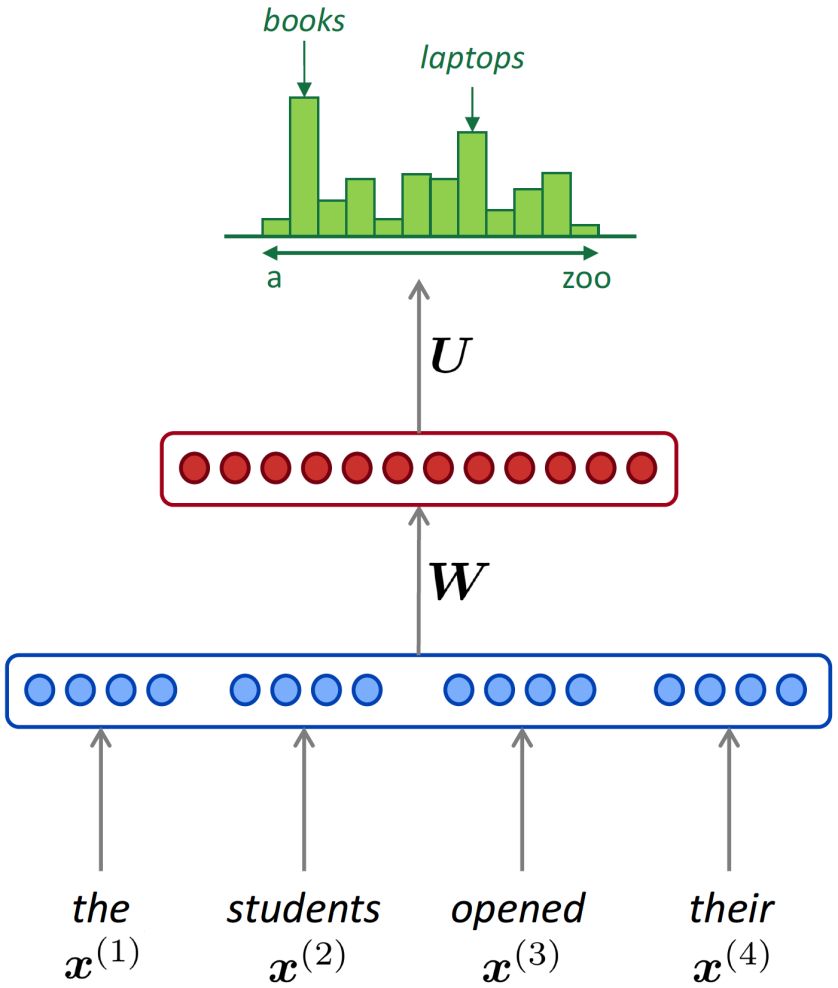
Improvements over *n*-gram LM:

- No sparsity problem
- Model size is O(*n*) not O(exp(*n*))

Remaining **problems**:

- Fixed window is too small
- Enlarging window enlarges $\boldsymbol{W}$
- Window can never be large enough!
- Each $\boldsymbol{x}^{(i)}$ uses different rows of $\boldsymbol{W}$. We don't share weights across the window.

We need a neural architecture that can process *any length input*

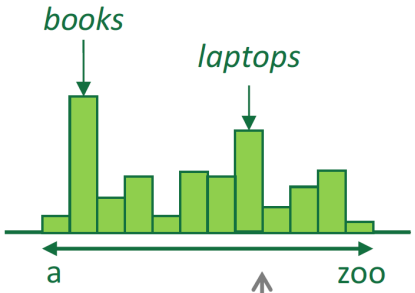


→ RNN Language Model

$$\hat{\boldsymbol{y}}^{(4)} = P(\boldsymbol{x}^{(5)}|\text{the students opened their})$$

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$



hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

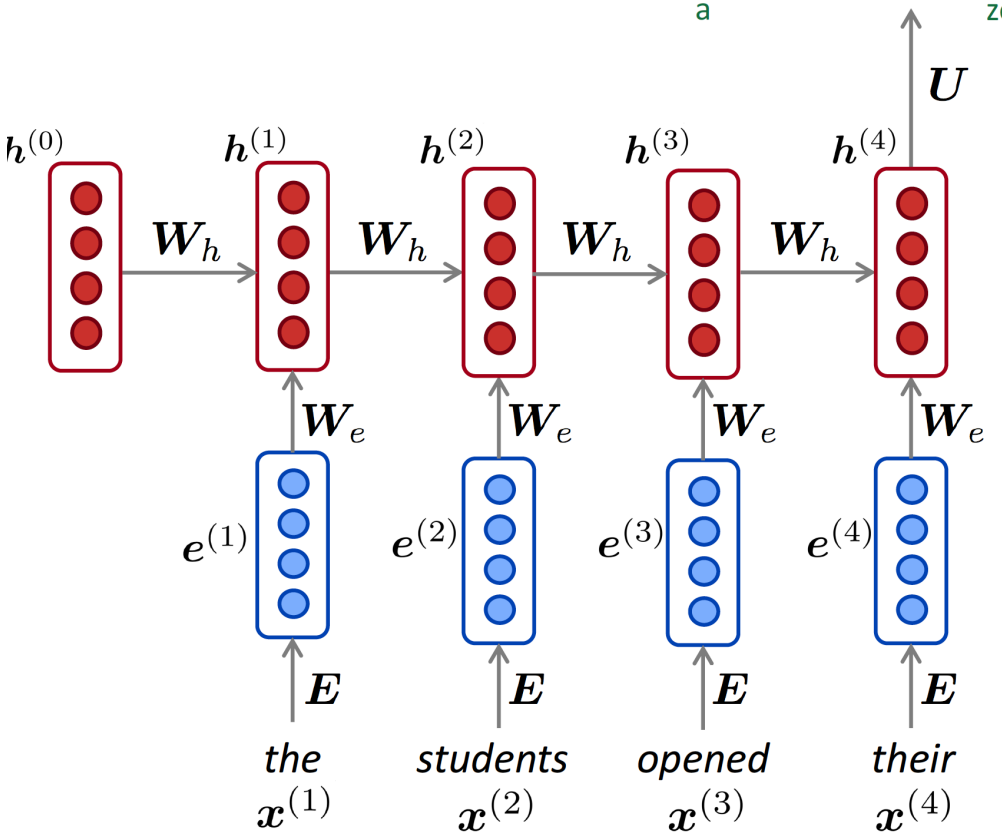
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$



***Note**: this input sequence could be much longer, but this slide doesn't have space!*

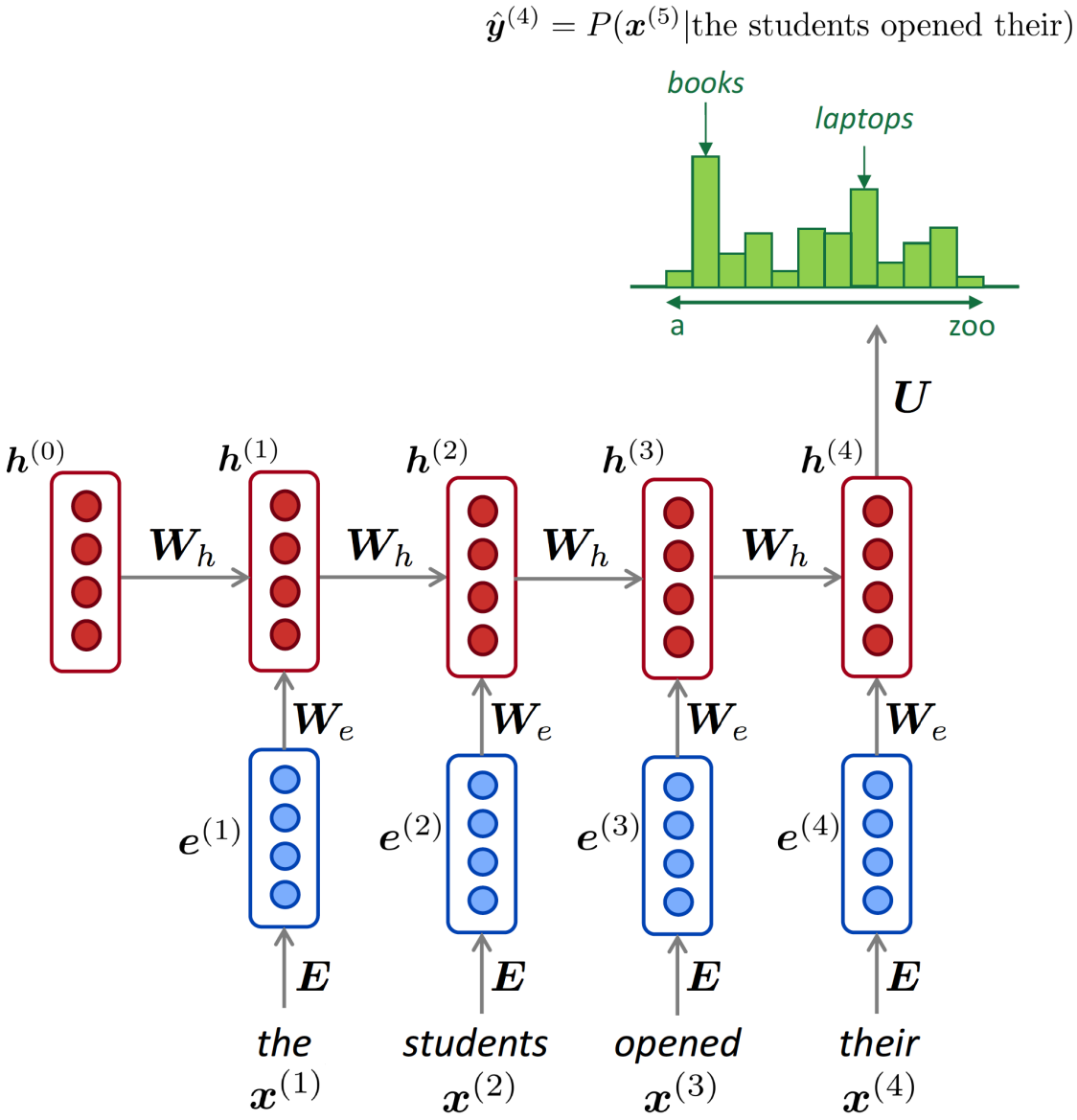
→ RNN Language Model

RNN **Advantages**:

- Can process any length input
- Model size doesn't increase for longer input
- Computation for step *t* can (in theory) use information from many steps back
- Weights are shared across timesteps → representations are shared

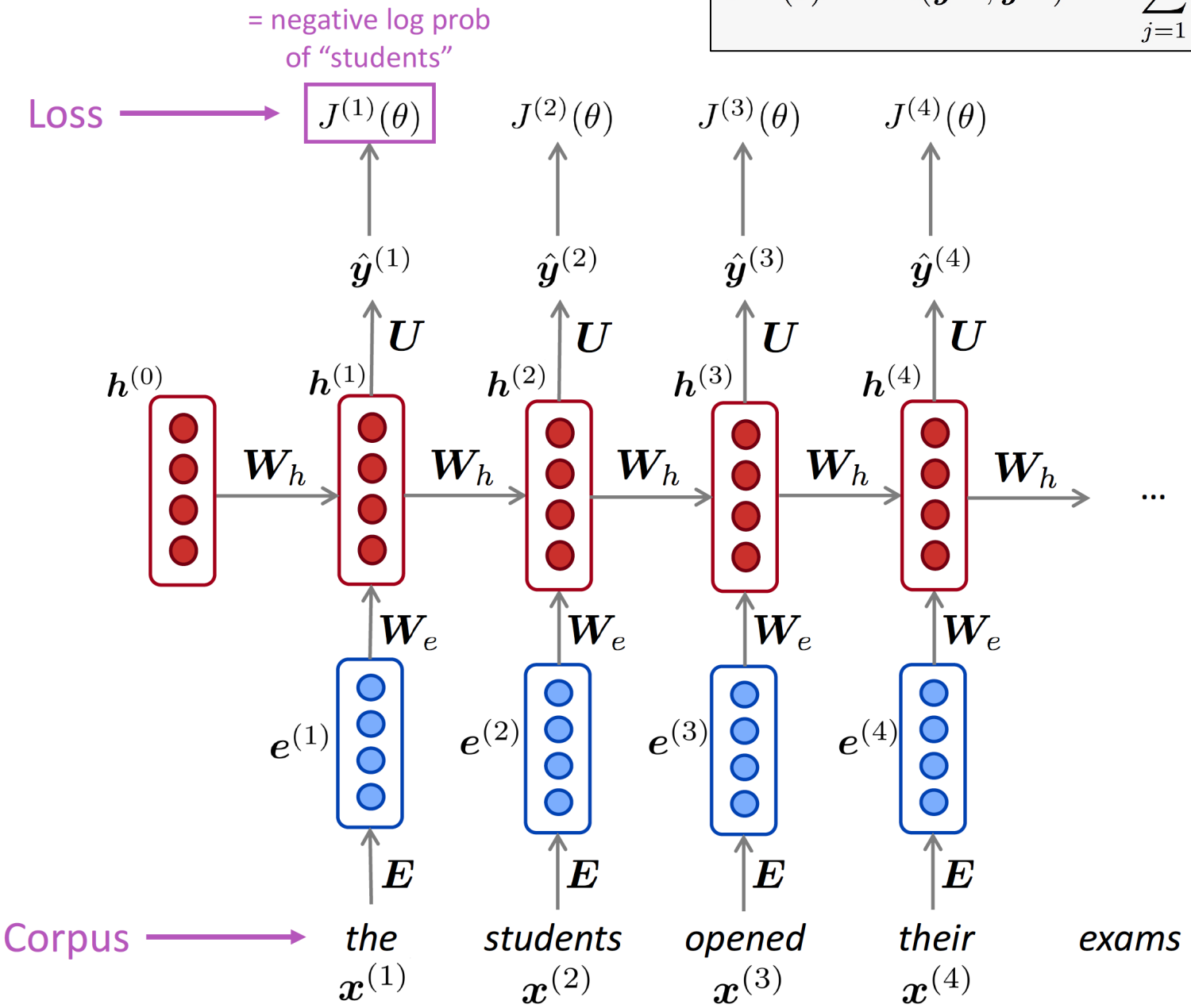
RNN **Disadvantages**:

- Recurrent computation is slow
- In practice, difficult to access information from many steps back

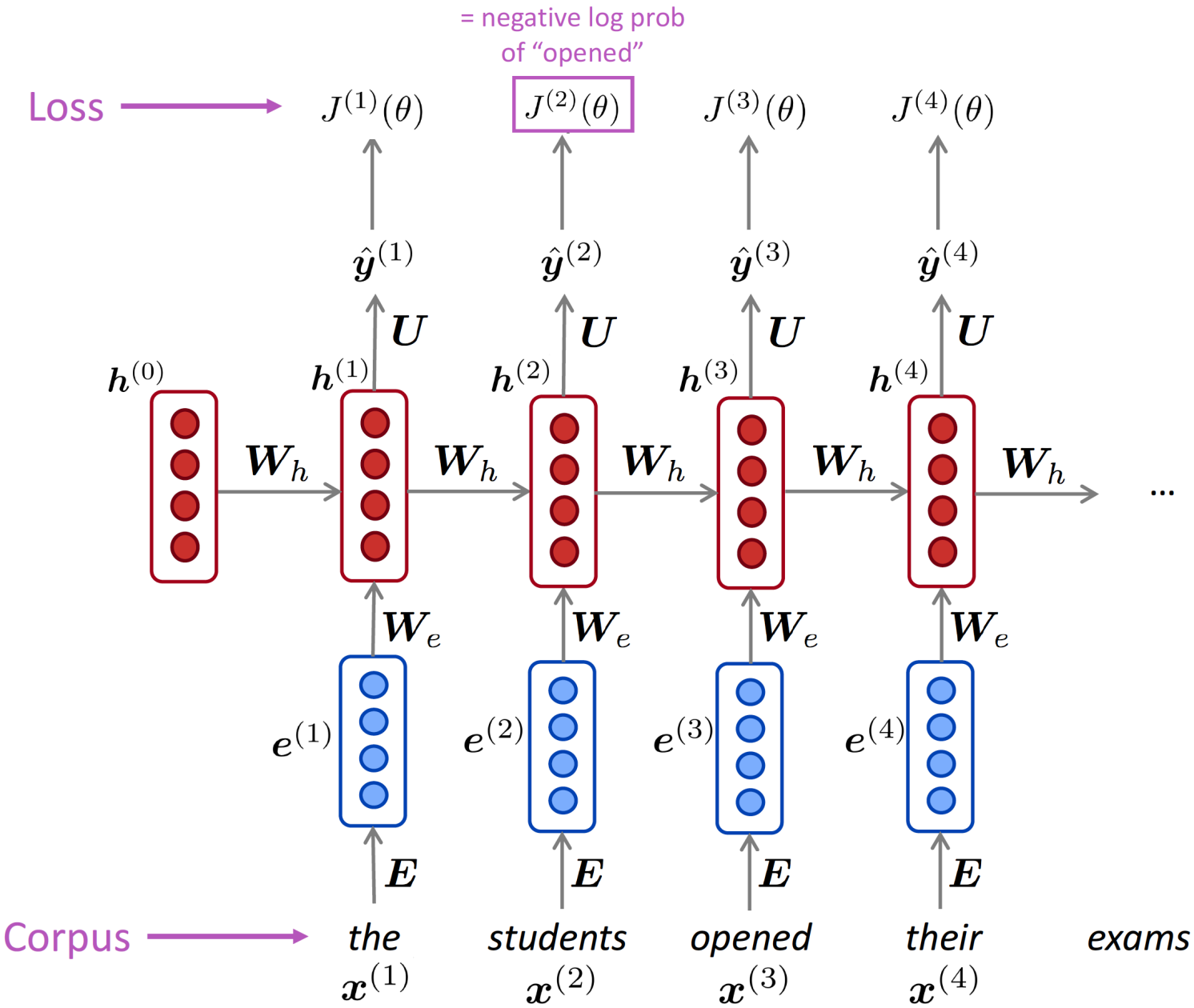


RNN LM: Training

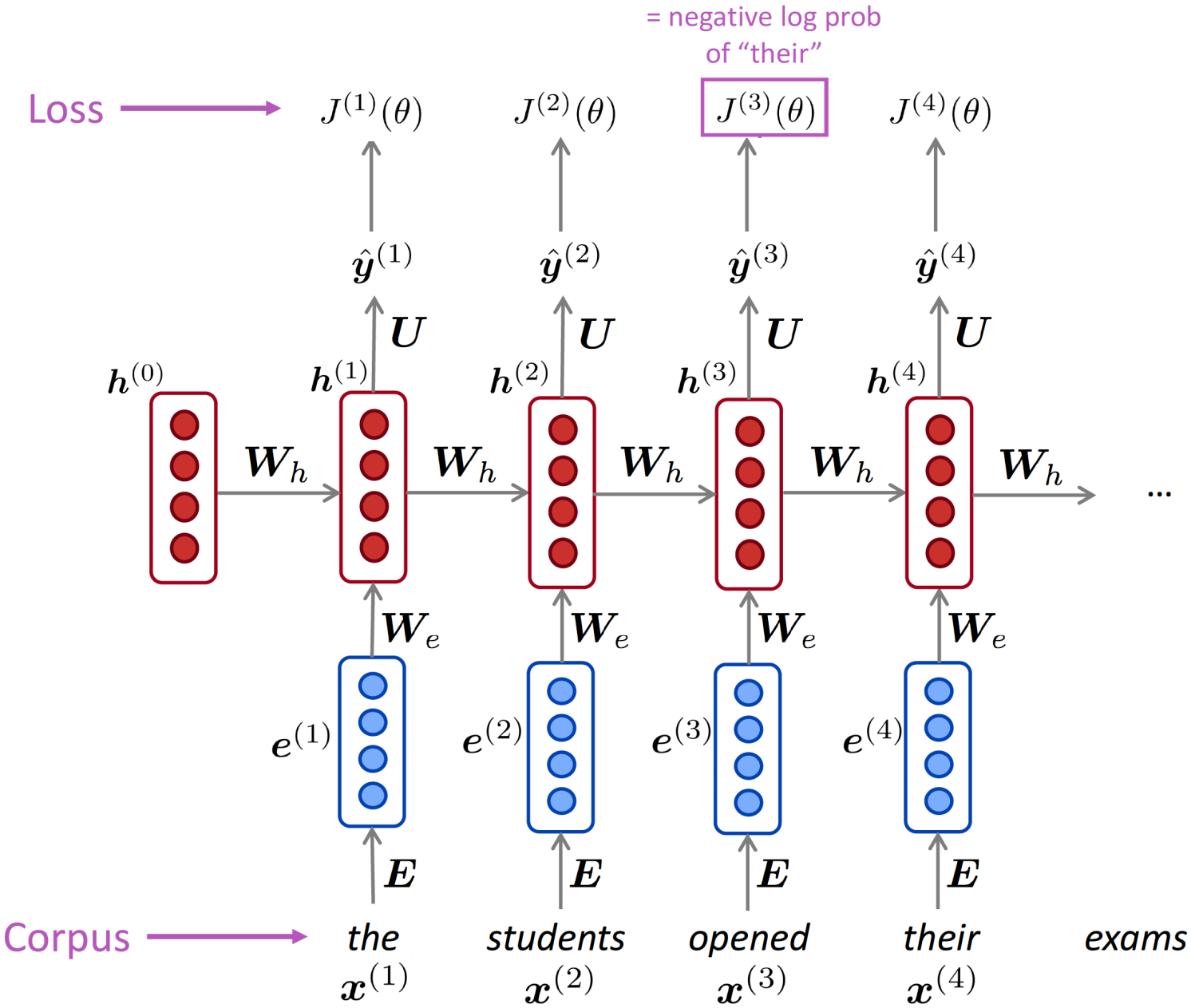
$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{j=1}^{|V|} y_j^{(t)} \log \hat{y}_j^{(t)}$$



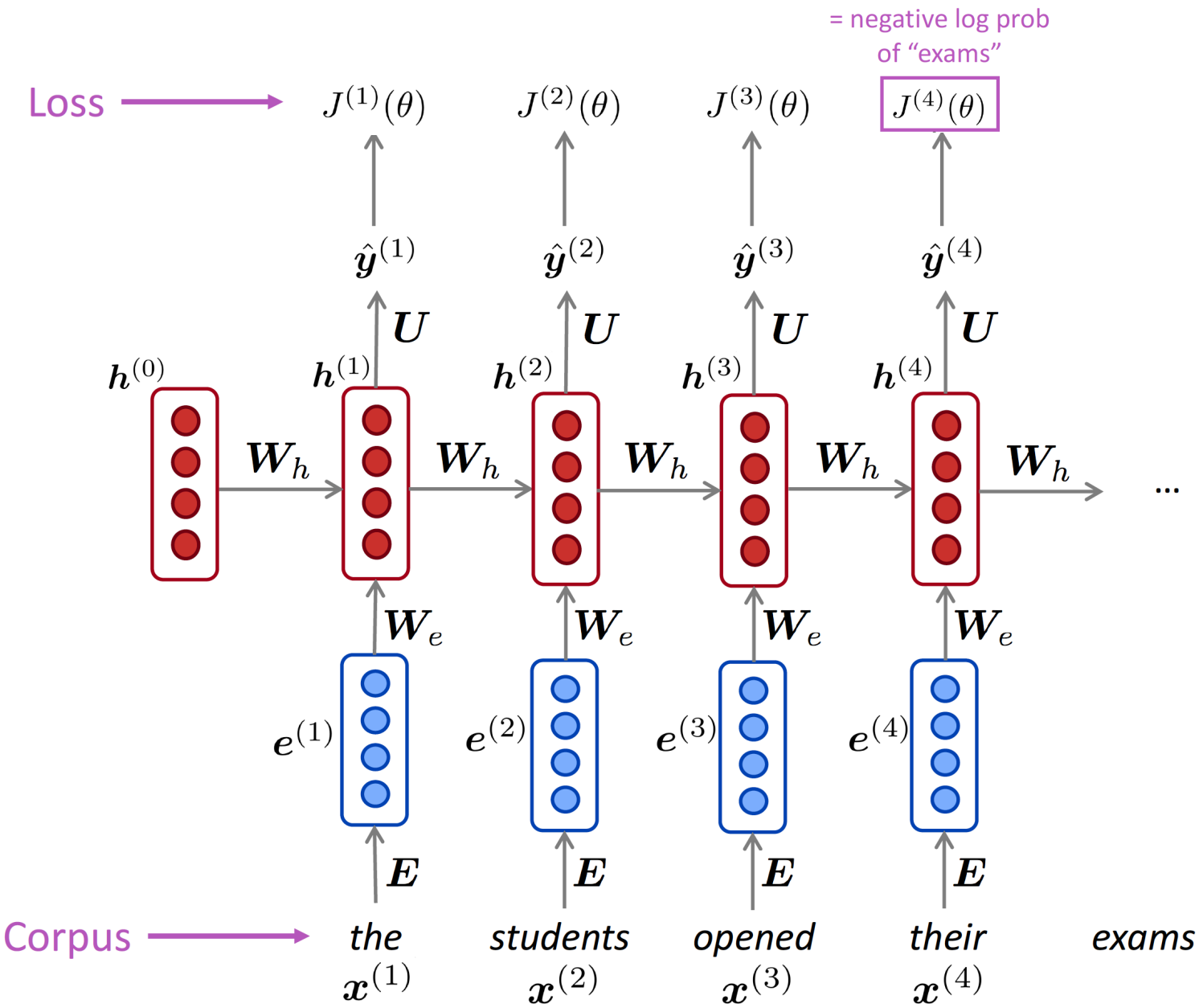
RNN LM: Training



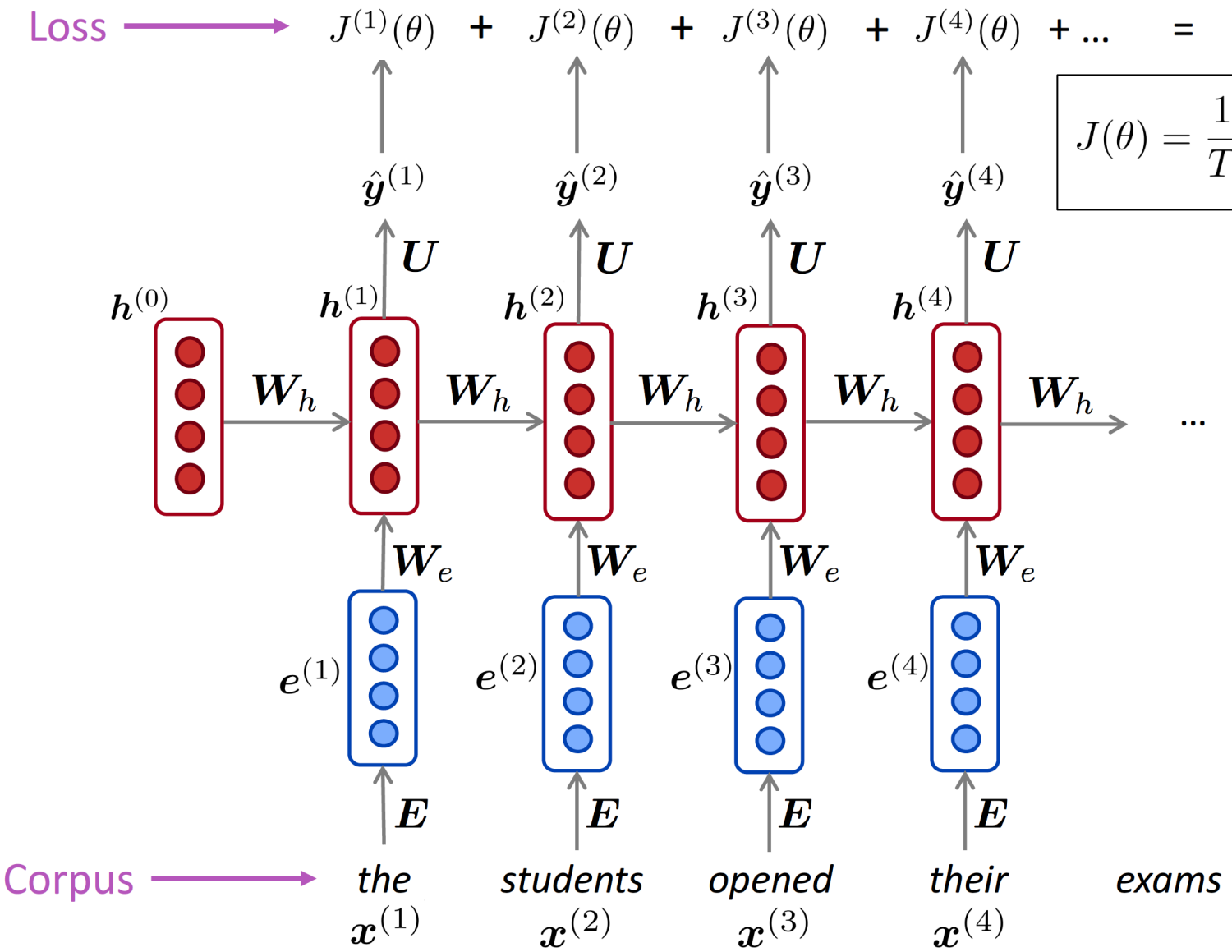
RNN LM: Training



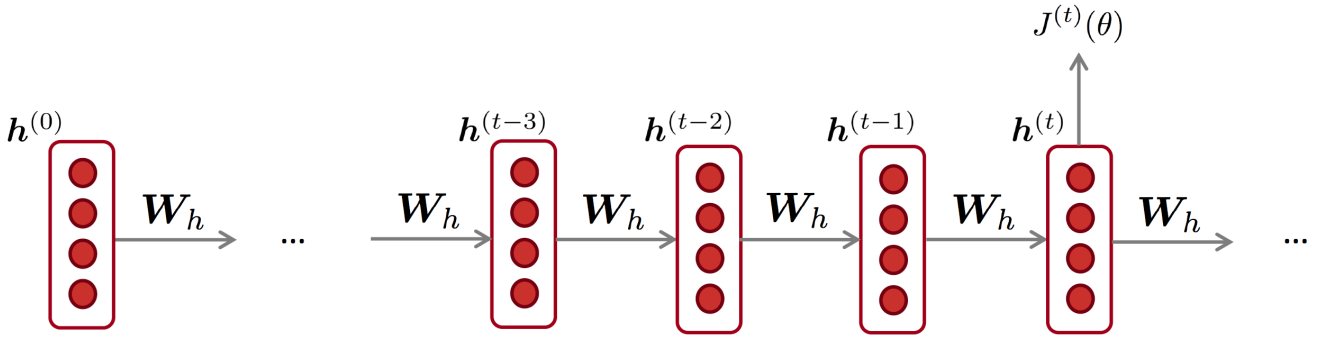
RNN LM: Training



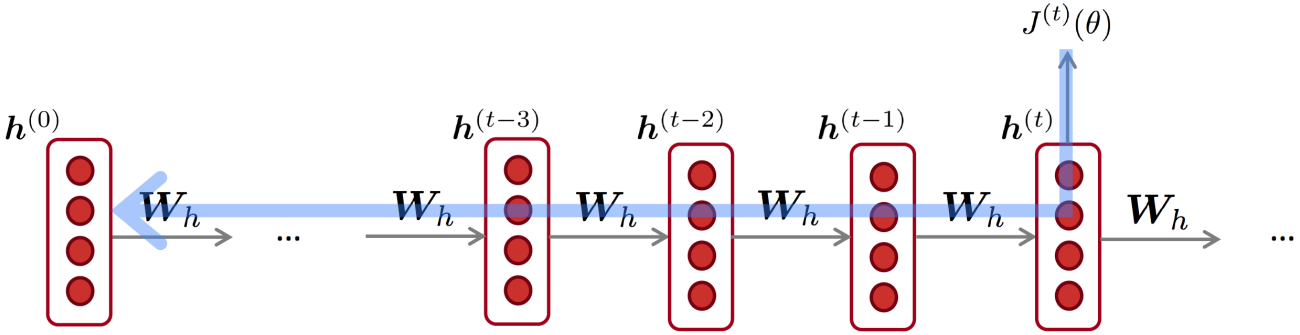
RNN LM: Training



RNN LM: BPTT



$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}} = \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}}\bigg|_{(i)}$$



RNN LM: Generating Text ☺

- Corpus: Obama speeches:

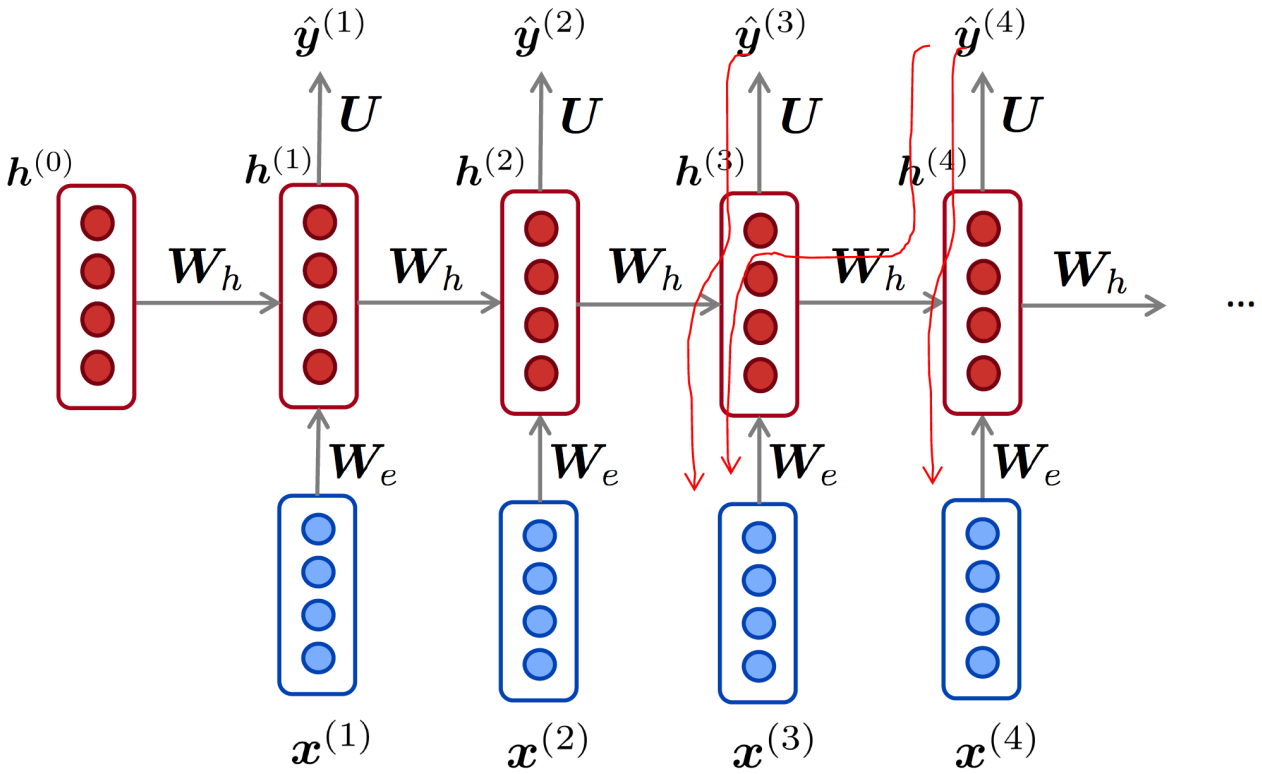
> *The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done.*

- Corpus: Harry Potter:

> "Sorry," Harry shouted, panicking—"I'll leave those brooms in London, are they?"
>
> "No idea," said Nearly Headless Nick, casting low close by Cedric, carrying the last bit of treacle Charms, from Harry's shoulder, and to answer him the common room perched upon it, four arms held a shining knob from when the spider hadn't felt it seemed. He reached the teams too.

Example 3 / Case Study 3: Vanishing Gradients

Simple RNN: Vanishing Gradients



$$h^{(t)} = W_h f\left(h^{(t-1)}\right) + W_e x^{(t)}$$
$$\hat{y}^{(t)} = U\, f(h^{(t)})$$

$$h_t = W f(h_{t-1}) + W^{(hx)} x_{[t]}$$
$$\hat{y}_t = W^{(S)} f(h_t)$$

multiply same matrix at each BPTT step → possibly leads to vanishing gradient; problem: possibly important contributions to / from the past vanish

Simple RNN: Vanishing Gradients

$$
\begin{aligned}
h_t &= W f(h_{t-1}) + W^{(hx)} x_{[t]} \\
\hat{y}_t &= W^{(S)} f(h_t)
\end{aligned}
$$

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

Simple RNN: Vanishing Gradients

$$
\begin{aligned}
h_t &= W f(h_{t-1}) + W^{(hx)} x_{[t]} \\
\hat{y}_t &= W^{(S)} f(h_t)
\end{aligned}
$$

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

this may become very small (or very large)

Simple RNN: Vanishing Gradients

$$
\begin{aligned}
h_t &= W f(h_{t-1}) + W^{(hx)} x_{[t]} \\
\hat{y}_t &= W^{(S)} f(h_t)
\end{aligned}
$$

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

this may become very small (or very large)

chain rule:

$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$$

Simple RNN: Vanishing Gradients

$$
\begin{aligned}
h_t &= W f(h_{t-1}) + W^{(hx)} x_{[t]} \\
\hat{y}_t &= W^{(S)} f(h_t)
\end{aligned}
$$

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

this may become very small (or very large)

chain rule: $$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$$

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq \|W^T\| \|\mathrm{diag}[f'(h_{j-1})]\| := \beta_W \beta_h$$

Simple RNN: Vanishing Gradients

$$h_t = W f(h_{t-1}) + W^{(hx)} x_{[t]}$$
$$\hat{y}_t = W^{(S)} f(h_t)$$

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

this may become very small (or very large)

chain rule: $$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$$

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq \|W^T\| \|\mathrm{diag}[f'(h_{j-1})]\| := \beta_W \beta_h$$

$$\left\| \frac{\partial h_t}{\partial h_k} \right\| = \left\| \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq (\beta_W \beta_h)^{t-k}$$

Simple RNN: Vanishing Gradients

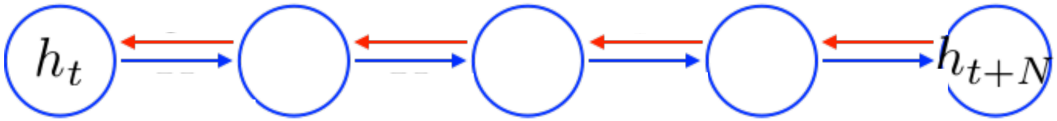
$$\left\| \frac{\partial h_t}{\partial h_k} \right\| = \left\| \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq (\beta_W \beta_h)^{t-k}$$

- observe this influence from / to the past go to zero: →
  - either there IS no such dependency OR
  - there is a wrong choice of parameters / architecture cannot model this influence properly

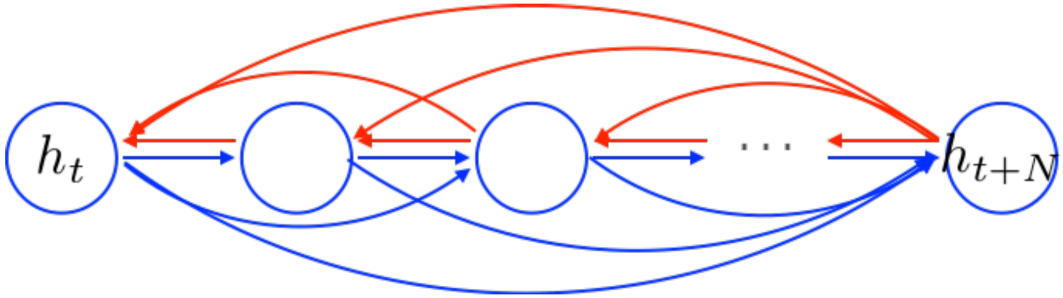
Contribution of Gated Nodes (GRU or LSTM)

- Simple RNN: $h_t = f\left(W^{(hh)} h_{t-1} + W^{(hx)} x_t\right)$

→ error must backpropagate through all intermediate nodes:



- idea: introduce shortcuts:



let gated units learn which shortcuts are necessary → 😊

Bibliography

(1) Yoav Goldberg: “A Primer on Neural Network Models for Natural Language Processing” http://www.cs.biu.ac.il/~yogo/nnlp.pdf (URL, May 2018) and at https://arxiv.org/abs/1510.00726v1 (URL, May 2018), 2015

(2) Richard Socher et al: “CS224n: Natural Language Processing with Deep Learning”, Lecture Materials (slides and links to background reading) http://web.stanford.edu/class/cs224n/ (URL, May 2018), 2018

Recommendations for Studying

- minimal approach:

  work with the slides and understand their contents! Think beyond instead of merely memorizing the contents

- standard approach:

  minimal approach + study the corresponding lecture slides from [2] for additional details omitted in our slides

- interested / deeply interested student's approach:

  standard approach + read some or all of the recommended background reading of [2] for lecture 10