Language Models – Basics - 2

``It appears then that a sufficiently complex stochastic process will give a satisfactory representation of a discrete source.''

``A second method is to delete a certain fraction of the letters from a sample of English text and then let someone attempt to restore them. If they can be restored when 50% are deleted the redundancy must be greater than 50%.''

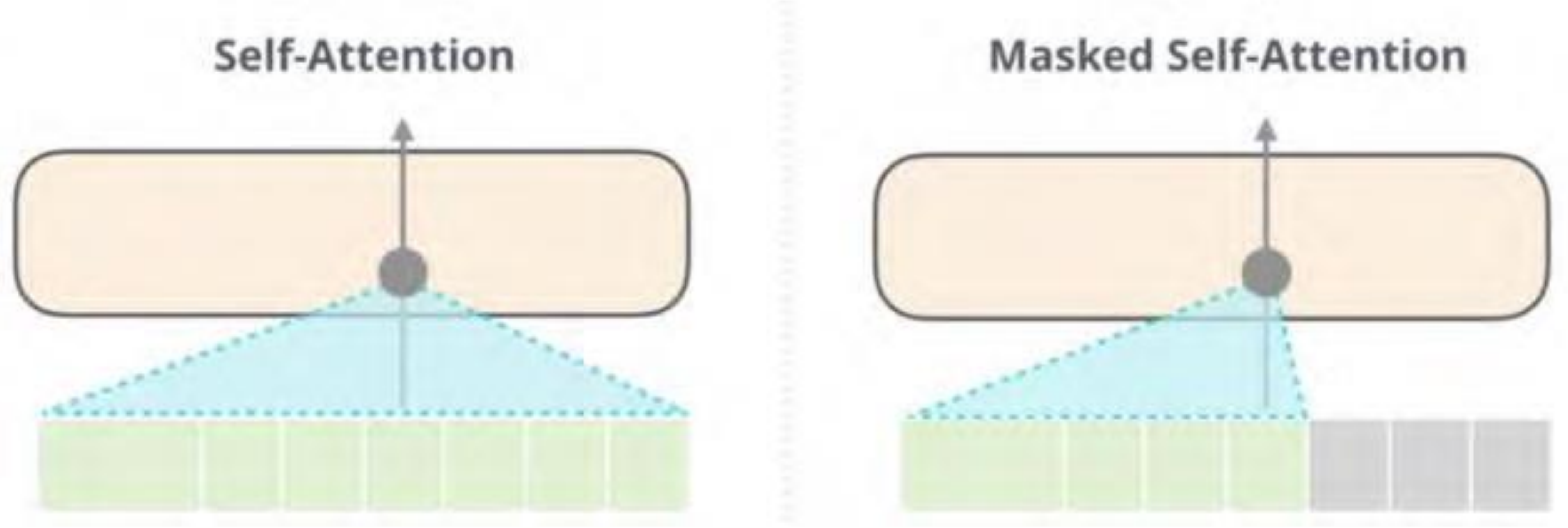
LMs: “fill in the blank”

- Think of this as a “fill in the blank” problem.

$$P(w_n|w_1, w_2, w_3, \ldots, w_{n-1})$$

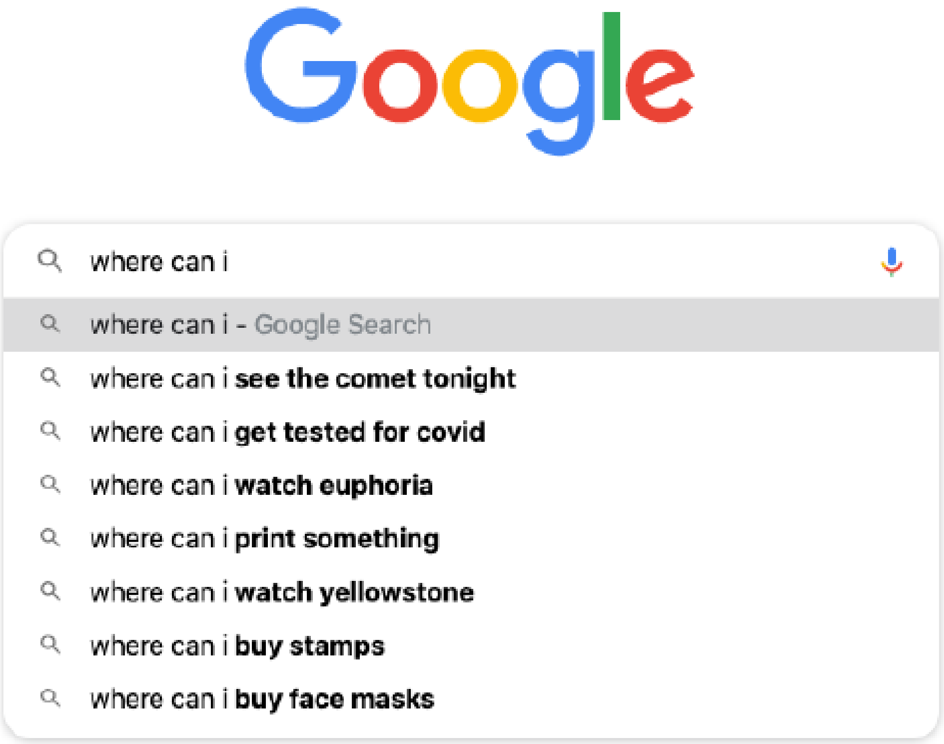
“He picked up the bat and hit the _____”

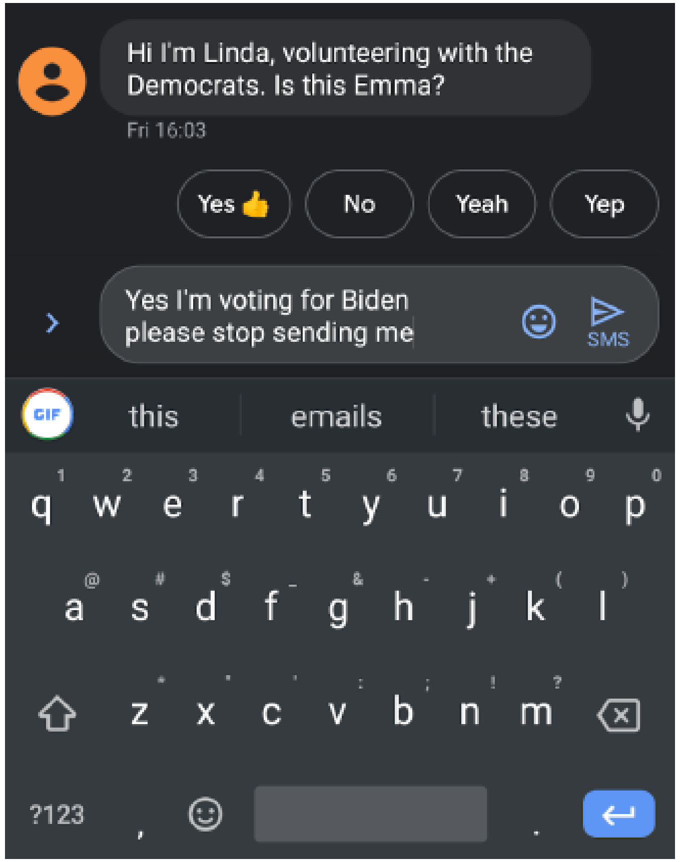
Ball? Poetry?


Self-Attention
Masked Self-Attention

Probabilistic language models

- Today's goal: assign a probability to a sentence. Why?





Probabilistic language models

- Today's goal: assign a probability to a sentence. Why?
  - Machine translation:
    P(*high winds tonight*) > P(*large winds tonight*)
  - Spelling correction:
    P(*I'll be five minutes late*) > P(*I'll be five minuets late*)
  - Speech recognition:
    P(*I saw a van*) > P(*eyes awe of an*)
  - Summarization, question answering, …

Probabilistic language models

- Goal: compute the probability of a sentence (or sequence of words):

$$P(\mathbf{w}) = P(w_1, w_2, w_3, \ldots, w_n)$$

- Related task: probability of the next word:

$$P(w_5 \mid w_4, w_3, w_2, w_1)$$

- A model that computes either of these is called a **language model** (or **LM**).

Statistical Language Models (LM)

To calculate $P(\text{word } w \mid \text{history } h)$

or $P(\overline{W})$, $\overline{W} = w_1, w_2, \ldots, w_n$ a 'n-word' sequence

- P(Random variable X_i taking the value "the")

$$P(X_i = \text{"the"}) \Rightarrow P(\text{"the"}) \text{ or } P(\text{the})$$

- Sequence of 'n' words $\overline{W} = w_1, w_2, \ldots w_n$ or $w_{1:n}$

$$w_{1:1} \Rightarrow w_1$$
$$w_{1:2} \Rightarrow w_1 w_2$$
$$\vdots$$
$$w_{1:n-1} \Rightarrow w_1 w_2 \ldots \ldots w_{n-1}$$
$$w_{1:n} \Rightarrow w_1 w_2 \ldots \ldots w_{n-1} w_n$$

Joint Probability of each word in a sequence having a particular value

$$P(X_1 = w_1,\ X_2 = w_2, \ldots, X_n = w_n)$$

$$\Rightarrow P(w_1, w_2, \ldots, w_{n-1}, w_n)$$

What is $P(w_1, w_2, \ldots, w_n)$? [Prob of sentence W]

$$P(X_1 \ldots X_n) = P(X_1)\,P(X_2/X_1)\,P(X_3|X_1X_2)\ldots P(X_n|X_{1:n-1})$$

$$= \prod_{k=1}^{n} P(X_k | X_{1:k-1})$$

From $P(X,Y) = P(X/Y)\cdot P(Y)$

How By CHAIN RULE OF PROBABILITY

$$P(A,B,C,D) = \underbrace{\underbrace{P(A)\cdot P(B/A)}_{P(A,B)}\cdot P(C/A,B)}_{P(A,B,C)}\cdot P(D/A,B,C)$$

Applying Chain-Rule to Words

$$P(\overline{W}) = P(w_{1:n}) = P(w_1)P(w_2/w_{1:1})P(w_3/w_{1:2})\ldots P(w_n/w_{1:n-1})$$

$$= \prod_{k=1}^{n} P(w_k \mid w_{1:k-1})$$

Problem: How to get estimates (e.g. MLE) of these terms

i.e. Compute the "exact" probability of a word w_n given a long sequence of preceding words

$w_{1:n-1}$ i.e. $P(w_n \mid w_{1:n-1})$

or in general $P(w_k \mid w_{1:k-1})$

Any particular "Context" (or history "h") might have never occured before. (in the trg corpus)

Instead : "N- gram model"

Approximate the history $w_{1:k-1}$ by just the last few words

e.g Bigram Model $\Rightarrow$ Last '1' word.

$$P(w_n \mid w_{1:n-1}) \approx P(w_n \mid w_{n-1})$$

Approximate history

$$\overbrace{w_1 \mid w_2 \mid w_3 \mid \ldots \mid w_{n-1}}^{h:\ \text{history}} \mid w_n$$

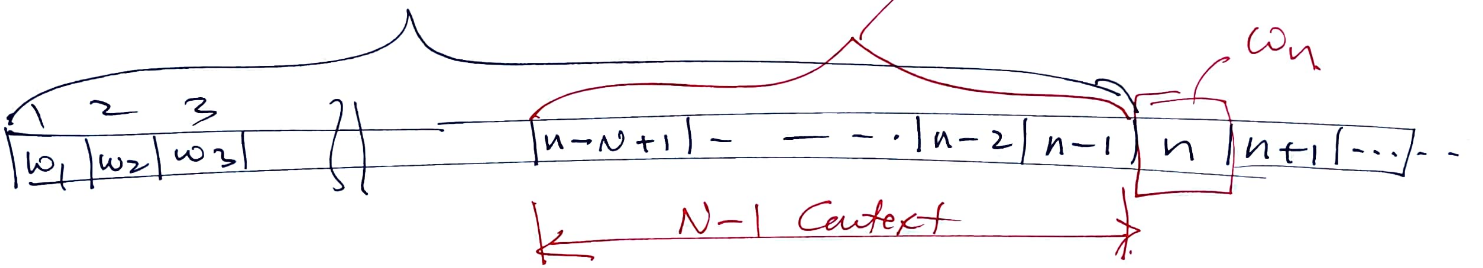
e.g (Walden Pond's water is so transparent that) — the w_n

$\Rightarrow P(\text{the} \mid \text{history}) = P(\text{the} \mid \text{that})$

$\Rightarrow$ MARKOV ASSUMPTION of Order-1.

Likewise Markov Assumption of Order N−1 yields N-gram Model or approximation

$$P(w_n \mid w_{1:n-1}) = P(w_n \mid w_{n-N+1:n-1})$$



Note #words in "truncated" Context

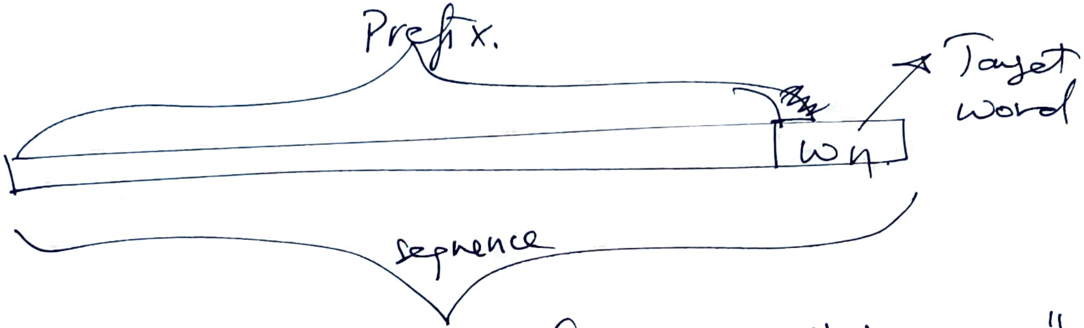
$$= n-1 - (n - N + 1) + 1$$

$$= \cancel{n} - 1 - \cancel{n} + N - \cancel{1} + \cancel{1}$$

$= N-1$ words $\rightsquigarrow$ Markov Assumption of order N−1 $\Rightarrow$ N-gram probabilities

N-grams

	$P(W) = P(w_{1:n})$	MLE Maximum Likelihood Estimate	
Unigram $N=1$	$\prod_{k=1}^{n} P(w_k)$	$P(w_n) = \dfrac{C(w_n)}{\sum_x C(x)}$	Total Count of words in the Corpus / Document
Bigram. $N=2$.	$\prod_{k=1}^{n} P(w_k \mid w_{k-1})$	$P(w_n \mid w_{n-1}) = \dfrac{C(w_{n-1}, w_n)}{C(w_{n-1})}$	$\sum_w C(w_{n-1}, w)$ Marginal Distn. from joint distn: Integrate out the unwanted variable.
Trigram $N=3$.	$\prod_{k=1}^{n} P(w_k \mid w_{k-1}, w_{k-2})$	~~P(~~	
N-gram any N!	$\prod_{k=1}^{n} P(w_k \mid w_{k-N+1:k-1})$	$P(w_n \mid w_{n-N+1:n-1}) = \dfrac{C(w_{n-N+1:n-1}, w_n)}{C(w_{n-N+1:n-1})}$	Relative Frequency of Occurrences



$$\frac{\text{observed frequency of "sequence"}}{\text{observed frequency of "prefix"}}$$

1) **Problem:** How to deal with "UNKNOWN Words"
=> OOVs or Out-of-Vocabulary words.

- Smoothing
- Backoff
- Interpolation

Backoff: Use trigram
— If not available — Backoff to (use) Bigram
— If not available — Backoff to Unigram

Interpolation ⇒ Mix the Probability Estimates

Trigram

$$\hat{P}(w_n \mid w_{n-1}\, w_{n-2}) = \lambda_1 P(w_n) + \lambda_2 P(w_n \mid w_{n-1}) + \lambda_3 P(w_n \mid w_{n-1}\, w_{n-1})$$

$$\sum_i \lambda_i = 1$$

Neural LMs

LMs: Probability Distribution over sequences of 'n' tokens.

$$\{t_1\ t_2 \ldots\ t_n\}$$

Given such a sequence, an LM assigns a probability

$$P(t_1, t_2 \ldots t_n)$$

to the whole sequence by modeling the

$$\text{Prob}(\text{token } t_k \mid \text{history } t_1\ t_2 \ldots t_{k-1})$$

ie,

$$P(t_1\ t_2 \ldots t_n) = \prod_{k=1}^{n} P(t_k \mid t_1\ t_2 \ldots t_{k-1})$$

(2)

$$P(t_1 t_2 \dots t_n) = \prod_{k=1}^{n} P(t_k \mid t_1 t_2 \dots t_{k-1})$$

→ Trained by Minimizing the Negative Log-Likelihood,

$$\sum_{k=1}^{n} -\log(P(t_1 t_2 \dots t_k ; \theta_t, \theta_{rnn}, \theta_s))$$

Parameters to be optimized

θ_t , θ_{rnn} , θ_s

$\Rightarrow$

θ_t : Look-up table / Word-to-Vec embedding layer (3)

→ maps each token into a vector of fixed dimension.

→ Word2Vec or TB result on a given vocab of words.
↳ $|V|$

θ_{rnn} : RECURRENT NEURAL NETWORK (RNN, LSTM, GRU..)

→ Summarizes the sequence of history

e.g. t_1 t_2 - - - . t_{k-1}

up to the current time-step (e.g t_k)

θ_s: Softmax layer appended ~~at~~ the O/P of EACH
RNN time step for estimating the probability
distribution over the tokens (Posterior Vector of dim $|V|$)
@ $k-1$ ⟹ max prob of t_k (ground truth at k)

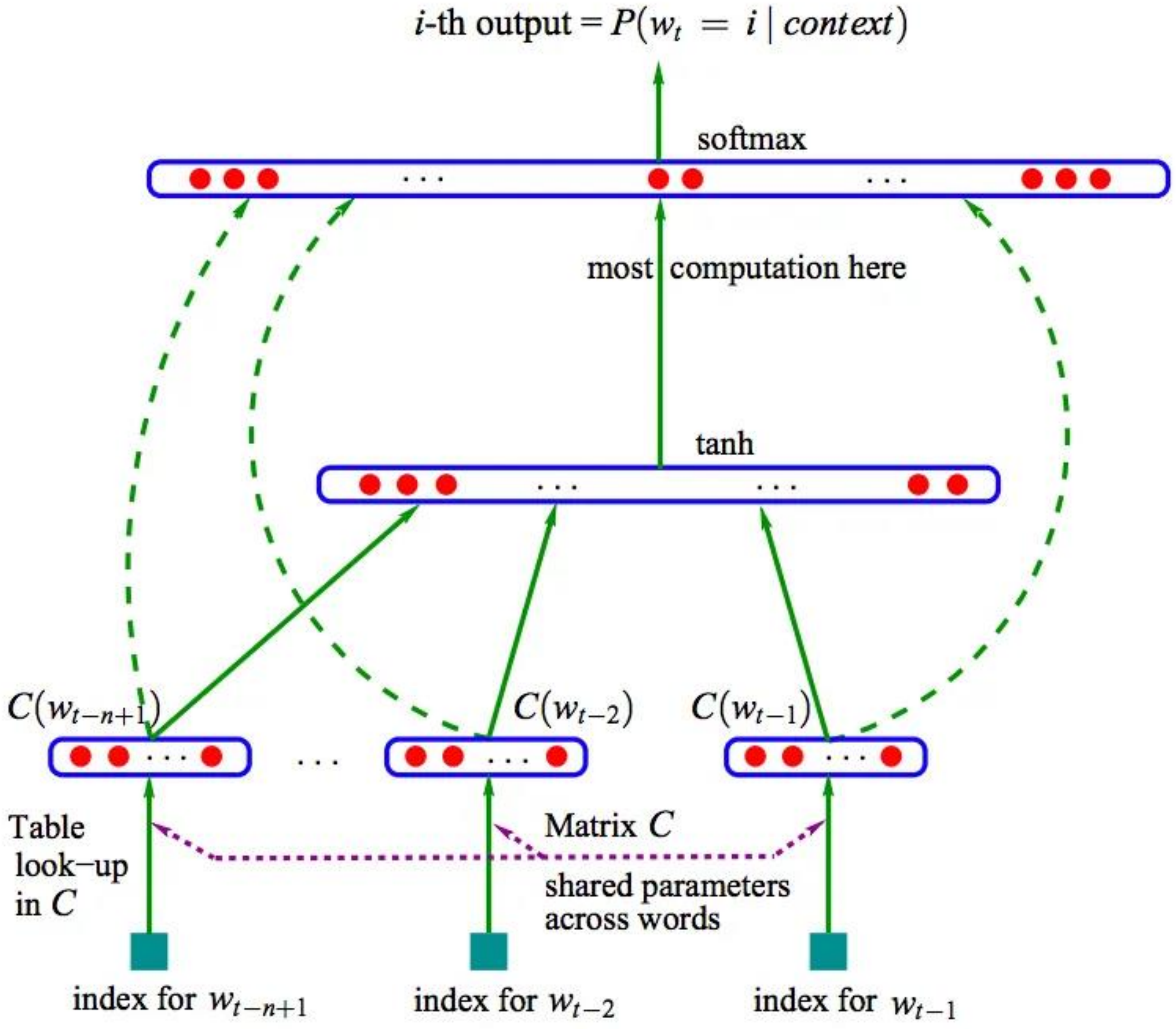


Figure 1: Neural architecture: $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i-th word feature vector.

A fixed-window neural Language Model

output distribution

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

hidden layer

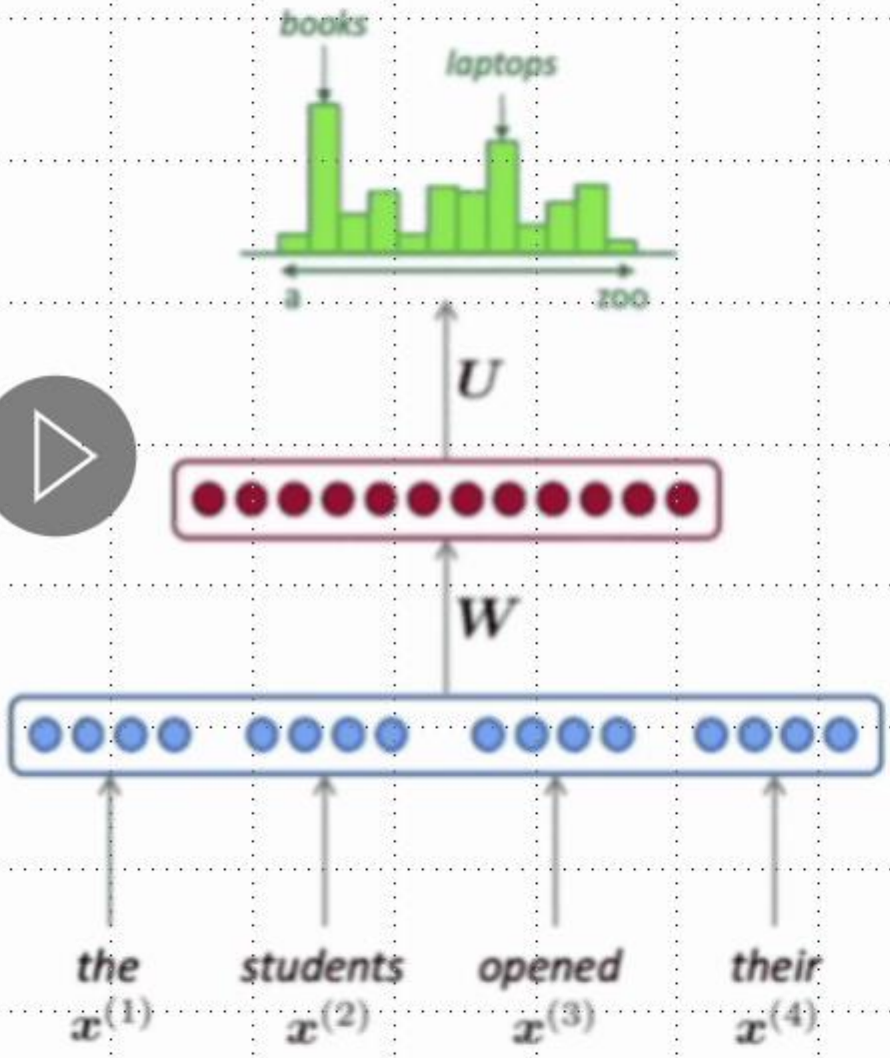
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

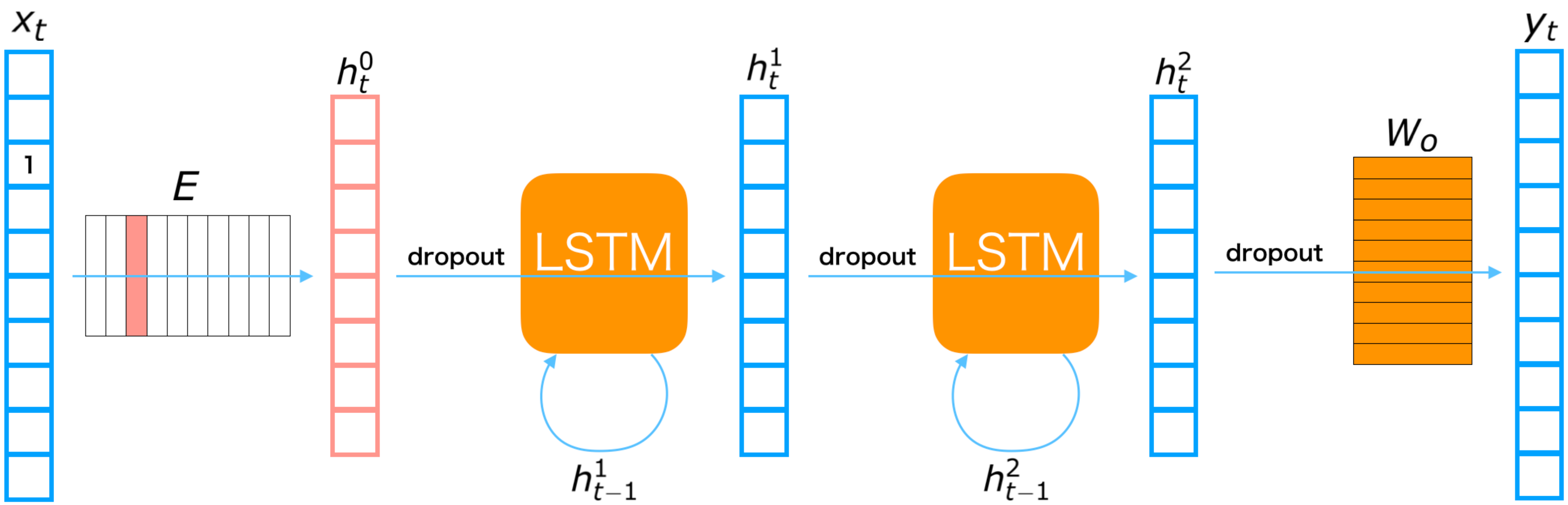
concatenated word embeddings

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

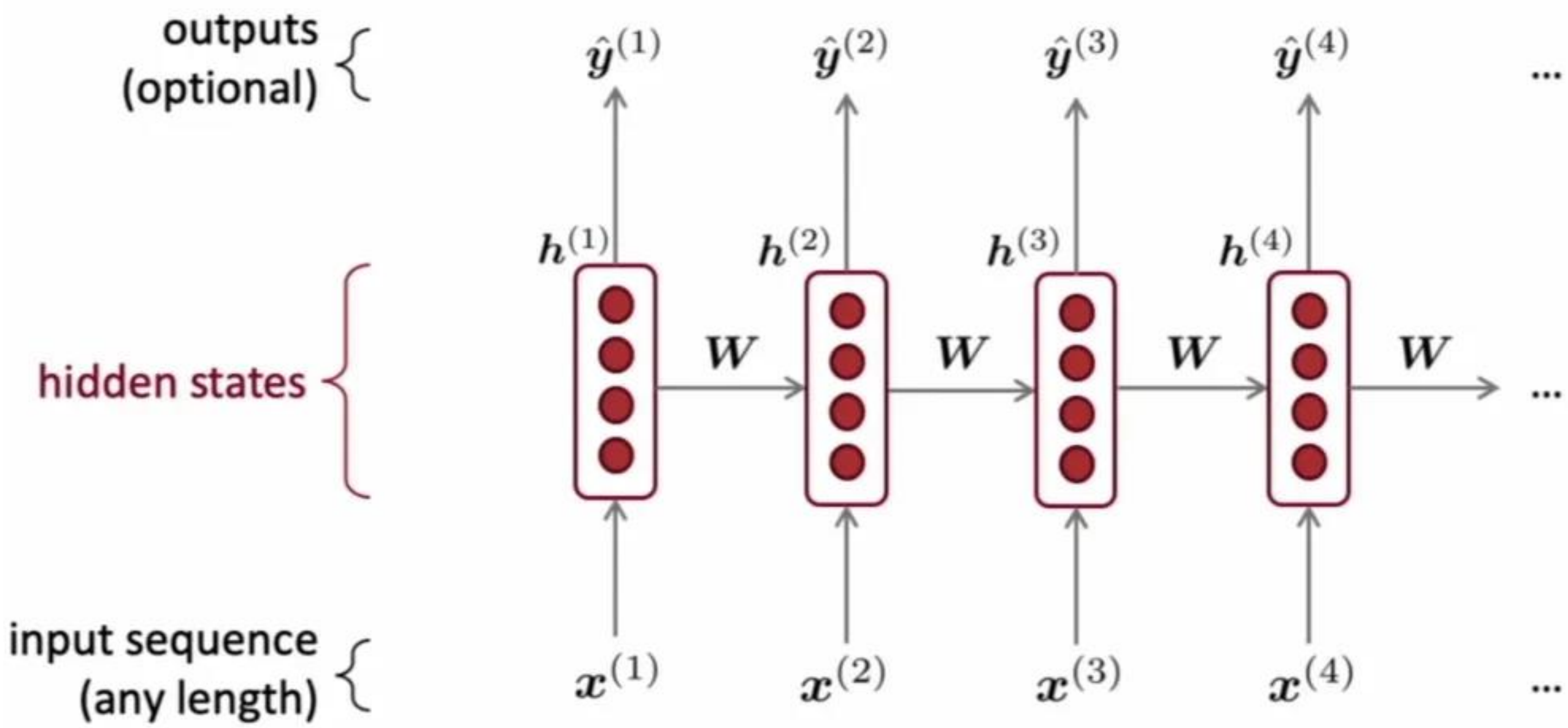
words / one-hot vectors

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





x_t
1
E
h_t^0
dropout
LSTM
h_{t-1}^1
h_t^1
dropout
LSTM
h_{t-1}^2
h_t^2
dropout
W_o
y_t


outputs (optional)
$\hat{y}^{(1)}$
$\hat{y}^{(2)}$
$\hat{y}^{(3)}$
$\hat{y}^{(4)}$
...
$h^{(1)}$
$h^{(2)}$
$h^{(3)}$
$h^{(4)}$
hidden states
W
W
W
W
...
input sequence (any length)
$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
...

A RNN Language Model

output distribution

$\hat{y}^{(t)} = \mathrm{softmax}\left(Uh^{(t)} + b_2\right) \in \mathbb{R}^{|V|}$

hidden states

$h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_e e^{(t)} + b_1\right)$

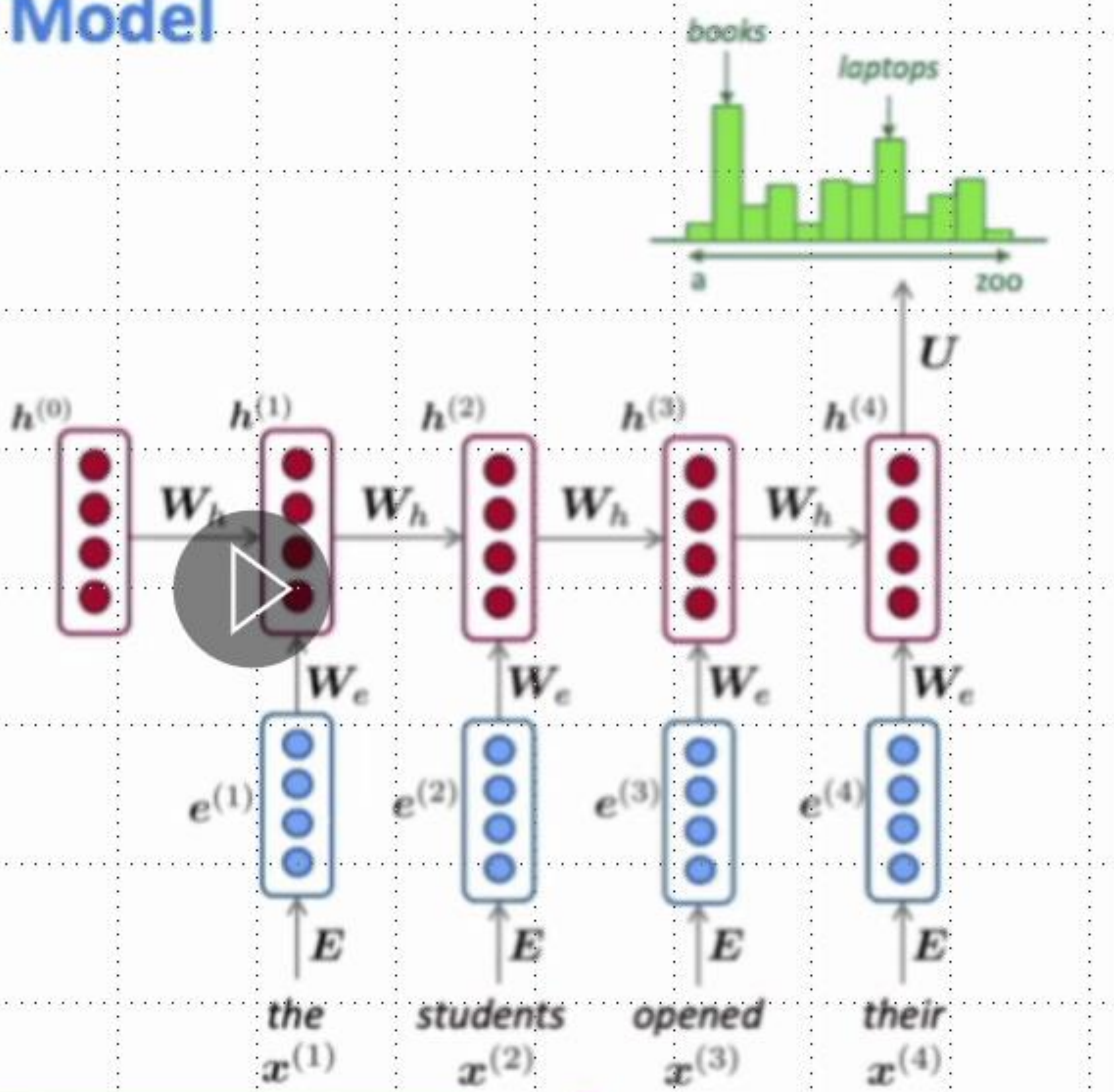
$h^{(0)}$ is the initial hidden state

word embeddings

$e^{(t)} = Ex^{(t)}$

words / one-hot vectors

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



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

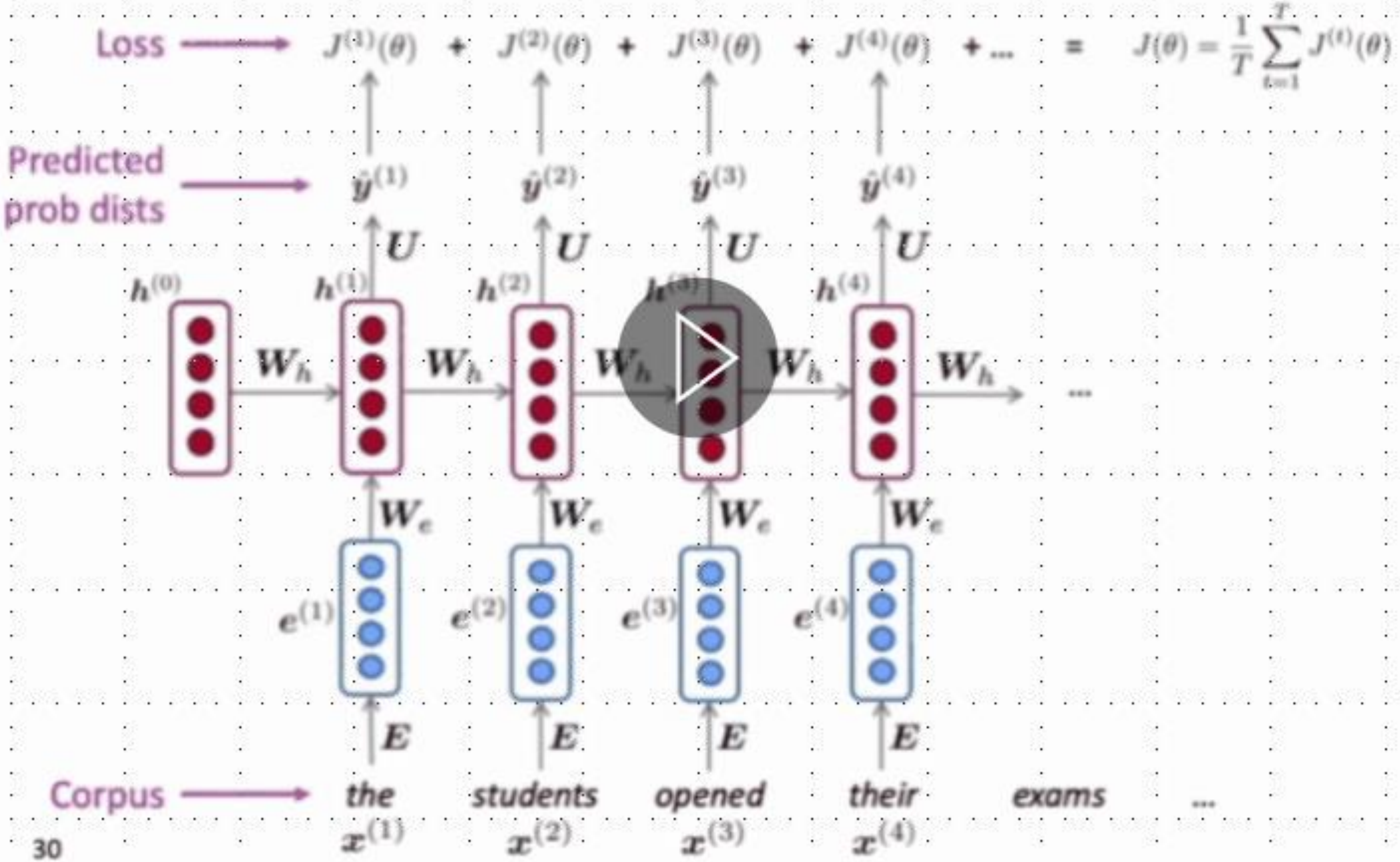
Loss function on step t is cross-entropy between predicted probability distribution $\hat{y}^{(t)}$, and the true next word $y^{(t)}$ (one-hot for $x^{(t+1)}$):

$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{w \in V} \boldsymbol{y}_w^{(t)} \log \hat{\boldsymbol{y}}_w^{(t)} = -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

Average this to get overall loss for entire training set:

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J^{(t)}(\theta) = \frac{1}{T}\sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

Training a RNN Language Model



Thank you !!