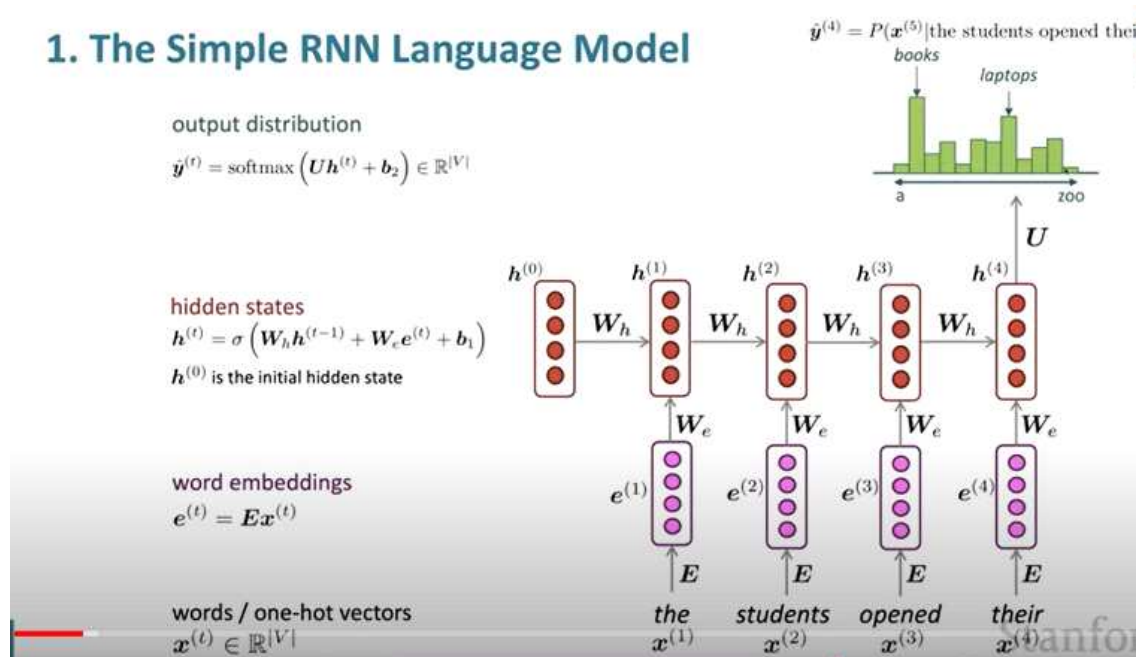


1. The Simple RNN Language Model

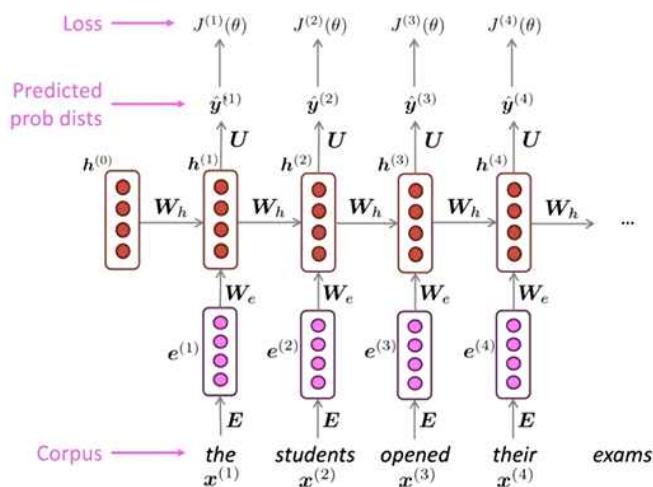


1. Training RNNs

big corpus of text \rightarrow feed into RNN-LM; for every step $t \rightarrow$ Loss function/cross entropy \rightarrow Average of overall loss

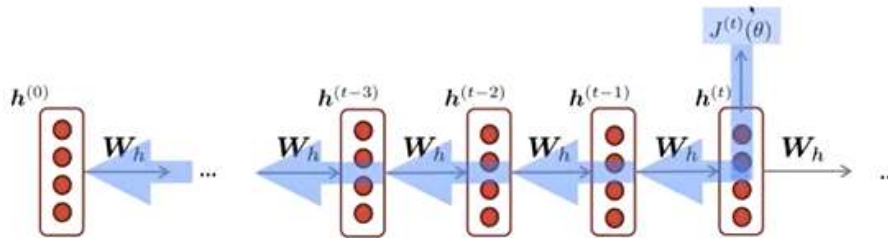
$$J^{(t)}(\theta) = CE(y^{(t)}, \hat{y}^{(t)}) = - \sum_{w \in V} y_w^{(t)} \log \hat{y}_w^{(t)} = - \log \hat{y}_{x_{t+1}}^{(t)}$$

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



2. Uses of RNNs

Backpropagation for RNNs



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

Question: How do we calculate this?

Answer: Backpropagate over timesteps $i=t, \dots, 0$, summing gradients as you go. This algorithm is called **"backpropagation through time"** [Werbos, P.G., 1988, *Neural Networks 1*, and others]

=> generating text with RNN(repeated sampling) eg. Harry Potter, recipes,

● Evaluating

- The standard **evaluation metric** for Language Models is **perplexity**.

$$\text{perplexity} = \prod_{t=1}^T \left(\frac{1}{P_{\text{LM}}(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)})} \right)^{1/T}$$

Inverse probability of corpus, according to Language Model

Normalized by number of words

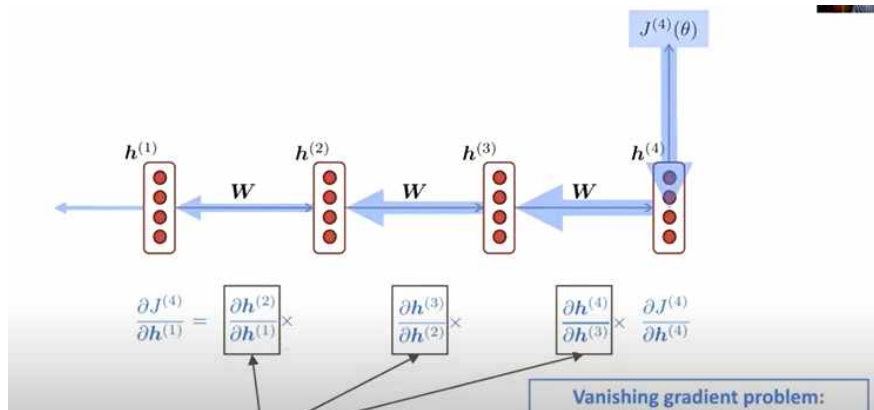
- This is equal to **the exponential of the cross-entropy loss** $J(\theta)$:

$$= \prod_{t=1}^T \left(\frac{1}{\hat{y}_{\mathbf{x}_{t+1}}^{(t)}} \right)^{1/T} = \exp \left(\frac{1}{T} \sum_{t=1}^T -\log \hat{y}_{\mathbf{x}_{t+1}}^{(t)} \right) = \exp(J(\theta))$$

sequence tagging, sentiment classification, language encoder module, generating text ...

3. Problems with RNNs

Vanishing and exploding gradients



Vanishing gradient proof sketch

- Recall: $h^{(t)} = \sigma(W_h h^{(t-1)} + W_x x^{(t)} + b_1)$
- What if σ were the identity function, $\sigma(x) = x$?

$$\frac{\partial h^{(t)}}{\partial h^{(t-1)}} = \text{diag}(\sigma'(W_h h^{(t-1)} + W_x x^{(t)} + b_1)) W_h \quad (\text{chain rule})$$

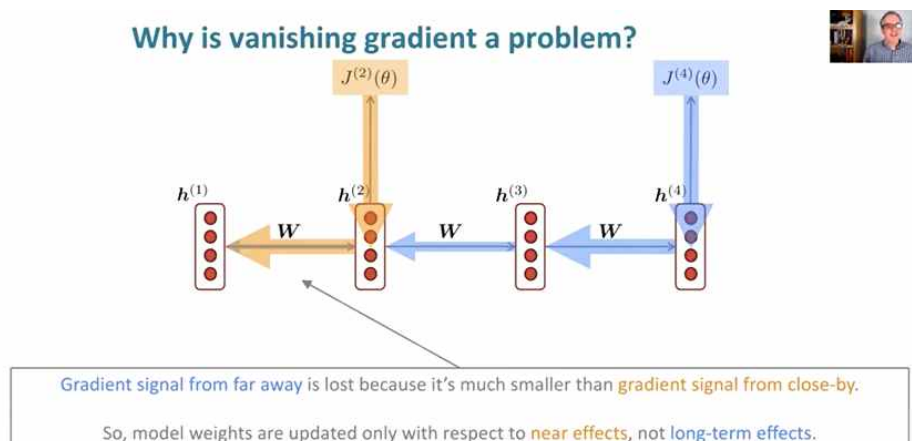
$$= I W_h = W_h$$
- Consider the gradient of the loss $J^{(i)}(\theta)$ on step i , with respect to the hidden state $h^{(j)}$ on some previous step j . Let $\ell = i - j$

$$\frac{\partial J^{(i)}(\theta)}{\partial h^{(j)}} = \frac{\partial J^{(i)}(\theta)}{\partial h^{(i)}} \prod_{j < t \leq i} \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \quad (\text{chain rule})$$

$$= \frac{\partial J^{(i)}(\theta)}{\partial h^{(i)}} \prod_{j < t \leq i} W_h = \frac{\partial J^{(i)}(\theta)}{\partial h^{(i)}} \boxed{W_h^\ell} \quad (\text{value of } \frac{\partial h^{(t)}}{\partial h^{(t-1)}})$$

If W_h is "small", then this term gets exponentially problematic as ℓ becomes large

Why is vanishing gradient a problem?



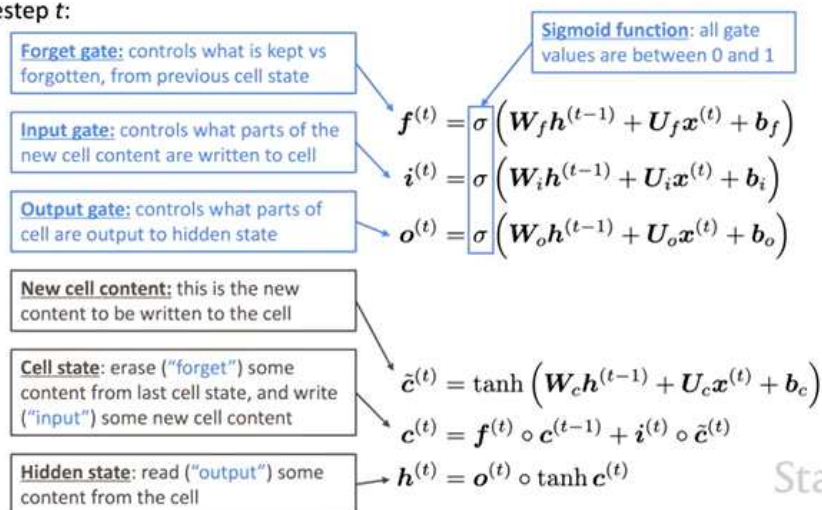
4. LSTMs

A type of RNN propped as a solution to the vanishing gradients problem

- On step t , there is a **hidden state** $h^{(t)}$ and a **cell state** $c^{(t)}$
 - Both are vectors length n
 - The cell stores **long-term information**
 - The LSTM can **read**, **erase**, and **write** information from the cell
 - The cell becomes conceptually rather like RAM in a computer

Long Short-Term Memory (LSTM)

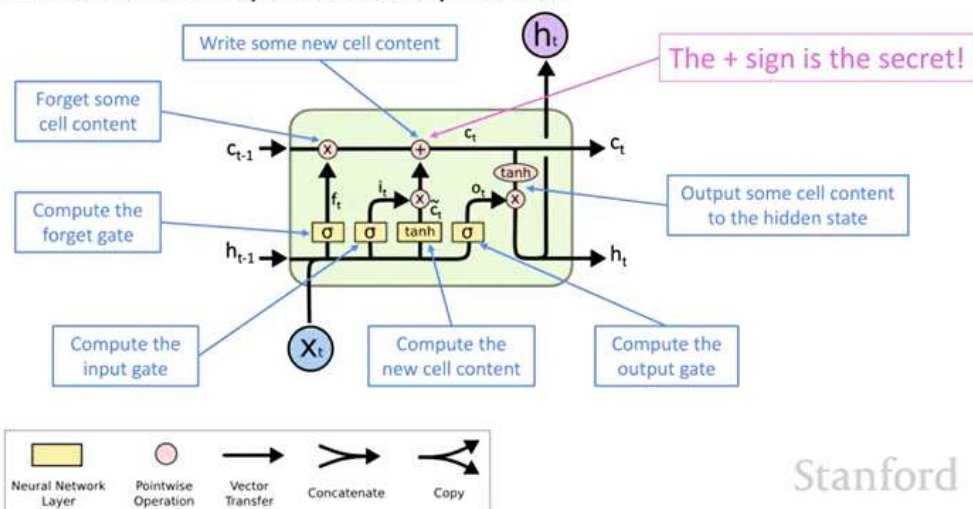
We have a sequence of inputs $x^{(t)}$, and we will compute a sequence of hidden states $h^{(t)}$ and cell states $c^{(t)}$. On timestep t :



Stanford

Long Short-Term Memory (LSTM)

You can think of the LSTM equations visually like this:

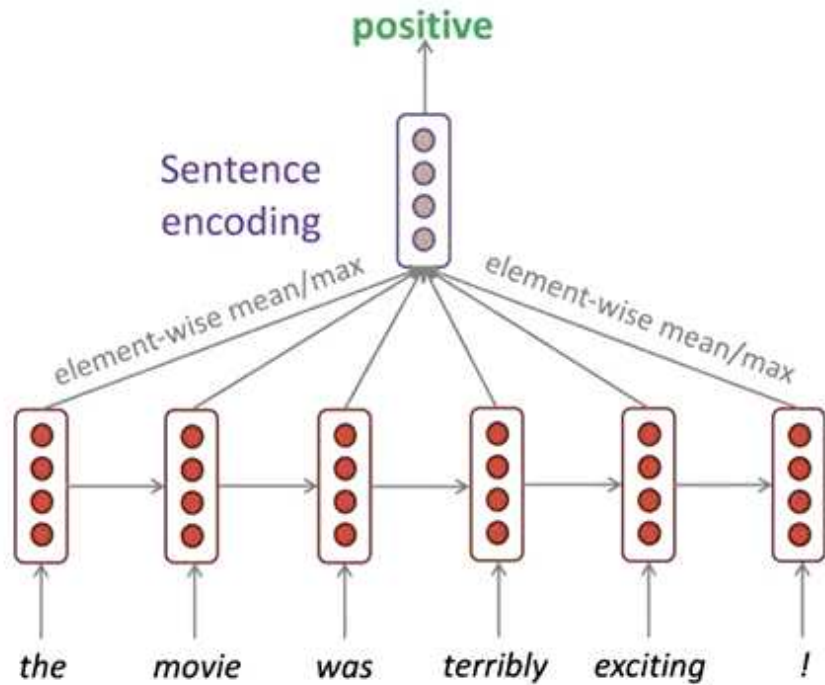


Stanford

Source: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

5. bidirectional RNNs and multi-layer RNNs

Task: Sentiment Classification



Bidirectional RNNs

