Topic Models
& Recurrent Neural Networks

Kuan-Yu Chen (陳冠宇)

2018/04/12 @ TR-409, NTUST

Semantic?

- In the context of NLP, long-span information is important!
  - Especially for language modeling

一 枝 美麗 的 玫瑰 在 花盆

一 隻 可愛 的 小貓 在 花園

一 支 高貴 的 鋼筆 在 拍賣

- N-gram models are not an efficient strategy to capture the long-span information
  - From literal matching to semantic mapping

Probabilistic Latent Semantic Analysis

- **Probabilistic Latent Semantic Analysis** also called
  - Probabilistic Latent Semantic Indexing (PLSI)
  - Aspect Model

- PLSA is a probabilistic counterpart of LSA
  - $P(d_j)$: the probability of selecting document d_j
  - $P(w_i|T_k)$: the probability of word w_i condition on a latent factor/topic T_k
    - **Aspect!**
  - $P(T_k|d_j)$: the probability of a latent factor/topic T_k generated by document d_j

PLSA – 1

- The PLSA model is a latent variable model for co-occurrence data (i.e., each pair of word w_i and document d_j) which associates an unobserved class variable (i.e., latent factor T_k)

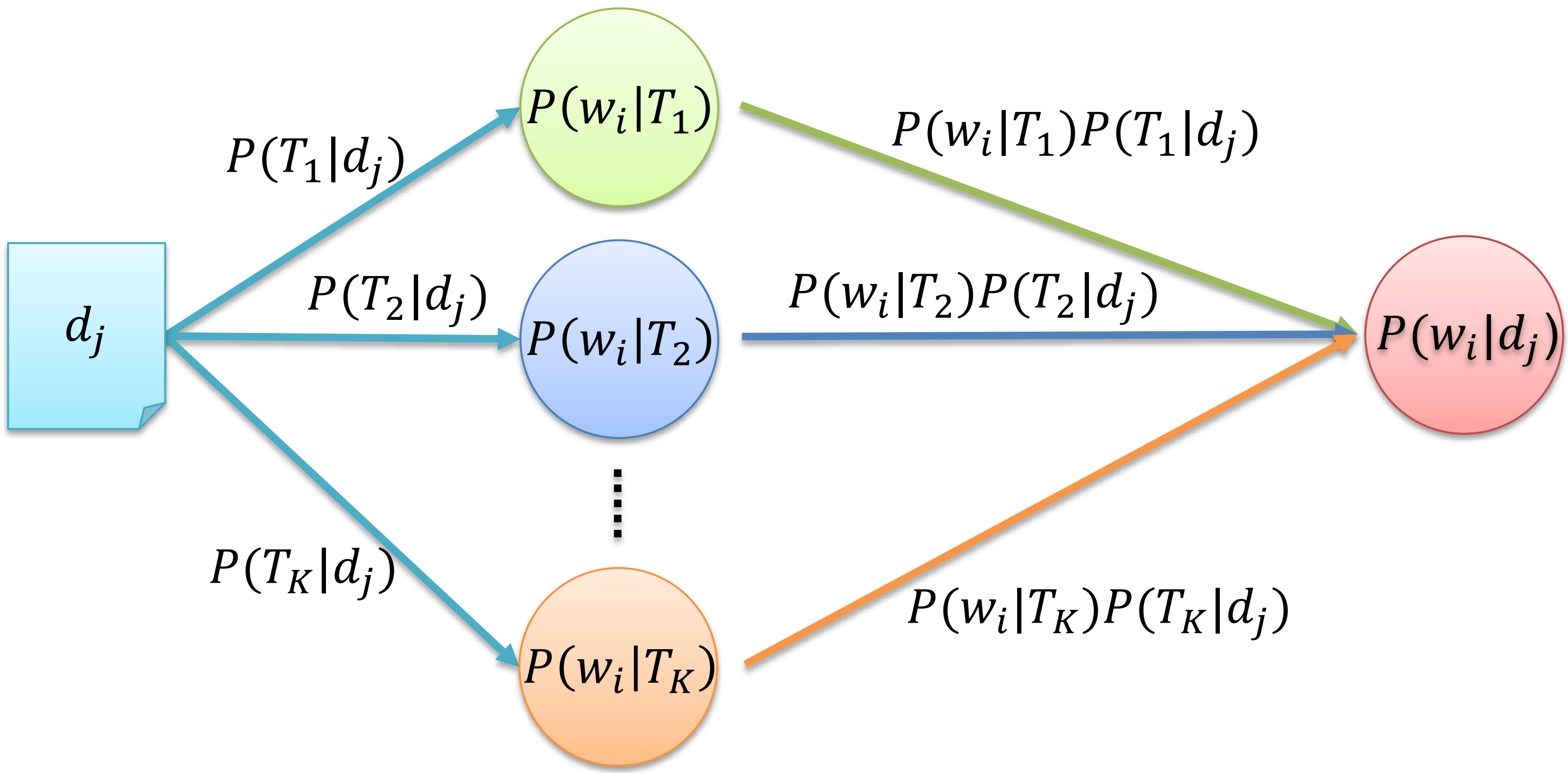
$$P(w_i, d_j) = P(d_j)P(w_i|d_j) = P(d_j)\sum_{k=1}^{K} P(w_i|T_k)P(T_k|d_j)$$

$$\begin{aligned}
P(w_i|d_j) &= \sum_{k=1}^{K} P(w_i, T_k|d_j) = \sum_{k=1}^{K} \frac{P(w_i, T_k, d_j)}{P(d_j)} \\
&= \sum_{k=1}^{K} \frac{P(w_i, d_j|T_k)P(T_k)}{P(d_j)} \\
&= \sum_{k=1}^{K} \frac{P(w_i\,|T_k)P(d_j|T_k)P(T_k)}{P(d_j)} \\
&= \sum_{k=1}^{K} \frac{P(w_i\,|T_k)P(d_j, T_k)}{P(d_j)} = \sum_{k=1}^{K} P(w_i\,|T_k)P(T_k|d_j)
\end{aligned}$$

Conditional Independence Assumption
document and word are independent conditioned on the state of the associated latent variable

PLSA – 2

- Thus, the modeling goal is to identify conditional probability mass functions $P(w_i|T_k)$ such that the document-specific word distributions $P\left(w_i\middle|d_j\right)$ are as faithfully as possible approximated by convex combinations of these aspects



PLSA – 3

- The training objective is defined to maximize the total log-likelihood of a given training collection
  - The model parameters are $P(d_j)$, $P(w_i|T_k)$, and $P(T_k|d_j)$

$$\mathcal{L} = \sum_{w_i \in V} \sum_{d_j \in \mathbf{D}} c(w_i, d_j) log P(w_i, d_j)$$

$$= \sum_{w_i \in V} \sum_{d_j \in \mathbf{D}} c(w_i, d_j) log \left(P(d_j) \sum_{k=1}^{K} P(w_i|T_k) P(T_k|d_j) \right)$$

PLSA – 4

- By using the Expectation-Maximization algorithm
  - E-step

$$P\left(T_k \middle| w_i, d_j\right) = \frac{P(w_i|T_k)P\left(T_k \middle| d_j\right)}{\sum_{k=1}^{K} P(w_i|T_k)P\left(T_k \middle| d_j\right)}$$

  - M-step

$$P(w_i|T_k) = \frac{\sum_{d_j \in \mathbf{D}} c(w_i, d_j)P\left(T_k \middle| w_i, d_j\right)}{\sum_{i'=1}^{|V|} \sum_{d_j \in \mathbf{D}} c(w_{i'}, d_j)P\left(T_k \middle| w_{i'}, d_j\right)}$$

$$P(T_k|d_j) = \frac{\sum_{i=1}^{|V|} c(w_i, d_j)P\left(T_k \middle| w_i, d_j\right)}{\sum_{i'=1}^{|V|} c(w_{i'}, d_j)}$$

PLSA – 5

- Consequently, for a given word sequence, $w_1, w_2, \dots, w_T$, the joint probability in a language can be calculated by using PLSA

$$P(w_1, w_2, \dots, w_T) = P(w_1) \prod_{t=2}^{T} P(w_t | w_1, w_2, \dots, w_{t-1})$$

$$= P(w_1) \prod_{t=2}^{T} \left(\sum_{k=1}^{K} P(w_t | T_k) P(T_k | w_1, w_2, \dots, w_{t-1}) \right)$$

  - Usually, we can combine the PLSA with the traditional n-gram models
    - Semantic matching and literal term matching

$$P(w_t | w_1, w_2, \dots, w_{t-1}) = \alpha \cdot P(w_t | w_{t-n+1}, \dots, w_{t-1}) +$$

$$(1-\alpha) \cdot \sum_{k=1}^{K} P(w_t | T_k) P(T_k | w_1, w_2, \dots, w_{t-1})$$

PLSA – 6

- For a new history of words, $w_1, w_2, \ldots, w_{t-1} = H_1^{t-1}$, the **fold-in** strategy can be perform to obtain the topic distribution
  - The word distribution for each topic $P(w_i|T_k)$ is fixed
  - E-step

$$P(T_k \middle| w_i, H_1^{t-1}) = \frac{P(w_i|T_k)P(T_k \middle| H_1^{t-1})}{\sum_{k=1}^{K} P(w_i|T_k)P(T_k \middle| H_1^{t-1})}$$

  - M-step

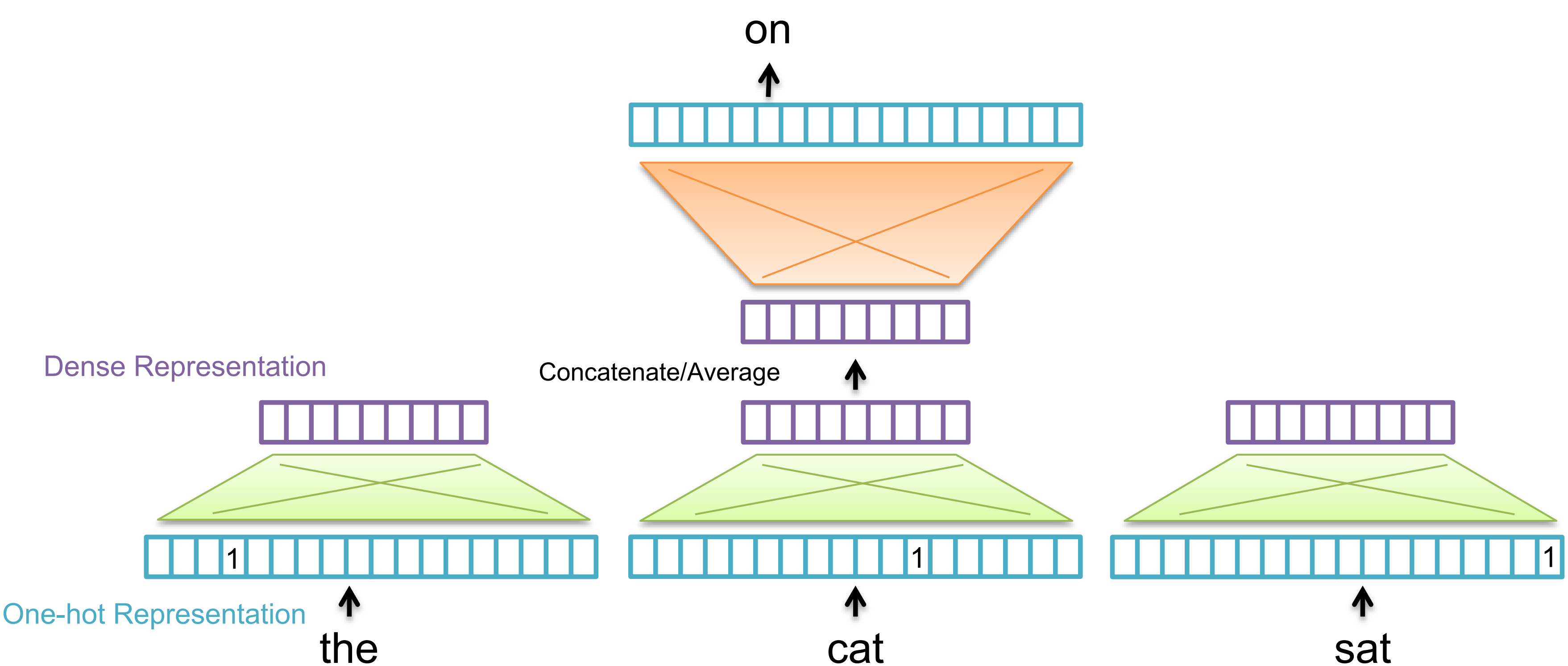
$$P(T_k|H_1^{t-1}) = \frac{\sum_{i=1}^{|V|} c(w_i, H_1^{t-1})P(T_k \middle| w_i, H_1^{t-1})}{\sum_{i'=1}^{|V|} c(w_{i'}, H_1^{t-1})}$$

Y. Bengio, *et al.*, "A Neural Probabilistic Language Model," in *Proc. of NIPS*, 2001.

Revisiting NNLM – 1

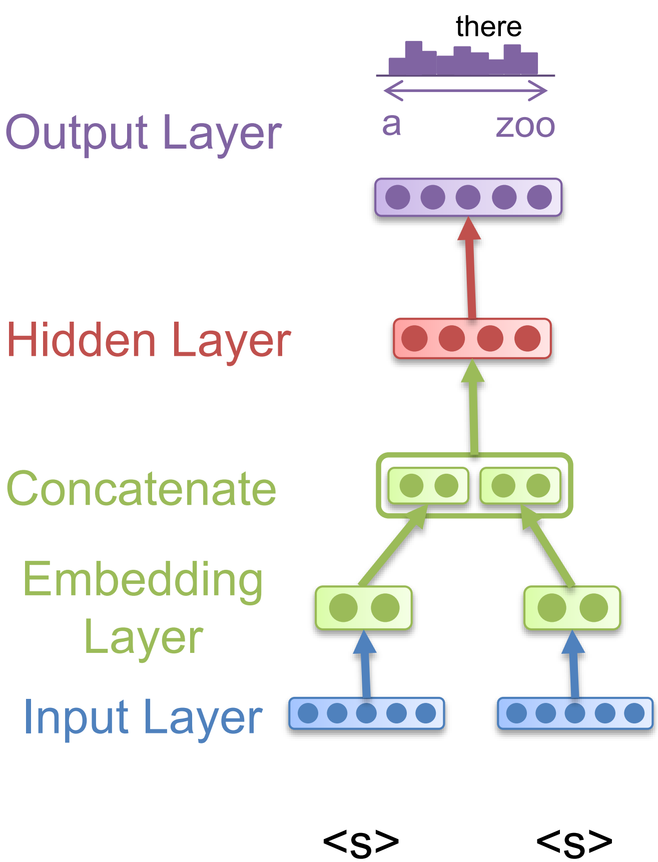
- The Neural Network Language Mode (NNLM) estimated a statistical (n-gram) language model for **predicting future words**

$$P(w_1, w_2, \dots, w_T) \approx \prod_{t=1}^{T} P(w_t | w_{t-n+1}, \dots, w_{t-1})$$



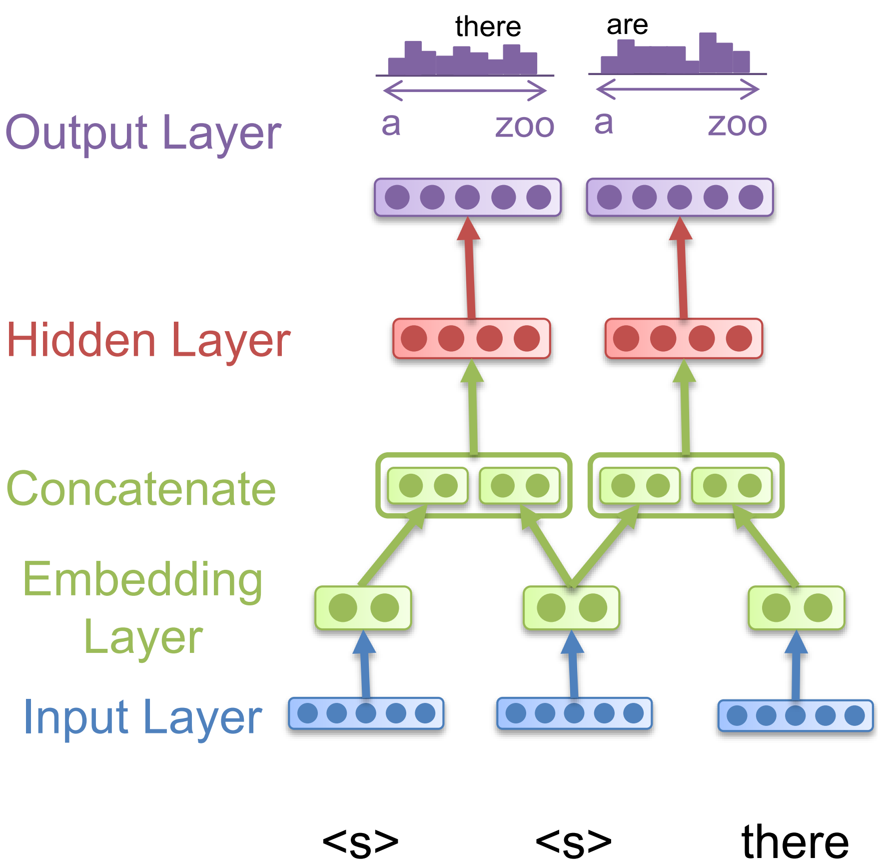
Revisiting NNLM – 2.

$P(there\ are\ books\ on\ the\ table)$
$\approx \boldsymbol{P(there)} P(are|there) P(books|there\ are) P(on|are\ books)$
$P(the|books\ on) P(table|on\ the)$



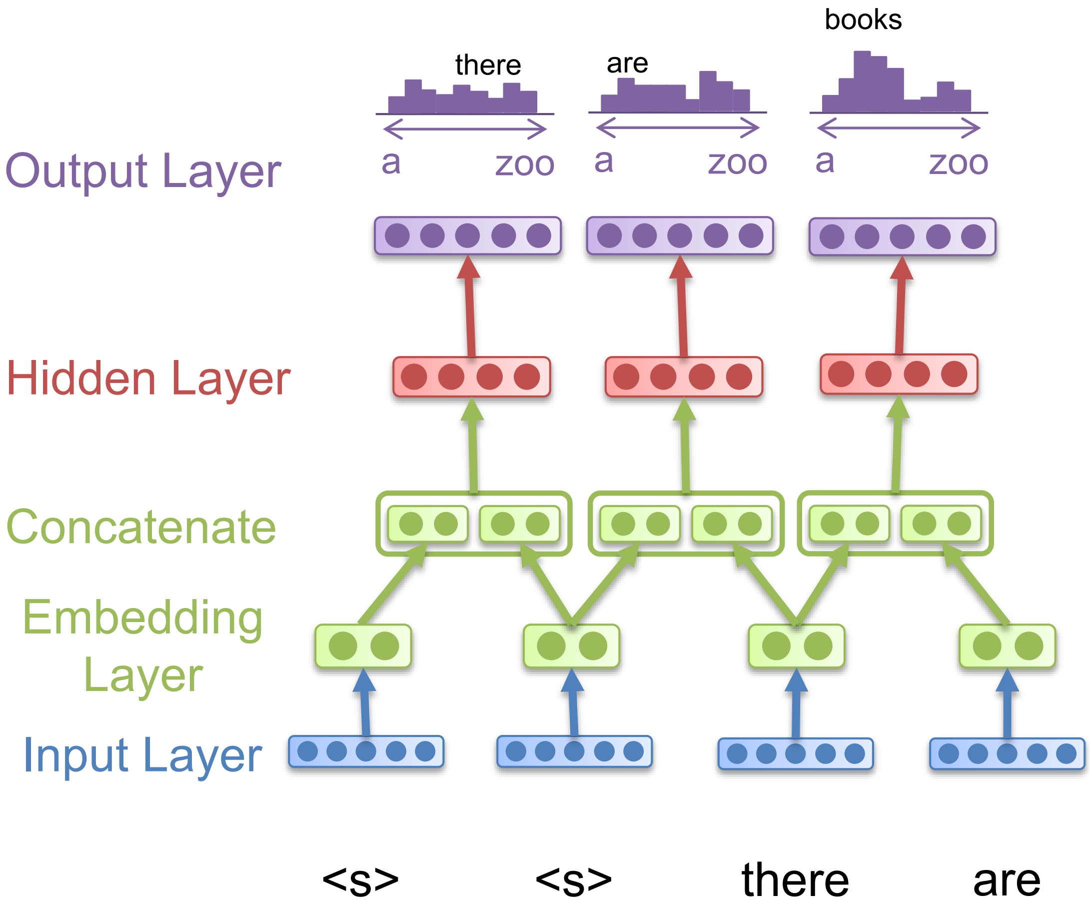
Revisiting NNLM – 2..

$P(there\ are\ books\ on\ the\ table)$
$\approx P(there)\color{red}{P(are|there)}P(books|there\ are)P(on|are\ books)$
$P(the|books\ on)P(table|on\ the)$



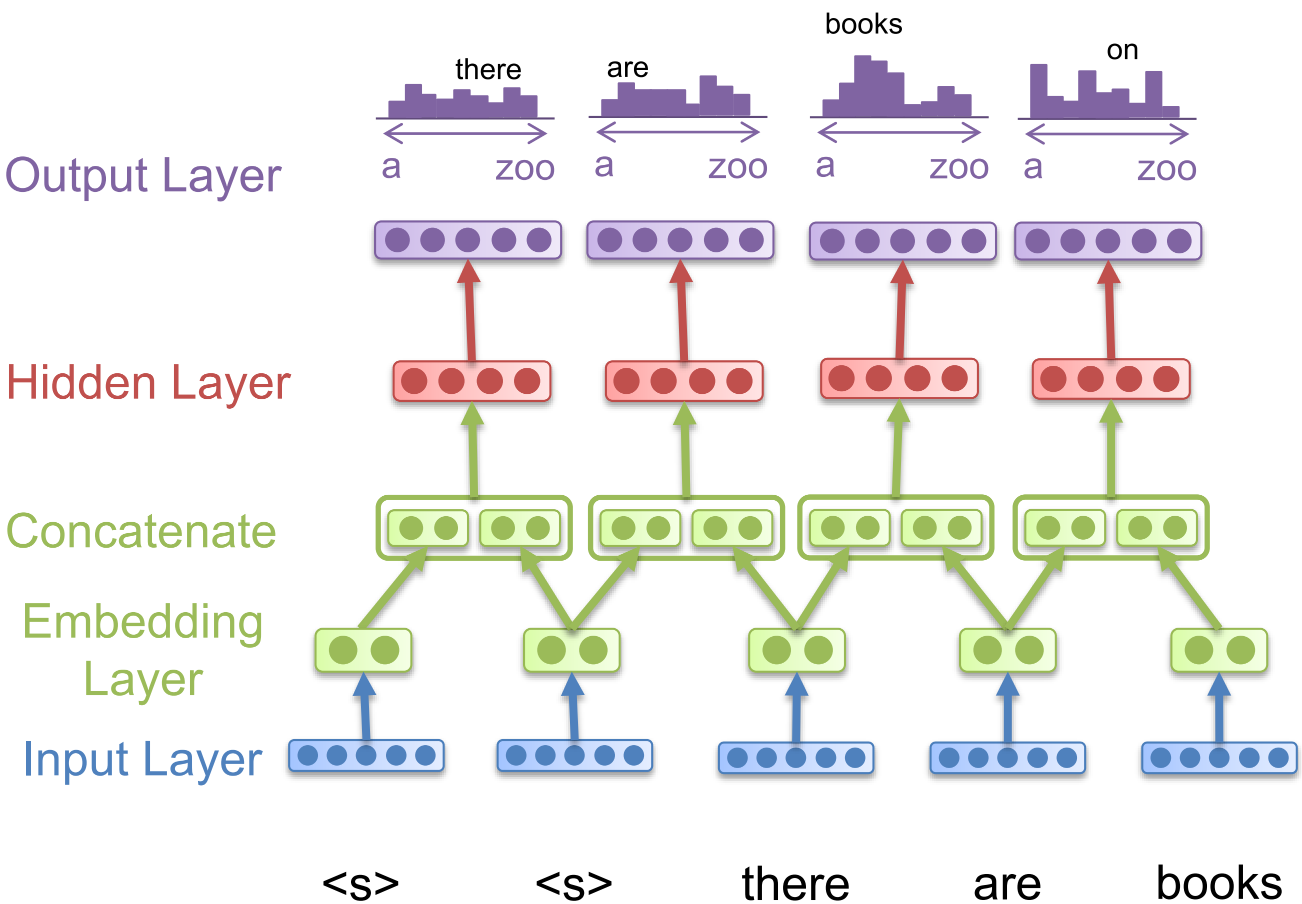
Revisiting NNLM – 2...

$$P(\mathit{there\ are\ books\ on\ the\ table})$$
$$\approx P(\mathit{there})P(\mathit{are}|\mathit{there})\boldsymbol{P(\mathit{books}|\mathit{there\ are})}P(\mathit{on}|\mathit{are\ books})$$
$$P(\mathit{the}|\mathit{books\ on})P(\mathit{table}|\mathit{on\ the})$$



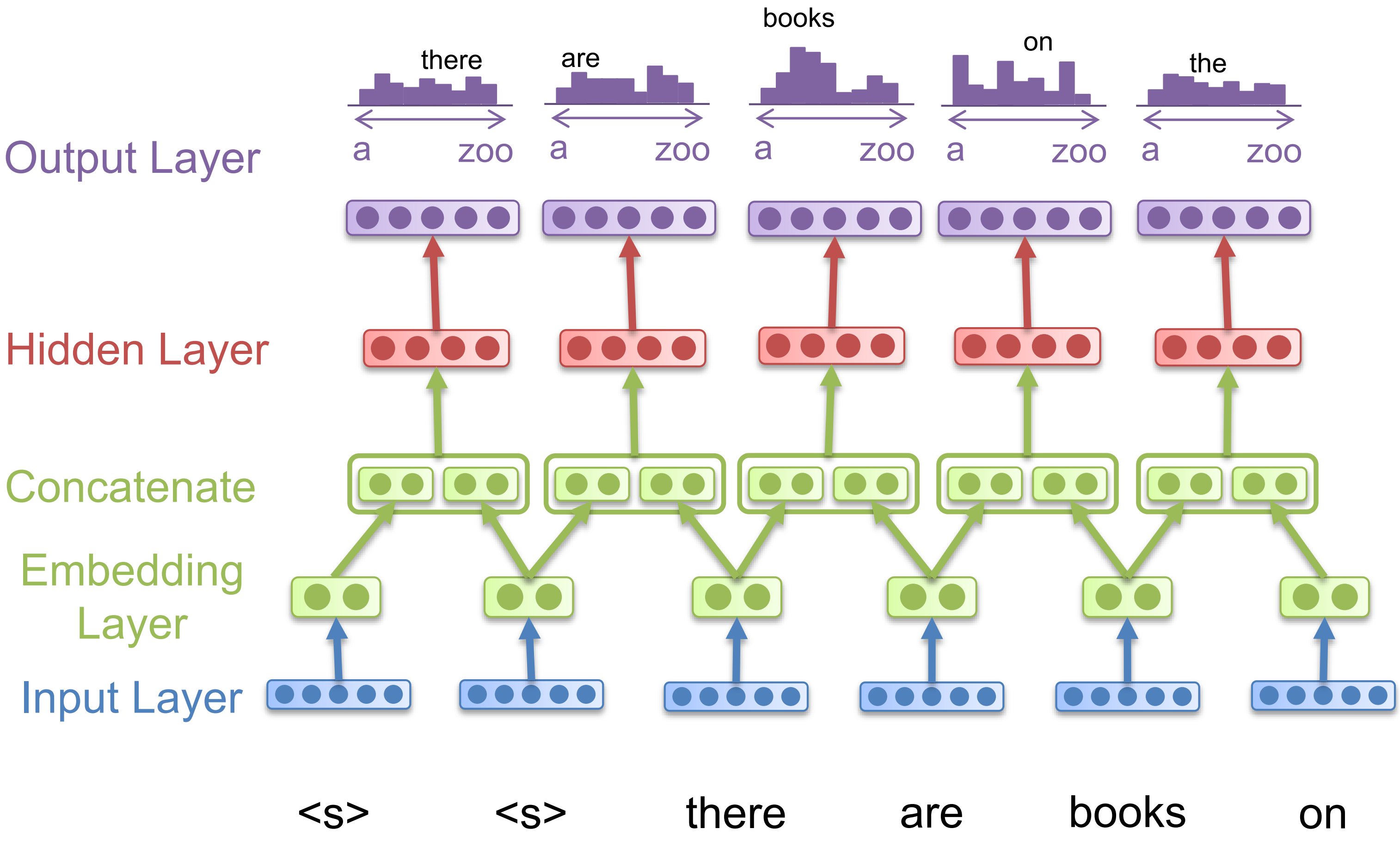
Revisiting NNLM – 2....

$P(there\ are\ books\ on\ the\ table)$
$\approx P(there)P(are|there)P(books|there\ are)\boldsymbol{P(on|are\ books)}$
$P(the|books\ on)P(table|on\ the)$



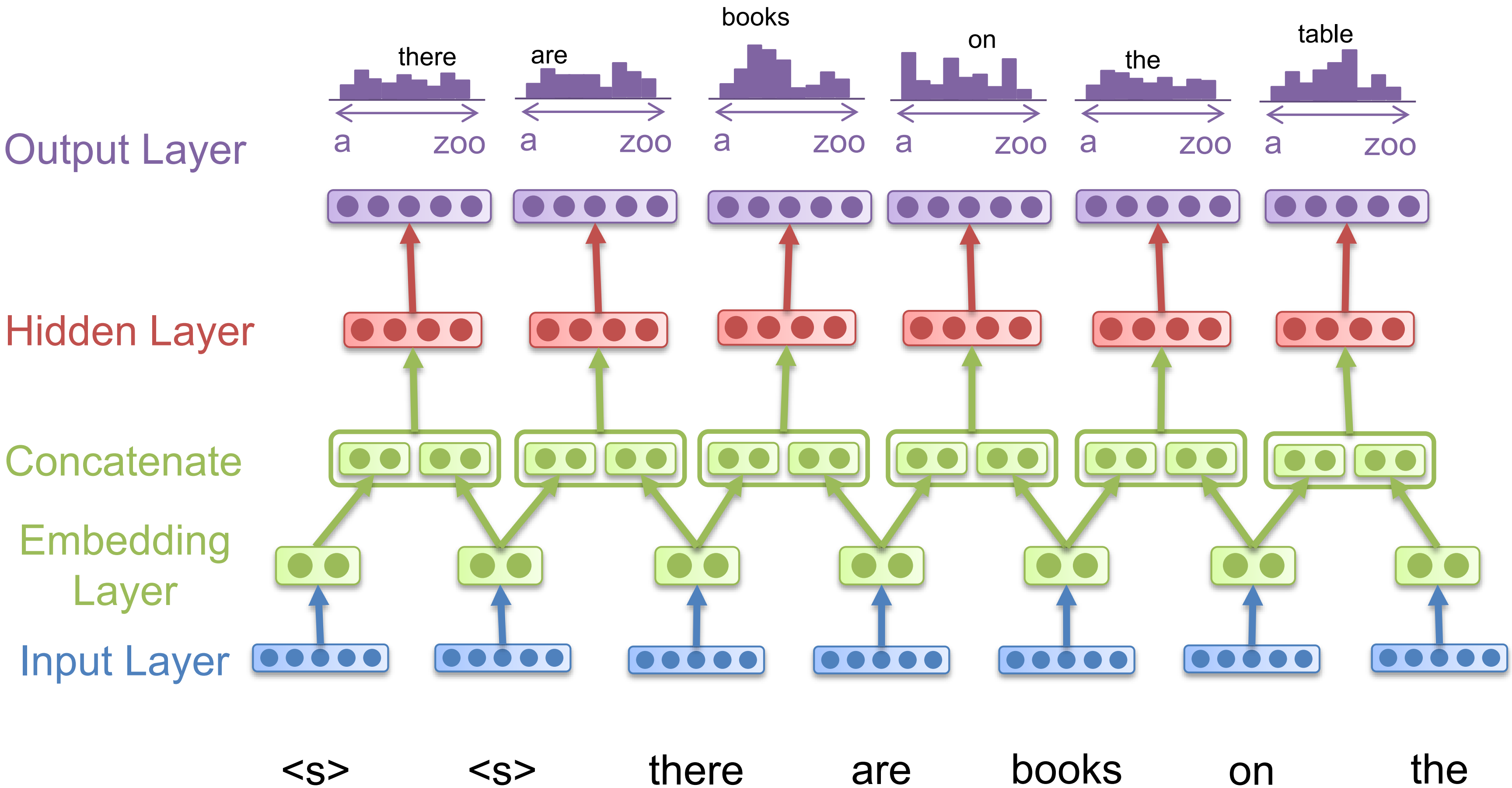
Revisiting NNLM – 2…..

$$P(\mathit{there\ are\ books\ on\ the\ table}) \approx P(\mathit{there})P(\mathit{are}|\mathit{there})P(\mathit{books}|\mathit{there\ are})P(\mathit{on}|\mathit{are\ books})$$
$$\boldsymbol{P(\mathit{the}|\mathit{books\ on})}P(\mathit{table}|\mathit{on\ the})$$



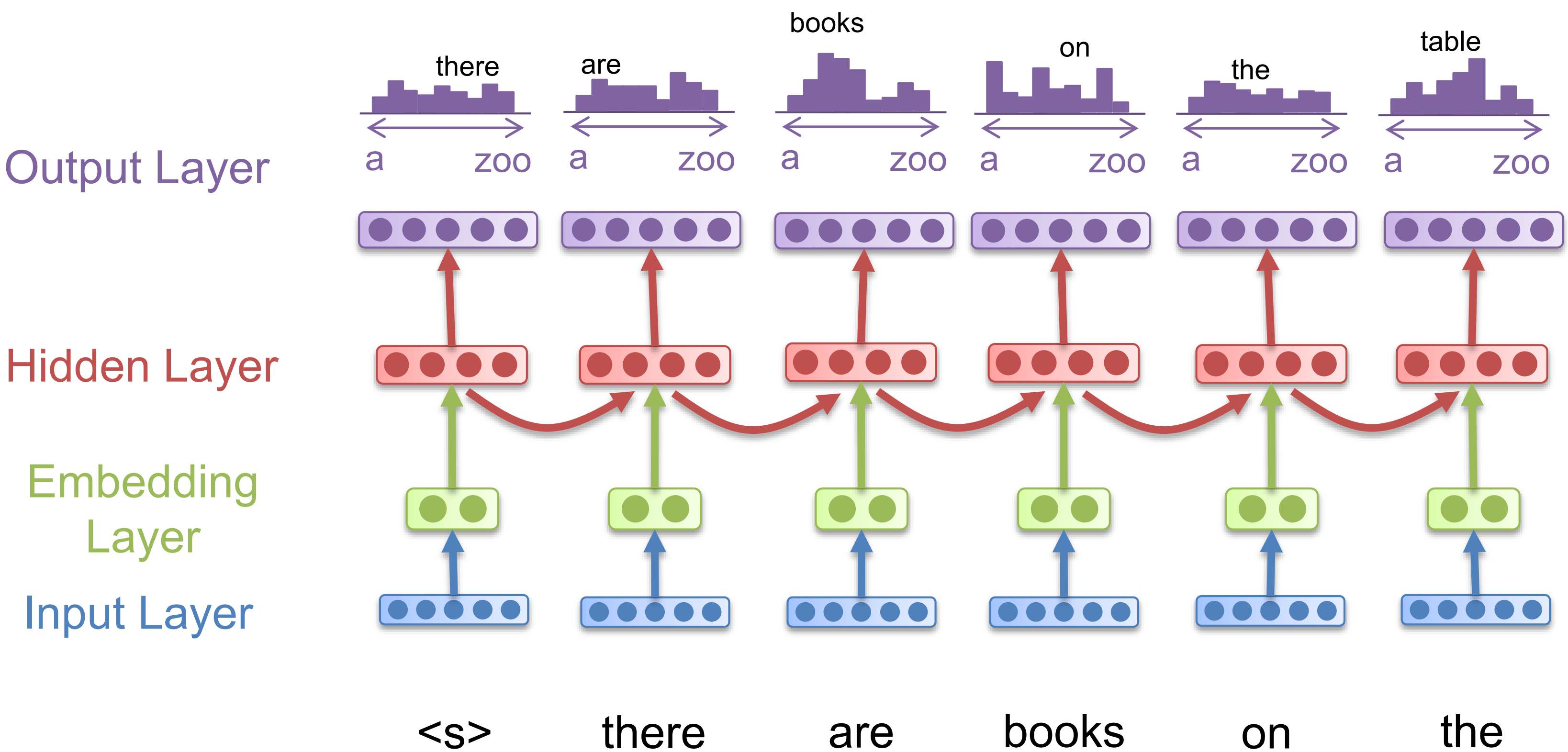
Revisiting NNLM – 2……

$$P(\textit{there are books on the table})$$
$$\approx P(\textit{there})P(\textit{are}|\textit{there})P(\textit{books}|\textit{there are})P(\textit{on}|\textit{are books})$$
$$P(\textit{the}|\textit{books on})\boldsymbol{P(\textit{table}|\textit{on the})}$$



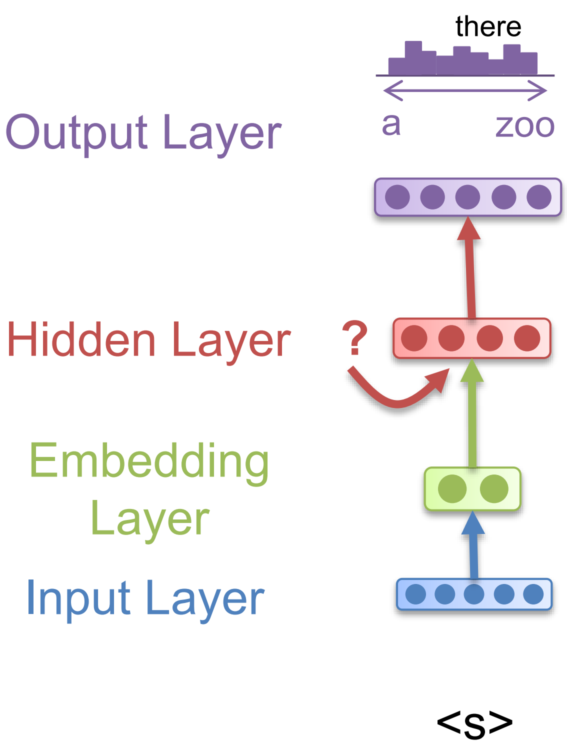
From NNLM to RNNLM

- The hidden state can encapsulate the information of word usage (ordering)
  - Leverage the information!!



RNNLM.

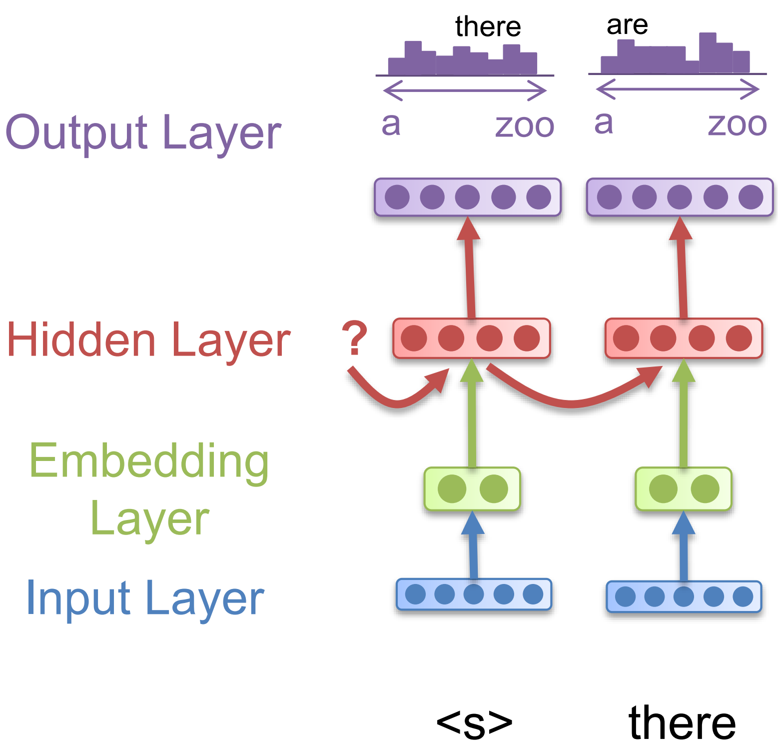
$$P(there\ are\ books\ on\ the\ table)$$
$$\approx \boldsymbol{P(there)}P(are|there)P(books|there\ are)P(on|there\ are\ books)$$
$$P(the|there\ are\ books\ on)P(table|there\ are\ books\ on\ the)$$



RNNLM..

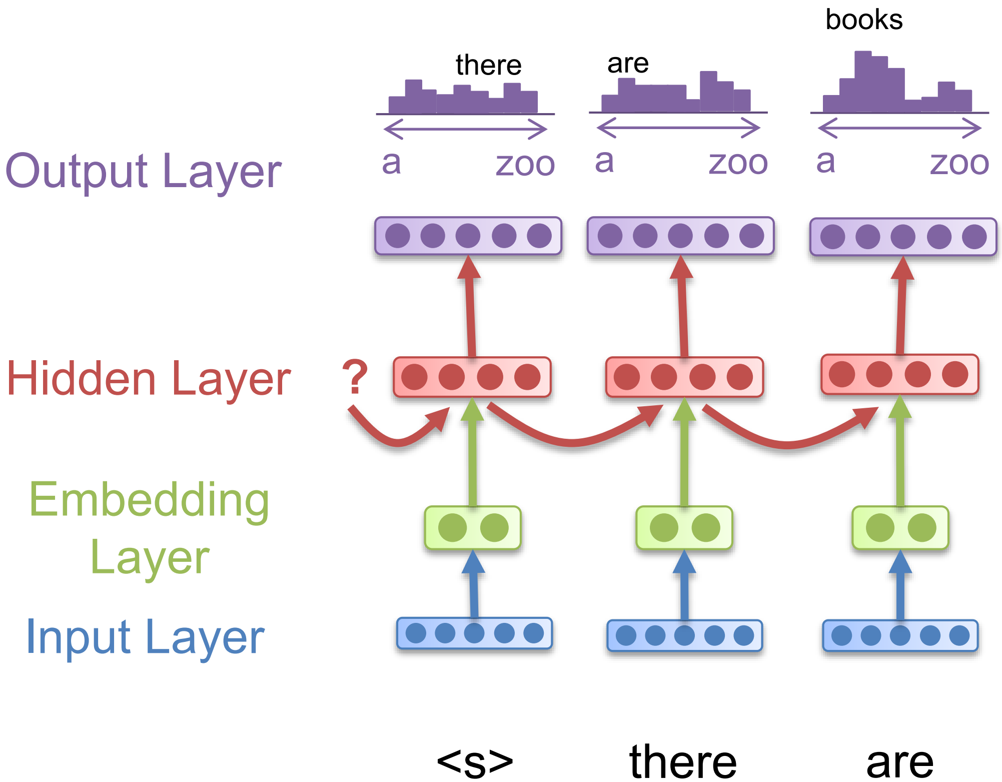
$P(there\ are\ books\ on\ the\ table)$

$\approx P(there)\boldsymbol{P(are|there)}P(books|there\ are)P(on|there\ are\ books)$
$P(the|there\ are\ books\ on)P(table|there\ are\ books\ on\ the)$



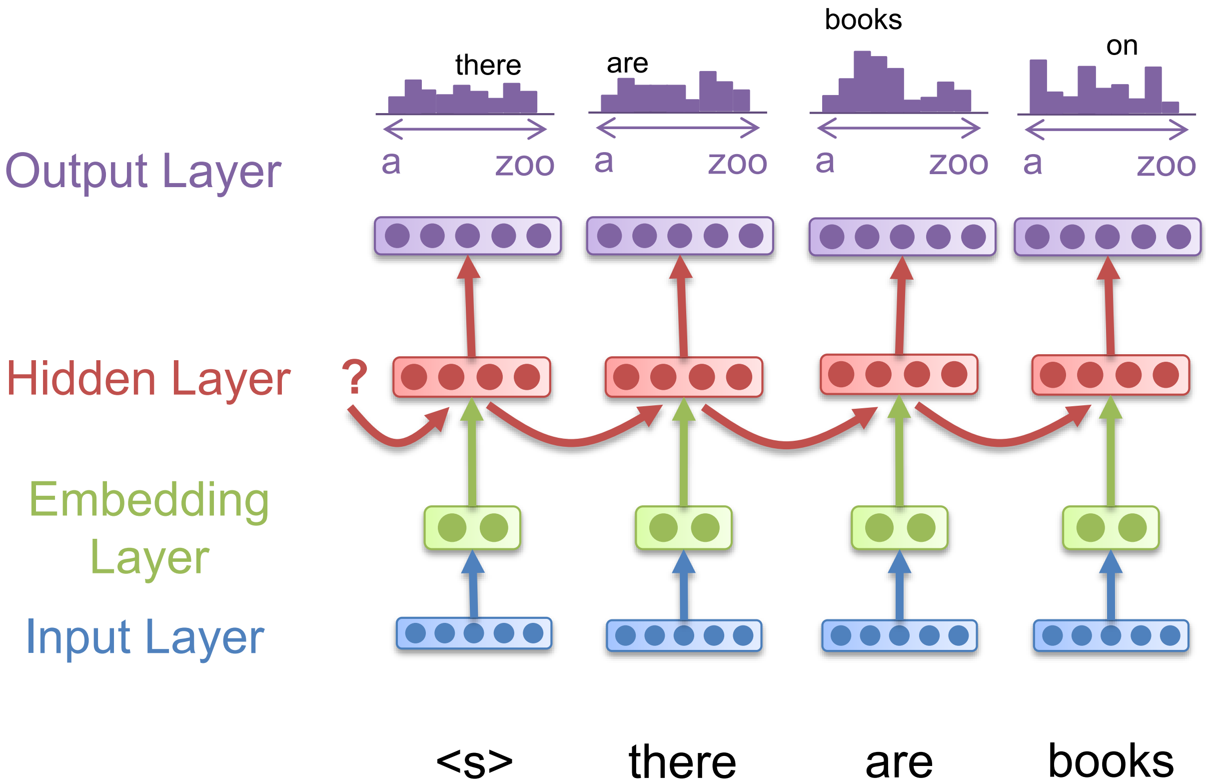
RNNLM…

$$P(\textit{there are books on the table})$$
$$\approx P(\textit{there})P(\textit{are}|\textit{there})\boldsymbol{P(\textit{books}|\textit{there are})}P(\textit{on}|\textit{there are books})$$
$$P(\textit{the}|\textit{there are books on})P(\textit{table}|\textit{there are books on the})$$



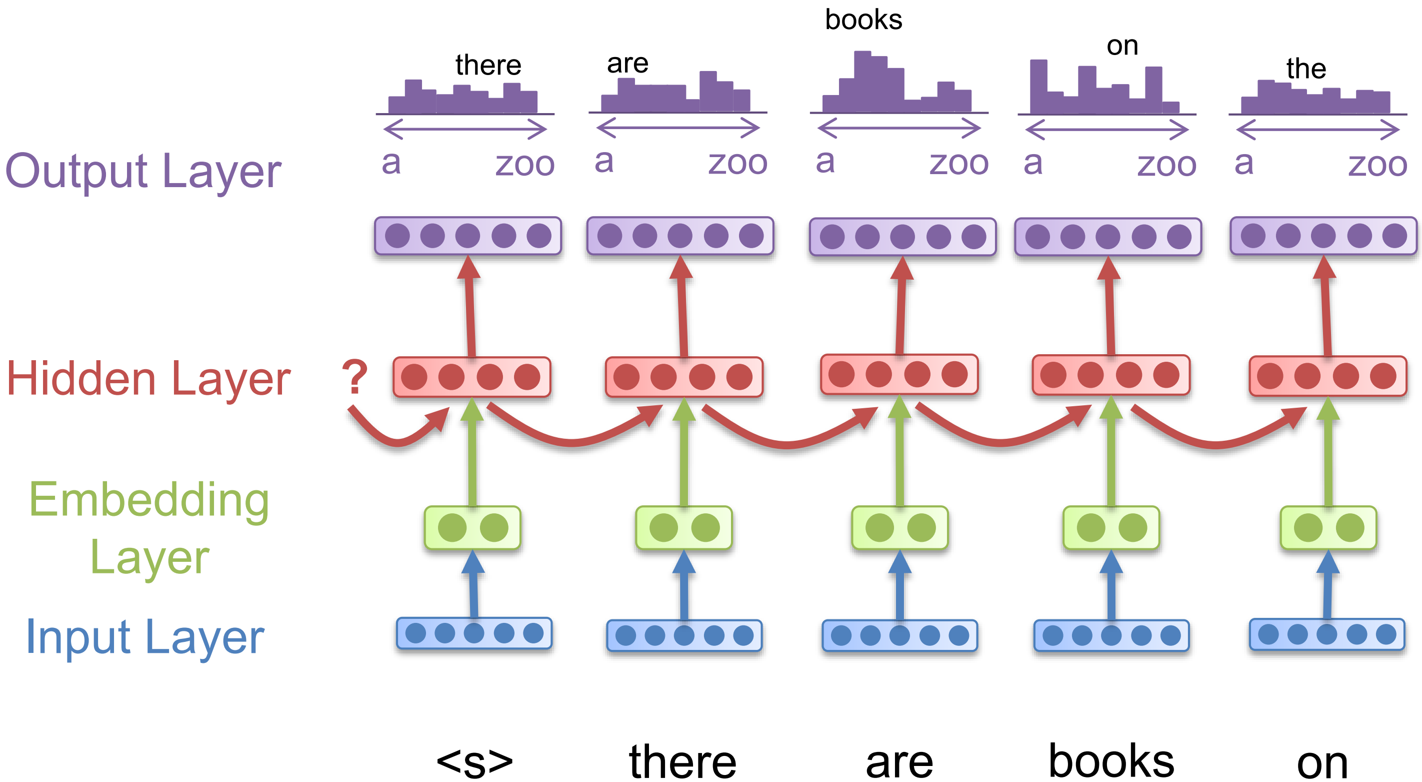
RNNLM….

$$P(there\ are\ books\ on\ the\ table)$$
$$\approx P(there)P(are|there)P(books|there\ are)P(on|there\ are\ books)$$
$$P(the|there\ are\ books\ on)P(table|there\ are\ books\ on\ the)$$



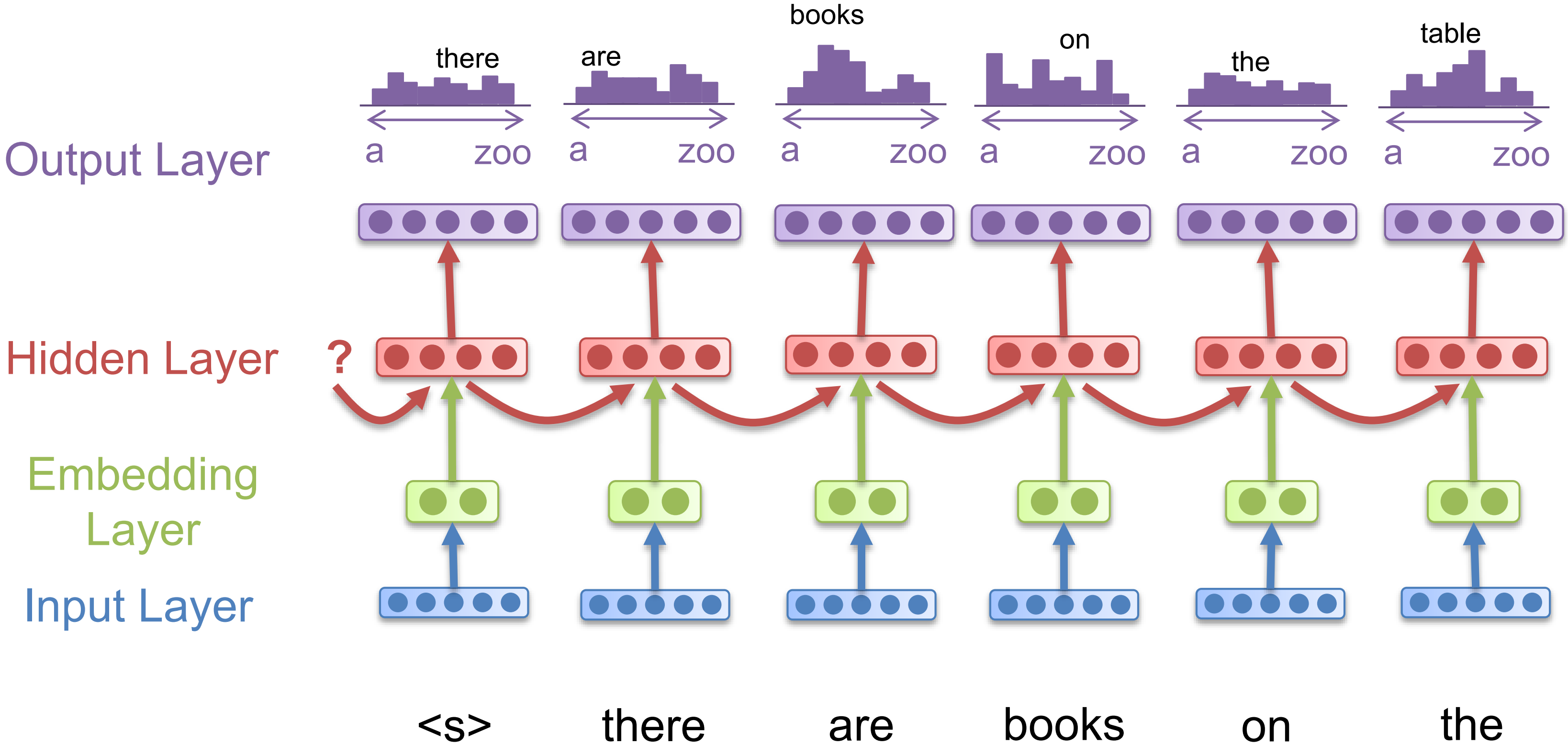
RNNLM.....

$$P(there\ are\ books\ on\ the\ table)$$
$$\approx P(there)P(are|there)P(books|there\ are)P(on|there\ are\ books)$$
$$\color{red}{\boldsymbol{P(the|there\ are\ books\ on)}}P(table|there\ are\ books\ on\ the)$$



RNNLM……

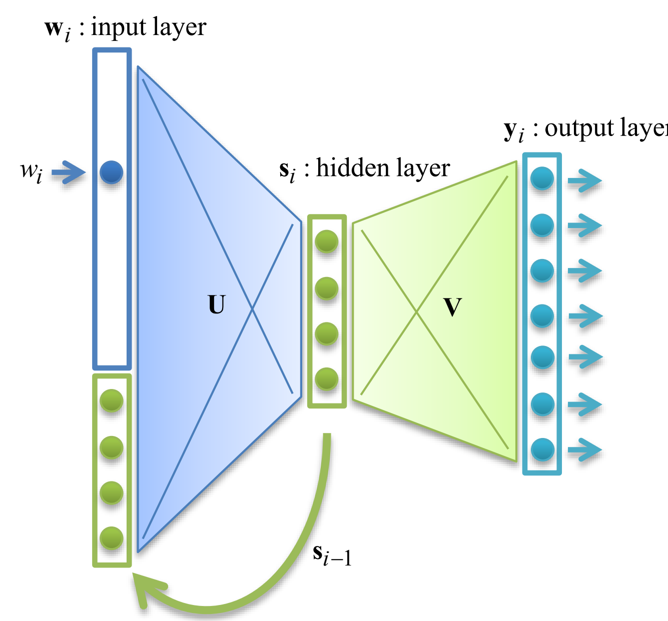
$$P(\mathit{there\ are\ books\ on\ the\ table})$$
$$\approx P(\mathit{there})P(\mathit{are}|\mathit{there})P(\mathit{books}|\mathit{there\ are})P(\mathit{on}|\mathit{there\ are\ books})$$
$$P(\mathit{the}|\mathit{there\ are\ books\ on})\boldsymbol{P(\mathit{table}|\mathit{there\ are\ books\ on\ the})}$$



Recurrent Neural Network LM

- RNNLM has recently emerged as a promising modeling framework for several tasks
  - Both **word usage cues** and **long-span structural information** of word co-occurrence relationships can be take into account naturally

- The limitations of the feed-forward NNLM
  - Need to specify the context length
  - RNN can efficiently represent more complex patterns than shallow NNs

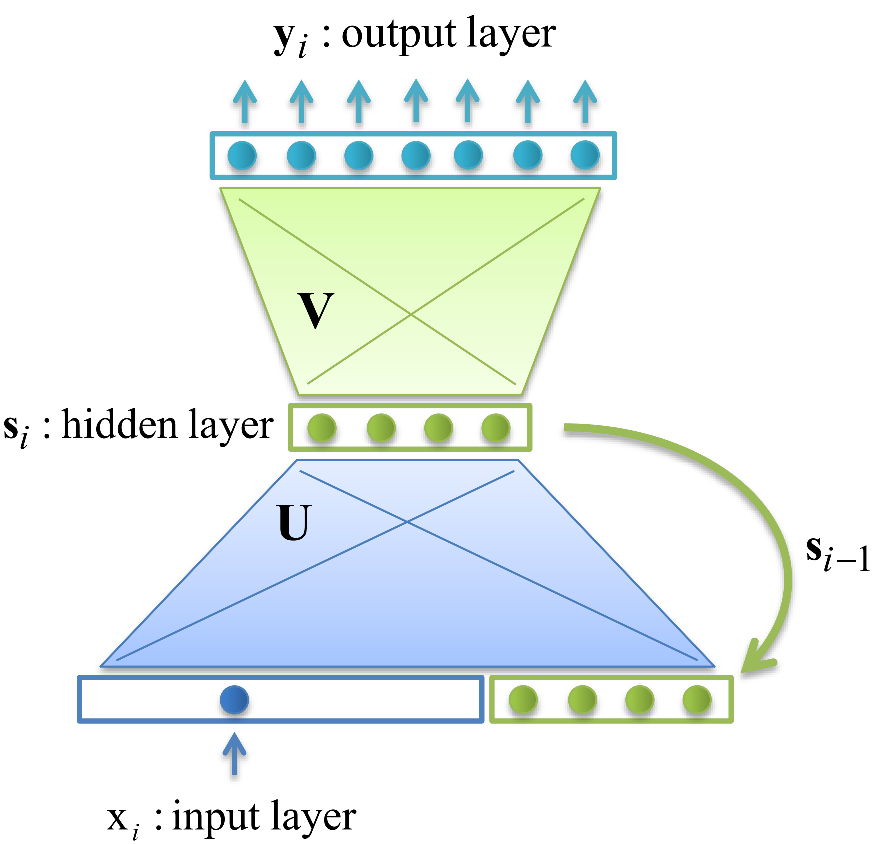


Compared with Topic Modeling

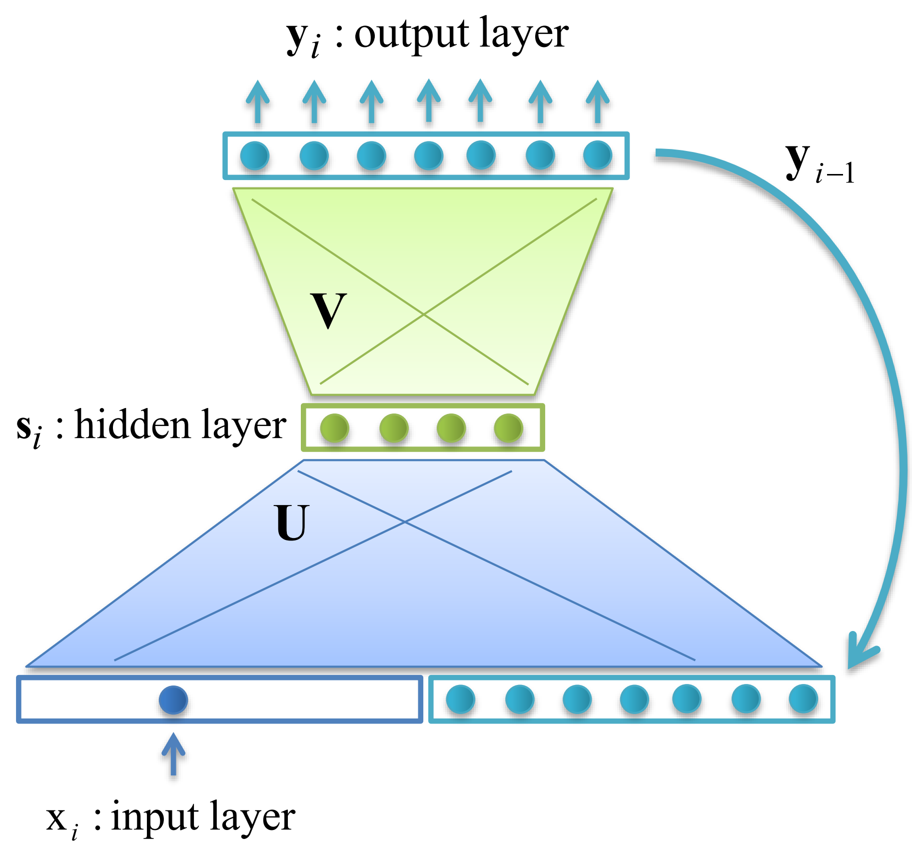
	RNNLM	Topic Models
Local and/or Long-span Information	Both bigram and long-span information	Long-span
Capture the Long-span Information	By the Hidden State	By EM Algorithm
The Combination Weight between Local and Long-span Information	Automatic Adaptation	Empirical Setting
The Importance of Each Word in the History	Automatic Learned	Equal Weight
Interpretability	No	Yes

Elman & Jordan Types

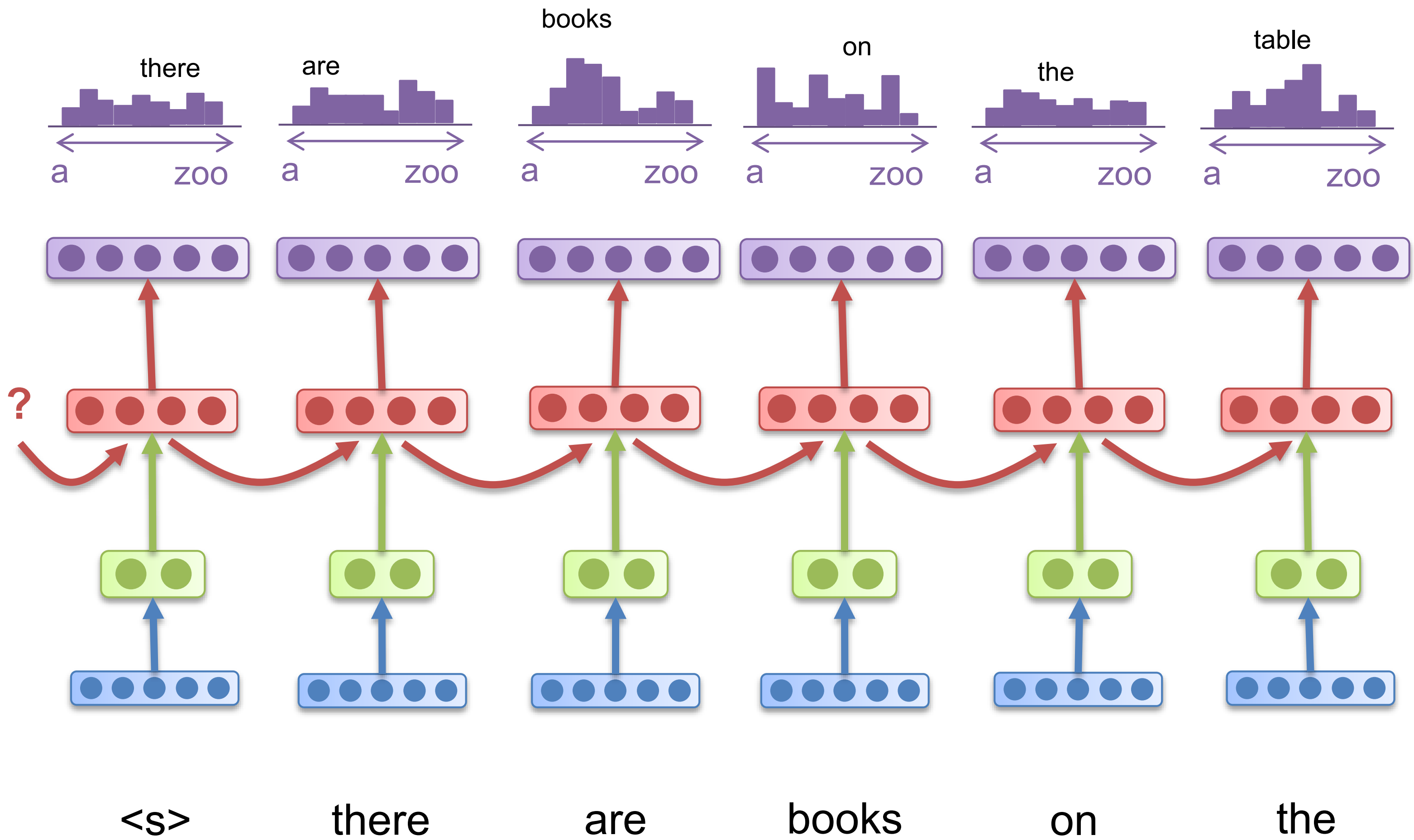
Elman Type RNN
SimpleRNN
Vanilla RNN



Jordan Type RNN

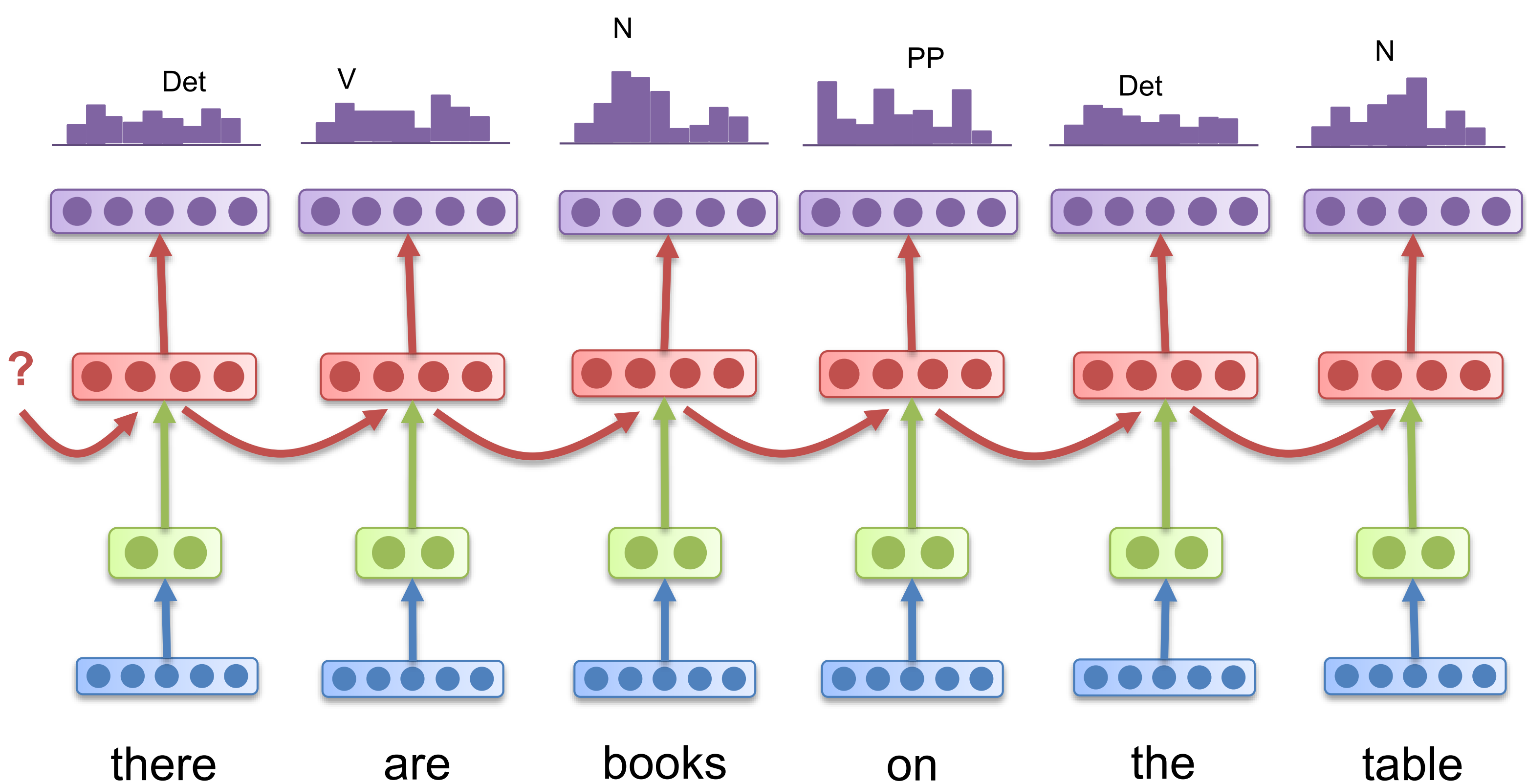


RNN for LM



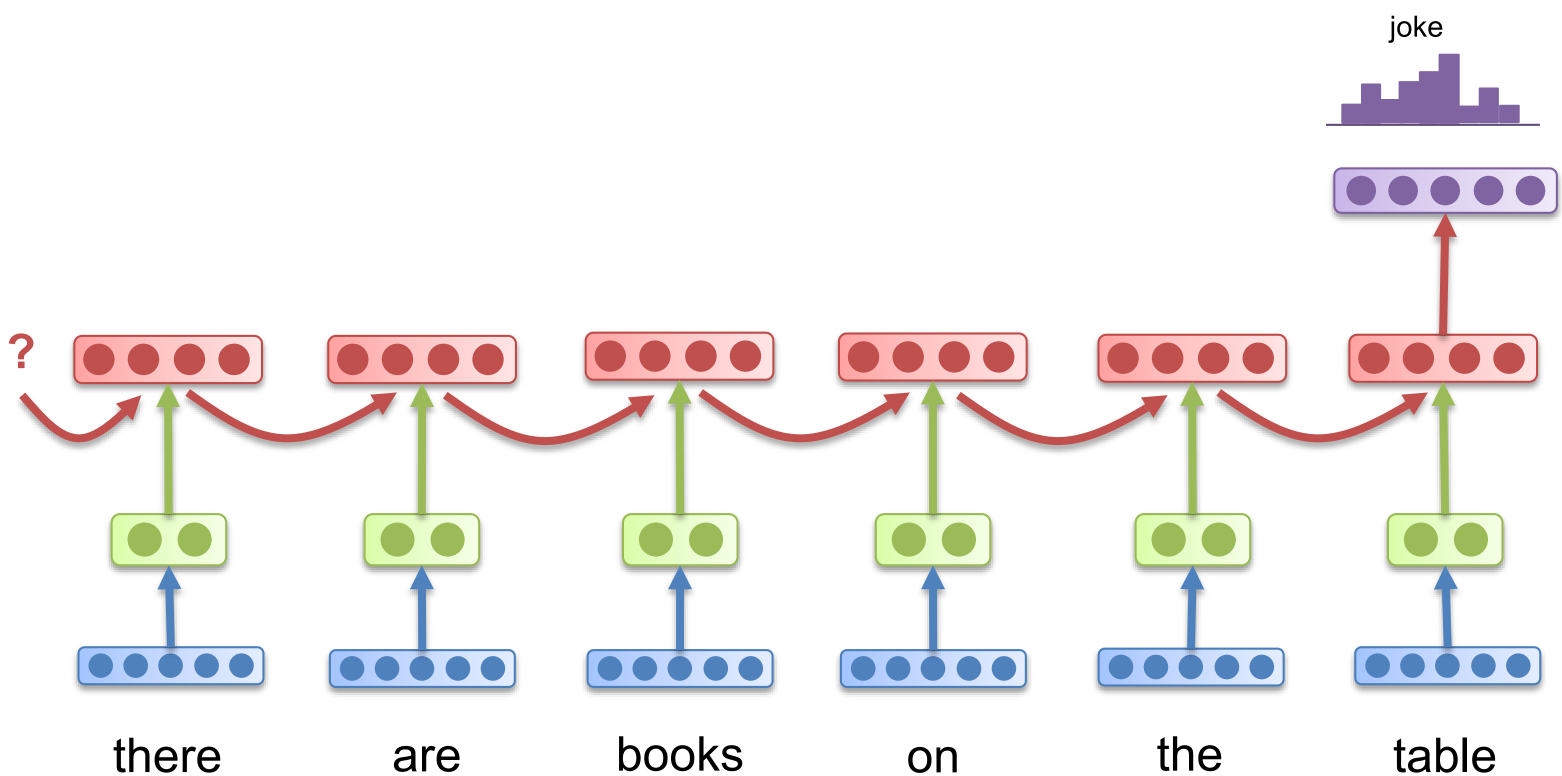
there
are
books
on
the
table
a
zoo
a
zoo
a
zoo
a
zoo
a
zoo
a
zoo
?
<s>
there
are
books
on
the

RNN for Tagging

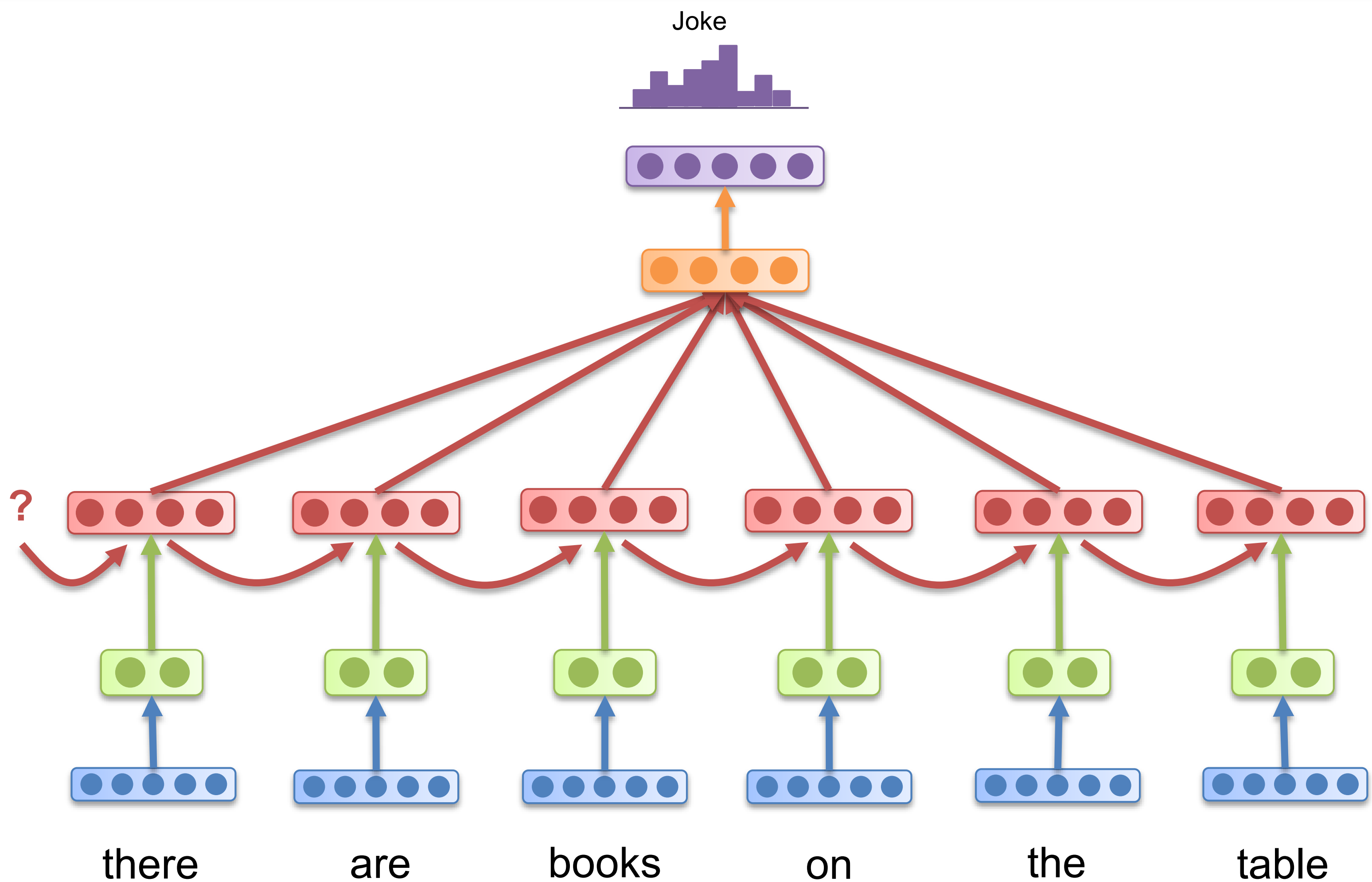


RNN for Classification – 1

- Forward RNN

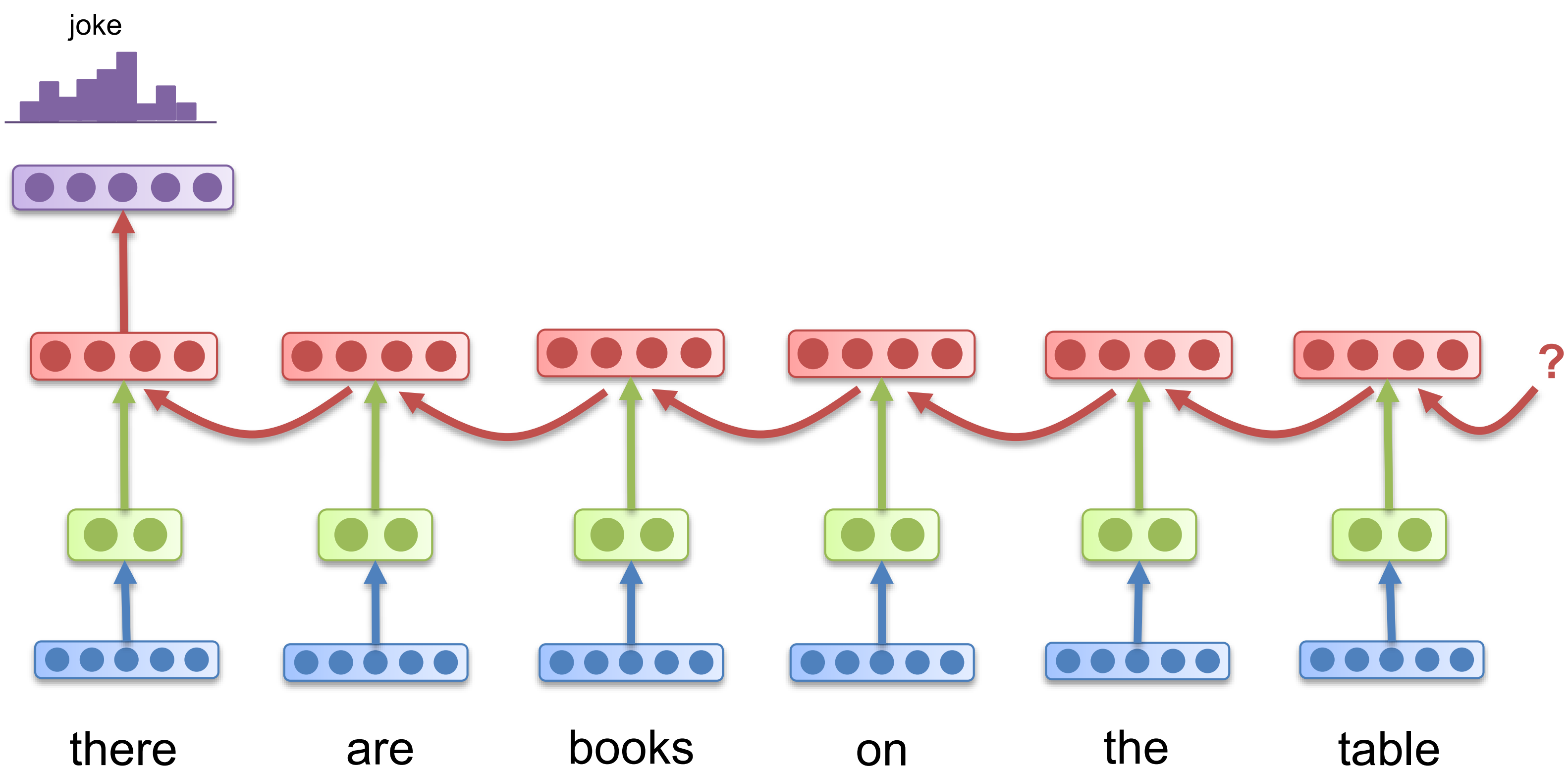


RNN for Classification – 2



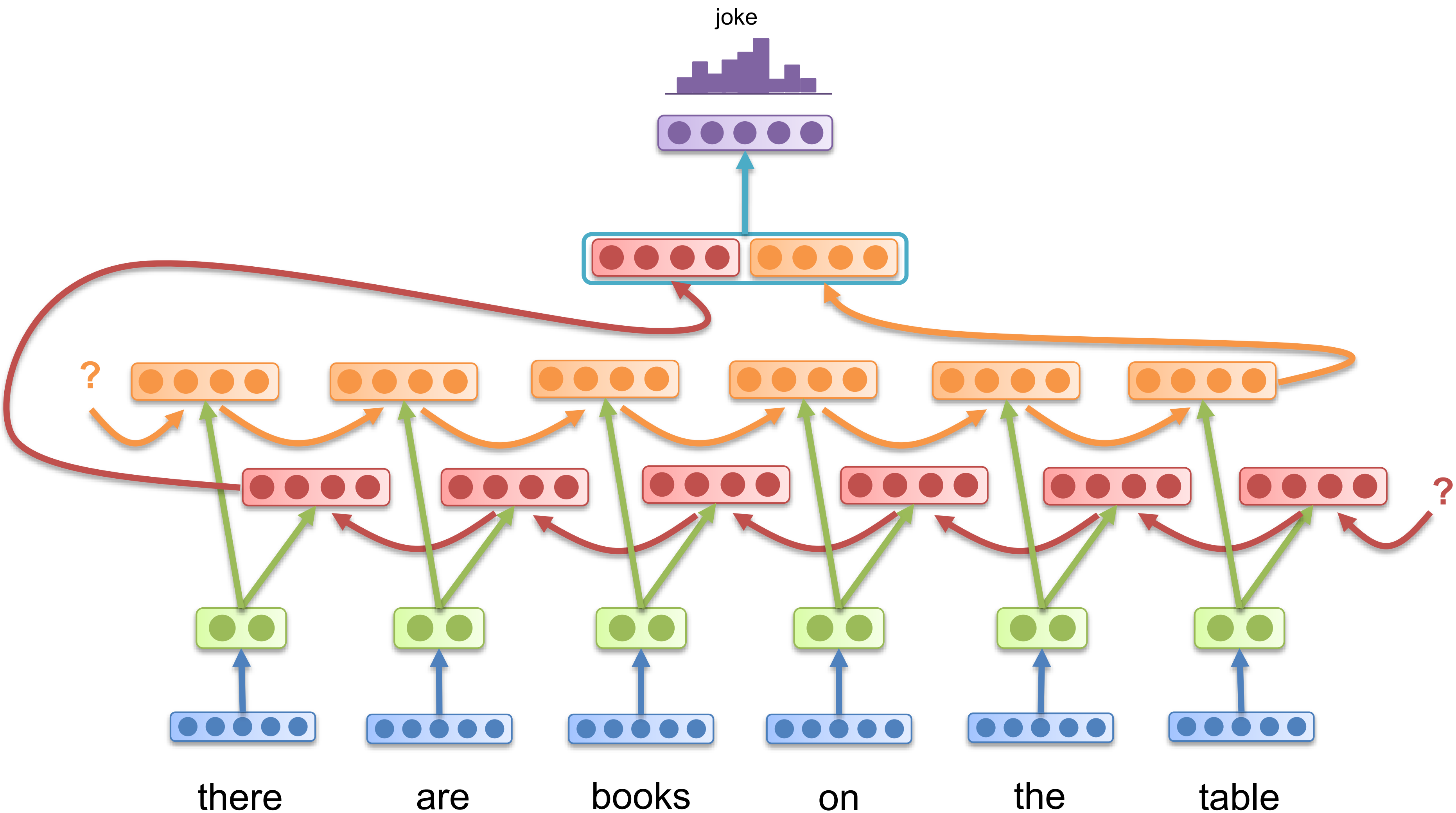
RNN for Classification – 3

- Backward RNN



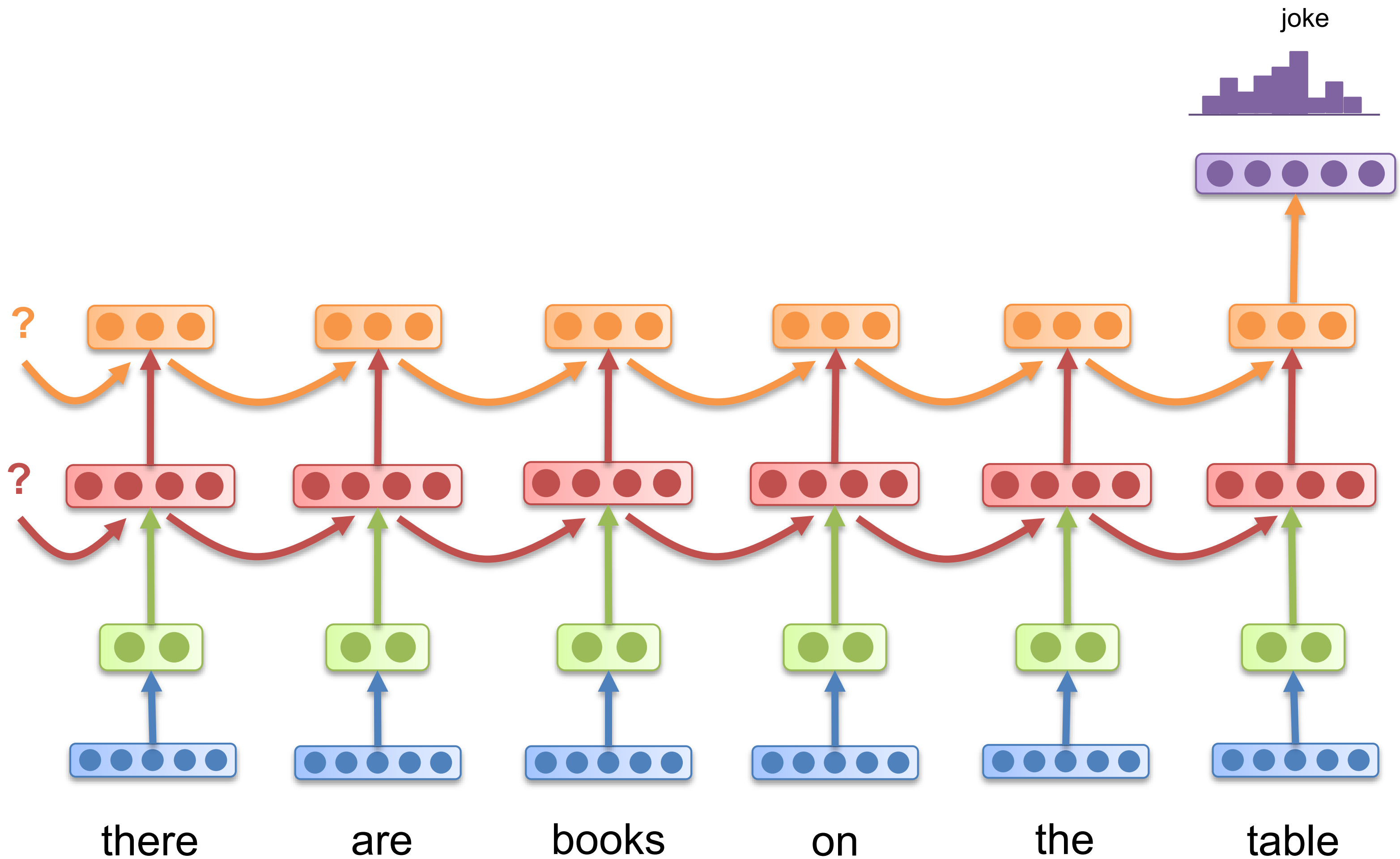
RNN for Classification – 4

- Bi-directional RNN!!



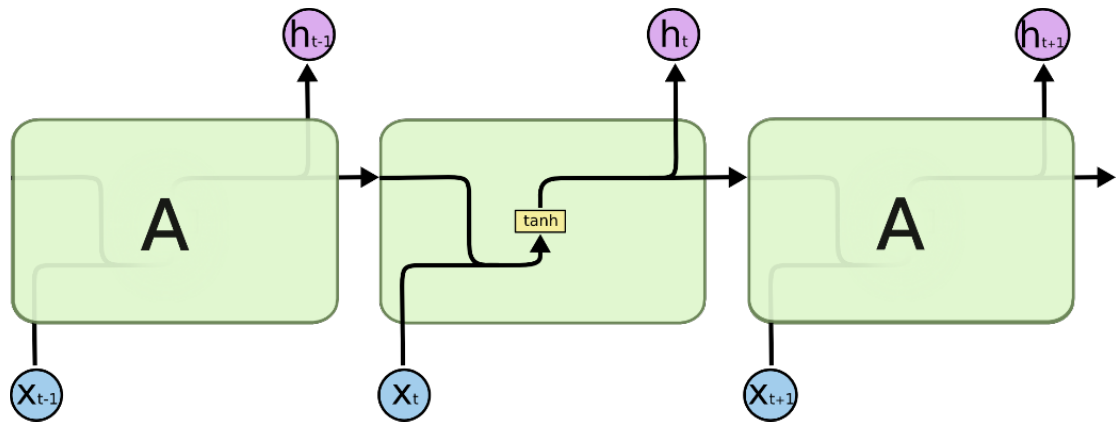
RNN for Classification – 5

- Multi-Layers RNN

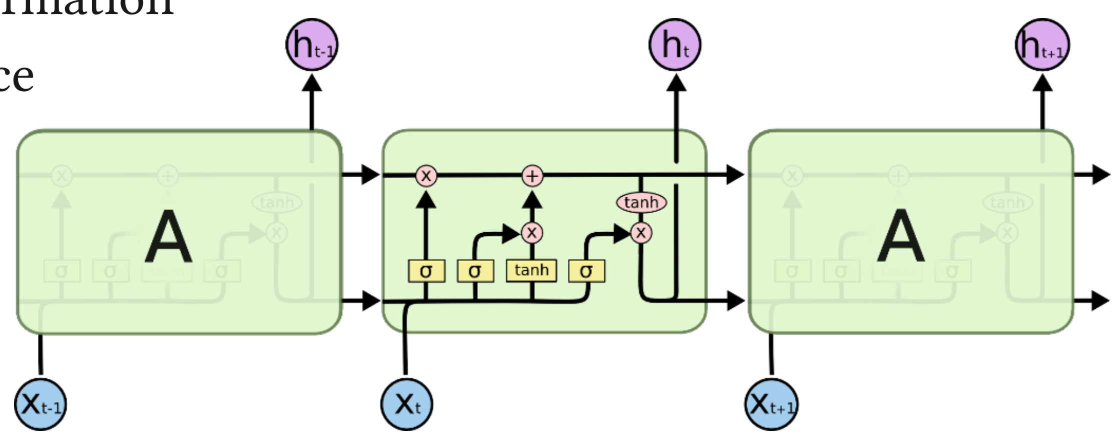


Long Short-Term Memory (LSTM)

- Learning to Forget!
  - RNN
    - The classic model



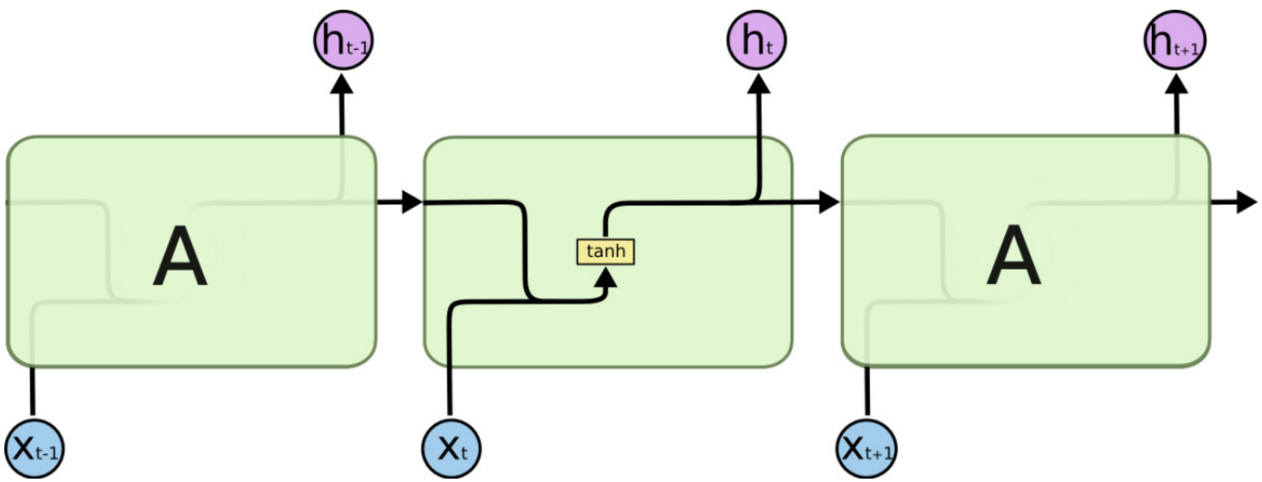
  - LSTM
    - Learning to forget
    - Capture longer information
    - Very slow in practice

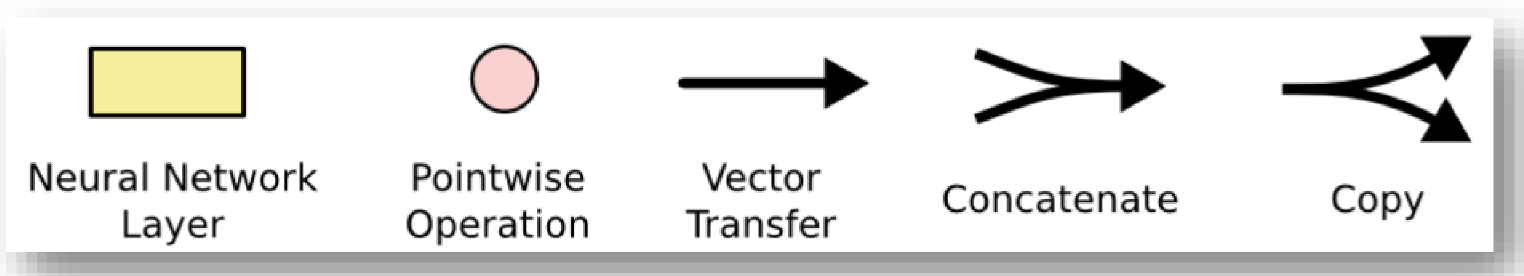


Vanilla RNN

- RNN is hard to capture long-term dependencies

$$h_t = \tanh(W\,[h_{t-1}, x_t] + b)$$

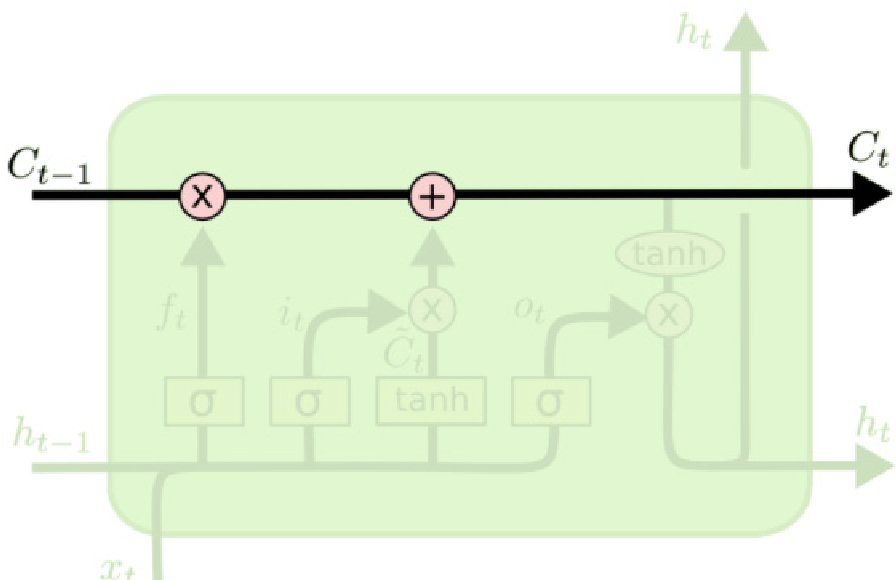




http://colah.github.io/posts/2015-08-Understanding-LSTMs/

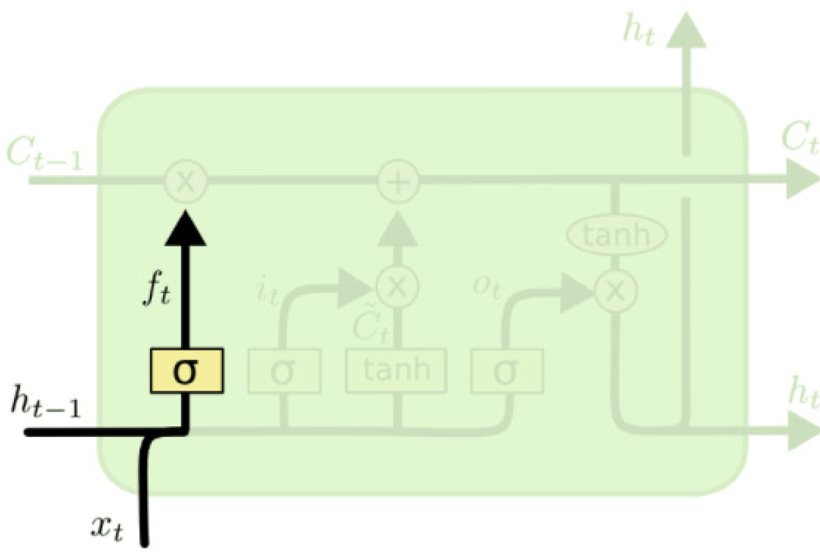
LSTM.

- The key to LSTMs is the **cell state**
  - The horizontal line running through the top of the diagram
  - It's very easy for information to just flow along it **unchanged**

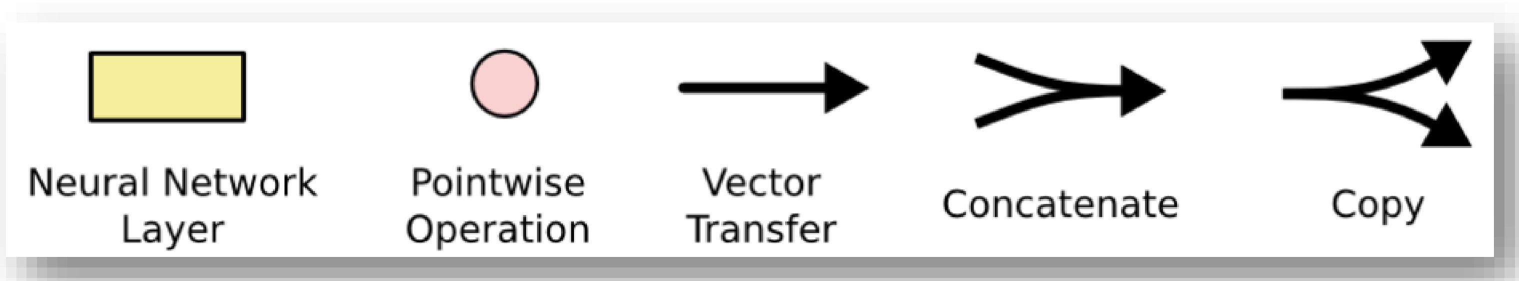


LSTM..

- The **forget gate** is to decide what information we're going to throw away from the cell state
  - $f_t = 1$: completely keep the information
  - $f_t = 0$: completely get rid of the information



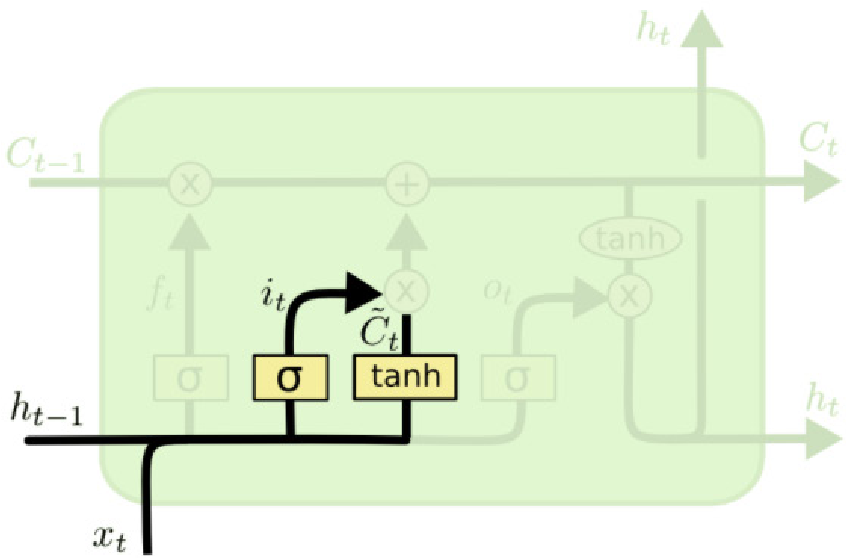
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$



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

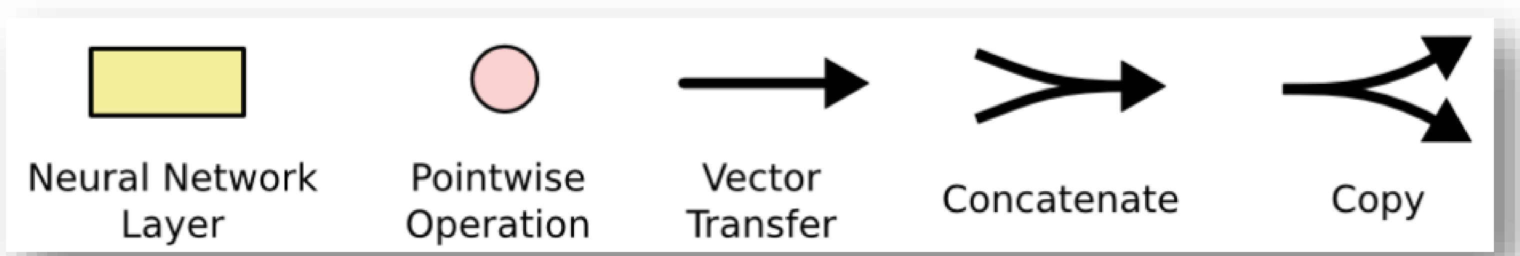
LSTM…

- The **input gate** is to decide which value we will update
  - A candidate vector, which contains the new information, will also be create



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

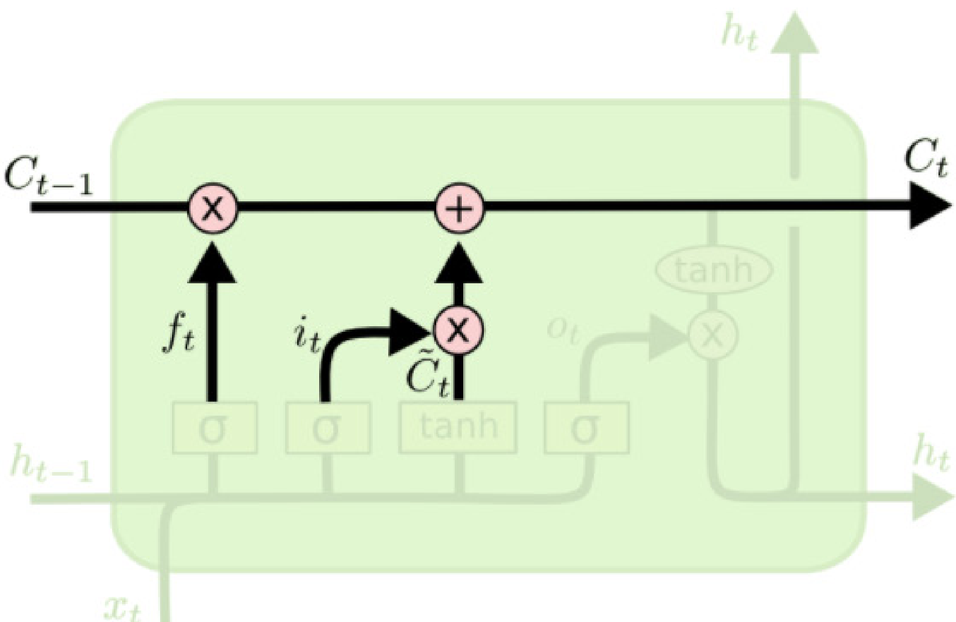
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$



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

LSTM….

- Update the cell state!

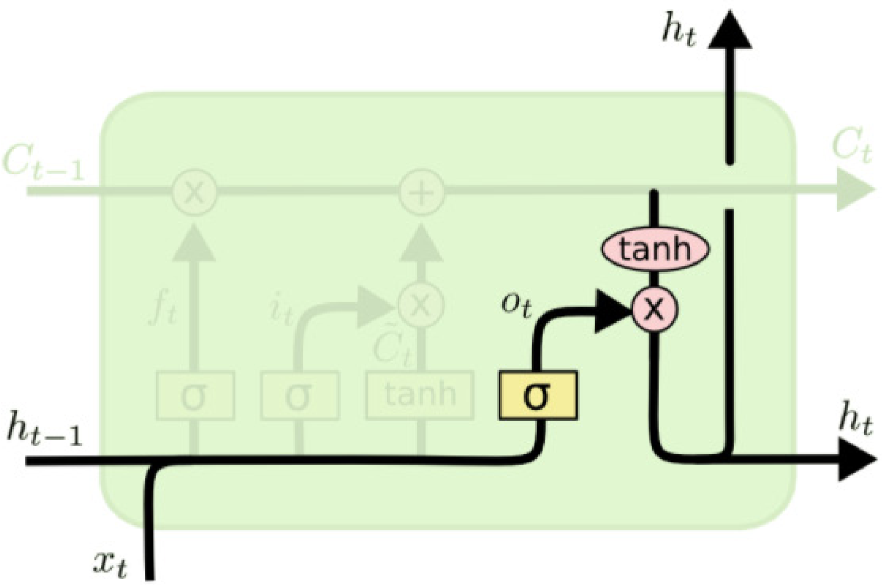


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

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

LSTM.....

- The **output gate** is a filter to select what information the model is going to output

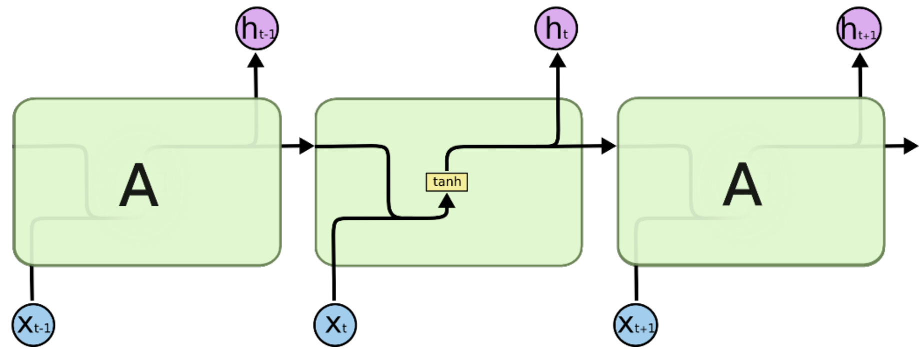


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

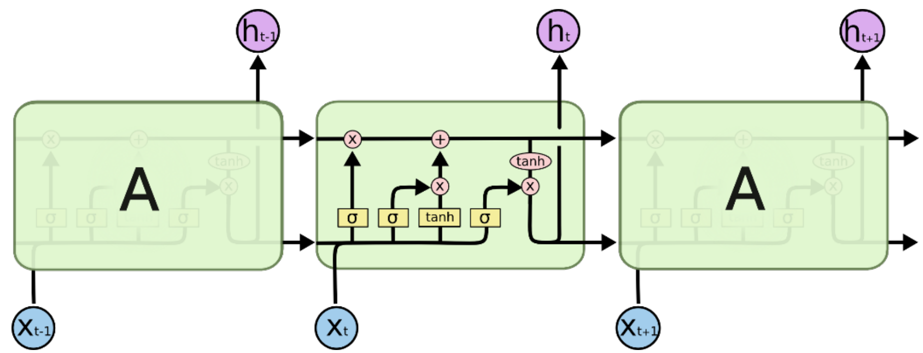
$$h_t = o_t * \tanh\left(C_t\right)$$

Gated Recurrent Unit (GRU)

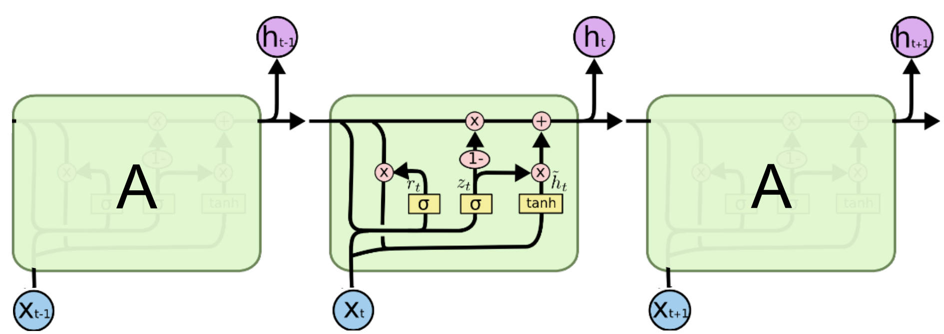
- RNN
  - The classic model



- LSTM
  - Learning to forget
  - Capture longer information
  - Very slow in practice



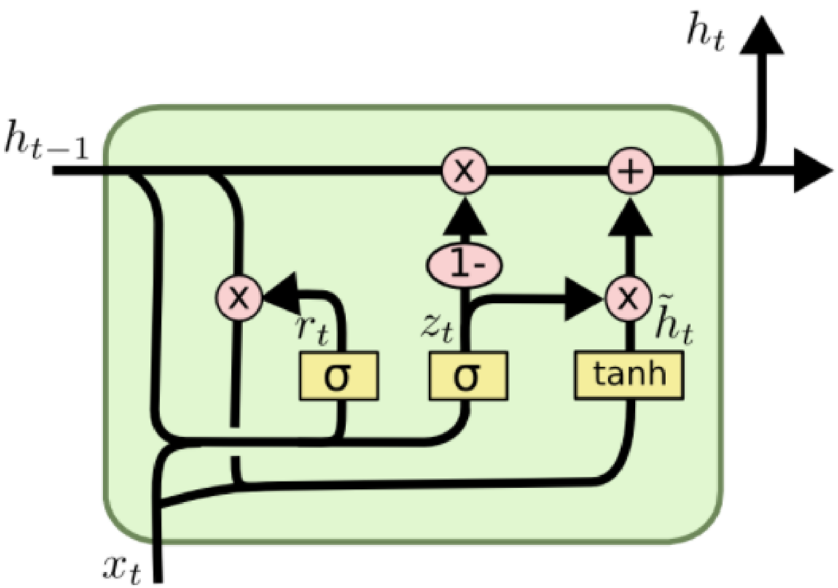
- GRU
  - A balanced choice



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

GRU

- GRU combines the forget and input gates into a **update gate**
- It also merges the cell state and hidden state, and a **reset gate** is used to control the previous information

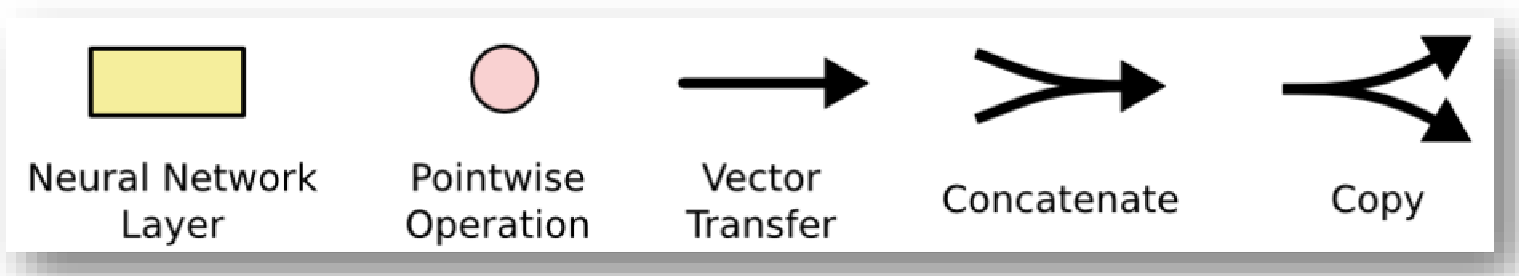


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$ **Update Gate**

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$ **Reset Gate**

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$



Questions?

kychen@mail.ntust.edu.tw