Natural Language Processing with Deep Learning
LSTM, GRU, and applications in summarization and contextualized word embeddings

Navid Rekab-Saz
navid.rekabsaz@jku.at

Institute of Computational Perception

JOHANNES KEPLER UNIVERSITY LINZ

Institute of Computational Perception

Agenda

- Vanishing/Exploding gradient
- RNNs with Gates: LSTM, GRU
- Contextualized word embeddings with RNNs
- Extractive summarization with RNNs

Some slides are adopted from http://web.stanford.edu/class/cs224n/

Element-wise Multiplication

- $\boldsymbol{a} \odot \boldsymbol{b} = \boldsymbol{c}$
  - dimensions: $1 \times d \odot 1 \times d = 1 \times d$

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \odot \begin{bmatrix} 3 & 0 & -2 \end{bmatrix} = \begin{bmatrix} 3 & 0 & -6 \end{bmatrix}$$

- $\boldsymbol{A} \odot \boldsymbol{B} = \boldsymbol{C}$
  - dimensions: $l \times m \odot l \times m = l \times m$

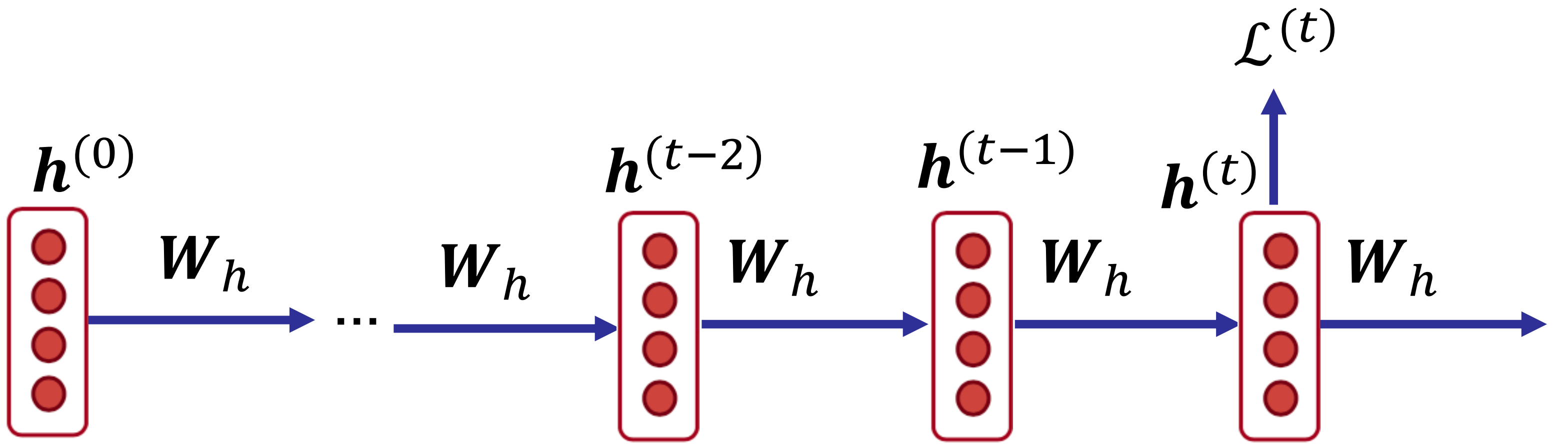
$$\begin{bmatrix} 2 & 3 \\ 0 & 1 \\ 1 & -1 \end{bmatrix} \odot \begin{bmatrix} -1 & 0 \\ 0 & 2 \\ 0.5 & -1 \end{bmatrix} = \begin{bmatrix} -2 & 0 \\ 0 & 2 \\ 0.5 & 1 \end{bmatrix}$$

Agenda

- **Vanishing/Exploding gradient**
- RNNs with Gates: LSTM, GRU
- Contextualized word embeddings with RNNs
- Extractive summarization with RNNs

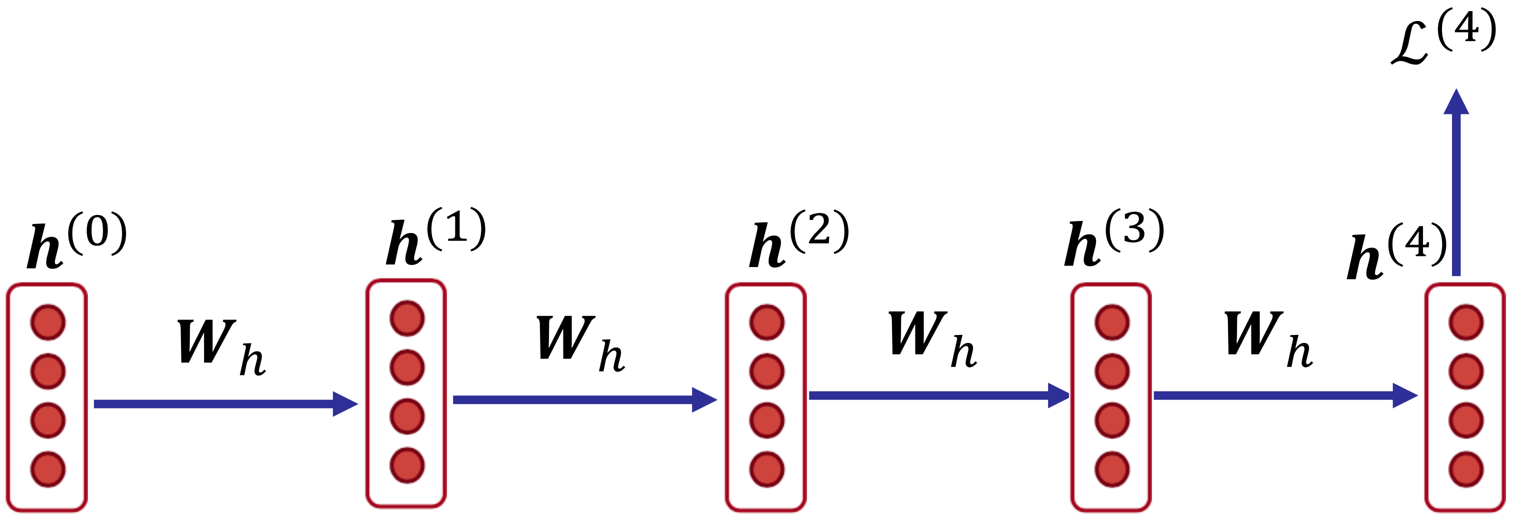
Recap: Backpropagation Through Time (BPTT)

- Unrolling RNN (simplified)



$$\frac{\partial \mathcal{L}^{(t)}}{\partial \boldsymbol{W}_h} = ?$$

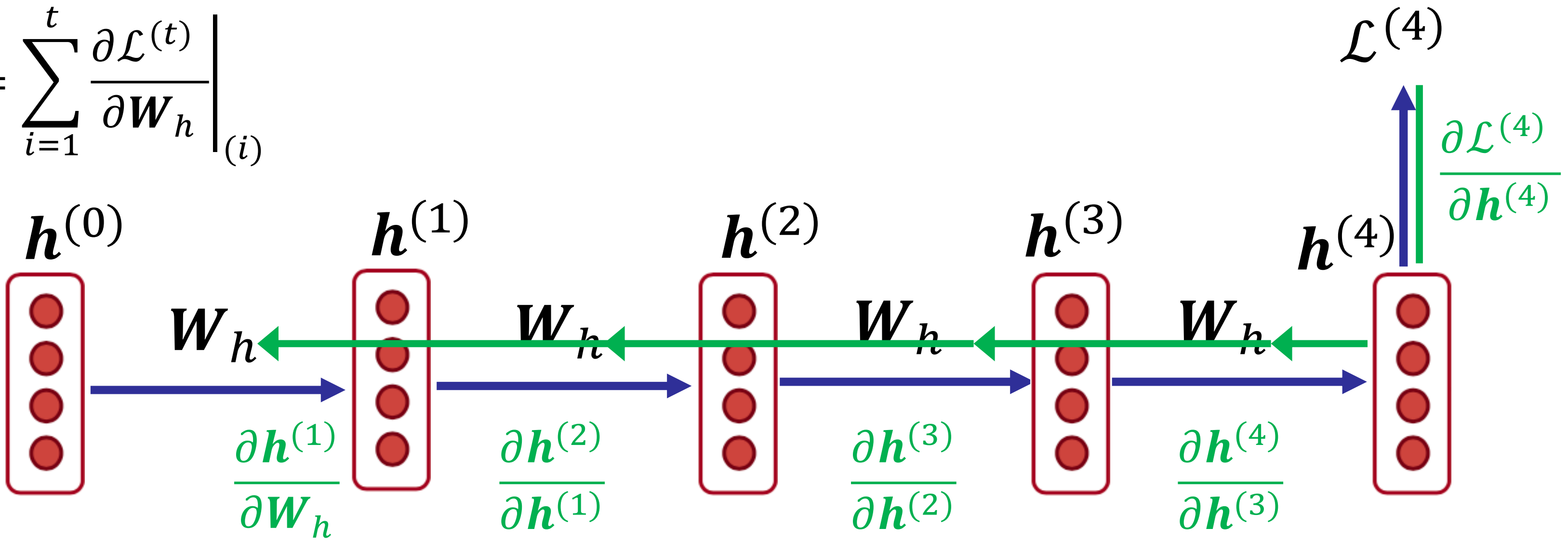
Recap: Backpropagation Through Time (BPTT)



$$\frac{\partial \mathcal{L}^{(4)}}{\partial \boldsymbol{W}_h} = ?$$

Recap: Backpropagation Through Time (BPTT)

$$\frac{\partial \mathcal{L}^{(t)}}{\partial \boldsymbol{W}_h} = \sum_{i=1}^{t} \left.\frac{\partial \mathcal{L}^{(t)}}{\partial \boldsymbol{W}_h}\right|_{(i)}$$



$\mathcal{L}^{(4)}$
$\frac{\partial \mathcal{L}^{(4)}}{\partial \boldsymbol{h}^{(4)}}$
$\boldsymbol{h}^{(0)}$
$\boldsymbol{h}^{(1)}$
$\boldsymbol{h}^{(2)}$
$\boldsymbol{h}^{(3)}$
$\boldsymbol{h}^{(4)}$
$\boldsymbol{W}_h$
$\boldsymbol{W}_h$
$\boldsymbol{W}_h$
$\boldsymbol{W}_h$
$\frac{\partial \boldsymbol{h}^{(1)}}{\partial \boldsymbol{W}_h}$
$\frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}}$
$\frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}}$
$\frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}}$

$$\left.\frac{\partial \mathcal{L}^{(4)}}{\partial \boldsymbol{W}_h}\right|_{(4)} = \frac{\partial \mathcal{L}^{(4)}}{\partial \boldsymbol{h}^{(4)}}\frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{W}_h}$$

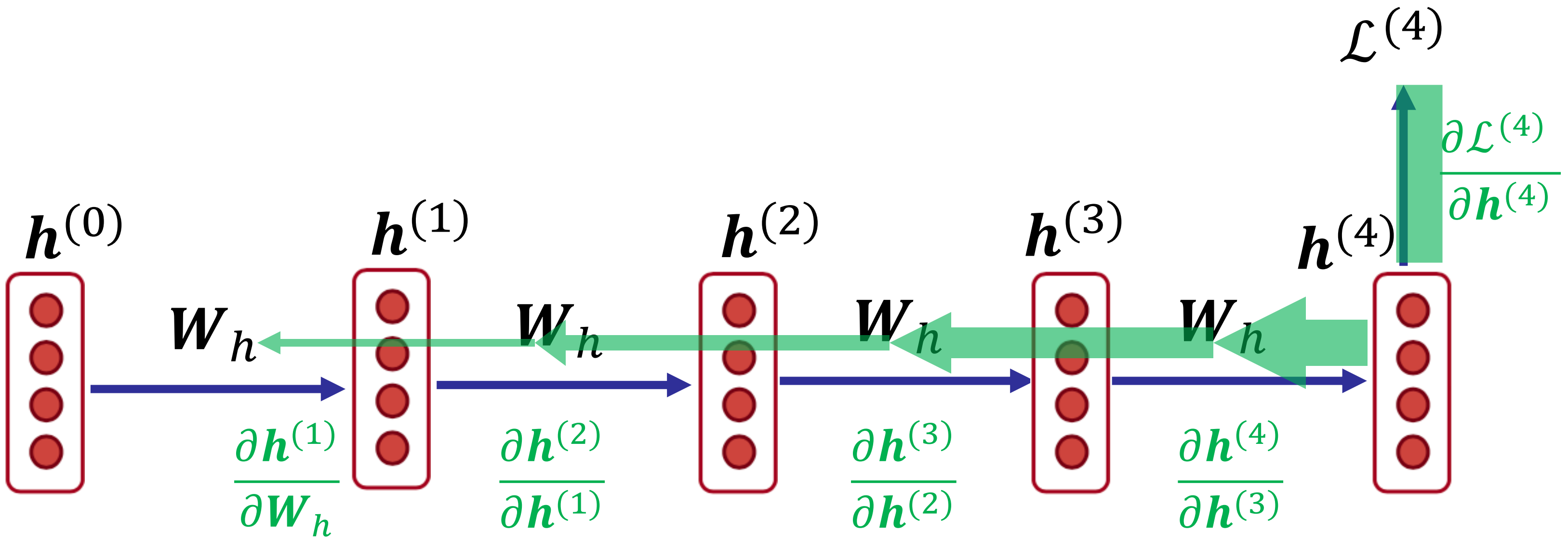
$$\left.\frac{\partial \mathcal{L}^{(4)}}{\partial \boldsymbol{W}_h}\right|_{(3)} = \frac{\partial \mathcal{L}^{(4)}}{\partial \boldsymbol{h}^{(4)}}\frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}}\frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{W}_h}$$

$$\left.\frac{\partial \mathcal{L}^{(4)}}{\partial \boldsymbol{W}_h}\right|_{(2)} = \frac{\partial \mathcal{L}^{(4)}}{\partial \boldsymbol{h}^{(4)}}\frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}}\frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}}\frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{W}_h}$$

$$\left.\frac{\partial \mathcal{L}^{(4)}}{\partial \boldsymbol{W}_h}\right|_{(1)} = \frac{\partial \mathcal{L}^{(4)}}{\partial \boldsymbol{h}^{(4)}}\frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}}\frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}}\frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}}\frac{\partial \boldsymbol{h}^{(1)}}{\partial \boldsymbol{W}_h}$$

$$\left.\frac{\partial \mathcal{L}^{(t)}}{\partial \boldsymbol{W}_h}\right|_{(i)} = \frac{\partial \mathcal{L}^{(t)}}{\partial \boldsymbol{h}^{(t)}}\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}}\cdots\frac{\partial \boldsymbol{h}^{(i)}}{\partial \boldsymbol{W}_h}$$

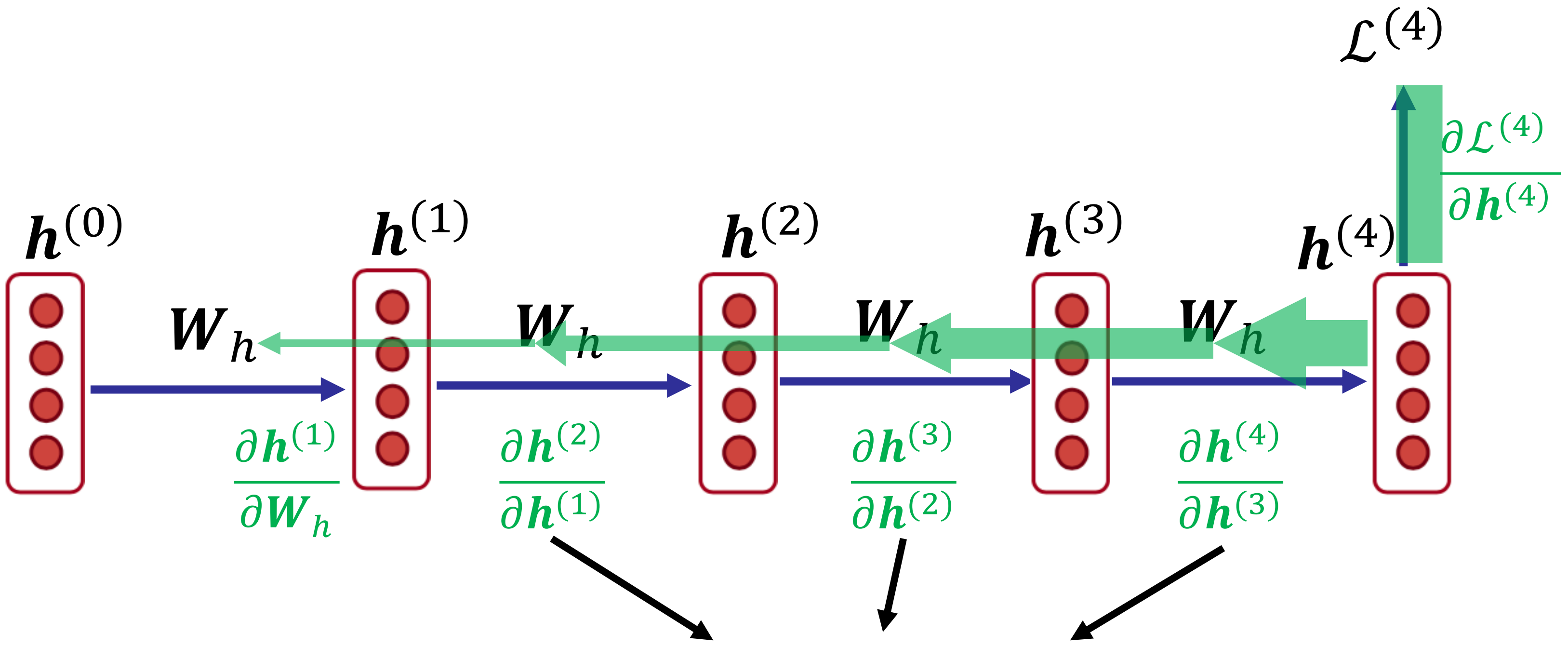
Vanishing/Exploding gradient


$\mathcal{L}^{(4)}$
$\frac{\partial \mathcal{L}^{(4)}}{\partial \boldsymbol{h}^{(4)}}$
$\boldsymbol{h}^{(0)}$
$\boldsymbol{h}^{(1)}$
$\boldsymbol{h}^{(2)}$
$\boldsymbol{h}^{(3)}$
$\boldsymbol{h}^{(4)}$
$\boldsymbol{W}_h$
$\boldsymbol{W}_h$
$\boldsymbol{W}_h$
$\boldsymbol{W}_h$
$\frac{\partial \boldsymbol{h}^{(1)}}{\partial \boldsymbol{W}_h}$
$\frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}}$
$\frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}}$
$\frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}}$

- In practice, the gradient regarding each time step becomes smaller and smaller as it goes back in time → Vanishing gradient

- While less often, this may also happen other way around: the gradient regarding further time steps becomes larger and larger→ Exploding gradient

Vanishing/Exploding gradient – why?



$\mathcal{L}^{(4)}$
$\frac{\partial \mathcal{L}^{(4)}}{\partial \boldsymbol{h}^{(4)}}$
$\boldsymbol{h}^{(0)}$
$\boldsymbol{h}^{(1)}$
$\boldsymbol{h}^{(2)}$
$\boldsymbol{h}^{(3)}$
$\boldsymbol{h}^{(4)}$
$\boldsymbol{W}_h$
$\boldsymbol{W}_h$
$\boldsymbol{W}_h$
$\boldsymbol{W}_h$
$\frac{\partial \boldsymbol{h}^{(1)}}{\partial \boldsymbol{W}_h}$
$\frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}}$
$\frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}}$
$\frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}}$

If these gradients are small, their multiplication gets smaller. As we go further back, the final gradient contains more of these!

$$\left.\frac{\partial \mathcal{L}^{(4)}}{\partial \boldsymbol{W}_h}\right|_{(1)} = \frac{\partial \mathcal{L}^{(4)}}{\partial \boldsymbol{h}^{(4)}} \frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}} \frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \frac{\partial \boldsymbol{h}^{(1)}}{\partial \boldsymbol{W}_h}$$

Vanishing/Exploding gradient – why?

- What is $\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}}$?!
- Recall the definition of RNN:

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

- Let's replace sigmoid (σ) with a simple linear activation ($y = x$) function.

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

- In this case:

$$\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}} = \boldsymbol{W}_h$$

Vanishing/Exploding gradient – why?

- Recall the BPTT formula (for the simplified case):

$$\left.\frac{\partial \mathcal{L}^{(t)}}{\partial \boldsymbol{W}_h}\right|_{(i)} = \frac{\partial \mathcal{L}^{(t)}}{\partial \boldsymbol{h}^{(t)}} \underbrace{\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}} \cdots \frac{\partial \boldsymbol{h}^{(i+1)}}{\partial \boldsymbol{h}^{(i)}}} \frac{\partial \boldsymbol{h}^{(i)}}{\partial \boldsymbol{W}_h}$$

- Given $l = t - i$, the BPTT formula can be rewritten as:

$$\left.\frac{\partial \mathcal{L}^{(t)}}{\partial \boldsymbol{W}_h}\right|_{(i)} = \frac{\partial \mathcal{L}^{(t)}}{\partial \boldsymbol{h}^{(t)}} \boxed{(\boldsymbol{W}_h)^l} \frac{\partial \boldsymbol{h}^{(i)}}{\partial \boldsymbol{W}_h}$$

$\downarrow$

If weights in $\boldsymbol{W}_h$ are small (i.e. eigenvalues of $\boldsymbol{W}_h$ are smaller then 1), these term gets *exponentially* smaller

Why is vanishing/exploding gradient a problem?

- Vanishing gradient
  - Gradient signal from faraway “fades away” and becomes insignificant in comparison with the gradient signal from close-by
  - Long-term dependencies are not captured, since model weights are updated only with respect to near effects

  → one approach to address it: RNNs with gates – LSTM, GRU

- Exploding gradient
  - Gradients become too big → SGD update steps become too large
  - This causes (large loss values and) large updates on parameters, and eventually unstable training

  → main approach to address it: Gradient clipping

Gradient clipping

- Gradient clipping: if the norm of the gradient is greater than some threshold, scale the gradient down

Algorithm 1 Pseudo-code for norm clipping

```
$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ then
    $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if
```

- Intuition: take a step in the same direction, but in a smaller step

Problem with vanilla RNN – summary

- It is too difficult for the hidden state of vanilla RNN to learn and preserve information of several time steps
  - In particular as new contents are constantly added to the hidden state in every step

$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{h}^{(t-1)}\boldsymbol{W}_h + \underbrace{\boxed{\boldsymbol{x}^{(t)}\boldsymbol{W}_e}}_{\text{In every step, input vector “adds” new content to hidden state}} + \boldsymbol{b})$$

Agenda

- Vanishing & Exploding gradient
- **RNNs with Gates: LSTM, GRU**
- Contextualized word embeddings with LSTM
- Extractive summarization with LSTM

Gate vector

- Gate vector:
  - A vector with values between 0 and 1
  - Gate vector acts as "gate-keeper", such that it controls the content flow of another vector
- Gate vectors are typically defined using sigmoid:

$$\boldsymbol{g} = \sigma(some\ vector)$$

… and are applied to a vector $\boldsymbol{v}$ with element-wise multiplication to control its contents:

$$\boldsymbol{g} \odot \boldsymbol{v}$$

- For each element (feature) i of the vectors:
  - If g_i is 1 $\rightarrow v_i$ remains the same; everything passes; *open* gate!
  - If g_i is 0 $\rightarrow v_i$ becomes 0; nothing passes; *closed* gate!

Long Short-Term Memory (LSTM)

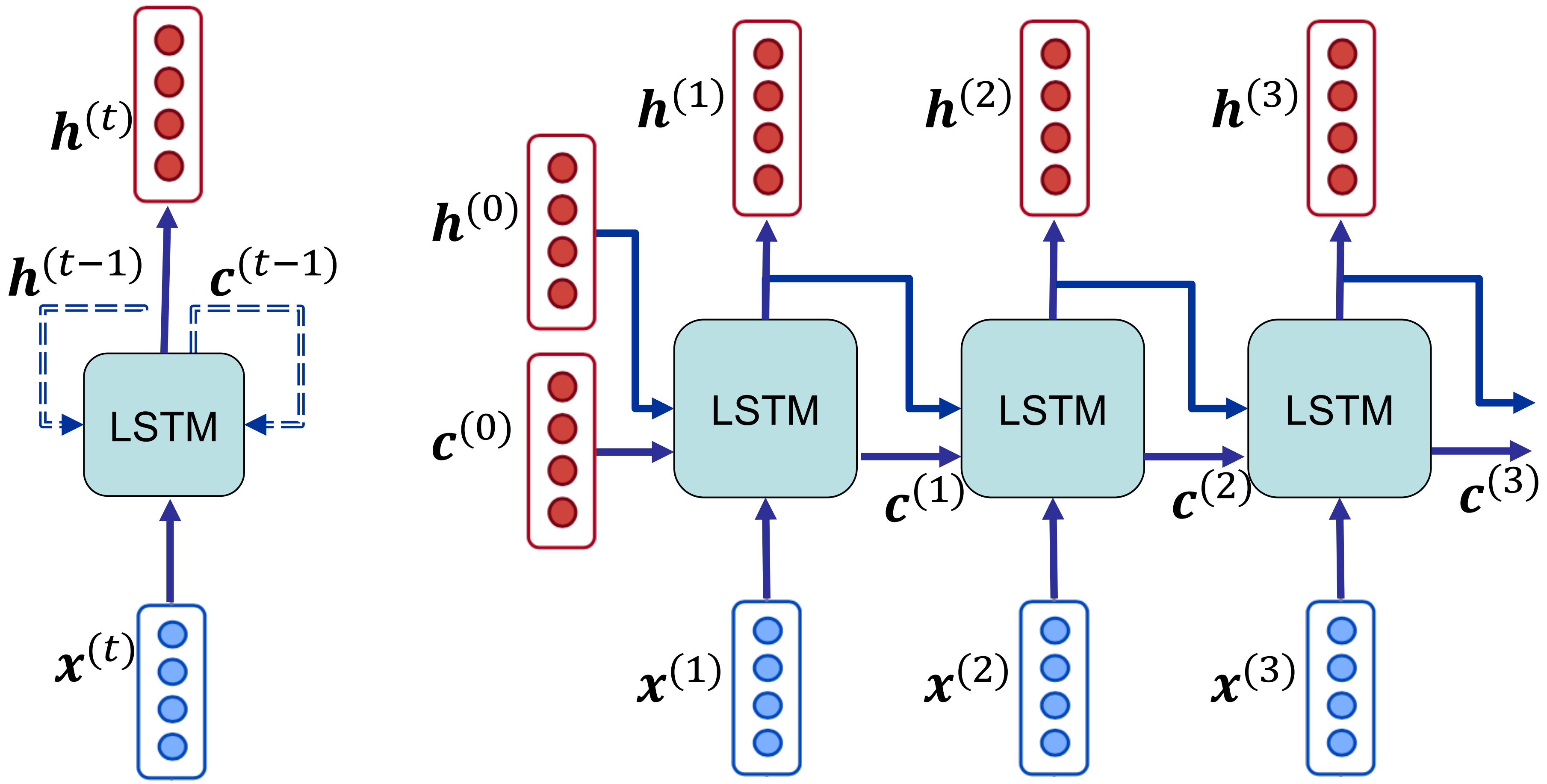
- Proposed by Hochreiter and Schmidhuber in 1997 as a solution to the vanishing gradients problem

- LSTM exploits a new vector cell state $\boldsymbol{c}^{(t)}$ to carry the memory of previous states
  - The cell state stores long-term information
  - As in vanilla RNN, hidden states $\boldsymbol{h}^{(t)}$ is used as output vector

- LSTM controls the process of reading, writing, and erasing information in/from memory states
  - These controls are done using gate vectors
  - Gates are dynamic and defined based on the input vector and hidden state

Hochreiter, Sepp, and Jürgen Schmidhuber. "**Long short-term memory**" *Neural computation* (1997) 17

LSTM – unrolled



LSTM definition – gates

- Gates are functions of input vector $\boldsymbol{x}^{(t)}$ and previous hidden state $\boldsymbol{h}^{(t-1)}$

$$\boldsymbol{i}^{(t)} = \text{function}(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)})$$
$$\boldsymbol{i}^{(t)} = \sigma(\boldsymbol{h}^{(t-1)}\boldsymbol{W}_{hi} + \boldsymbol{x}^{(t)}\boldsymbol{W}_{xi} + \boldsymbol{b}_i)$$

input gate: controls what parts of the new cell content are written to cell

$$\boldsymbol{f}^{(t)} = \text{function}(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)})$$
$$\boldsymbol{f}^{(t)} = \sigma(\boldsymbol{h}^{(t-1)}\boldsymbol{W}_{hf} + \boldsymbol{x}^{(t)}\boldsymbol{W}_{xf} + \boldsymbol{b}_f)$$

forget gate: controls what is kept vs forgotten, from previous cell state

$$\boldsymbol{o}^{(t)} = \text{function}(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)})$$
$$\boldsymbol{o}^{(t)} = \sigma(\boldsymbol{h}^{(t-1)}\boldsymbol{W}_{ho} + \boldsymbol{x}^{(t)}\boldsymbol{W}_{xo} + \boldsymbol{b}_o)$$

output gate: controls what parts of cell are output to hidden state

Model parameters (weights) are shown in red

LSTM definition – states

$$\tilde{\boldsymbol{c}}^{(t)} = \text{function}(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)})$$
$$\tilde{\boldsymbol{c}}^{(t)} = \tanh(\boldsymbol{h}^{(t-1)}\boldsymbol{W}_{hc} + \boldsymbol{x}^{(t)}\boldsymbol{W}_{xc} + \boldsymbol{b}_c)$$

new cell content: the new content to be used for cell and hidden (output) state

$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t)} + \boldsymbol{i}^{(t)} \odot \tilde{\boldsymbol{c}}^{(t)}$$

cell state: erases ("forgets") some content from last cell state, and writes ("inputs") some new cell content

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \odot \tanh(\boldsymbol{c}^{(t)})$$

hidden state: reads ("outputs") some content from the current cell state

Model parameters (weights) are shown in red

LSTM definition – all together

$$\boldsymbol{i}^{(t)} = \sigma(\boldsymbol{h}^{(t-1)}\boldsymbol{W}_{hi} + \boldsymbol{x}^{(t)}\boldsymbol{W}_{xi} + \boldsymbol{b}_i)$$

input gate: controls what parts of the new cell content are written to cell

$$\boldsymbol{f}^{(t)} = \sigma(\boldsymbol{h}^{(t-1)}\boldsymbol{W}_{hf} + \boldsymbol{x}^{(t)}\boldsymbol{W}_{xf} + \boldsymbol{b}_f)$$

forget gate: controls what is kept vs forgotten, from previous cell state

$$\boldsymbol{o}^{(t)} = \sigma(\boldsymbol{h}^{(t-1)}\boldsymbol{W}_{ho} + \boldsymbol{x}^{(t)}\boldsymbol{W}_{xo} + \boldsymbol{b}_o)$$

output gate: controls what parts of cell are output to hidden state

$$\tilde{\boldsymbol{c}}^{(t)} = \tanh(\boldsymbol{h}^{(t-1)}\boldsymbol{W}_{hc} + \boldsymbol{x}^{(t)}\boldsymbol{W}_{xc} + \boldsymbol{b}_c)$$

new cell content: the new content to be used for cell and hidden (output) state

$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t)} + \boldsymbol{i}^{(t)} \odot \tilde{\boldsymbol{c}}^{(t)}$$

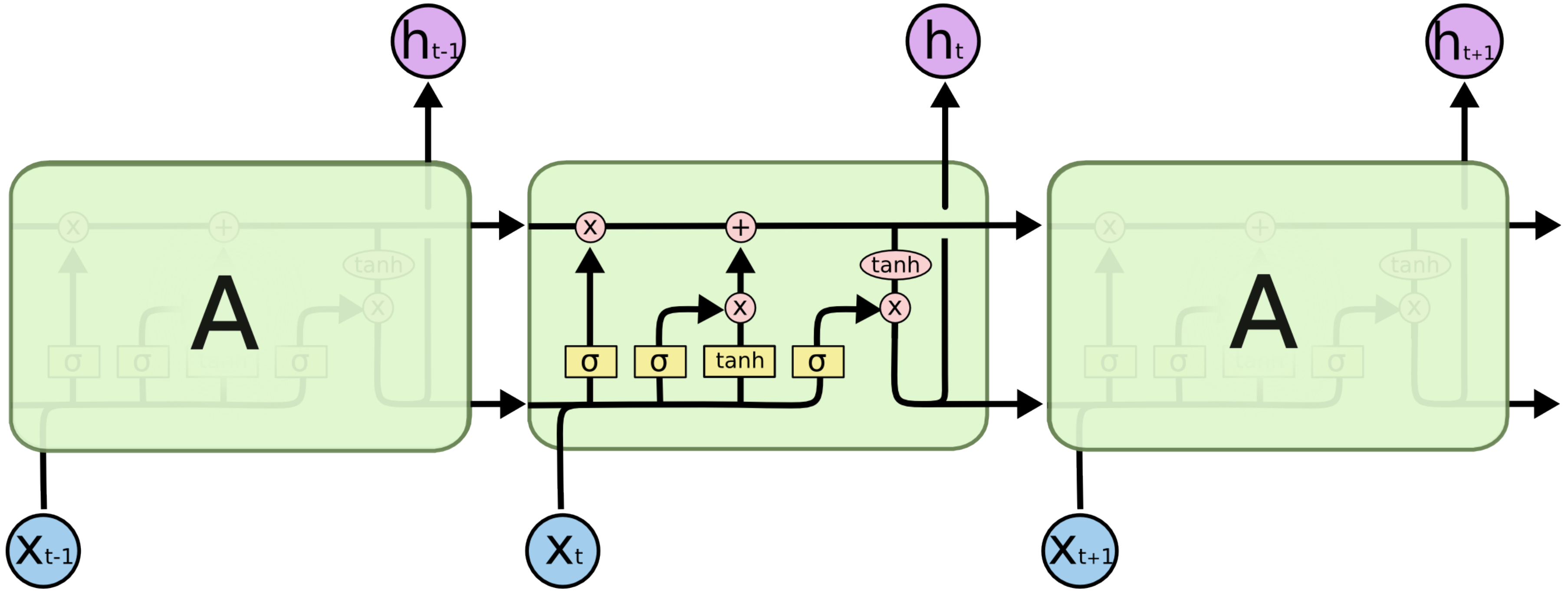
cell state: erases ("forgets") some content from last cell state, and writes ("inputs") some new cell content

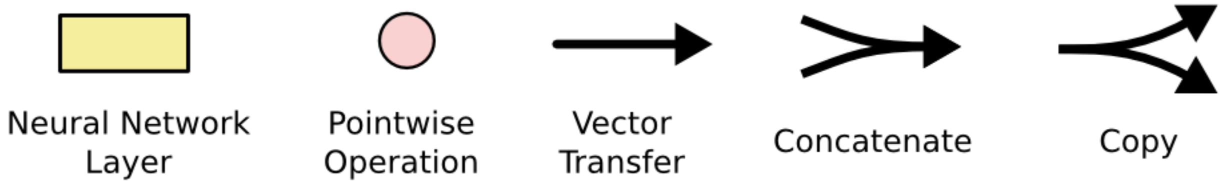
$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \odot \tanh(\boldsymbol{c}^{(t)})$$

hidden state: reads ("outputs") some content from the current cell state

Model parameters (weights) are shown in red

LSTM definition – visually!





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

Gated Recurrent Unit (GRU)

$$\boldsymbol{u}^{(t)} = \sigma(\boldsymbol{h}^{(t-1)}\boldsymbol{W}_{hu} + \boldsymbol{x}^{(t)}\boldsymbol{W}_{xu} + \boldsymbol{b}_u)$$

update gate: controls what parts of hidden state are updated vs preserved

$$\boldsymbol{r}^{(t)} = \sigma(\boldsymbol{h}^{(t-1)}\boldsymbol{W}_{hr} + \boldsymbol{x}^{(t)}\boldsymbol{W}_{xr} + \boldsymbol{b}_r)$$

reset gate: controls what parts of previous hidden state are used to compute new content

new hidden state content: (1) reset gate selects useful parts of previous hidden state. (2) Use this and current input to compute new hidden content.

$$\tilde{\boldsymbol{h}}^{(t)} = \tanh((\boldsymbol{r}^{(t)} \odot \boldsymbol{h}^{(t-1)})\,\boldsymbol{W}_{hh} + \boldsymbol{x}^{(t)}\boldsymbol{W}_{xh} + \boldsymbol{b}_h)$$

$$\boldsymbol{h}^{(t)} = \left(1 - \boldsymbol{u}^{(t)}\right) \odot \boldsymbol{h}^{(t-1)} + \boldsymbol{u}^{(t)} \odot \tilde{\boldsymbol{h}}^{(t)}$$

hidden state: update gate simultaneously controls what is kept from previous hidden state, and what is updated to new hidden state content

Model parameters (weights) are shown in red

RNNs with gates – counting parameters

- Parameters in LSTM (bias terms discarded)
  - $\boldsymbol{W}_{hi}, \boldsymbol{W}_{hf}, \boldsymbol{W}_{ho}, \boldsymbol{W}_{hc} \rightarrow h \times h \ * 4$
  - $\boldsymbol{W}_{xi}, \boldsymbol{W}_{xf}, \boldsymbol{W}_{xo}, \boldsymbol{W}_{xc} \rightarrow d \times h \ * 4$

- Parameters in GRU (bias terms discarded)
  - $\boldsymbol{W}_{hu}, \boldsymbol{W}_{hr}, \boldsymbol{W}_{hh} \rightarrow h \times h \ * 3$
  - $\boldsymbol{W}_{xu}, \boldsymbol{W}_{xr}, \boldsymbol{W}_{xh} \rightarrow d \times h \ * 3$

- If also considering encoder and decoder embeddings (e.g. in a Language Modeling network)
  - $\boldsymbol{E} \rightarrow |\mathbb{V}| \times d$
  - $\boldsymbol{U} \rightarrow h \times |\mathbb{V}|$

d and h are the number of dimensions of the input embedding and hidden vectors, respectively.

RNNs with gates – summary

- LSTM (and GRU) with dynamic gate mechanisms makes it easier to preserve necessary information over many timesteps

- LSTM does not *guarantee* that there is no vanishing/exploding gradient, but its large success in practice has shown that it can learn long-distance dependencies

- LSTM vs. GRU: LSTM is usually the default choice. Especially, when enough training data is available and capturing longer distances is important. GRU is faster and more suited for settings with low computation resources

Agenda

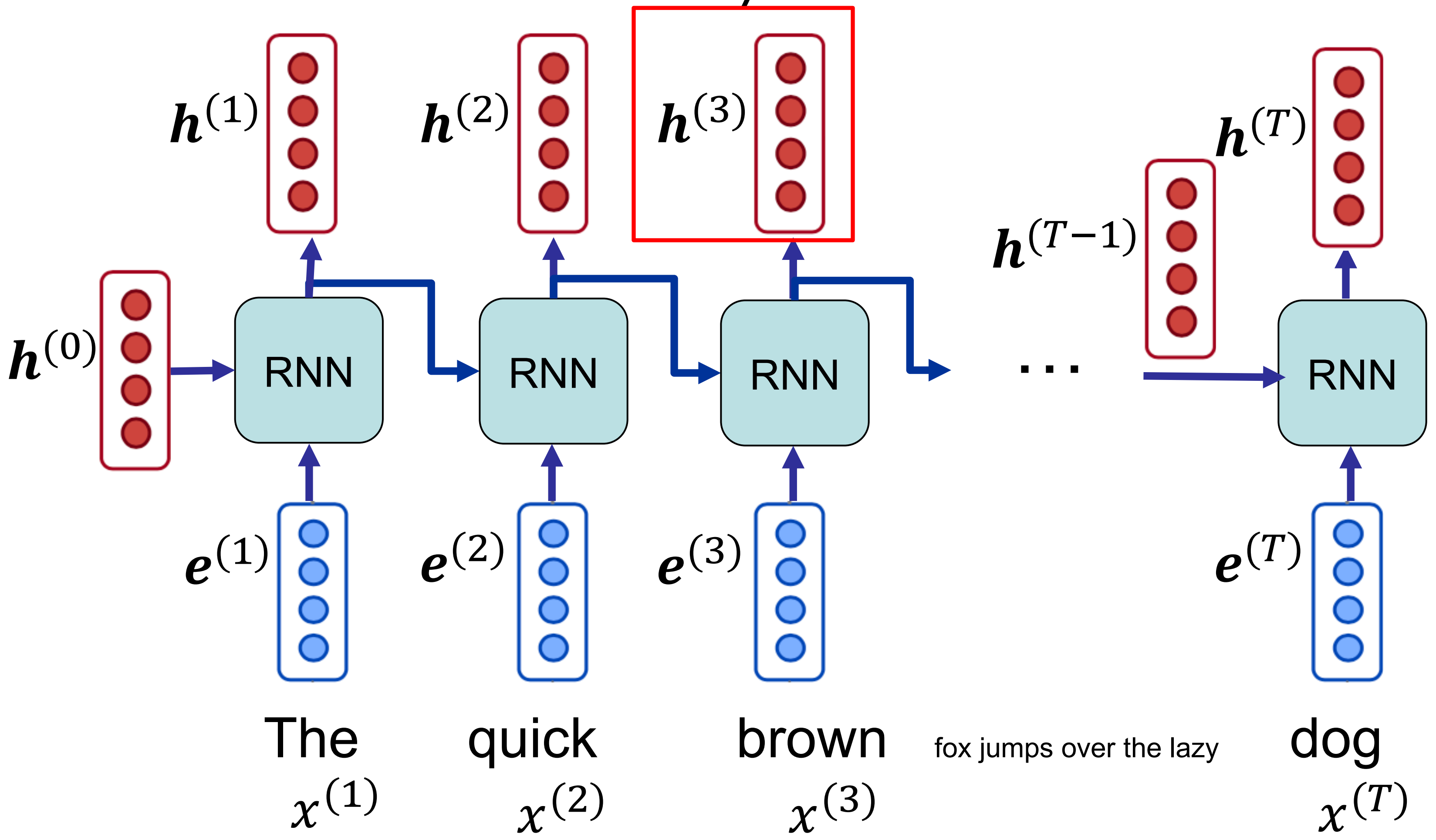
- Vanishing/Exploding gradient
- RNNs with Gates: LSTM, GRU
- **Contextualized word embeddings with RNNs**
- Extractive summarization with RNNs

Contextualized Word Embeddings

- Meaning of words can better be understood when we consider their contexts
  - Example of sense disambiguation when context is available:
    - what is *apple*? a fruit or the name of a company?
    - *"eating an apple"* vs. *"share of the apple company"*
- In contextualized word embedding, the representation of a word is defined based on the context, it appears in
- The input is a sequence of word embeddings and the output is a sequence of contextualized word embeddings

Contextualized Word Embeddings

The contextualized word embedding of "*brown*". However, it only has had access to the previous words (not the future ones)



Bidirectional RNNs

- Bidirectional RNN consists of two RNNs, one reads from the beginning to the end of sequence (forward), and the other reads from the end to the beginning (backward)

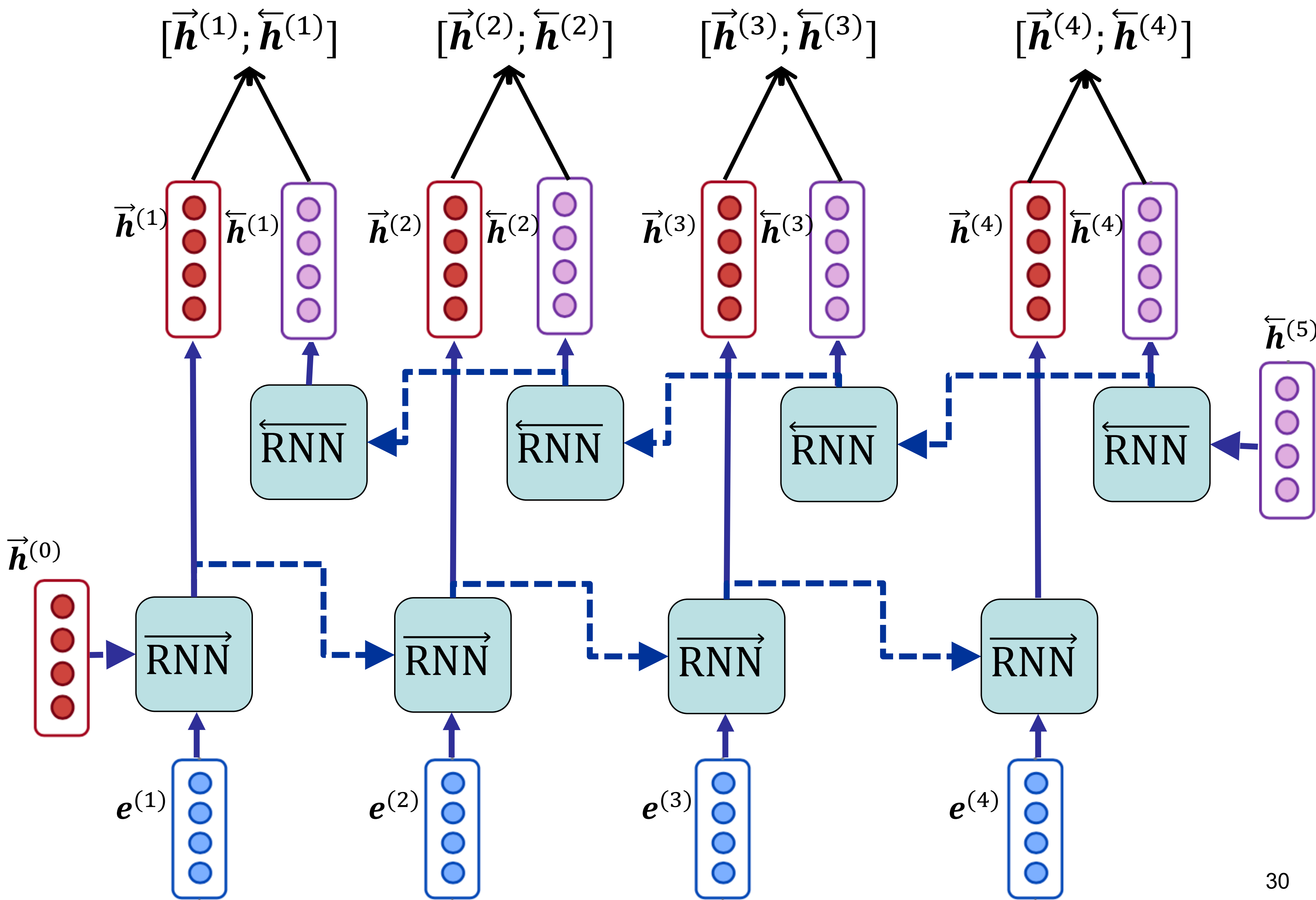
$$\overrightarrow{\boldsymbol{h}}^{(t)} = \overrightarrow{\mathrm{RNN}}\,(\overrightarrow{\boldsymbol{h}}^{(t-1)}, \boldsymbol{x}^{(t)})$$

$$\overleftarrow{\boldsymbol{h}}^{(t)} = \overleftarrow{\mathrm{RNN}}\,(\overleftarrow{\boldsymbol{h}}^{(t+1)}, \boldsymbol{x}^{(t)})$$

- Output at each time step is the concatenation of the outputs of both RNNs at that time step:

$$\boldsymbol{h}^{(t)} = [\overrightarrow{\boldsymbol{h}}^{(t)}; \overleftarrow{\boldsymbol{h}}^{(t)}]$$

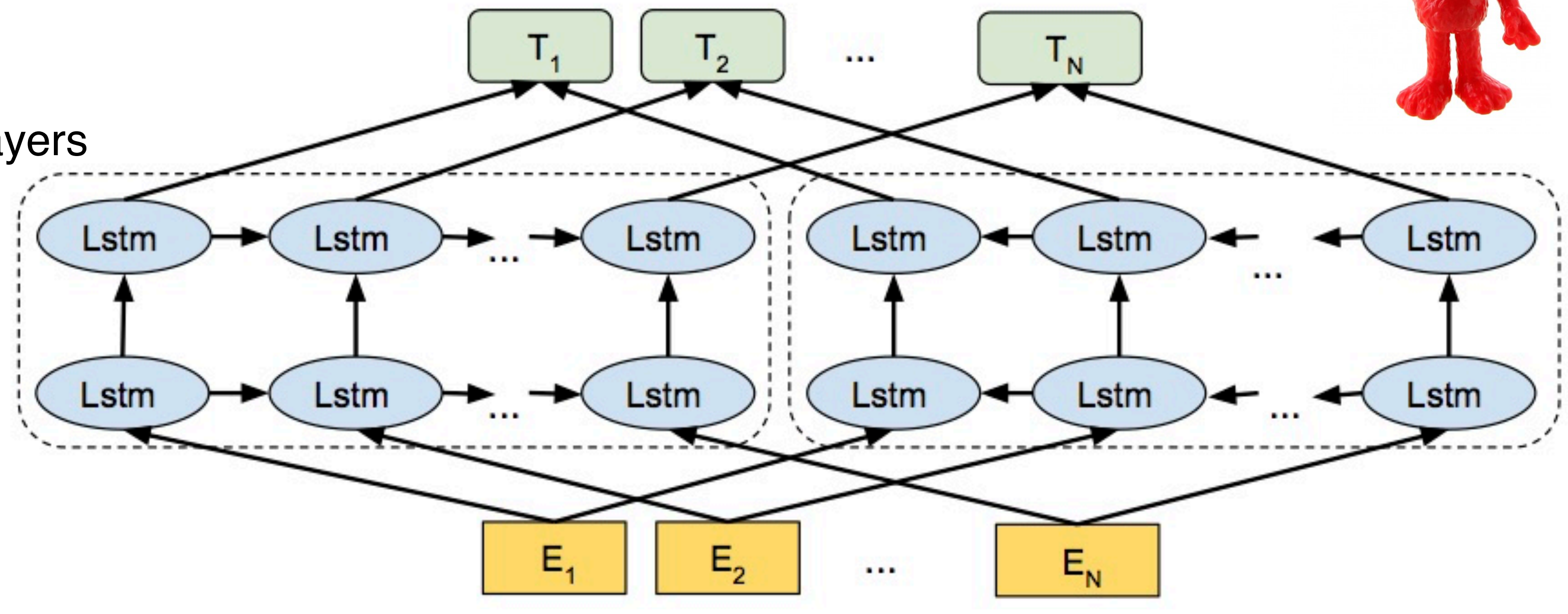
- *To remember:* Using bidirectional RNN is only possible when the entire sequence is available


$[\overrightarrow{h}^{(1)}; \overleftarrow{h}^{(1)}]$
$[\overrightarrow{h}^{(2)}; \overleftarrow{h}^{(2)}]$
$[\overrightarrow{h}^{(3)}; \overleftarrow{h}^{(3)}]$
$[\overrightarrow{h}^{(4)}; \overleftarrow{h}^{(4)}]$
$\overrightarrow{h}^{(1)}$
$\overleftarrow{h}^{(1)}$
$\overrightarrow{h}^{(2)}$
$\overleftarrow{h}^{(2)}$
$\overrightarrow{h}^{(3)}$
$\overleftarrow{h}^{(3)}$
$\overrightarrow{h}^{(4)}$
$\overleftarrow{h}^{(4)}$
$\overleftarrow{h}^{(5)}$
$\overleftarrow{\text{RNN}}$
$\overleftarrow{\text{RNN}}$
$\overleftarrow{\text{RNN}}$
$\overleftarrow{\text{RNN}}$
$\overrightarrow{h}^{(0)}$
$\overrightarrow{\text{RNN}}$
$\overrightarrow{\text{RNN}}$
$\overrightarrow{\text{RNN}}$
$\overrightarrow{\text{RNN}}$
$e^{(1)}$
$e^{(2)}$
$e^{(3)}$
$e^{(4)}$

ELMo (Embeddings from Language Models)

ELMo is a Multi-layer Bidirectional LSTM, trained on a Language Modeling objective

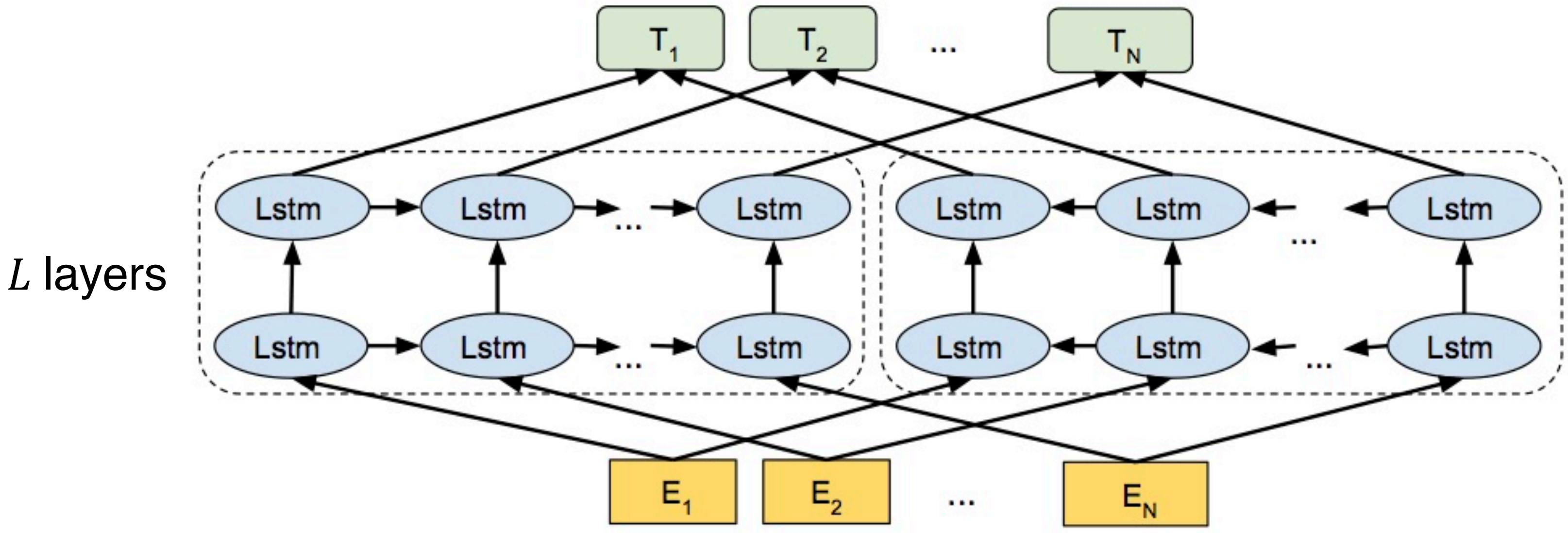
L layers



Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. **Deep Contextualized Word Representations**. In *Proc. of NAACL-HTL 2018*

ELMo – contextualized word embedding

- Given the set of input embeddings $\boldsymbol{e}_1, \boldsymbol{e}_2, \dots, \boldsymbol{e}_N$, in each layer j, ELMo outputs a set of contextualized word embeddings $\boldsymbol{h}_1^j, \boldsymbol{h}_2^j, \dots, \boldsymbol{h}_N^j$
- Contextualized embeddings in higher-levels capture semantic aspects (e.g., word senses), while embeddings in lower-levels model aspects of syntax (e.g., part-of-speech tagging)



ELMo in supervised tasks

- In supervised tasks, ELMo makes use of the embeddings in all layers (not only the last layer)!

- The final word embeddings are the weighted sum of the intermediary hidden states. Embedding of the word at position i:

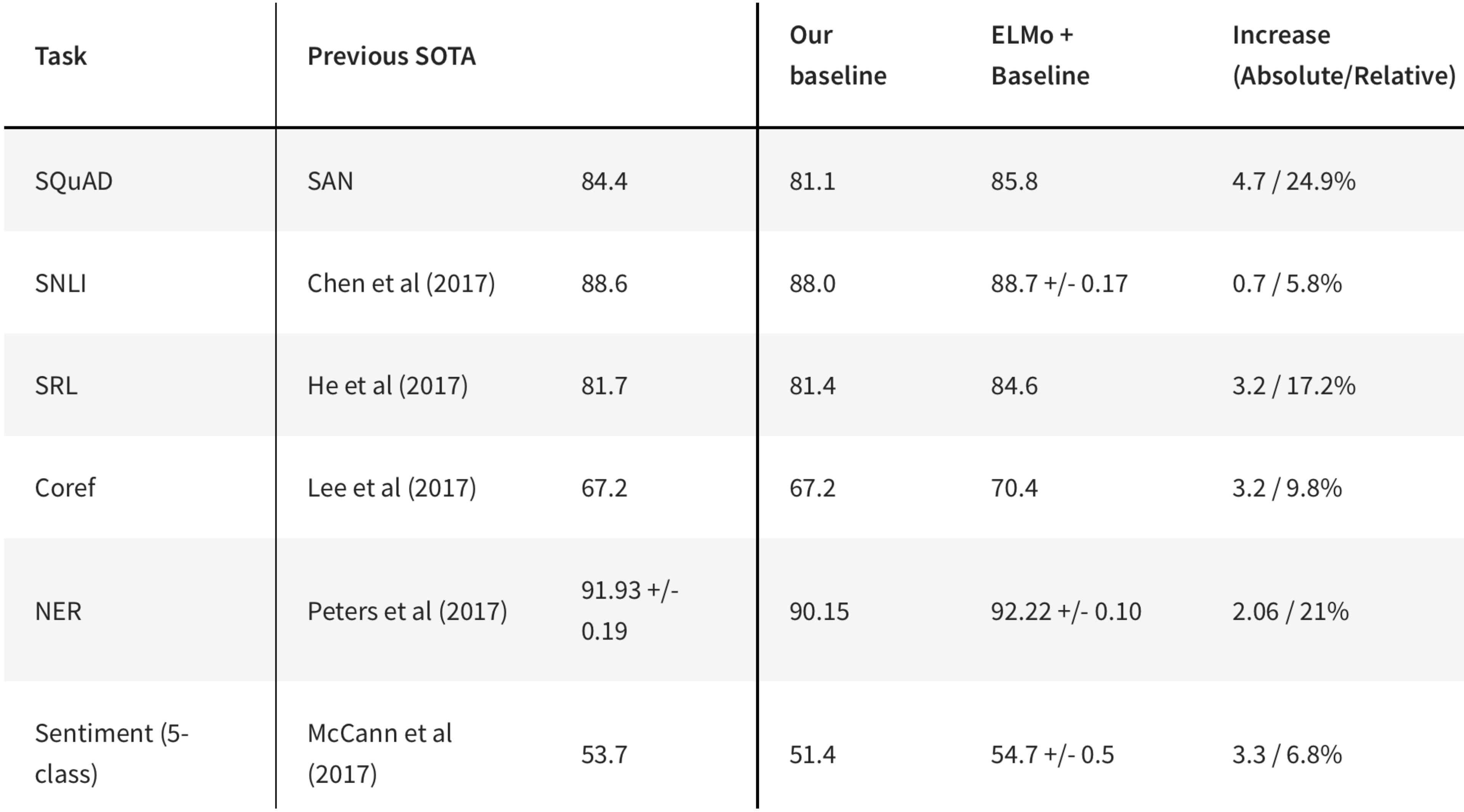
$$\gamma \sum_{j=1}^{L} \theta_j \boldsymbol{h}_i^j$$

  θ_j defines the weight (importance) of each layer

- θ_j values and γ are also model parameters, and are trained end-to-end with whole the model

Model parameters are shown in red

ELMo – key results



Task	Previous SOTA		Our baseline	ELMo + Baseline	Increase (Absolute/Relative)
SQuAD	SAN	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al (2017)	88.6	88.0	88.7 +/- 0.17	0.7 / 5.8%
SRL	He et al (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al (2017)	91.93 +/- 0.19	90.15	92.22 +/- 0.10	2.06 / 21%
Sentiment (5-class)	McCann et al (2017)	53.7	51.4	54.7 +/- 0.5	3.3 / 6.8%

https://allennlp.org/elmo

Agenda

- Vanishing/Exploding gradient
- RNNs with Gates: LSTM, GRU
- Contextualized word embeddings with RNNs
- **Extractive summarization with RNNs**

Text Summarization

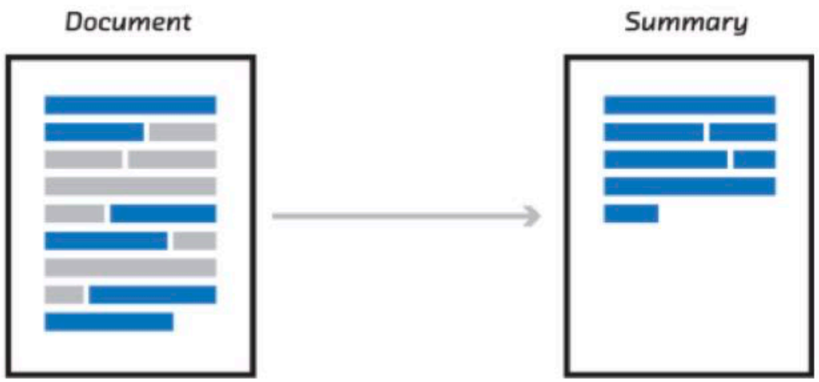
- The task of summarizing the key information content of a text (document) $X = \{x_1, x_2, \dots, x_N\}$ in summary $Y = \{y_1, y_2, \dots, y_M\}$
  - Summary is concise and (much) shorter than document

- Some datasets:
  - *Gigaword*: first one or two sentences of a news article
  - *CNN/DailyMail*: news article
  - *Wikihow*: full how-to article

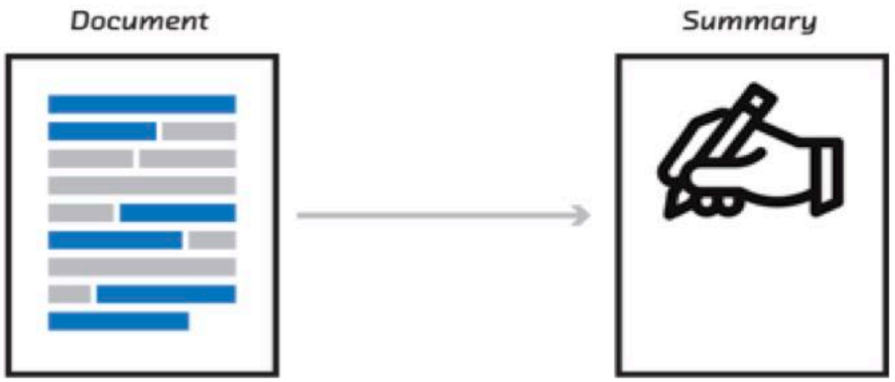
Summarization

- Extractive Summarization
  - Selecting sections (typically sentences) of the document
  - A model decides if a section of document should be selected for the summary
- Abstractive Summarization
  - Writing (generating) new summary text for the document
  - A language generation task conditioned on the document

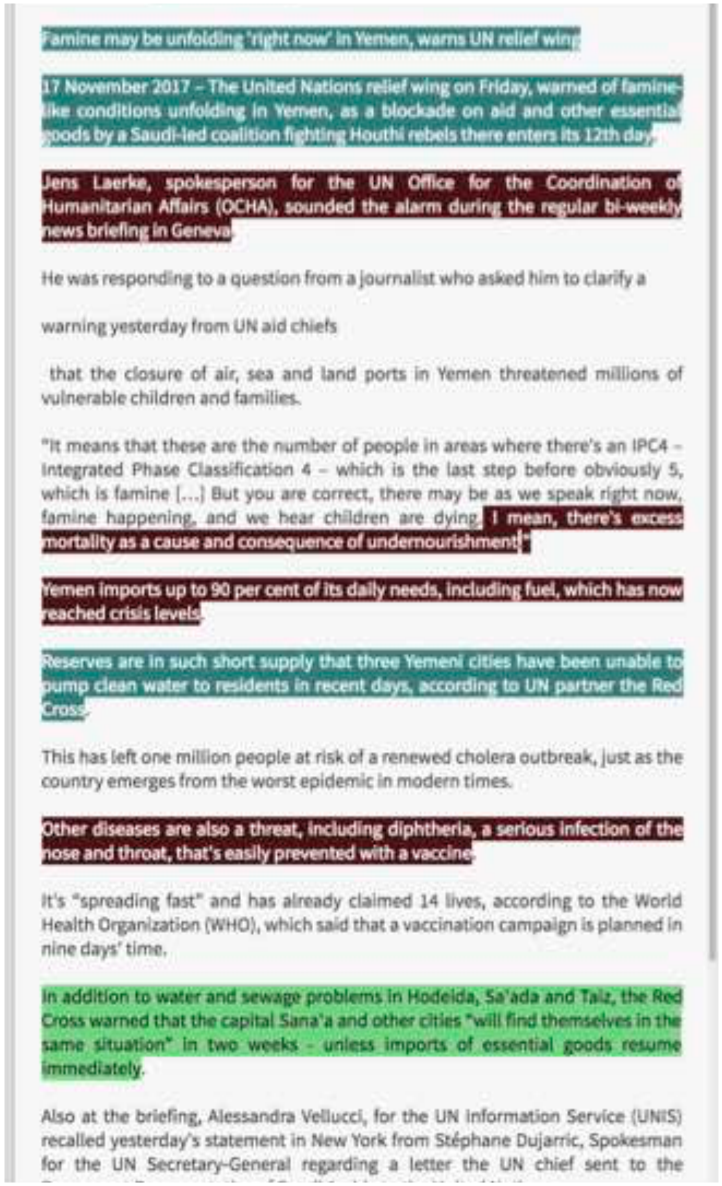
Extractive Summarization



Abstractive Summarization



Extractive Summarization



Famine may be unfolding 'right now' in Yemen, warns UN relief wing

17 November 2017 – The United Nations relief wing on Friday, warned of famine-like conditions unfolding in Yemen, as a blockade on aid and other essential goods by a Saudi-led coalition fighting Houthi rebels there enters its 12th day.

Jens Laerke, spokesperson for the UN Office for the Coordination of Humanitarian Affairs (OCHA), sounded the alarm during the regular bi-weekly news briefing in Geneva.

He was responding to a question from a journalist who asked him to clarify a

warning yesterday from UN aid chiefs

that the closure of air, sea and land ports in Yemen threatened millions of vulnerable children and families.

"It means that these are the number of people in areas where there's an IPC4 – Integrated Phase Classification 4 – which is the last step before obviously 5, which is famine [...] But you are correct, there may be as we speak right now, famine happening, and we hear children are dying. I mean, there's excess mortality as a cause and consequence of undernourishment."

Yemen imports up to 90 per cent of its daily needs, including fuel, which has now reached crisis levels.

Reserves are in such short supply that three Yemeni cities have been unable to pump clean water to residents in recent days, according to UN partner the Red Cross.

This has left one million people at risk of a renewed cholera outbreak, just as the country emerges from the worst epidemic in modern times.

Other diseases are also a threat, including diphtheria, a serious infection of the nose and throat, that's easily prevented with a vaccine.

It's "spreading fast" and has already claimed 14 lives, according to the World Health Organization (WHO), which said that a vaccination campaign is planned in nine days' time.

In addition to water and sewage problems in Hodeida, Sa'ada and Taiz, the Red Cross warned that the capital Sana'a and other cities "will find themselves in the same situation" in two weeks – unless imports of essential goods resume immediately.

Also at the briefing, Alessandra Vellucci, for the UN Information Service (UNIS) recalled yesterday's statement in New York from Stéphane Dujarric, Spokesman for the UN Secretary-General regarding a letter the UN chief sent to the

Abstractive Summarization

Document

SAN FRANCISCO, California (Reuters) -- Sony has cut the price of the PlayStation 3 by $100, or 17 percent, in the United States, a move that should boost the video game console's lackluster sales.
Starting Monday, the current PS3 60 gigabyte model will cost $499 -- a $100 price drop.
The PlayStation 3, which includes a 60-gigabyte hard drive and a Blu-ray high-definition DVD player, will now cost $500, or $20 more than the most expensive version of Microsoft's Xbox 360.
The PS3 still costs twice that of Nintendo's Wii console, whose $250 price and motion-sensing controller have made it a best-seller despite its lack of cutting-edge graphics and hard disk.
"Our initial expectation is that sales should double at a minimum," Jack Tretton, chief executive of Sony Computer Entertainment America, said in an interview.
"We've gotten our production issues behind us on the PlayStation 3, reaching a position to pass on the savings to consumers, and our attitude is the sooner the better."
…

Summary

- Sony drops price of current 60GB PlayStation 3 console by $100 in U.S.
- PS3 still costs twice that of Nintendo's best-selling Wii console, which is $250
- Some expect Microsoft to respond with its first price cuts on the Xbox 360
- Sony to revise PS3 console with bigger 80GB hard drive

Summarization – Evaluation

- **ROUGE-N**: overlap of *n*-grams between output and reference summary
  - **ROUGE-1**: the overlap of ***unigrams***
  - **ROUGE-2**: the overlap of ***bigrams***
  - …

$$\text{ROUGE-N} = \frac{\left| n\text{-grams}\,(\hat{Y}) \cap n\text{-grams}(Y) \right|}{\left| n\text{-grams}(Y) \right|}$$

Y and $\hat{Y}$ are the reference and output summary, respectively. $n\text{-grams}$ retrurns the set of all possible *n*-grams of the given text.

Summarization – Evaluation

- **ROUGE-L** is based on the length of the longest common subsequence between the output and reference summary

$$\text{ROUGE−L} = \frac{LCS\ (\hat{Y}, Y)}{|\hat{Y}|}$$

LCS is the longest common subsequence of the two given texts.

- ROUGE-L
  - does not require consecutive matches but in-sequence matches
  - reflects sentence structure
  - don’t need a predefined *n*-gram length

Summarization – Evaluation

- ROUGE (in the discussed definitions) is a recall-based measure
    - ROUGE can also be defined as a precision-based, as well as F-measure

Example

- Y: "police killed the gunman"
- $\hat{Y}1$: "police kill the gunman"
- $\hat{Y}2$: "the gunman kill police"

- $LCS\left(\hat{Y}1, Y\right)$ ="police the gunman"→ ROUGE−L($\hat{Y}1, Y$) = 0.75
- $LCS\left(\hat{Y}2, Y\right)$ = "the gunman" → ROUGE−L($\hat{Y}2, Y$) = 0.5
- However, ROUGE-2 of both $\hat{Y}1$ and $\hat{Y}2$ are the same
    - In both $\hat{Y}1$ and $\hat{Y}2$, "the gunman" is the only common bigram with the reference

Extractive Summarization – paper walkthrough

Problem definition

- Document D contains L sentences: $D = \{s_1, s_2, \dots, s_L\}$
- Each sentence s_i contains a set words, each annotated with $x^{(i)}$
- Extractive summarization objective: select l sentences of the document such that they provide the "*best*" summary (concise and comprehensive)
- Evaluation is done by comparing the output summary with the reference summary

Zhou, Q., Yang, N., Wei, F., Huang, S., Zhou, M., & Zhao, T.. **Neural Document Summarization by Jointly Learning to Score and Select Sentences**. In *Proceedings of ACL* 2018

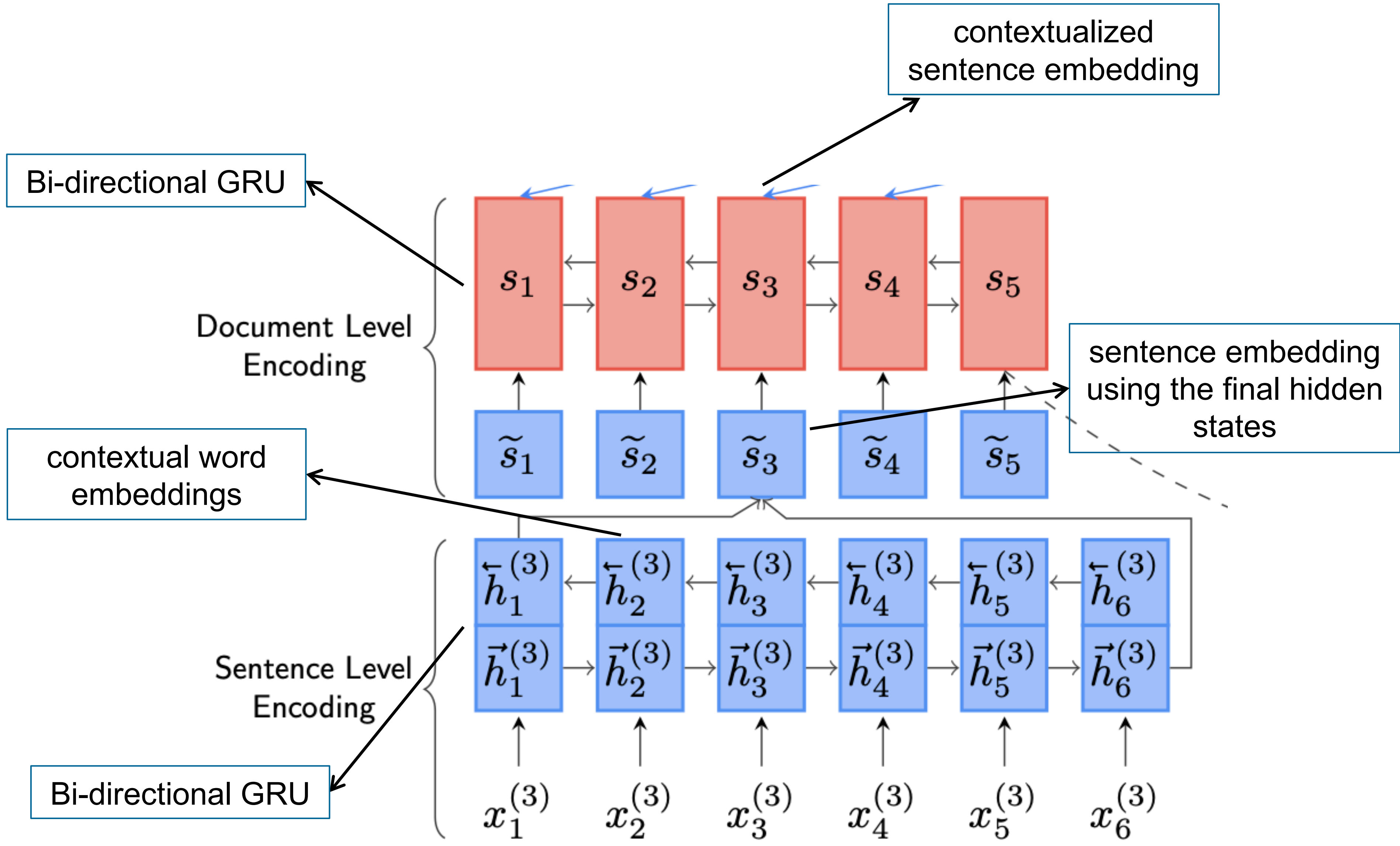
Core Ideas – NeuSum model

- At each time step, the model wants to decide which sentence to include in the summary (**sentence selection**)

- To do sentence selection, at each time step, the model assigns scores to sentences that are not included in summary (**sentence scoring**), and selects the one with the highest score

- The sentence scoring is based on the representation of each sentence, but also the content of the previously selected sentences
  - *Why on previously selected sentences?* Intuitively, if some contents are already included in the summary, the model should avoid selecting the sentences with similar contents

Sentence encoding


contextualized sentence embedding
Bi-directional GRU
Document Level Encoding
s_1 s_2 s_3 s_4 s_5
sentence embedding using the final hidden states
$\tilde{s}_1$ $\tilde{s}_2$ $\tilde{s}_3$ $\tilde{s}_4$ $\tilde{s}_5$
contextual word embeddings
$\overleftarrow{h}_1^{(3)}$ $\overleftarrow{h}_2^{(3)}$ $\overleftarrow{h}_3^{(3)}$ $\overleftarrow{h}_4^{(3)}$ $\overleftarrow{h}_5^{(3)}$ $\overleftarrow{h}_6^{(3)}$
$\overrightarrow{h}_1^{(3)}$ $\overrightarrow{h}_2^{(3)}$ $\overrightarrow{h}_3^{(3)}$ $\overrightarrow{h}_4^{(3)}$ $\overrightarrow{h}_5^{(3)}$ $\overrightarrow{h}_6^{(3)}$
Sentence Level Encoding
Bi-directional GRU
$x_1^{(3)}$ $x_2^{(3)}$ $x_3^{(3)}$ $x_4^{(3)}$ $x_5^{(3)}$ $x_6^{(3)}$

Sentence scoring and selection

- Sentence scoring learns a function δ that assigns a score to each sentence s_i at time step t. It uses:
  1. sentence embedding $\boldsymbol{s}_i$
  2. information of previously selected sentences, embedded in the vector $\boldsymbol{h}_t \rightarrow$ **current state of summary**

$$\delta(s_i) = \text{function}(\boldsymbol{h}_t, \boldsymbol{s}_i)$$

$$\delta(s_i) = \boldsymbol{w}_s \tanh(\boldsymbol{h}_t \boldsymbol{W}_q + \boldsymbol{s}_i \boldsymbol{W}_d + \boldsymbol{b}_i)$$

- $\delta(s_i)$ is calculated for the sentences that are not yet included in the summary
- The sentence with highest δ is added to summary

Sentence scoring

- How is current state of summary $\boldsymbol{h}_t$ calculated?
  - Using another GRU

- A GRU model outputs the new state of the summary using the previous state of summary and the last selected sentence

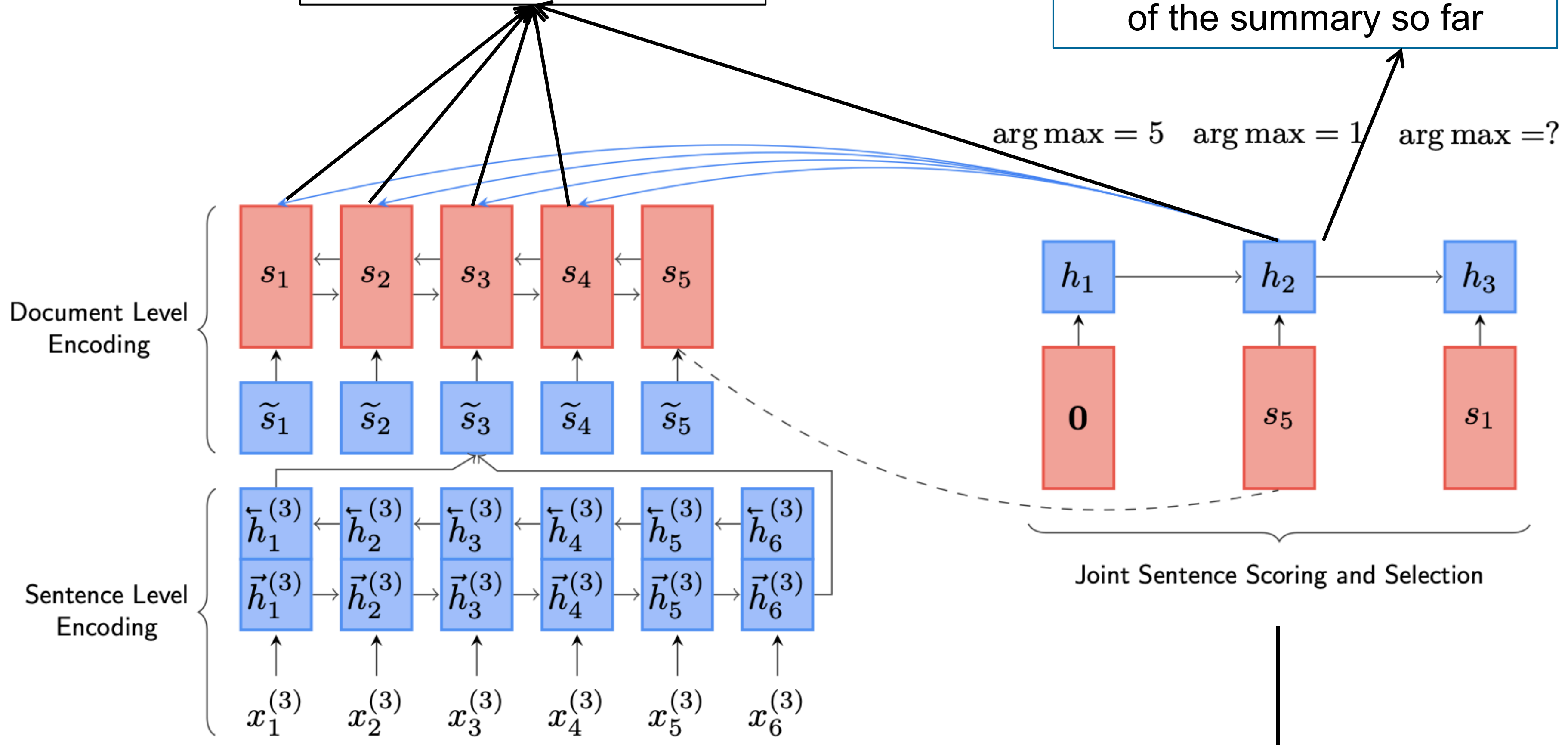
$$\boldsymbol{h}_t = \mathrm{GRU}(\boldsymbol{h}_{t-1}, \boldsymbol{s}_{t-1})$$

You can find such more details by looking into the paper:

- $\boldsymbol{h}_0$ is created based on the document embedding
- The model is optimized using the Kullback-Leibler divergence between the distribution of output scores and the distribution of ROUGE-2 F1 gains of sentences

$$\delta(s_i) = \text{function}(\boldsymbol{h}_t, \boldsymbol{s}_i)$$

$\boldsymbol{h}_t$ can be seen as an *aggregation* of the contents of the summary so far

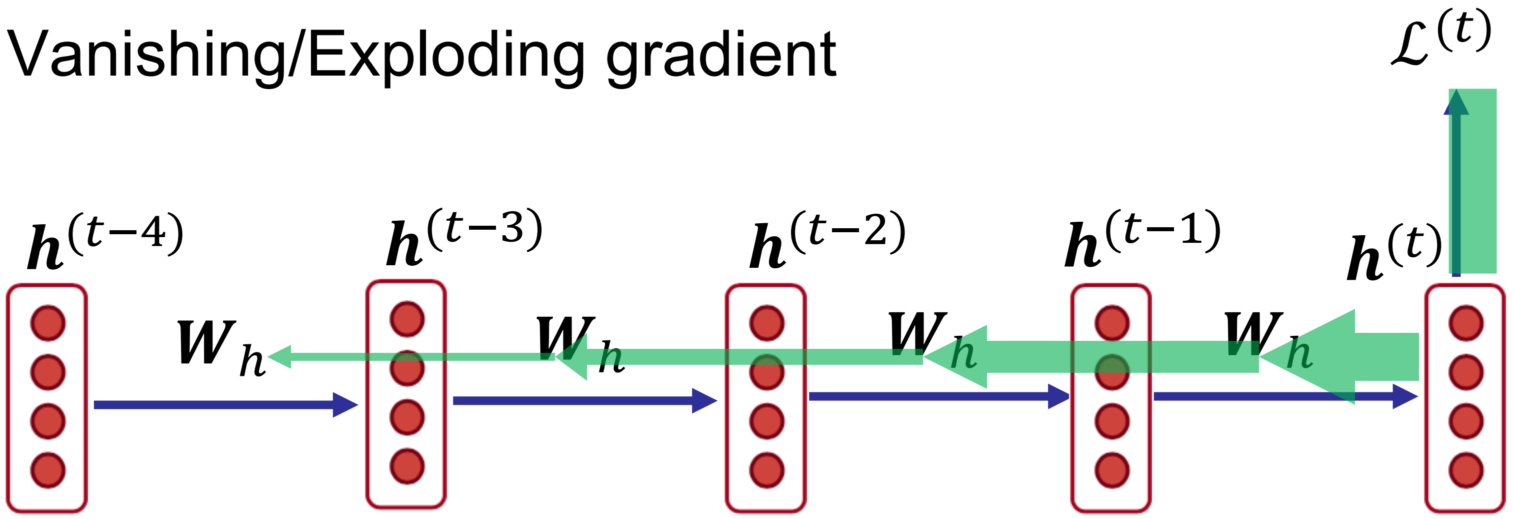


$\arg\max = 5$
$\arg\max = 1$
$\arg\max = ?$
Document Level Encoding
s_1 s_2 s_3 s_4 s_5
$\tilde{s}_1$ $\tilde{s}_2$ $\tilde{s}_3$ $\tilde{s}_4$ $\tilde{s}_5$
Sentence Level Encoding
$\overleftarrow{h}_1^{(3)}$ $\overleftarrow{h}_2^{(3)}$ $\overleftarrow{h}_3^{(3)}$ $\overleftarrow{h}_4^{(3)}$ $\overleftarrow{h}_5^{(3)}$ $\overleftarrow{h}_6^{(3)}$
$\vec{h}_1^{(3)}$ $\vec{h}_2^{(3)}$ $\vec{h}_3^{(3)}$ $\vec{h}_4^{(3)}$ $\vec{h}_5^{(3)}$ $\vec{h}_6^{(3)}$
$x_1^{(3)}$ $x_2^{(3)}$ $x_3^{(3)}$ $x_4^{(3)}$ $x_5^{(3)}$ $x_6^{(3)}$
h_1 h_2 h_3
$\mathbf{0}$ s_5 s_1
Joint Sentence Scoring and Selection

An unrolled GRU that outputs the next *state of summary*

Recap

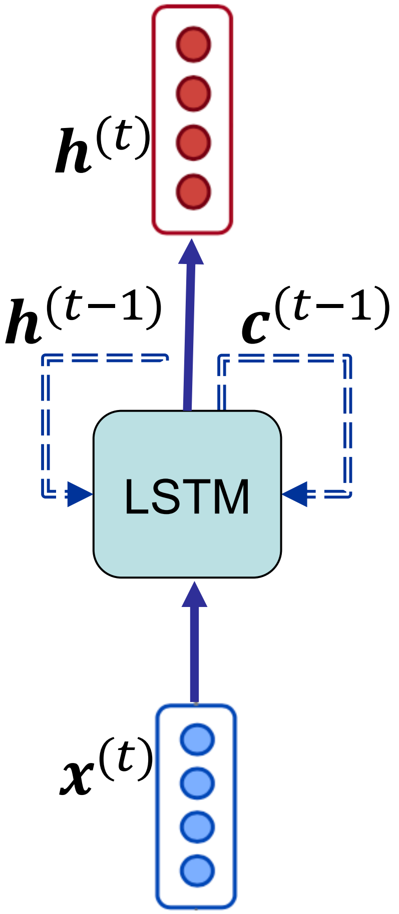
- Vanishing/Exploding gradient



- RNNs with gates
  - address the problem of vanishing/exploding gradient by controlling what to be stored in and forgotten from the memory states

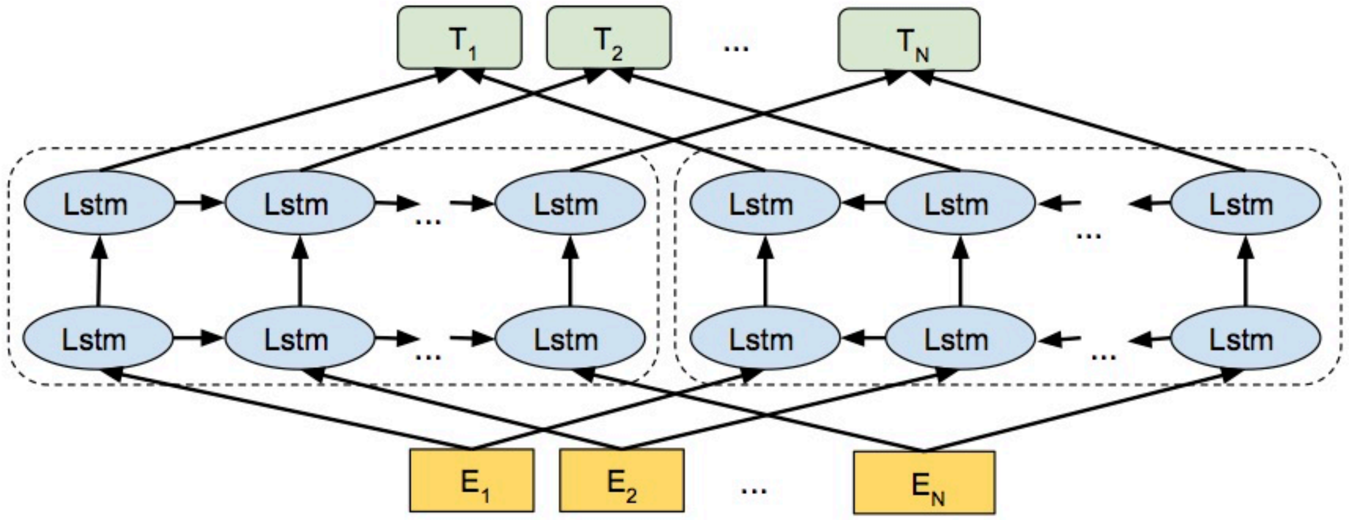
Recap

LSTM



ELMo

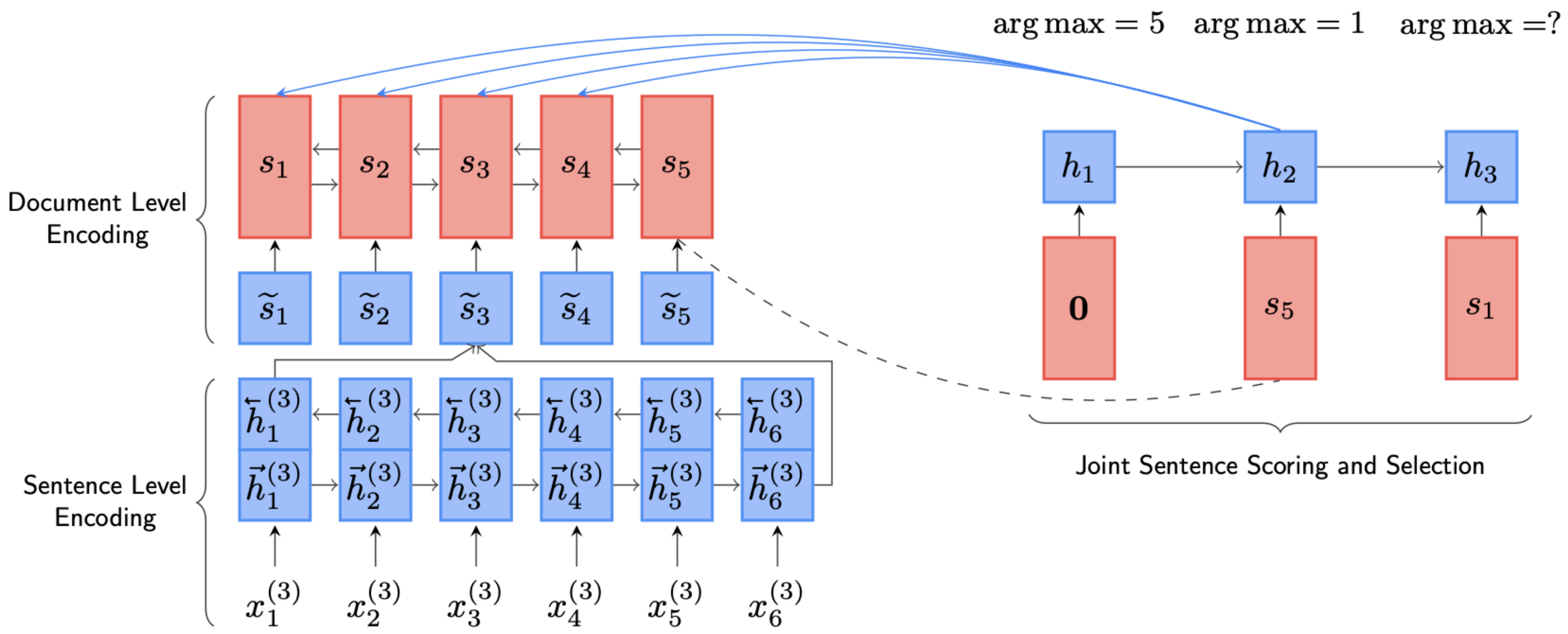
A contextualized word embedding model using multilayer bidirectional LSTM



Recap

Extractive summarization

NeuSum model



$\arg\max = 5$
$\arg\max = 1$
$\arg\max = ?$
Document Level Encoding
s_1 s_2 s_3 s_4 s_5
$\widetilde{s}_1$ $\widetilde{s}_2$ $\widetilde{s}_3$ $\widetilde{s}_4$ $\widetilde{s}_5$
Sentence Level Encoding
$\overleftarrow{h}_1^{(3)}$ $\overleftarrow{h}_2^{(3)}$ $\overleftarrow{h}_3^{(3)}$ $\overleftarrow{h}_4^{(3)}$ $\overleftarrow{h}_5^{(3)}$ $\overleftarrow{h}_6^{(3)}$
$\vec{h}_1^{(3)}$ $\vec{h}_2^{(3)}$ $\vec{h}_3^{(3)}$ $\vec{h}_4^{(3)}$ $\vec{h}_5^{(3)}$ $\vec{h}_6^{(3)}$
$x_1^{(3)}$ $x_2^{(3)}$ $x_3^{(3)}$ $x_4^{(3)}$ $x_5^{(3)}$ $x_6^{(3)}$
h_1 h_2 h_3
$\mathbf{0}$ s_5 s_1
Joint Sentence Scoring and Selection