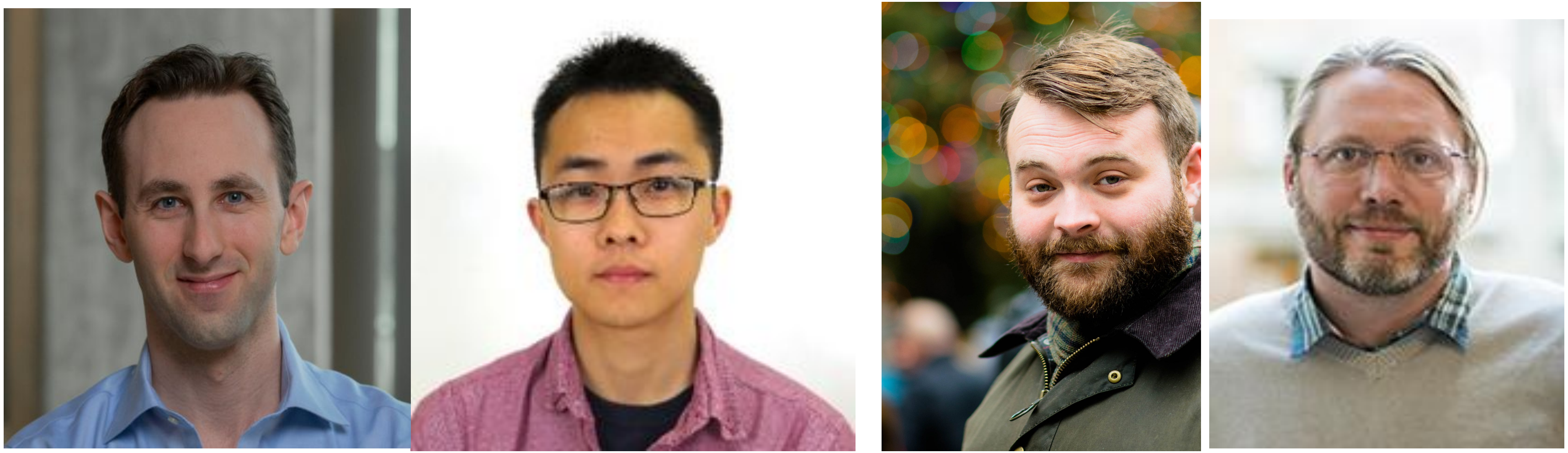
Long Short-Term Memory as a Dynamically Computed Element-wise Weighted Sum

Sara Khan Saumya Agarwal

Authors of the paper



(L-R) **Omer Levy, Kenton Lee, Nicholas FitzGerald, Luke Zettlemoyer**
Paul G. Allen School, University of Washington, Seattle, WA

ACL 2018, pages 732-739

Outline

- Abstract
- Quick Recap
- What do Memory Cells compute?
- Experiments
- Related Work
- Conclusion

ABSTRACT

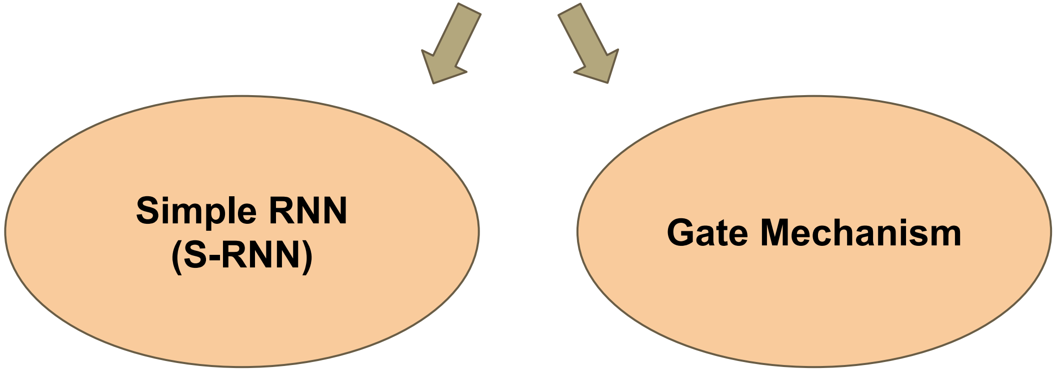
Aim: Why LSTMs are successful?

So far, we know,

- RNNs suffer from vanishing gradient (VG) problem
- To combat it, LSTMs are used
- Memory gates present in LSTM mitigate vanishing gradient problem

This study reveals

- Gates themselves are powerful recurrent models
- LSTMs can be viewed as combination of 2 components:



To prove it,

Ablations on different NLP applications

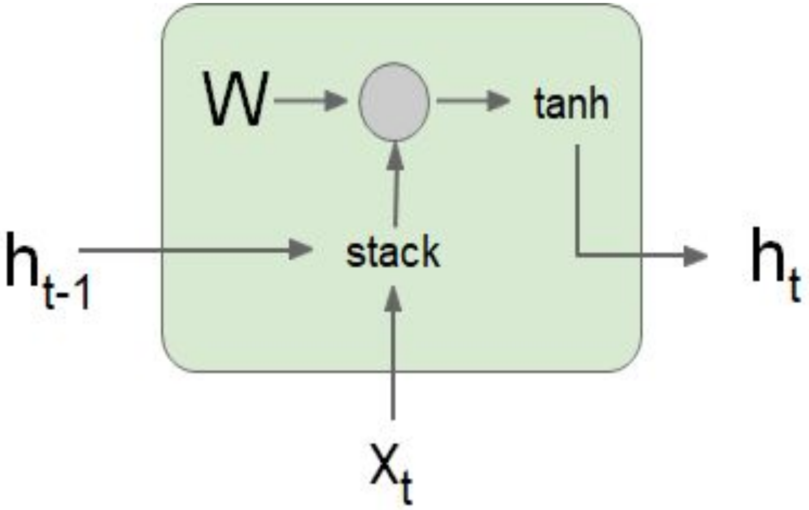
Language Modelling	Question Answering	Dependency Parsing	Machine Translation

Outcome:

In most settings, gating mechanism alone performs fairly equal to LSTM

Quick Recap (RNN & LSTMs)

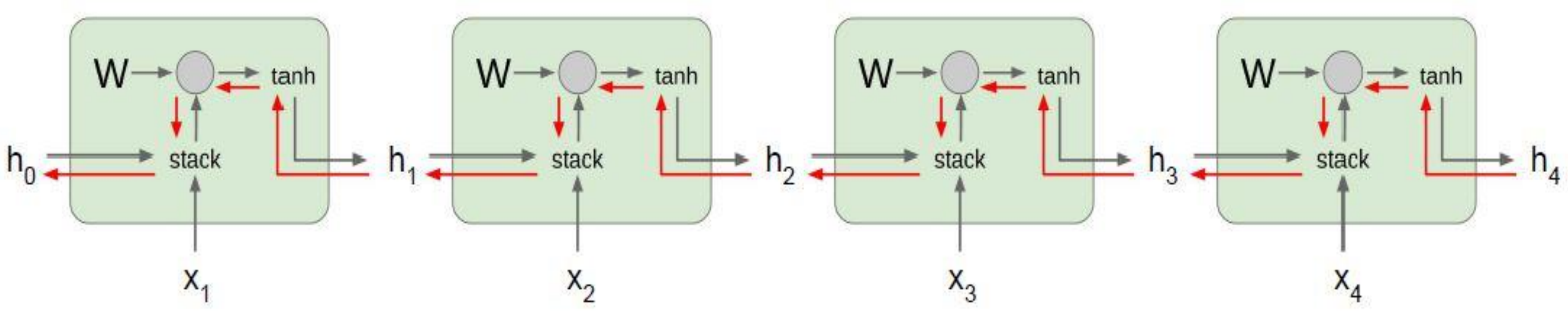
Basic structure of Simple RNN



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

Image Source: Google Images

But RNN suffers from vanishing gradient problem..



Alternative? LSTMs

Long Short Term Memory(LSTM)

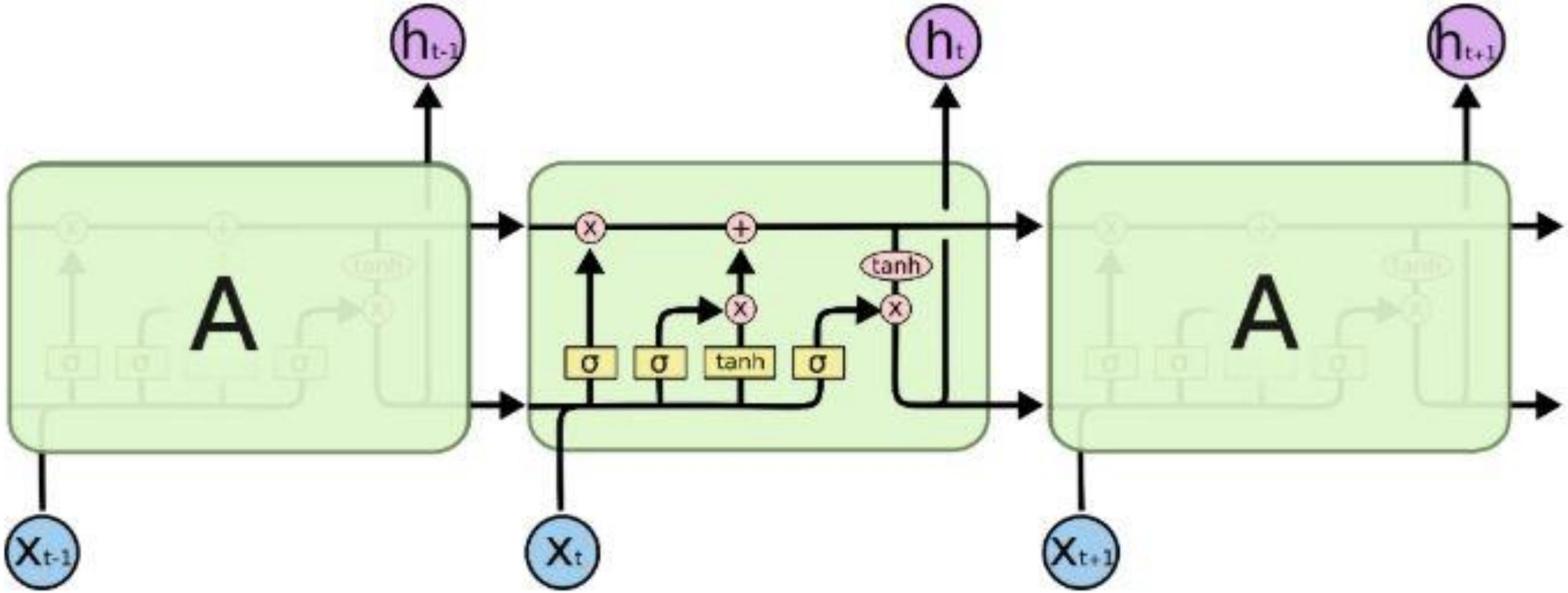
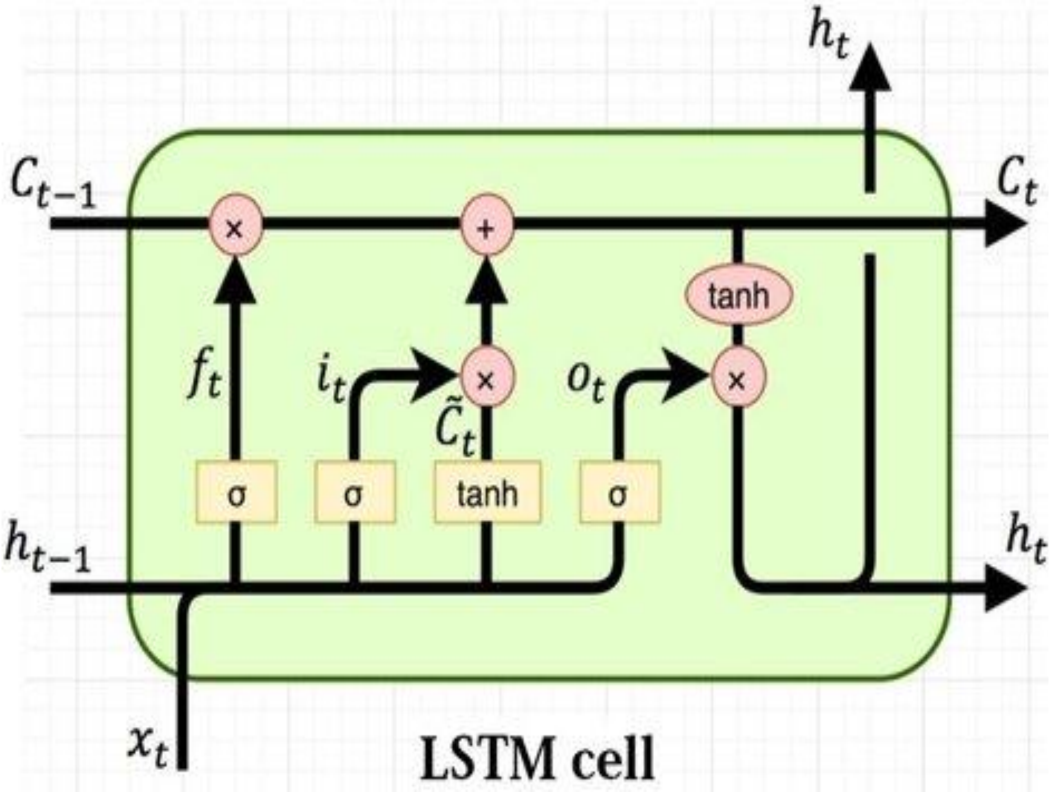


Image Source: Google Images



$$\tilde{C}_t = tanh(W_{ch}h_{t-1} + W_{cx}x_t)$$

$$i_t = \sigma(W_{ih}h_{t-1} + W_{ix}x_t)$$

$$f_t = \sigma(W_{fh}h_{t-1} + W_{fx}x_t)$$

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

$$o_t = \sigma(W_{oh}h_{t-1} + W_{ox}x_t)$$

$$h_t = o_t \circ tanh(C_t)$$

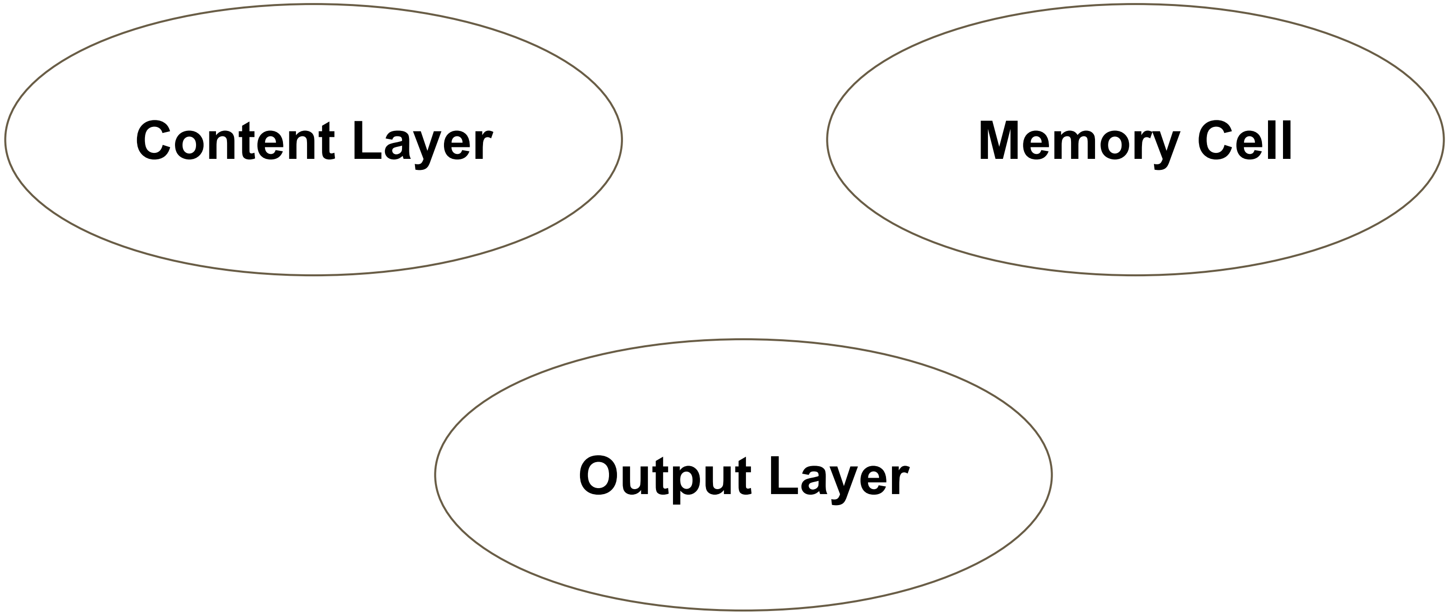
So far, we know LSTMs introduced to combat vanishing gradient problem.

Q) Can we explain why LSTMs are successful?

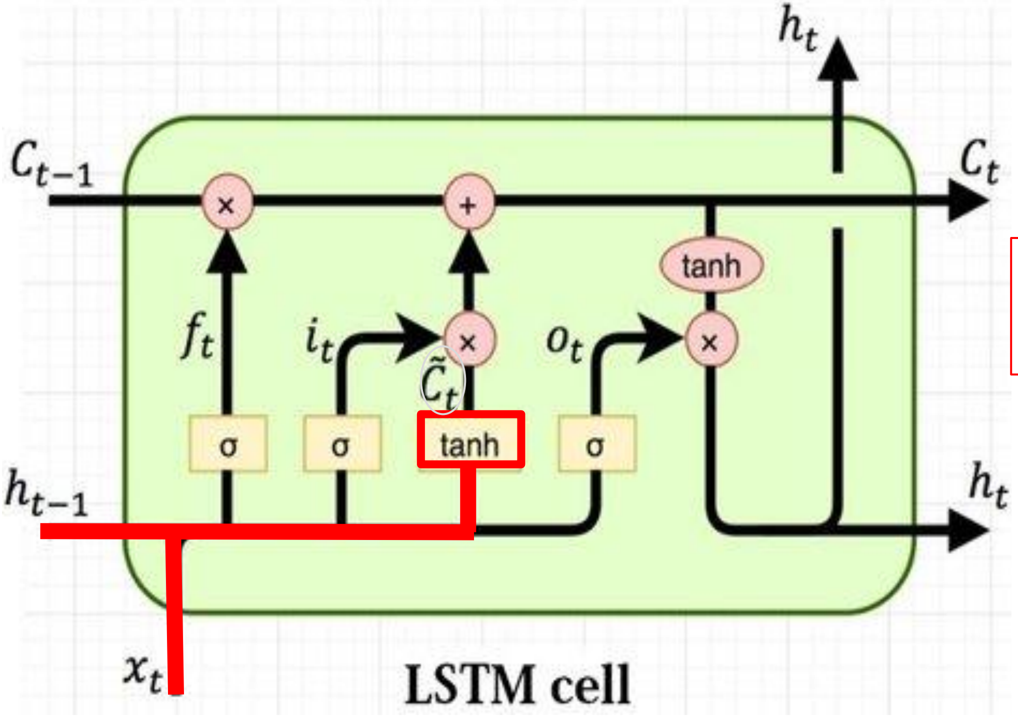
The answer is Yes

What do Memory Cells Compute?

LSTM as a hybrid of 3 sub-components:

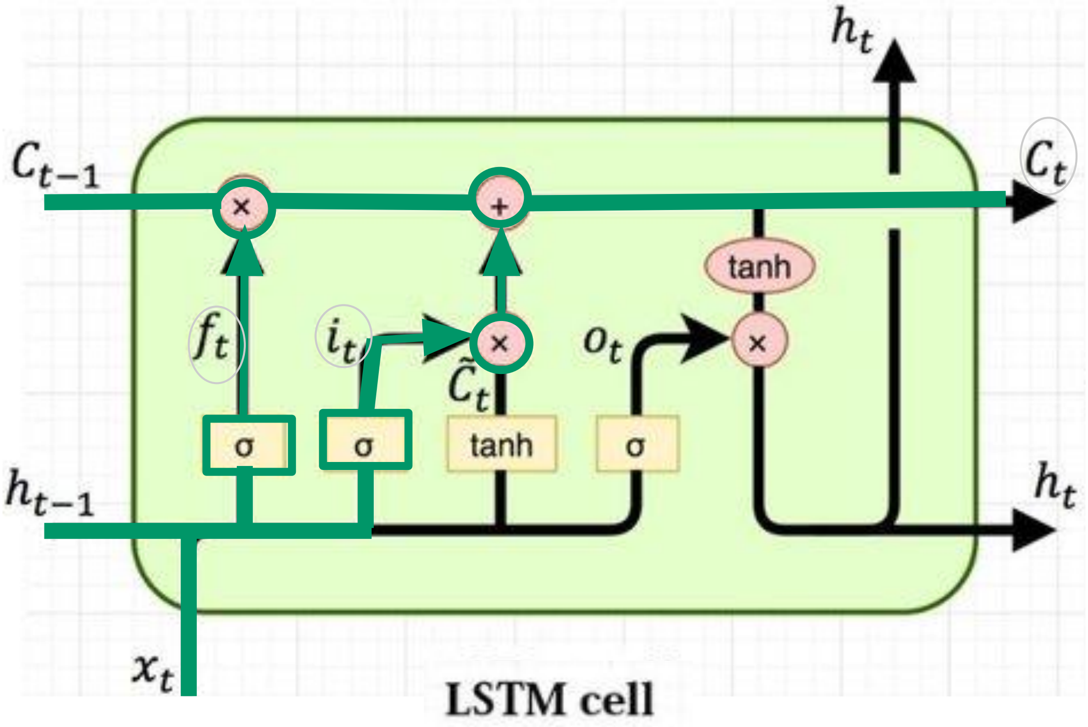


Content Layer



$$\tilde{C}_t = tanh(W_{ch}h_{t-1} + W_{cx}x_t)$$

Memory Cell

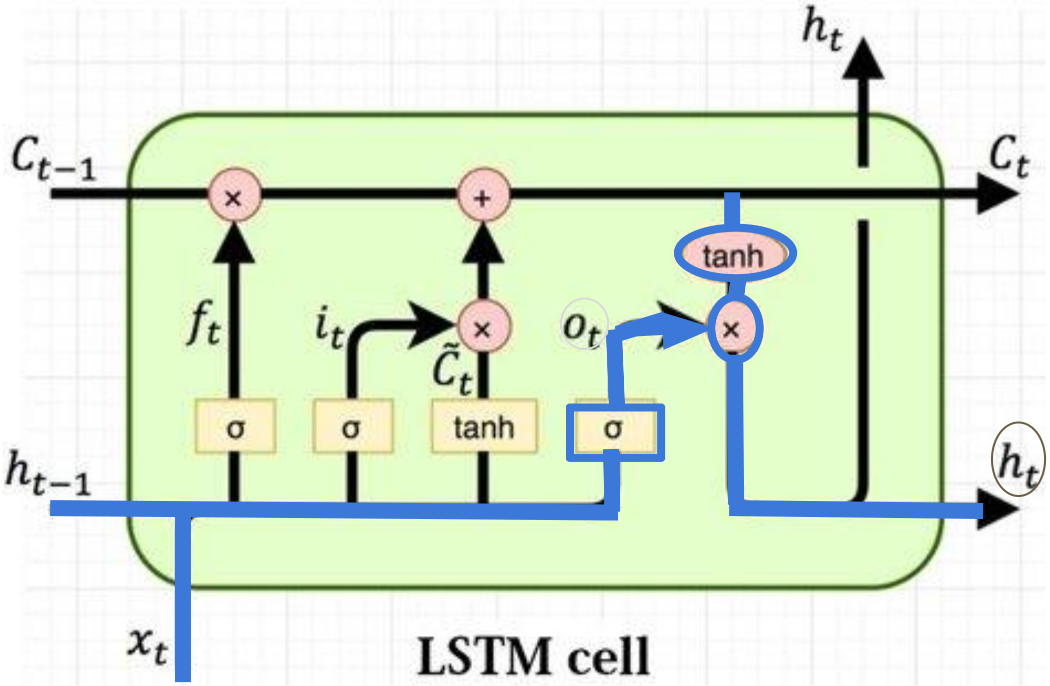


$$i_t = \sigma(W_{ih}h_{t-1} + W_{ix}x_t)$$

$$f_t = \sigma(W_{fh}h_{t-1} + W_{fx}x_t)$$

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

Output Layer



$$o_t = \sigma(W_{oh}h_{t-1} + W_{ox}x_t)$$

$$h_t = o_t \circ tanh(C_t)$$

Experiments

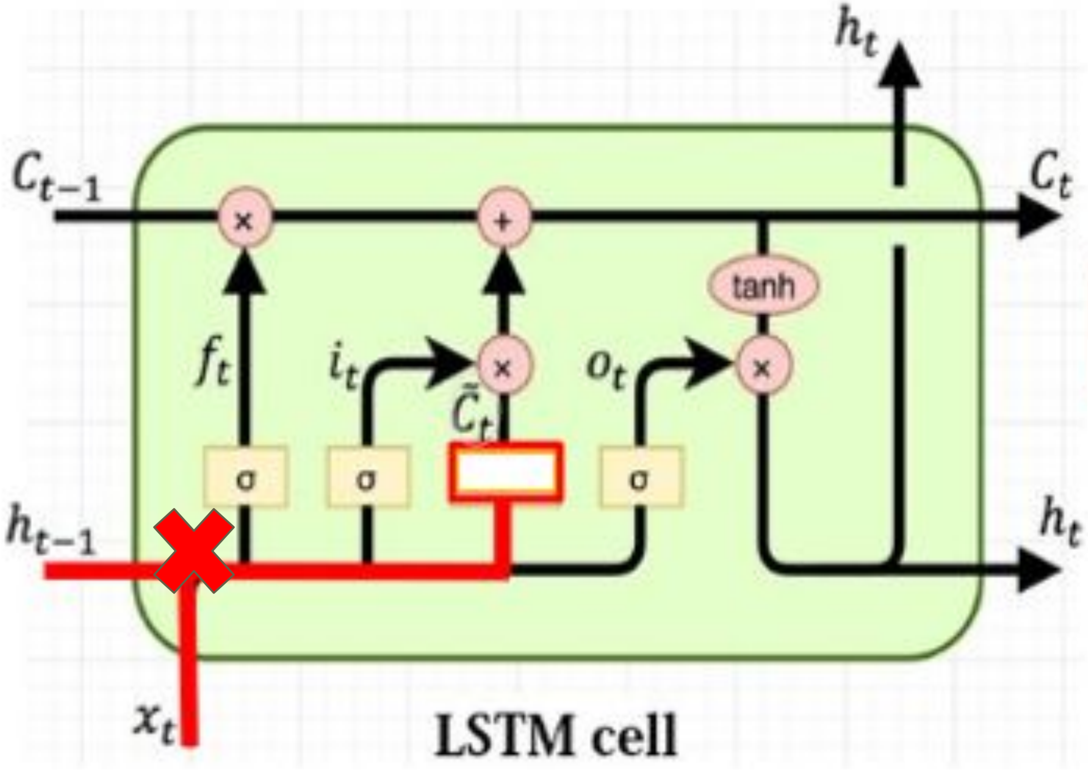
Simplified LSTM architectures

To test modeling power, the model has different ablations:

- LSTM-(S-RNN)
- LSTM- (GATES)
- LSTM-(S-RNN)-OUT
- LSTM-(S-RNN)-HIDDEN

LSTM-(S-RNN)

- Replace S-RNN in the Content Layer with simple linear transformation
- Replace equation in Subcomponent 1 Content Layer



Old formula

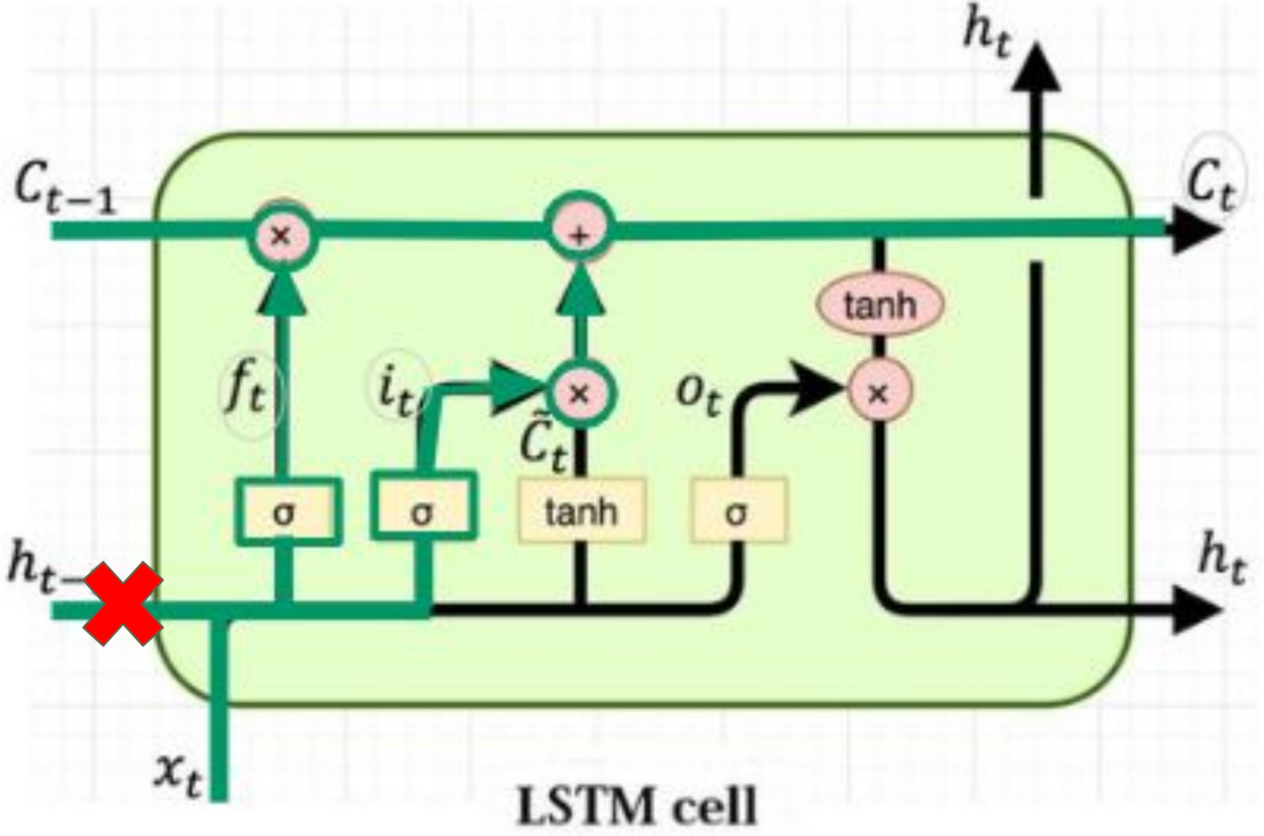
$$\tilde{C}_t = tanh(W_{ch}h_{t-1} + W_{cx}x_t)$$

New Formula

$$\tilde{C}_t = W_{cx}x_t$$

LSTM-GATES

- Removing the output gate from LSTM
- Replacing Equation from Output Layer Subcomponent 3



Old Formula

$$i_t = \sigma(W_{ih}h_{t-1} + W_{ix}x_t)$$

New Formula

$$i_t = \sigma(W_{ix}x_t)$$

LSTM-(S-RNN)-OUT

- Remove S-RNN and OUTPUT gate from LSTM
- New Model can be written as

$$h_t = \text{OUTPUT}\left(\sum_{j=0}^{t} w_j^t \circ \text{CONTENT}(x_j)\right)$$

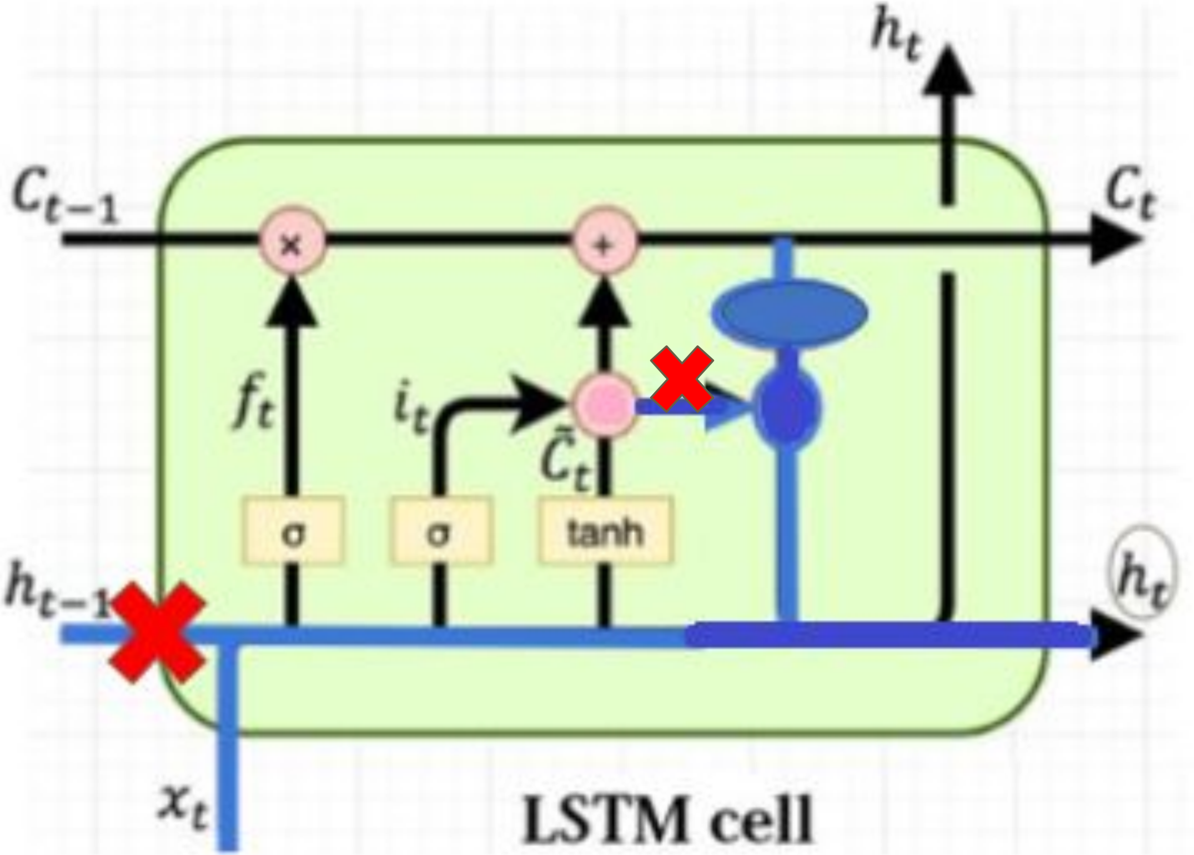
Element-Wise Weighted Sum

- Possible to show memory cell implicitly computes **an element-wise weighted sum** of all the previous layers
- Expanding the recurrence equation,

$$
\begin{aligned}
c_t &= i_t \circ \tilde{c}_t + f_t \circ c_{t-1} \\
&= \sum_{j=0}^{t} \left(i_j \circ \prod_{k=j+1}^{t} f_k \right) \circ \tilde{c}_j \\
&= \sum_{j=0}^{t} w_j^t \circ \tilde{c}_j
\end{aligned}
$$

LSTM-(S-RNN)-HIDDEN

- Ablate the hidden state from the gates
- Each gate is then computed using the formula



Old Formula

$$h_t = o_t \circ tanh(C_t)$$

New Formula

$$h_t = tanh(C_t)$$

Experimental Setup

- Consider 4 NLP cases

- Same hyperparameter and experimental setup for LSTM and simplified LSTM
- **Does Simplified LSTM perform equally or better?**

Language Models

- **Penn Treebank-** it is a dataset which tags the grammatical categories of each token in the text corpus

- **Two configurations:-** Medium and large are tested

Configuration	Model	Perplexity
PTB (Medium)	LSTM	83.9 ± 0.3
	– S-RNN	80.5
	– S-RNN – OUT	81.6
	– S-RNN – HIDDEN	83.3
	– GATES	140.9
PTB (Large)	LSTM	78.8 ± 0.2
	– S-RNN	76.0
	– S-RNN – OUT	78.5
	– S-RNN – HIDDEN	82.9
	– GATES	126.1

Table 1: Performance on language modeling benchmarks, measured by perplexity.

Question Answering

- Two different QA systems
  - -BiDAF contains 3 LSTMs
  - -phrase layer, modelling layer, span end encoder layer
- All LSTMs are replaced by there simplified counterparts
- Hyperparameters are not modified
  - -DrQA open source replace LSTMs leaving everything intact

System	Model	EM	F1
BiDAF	LSTM	67.9 ± 0.3	77.5 ± 0.2
	– S-RNN	68.4	78.2
	– S-RNN – OUT	67.4	77.2
	– S-RNN – HIDDEN	66.5	76.6
	– GATES	62.9	73.3
DrQA	LSTM	68.8 ± 0.2	78.2 ± 0.2
	– S-RNN	68.0	77.2
	– S-RNN – OUT	68.7	77.9
	– S-RNN – HIDDEN	67.9	77.2
	– GATES	56.4	66.5

Table 2: Performance on SQuAD, measured by exact match (EM) and span overlap (F1).

Dependency Parsing

- **Deep biaffine Dependency Parser** which relies on stacked bidirectional LSTMs to learn context sensitive word embeddings for determining arcs between a pair of words
- Existing hyperparameters are used and LSTMs are replaced.

Model	UAS	LAS
LSTM	90.60 ± 0.21	88.05 ± 0.33
– S-RNN	90.77	88.49
– S-RNN – OUT	90.70	88.31
– S-RNN – HIDDEN	90.53	87.96
– GATES	87.75	84.61

Table 3: Performance on the universal dependencies parsing benchmark, measured by unlabeled (UAS) and labeled attachment score (LAS).

Machine Translation

- Open NMT- train German to English Translation
- Default models and hyperparameters
- Replacing Bidirectional LSTMs encoder with simplified LSTMs

Model	BLEU
LSTM	38.19 ± 0.1
– S-RNN	37.84
– S-RNN – OUT	38.36
– S-RNN – HIDDEN	36.98
– GATES	26.52

Table 4: Performance on the WMT 2016 multi-modal English to German benchmark.

Related Work

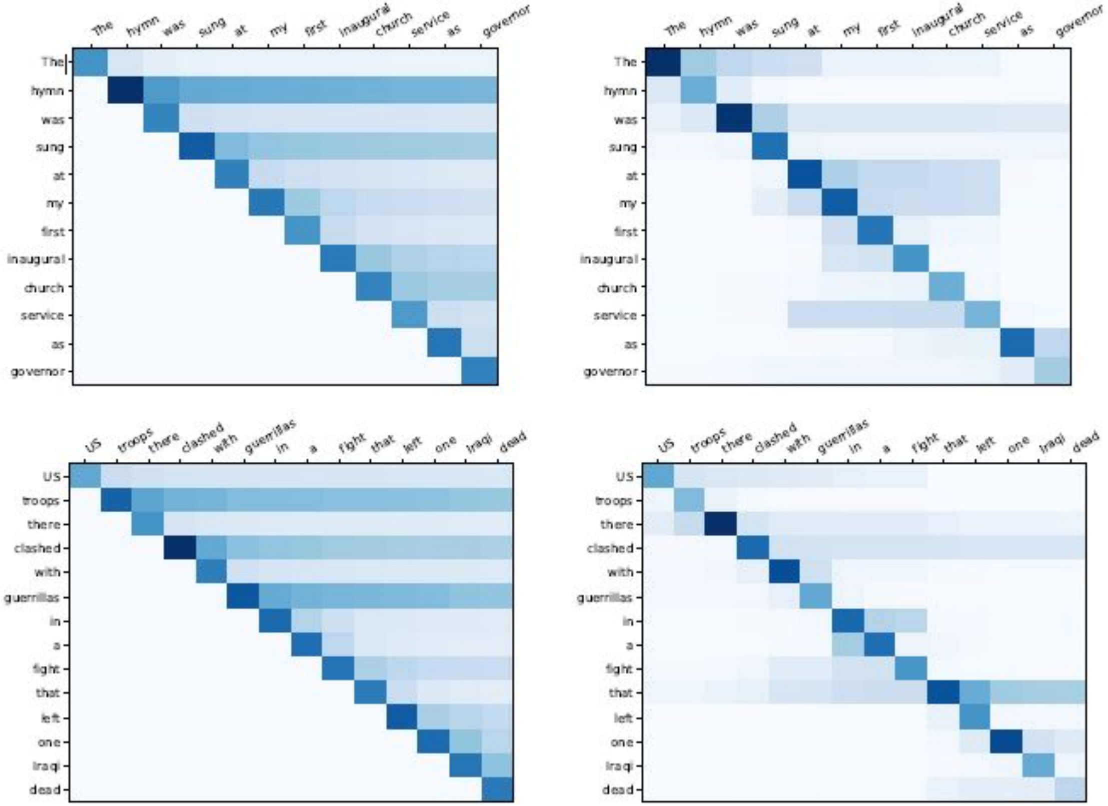
- Different variants of LSTMs explored
- LSTMs with rewiring of connections
- Quasi-recurrent models
- Recurrent additive networks
- **But, this is first study to provide comparison b/w LSTMs with and without recurrent layer**
- **Focuses on explaining what LSTMs are learning**

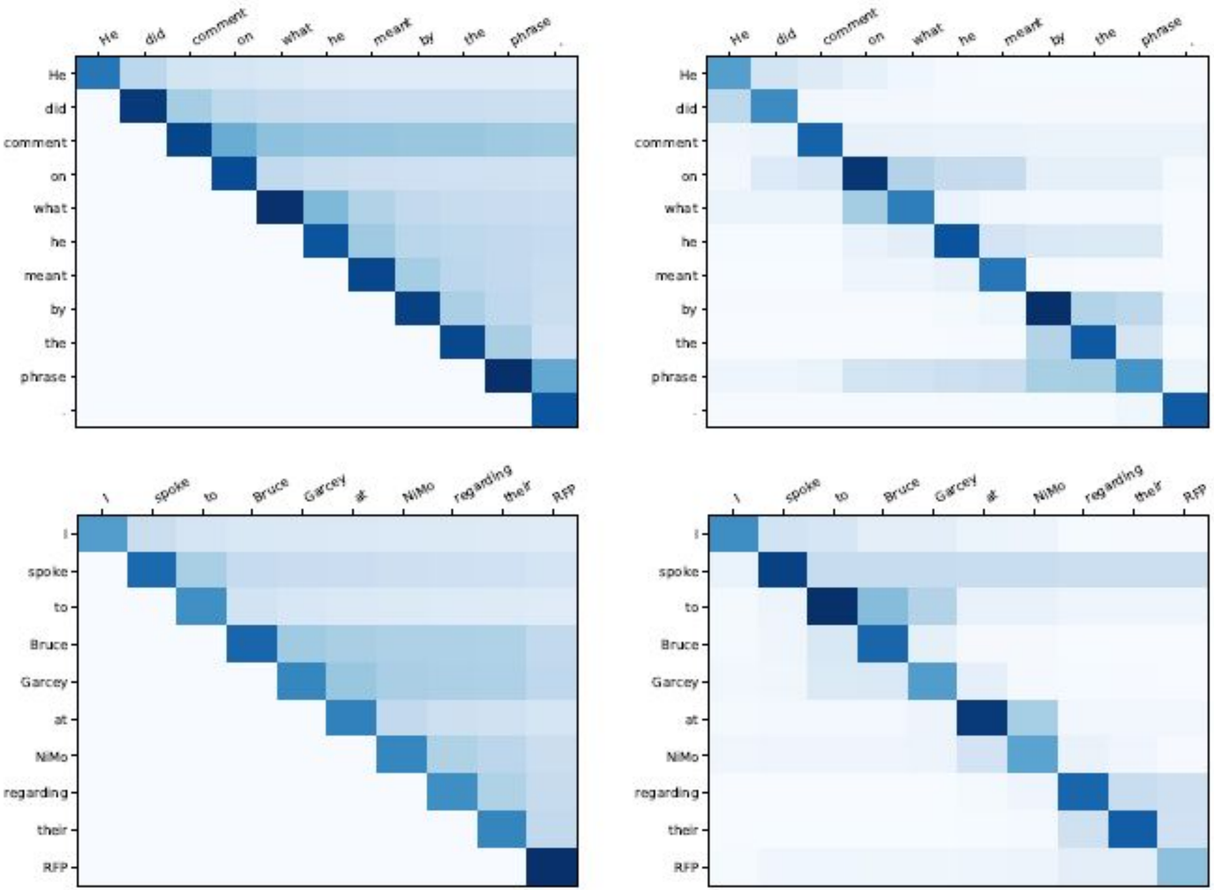
Conclusion

- Removing S-RNN → Little/no performance loss in LSTM
- Gating mechanism is important
- LSTMs powerful → Dynamically compute elementwise weighted sums of content layers.
- Ablating recurrence from gates → Considerable drop in performance

Thank You !
Questions?/Comments/Suggestions

Language model weights	Dependency parser weights




He did comment on what he meant by the phrase .
He did comment on what he meant by the phrase .
He did comment on what he meant by the phrase .
He did comment on what he meant by the phrase .
I spoke to Bruce Garcey at NiMo regarding their RFP
I spoke to Bruce Garcey at NiMo regarding their RFP
I spoke to Bruce Garcey at NiMo regarding their RFP
I spoke to Bruce Garcey at NiMo regarding their RFP

LSTM as Self-Attention

- LSTM weight are vectors, Attention computes scalar weights
- Weighted sum allows linear complexity rather than quadratic
- Attention has probabilistic interpretation due to softmax normalisation
- Sum of weights in LSTM can go to sequence lengths