Introduction to Deep NLP

Recurrent Neural Network (RNN)

Long Short-Term Memory (LSTM)

Attention Mechanism

ELMO

BERT

Transformer XL, XLNet

Recurrent Neural Network (RNN)

Example applications:

- Google Translate
- Apple Siri
- Baidu’s Text to Speech
- …

Language Model – word prediction

- **Chain Rules:** computing the joint probability of all the words conditioned by the previous words

$$P(w_1 w_2 \ \ldots . w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

P("its water is so transparent") =

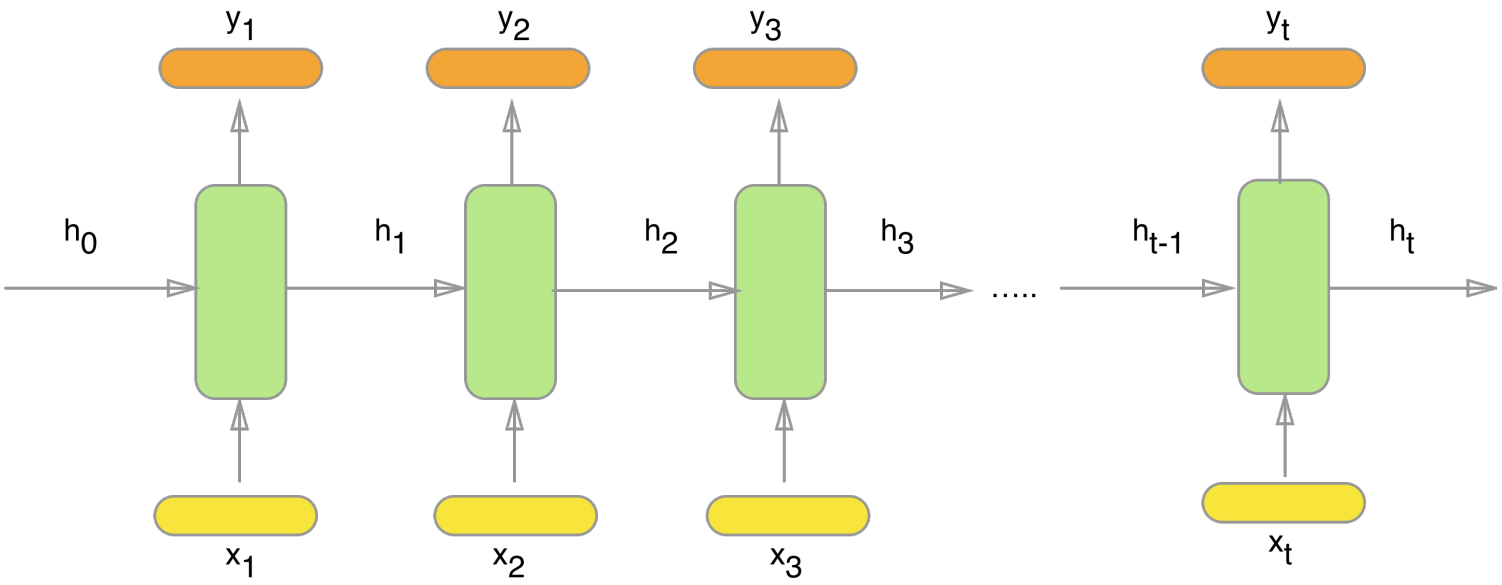
P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

- With our training corpus, we can compute the probability that something occurred by counting its occurrences and dividing by the total number

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

A Simple Recurrent Neural Network



$x_1, x_2, x_3, \ldots, x_t$: input words from the text
$y_1, y_2, y_3, \ldots, y_t$: the predicted next words
$h_0, h_1, h_2, h_3, \ldots, h_t$: the information for the previous input words.

1) $h_t = f(W^{(hh)}h_{t-1} + W^{(hx)}x_t)$

2) $y_t = softmax(W^{(S)}h_t)$

3) $J^{(t)}(\theta) = \sum_{i=1}^{|V|} (y'_{t_i} log\, y_{t_i})$

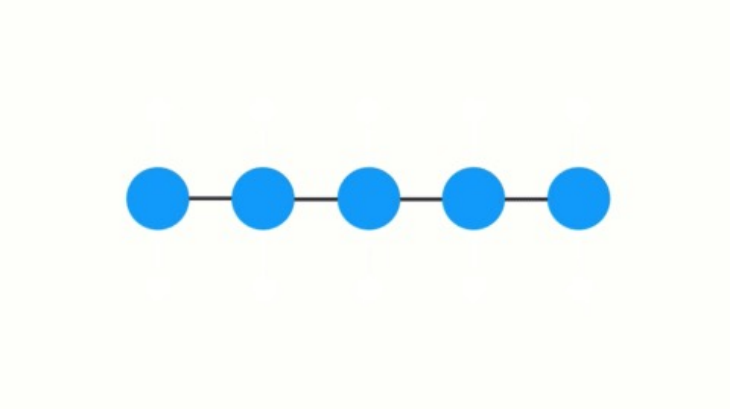
Source:

https://towardsdatascience.com/learn-how-recurrent-neural-networks-work-84e975feaaf7

A Simple Recurrent Neural Network: Problem

Vanishing/Exploding Gradient Problem

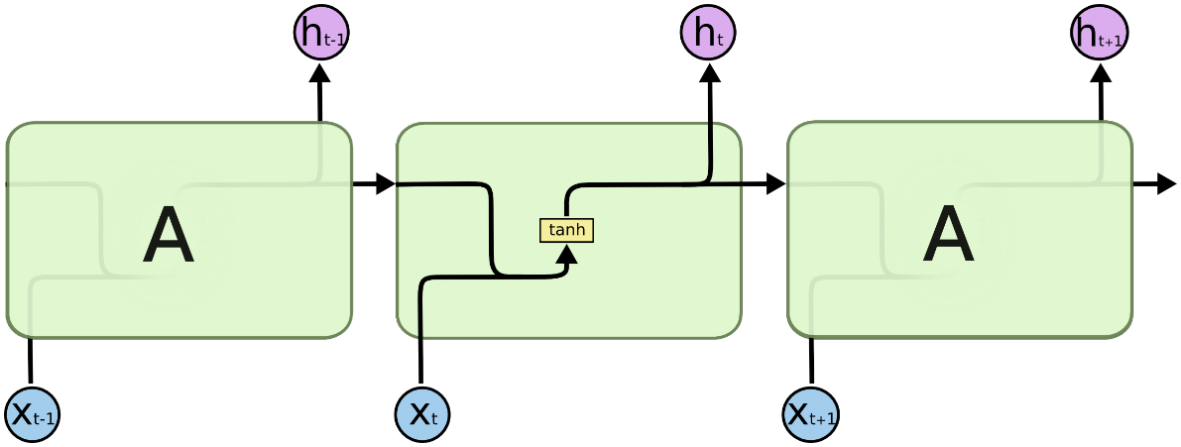
- Words from time steps far away are not as influential as they should be any more



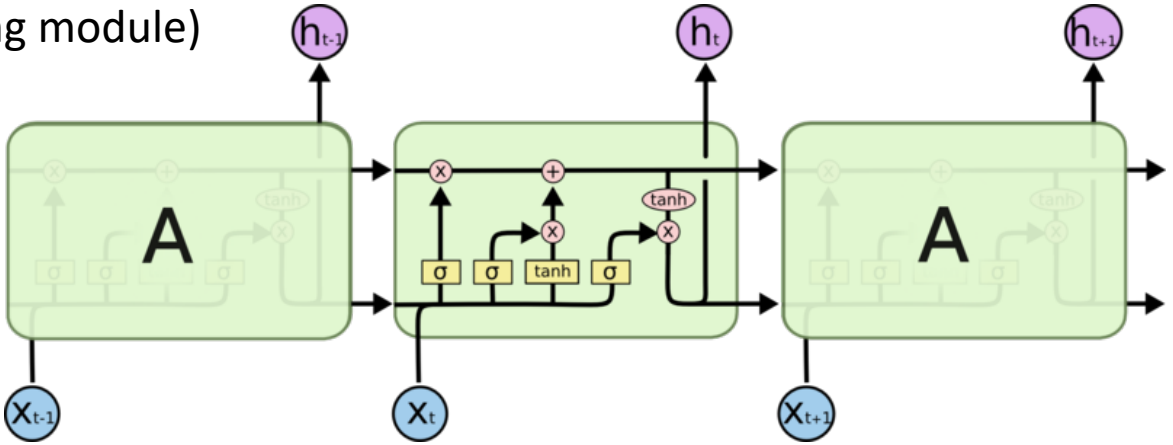
Example:

Michael and Jane met last Saturday. It was a nice sunny day when they saw each other in the park. Michael just saw the doctor two weeks ago. Jane came back from Norway last Monday. Jane offered her best wish to _____.

Long Short Term Memory (LSTM) Networks



RNN (the repeating module)



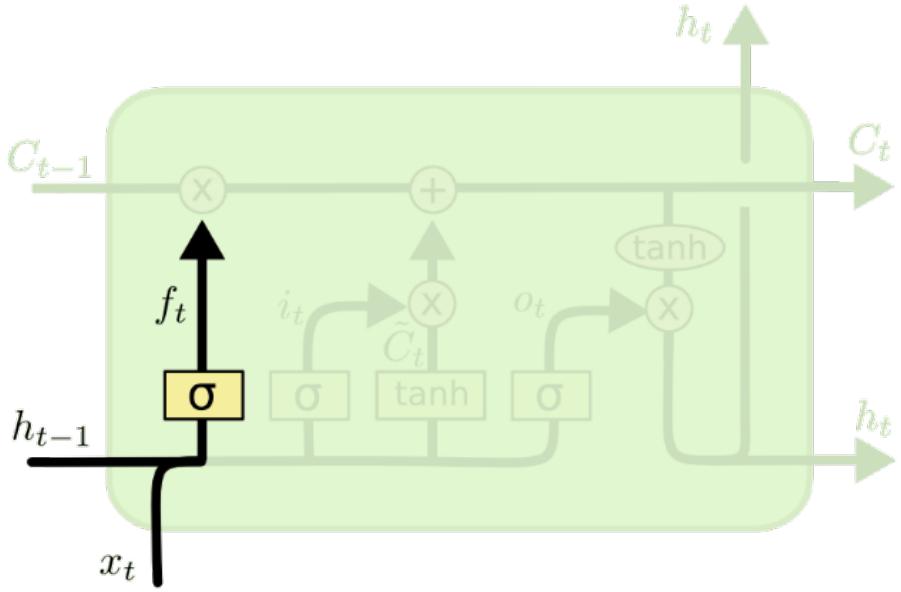
LSTM (the repeating module)

http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (image source)

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

LSTM Walk Through

- Forget gate layer: how much information from the previous time step will be kept?

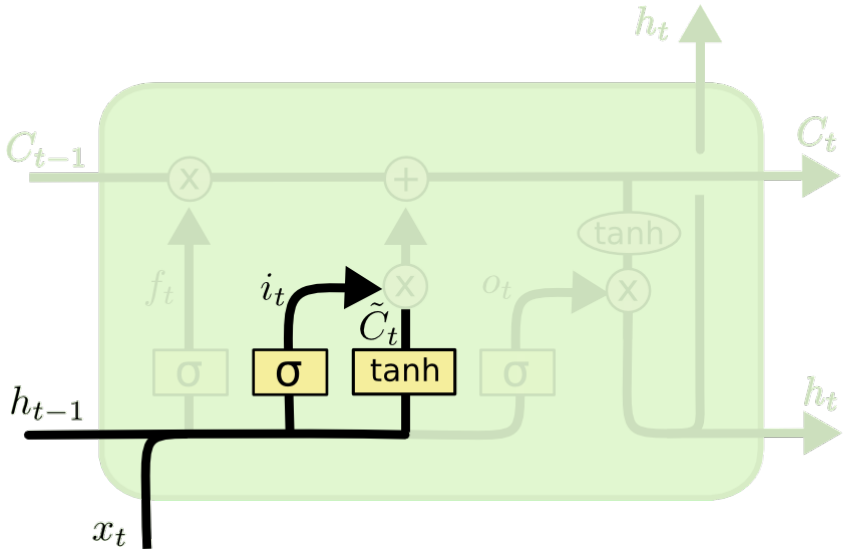


http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (image source)

Sigmoid function: outputs a number between 0 and 1

LSTM Walk Through

- Input gate layer: which values will be updated
- Tanh layer: a vector of new candidate values that could be added to the state



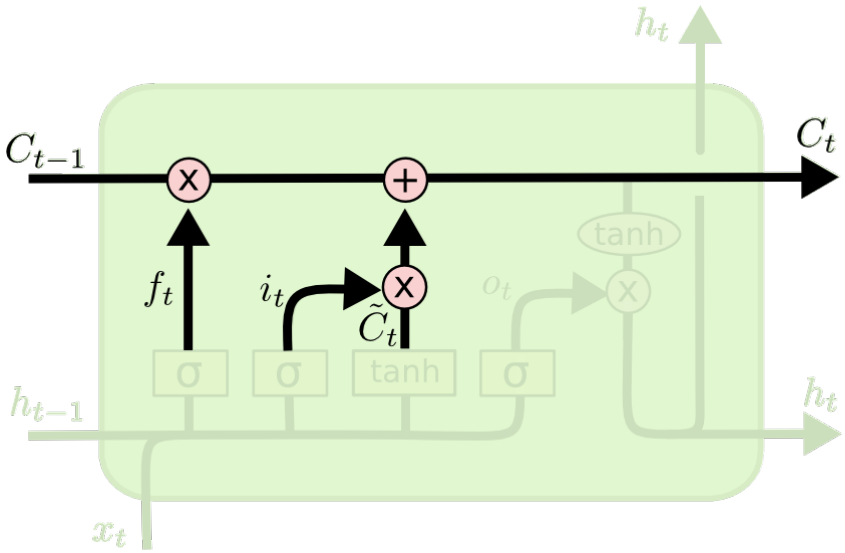
http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (image source)

Sigmoid function: outputs a number between 0 and 1

Tanh function (hyperbolic tangent function): outputs a number between -1 and 1

LSTM Walk Through

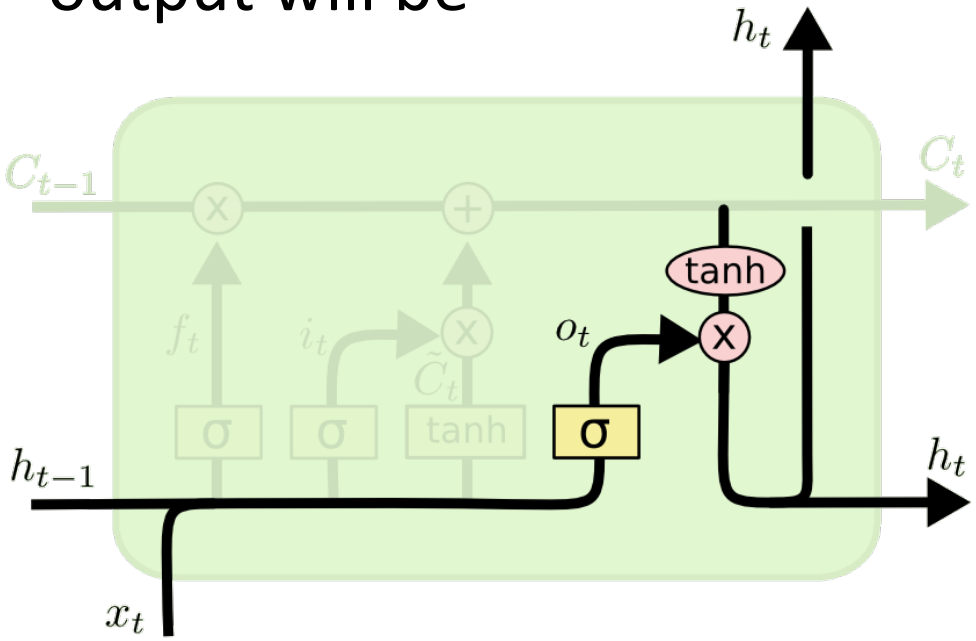
- Update the old cell state, C_{t-1}, into the new cell state C_t.



http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (image source)

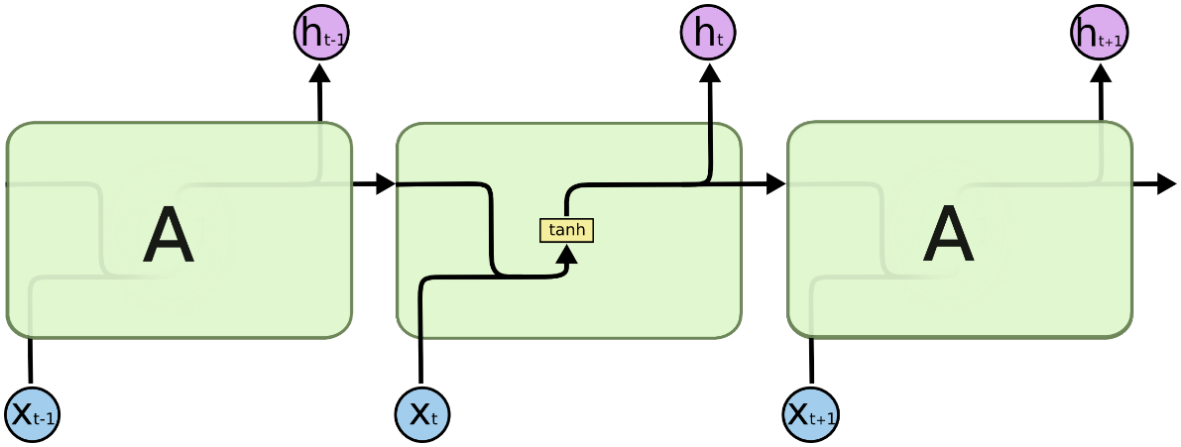
LSTM Walk Through

- Based on the cell state, we will decide what the output will be

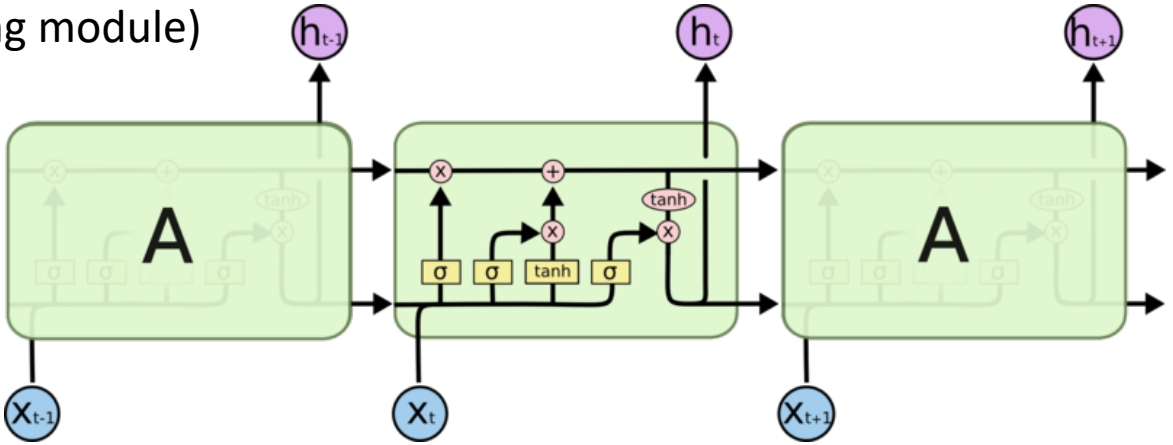


http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (image source)

Long Short Term Memory (LSTM) Networks



RNN (the repeating module)

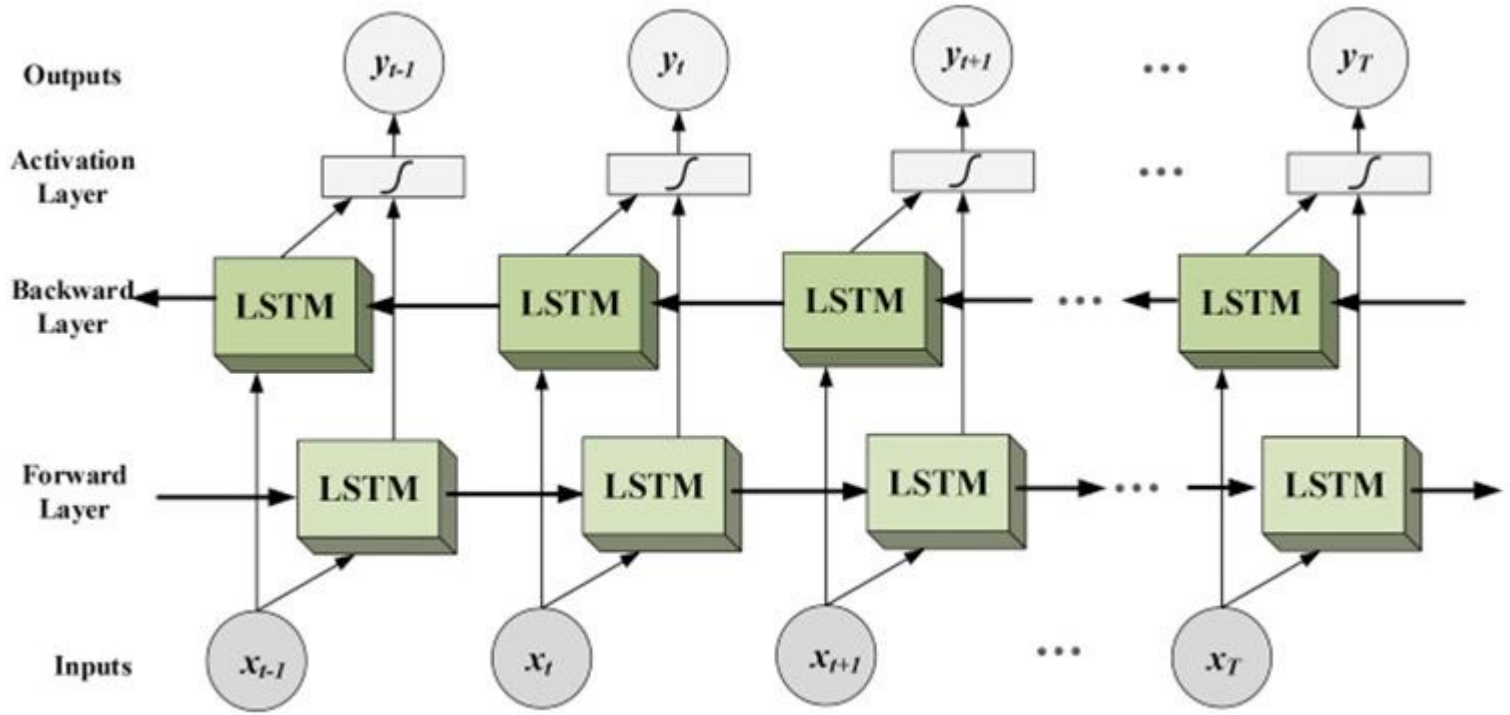


LSTM (the repeating module)

http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (image source)

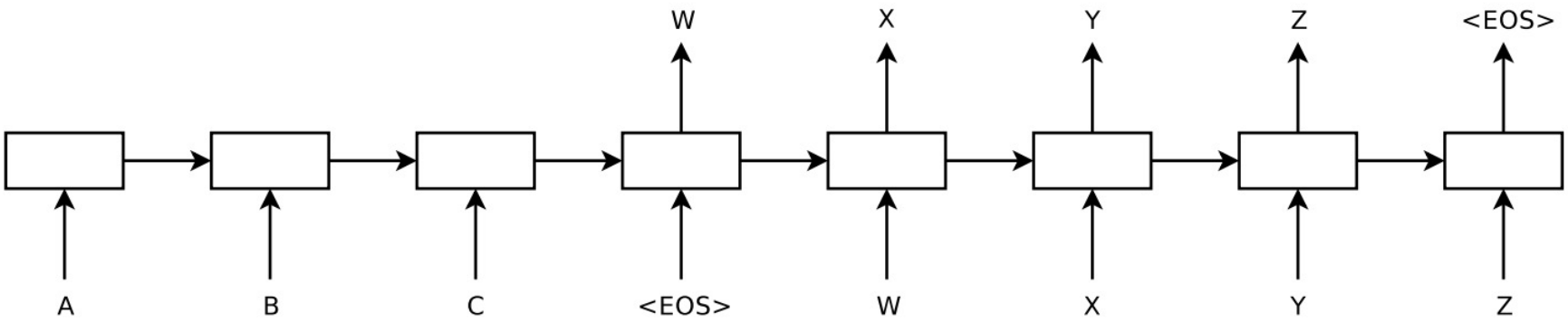
Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

Bidirectional LSTM



Source: https://www.i2tutorials.com/technology/deep-dive-into-bidirectional-lstm/

An Encoder-Decoder Architecture



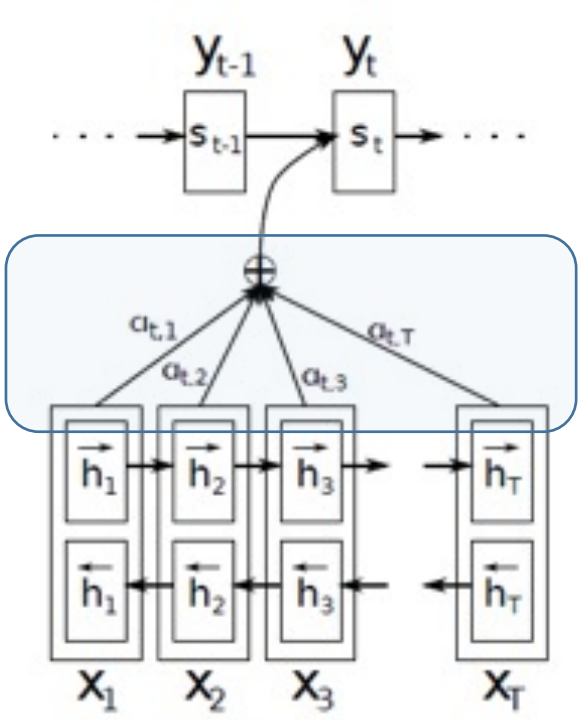
Encoder: a LSTM that encodes the input sequence to a fixed-length internal representation *W.*

Decoder: another LSTM that takes the internal representation *W* to extract the output sequence from that vector

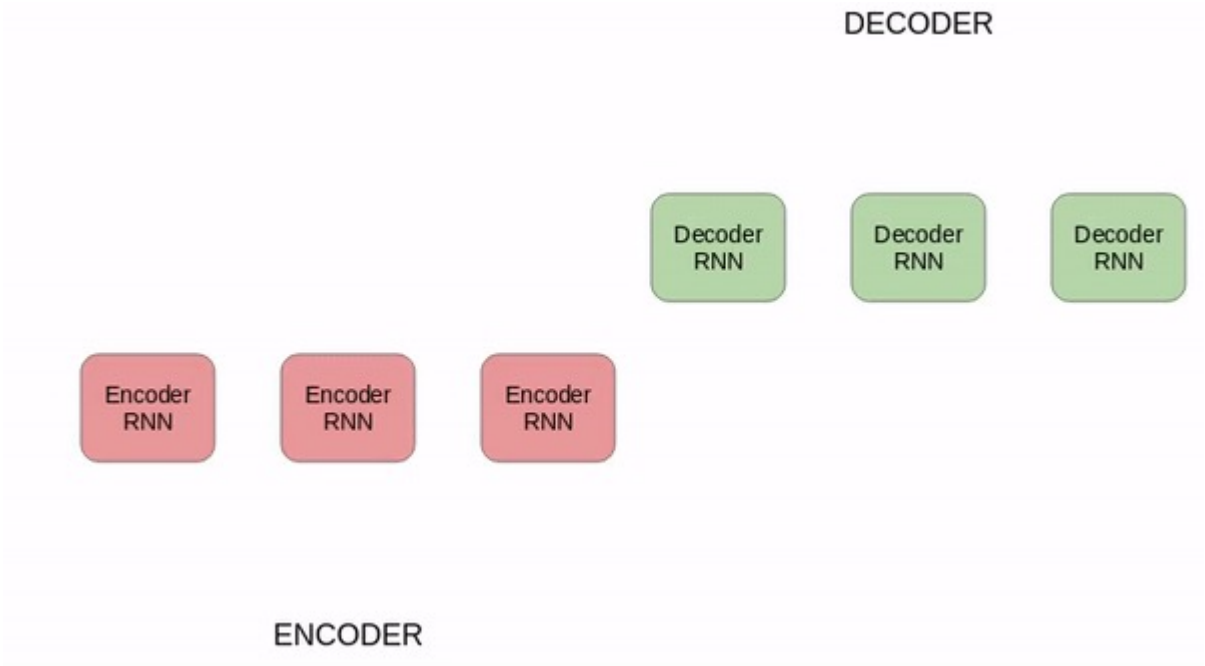
Source: https://machinelearningmastery.com/encoder-decoder-recurrent-neural-network-models-neural-machine-translation/

Attention Mechanism

Embeddings of all the words in the input (represented by hidden states) while creating the context vector.



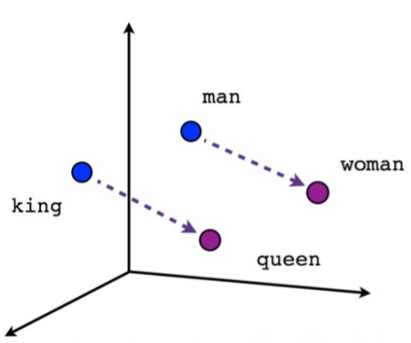
Example: German to English Translation with Seq2Seq



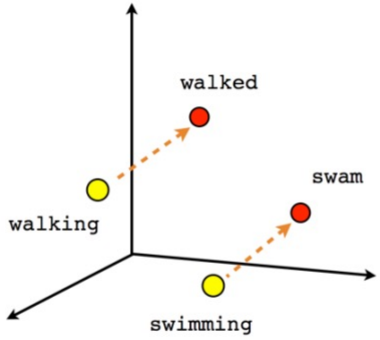
At a time step in the Encoder, the RNN takes a word vector (x_i) from the input sequence and a hidden state (H_{i-1}) from the previous time step; the hidden state is updated (H_i)

The context vector to the decoder is the hidden state from the last unit of the encoder (without the Attention mechanism) or the weighted sum of the hidden states of the encoder (with the Attention mechanism)

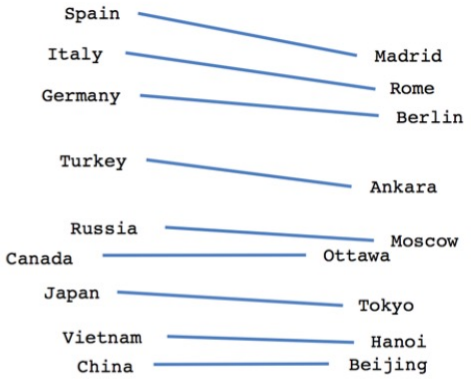
Problems with Word2Vec/GloVe



Male-Female



Verb tense

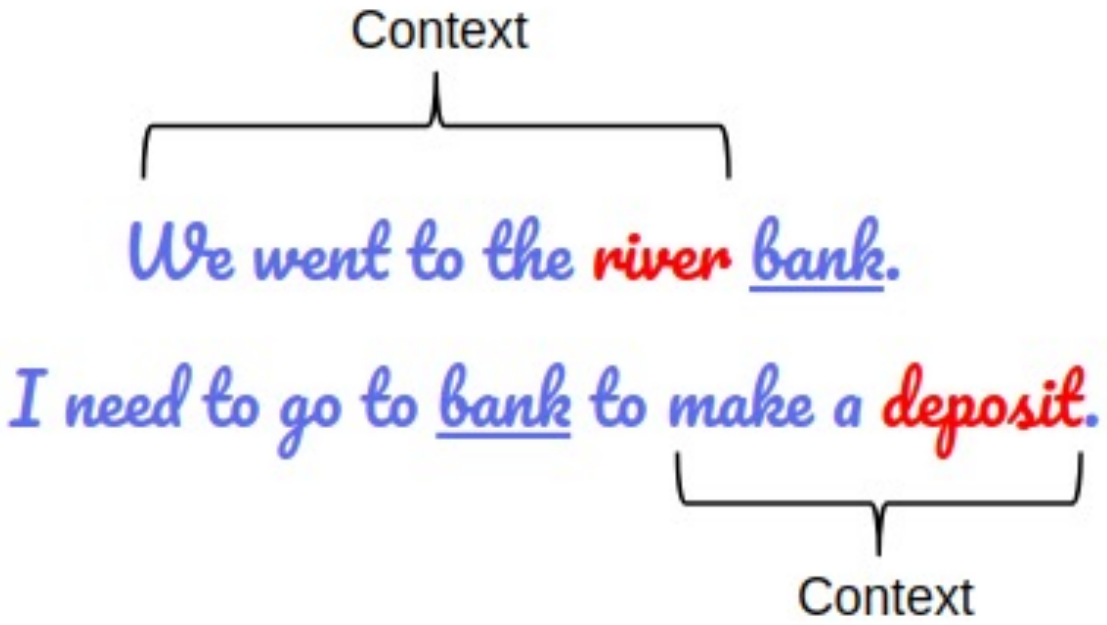


Country-Capital

Word Sense Disambiguation?

Image source: https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/?utm_source=blog&utm_medium=comprehensive-guide-attention-mechanism-deep-learning

Bidirectional Encoder Representations from Transformers (BERT)

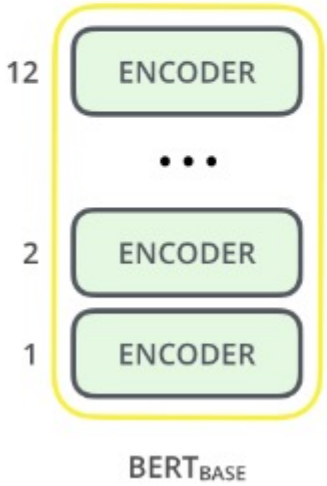


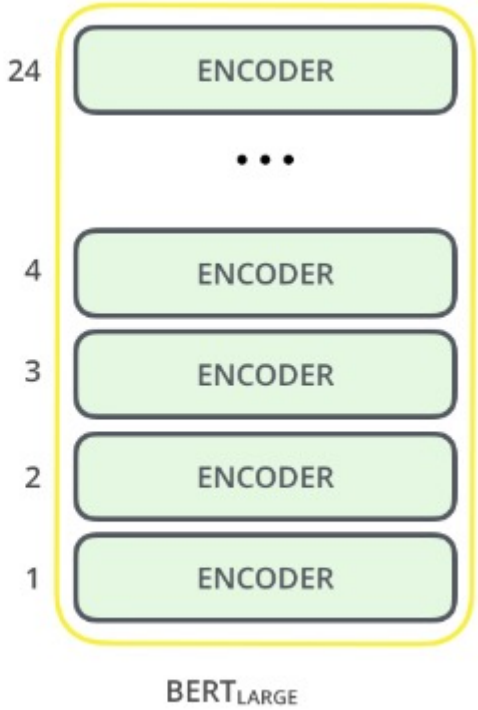
BERT captures both the left and right context

Source: https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/?utm_source=blog&utm_medium=comprehensive-guide-attention-mechanism-deep-learning

Bidirectional Encoder Representations from Transformers (BERT)

- BERT Base: 12 layers (transformer blocks), 12 attention heads, and 110 million parameters
- BERT Large: 24 layers (transformer blocks), 16 attention heads and, 340 million parameters





Source: https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/?utm_source=blog&utm_medium=comprehensive-guide-attention-mechanism-deep-learning

BERT: Text Preprocessing

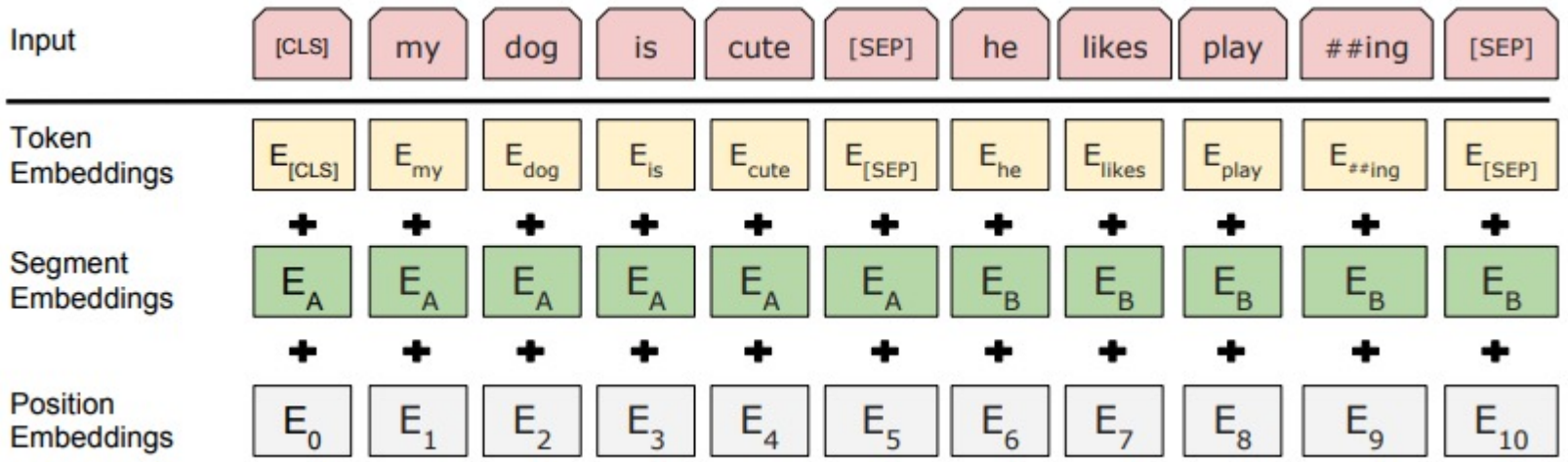


Image Source:
https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/?utm_source=blog&utm_medium=comprehensive-guide-attention-mechanism-deep-learning

BERT: Pre-trained on two NLP Tasks

- Masked Language Modeling

"This turns out to be the greatest thing that has ever happened to me"

"This turns out to be the [MASK] thing that has ever happened to me"

Training data:

- the selected words were replaced with
  - the masked token [MASK]: 80%
  - random words: 10%
- 10% of the time the words were left unchanged

Source:
https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/?utm_source=blog&utm_medium=comprehensive-guide-attention-mechanism-deep-learning

BERT: Pre-trained on two NLP Tasks

- Next Sentence Prediction

Given sentences A and B, is B the sentence after A or not?

Training data
- 50% “IsNext”
- 50% “IsNotNext” (a random sentence from the corpus)

Source:
https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/?utm_source=blog&utm_medium=comprehensive-guide-attention-mechanism-deep-learning

BERT: Pre-trained on two NLP Tasks

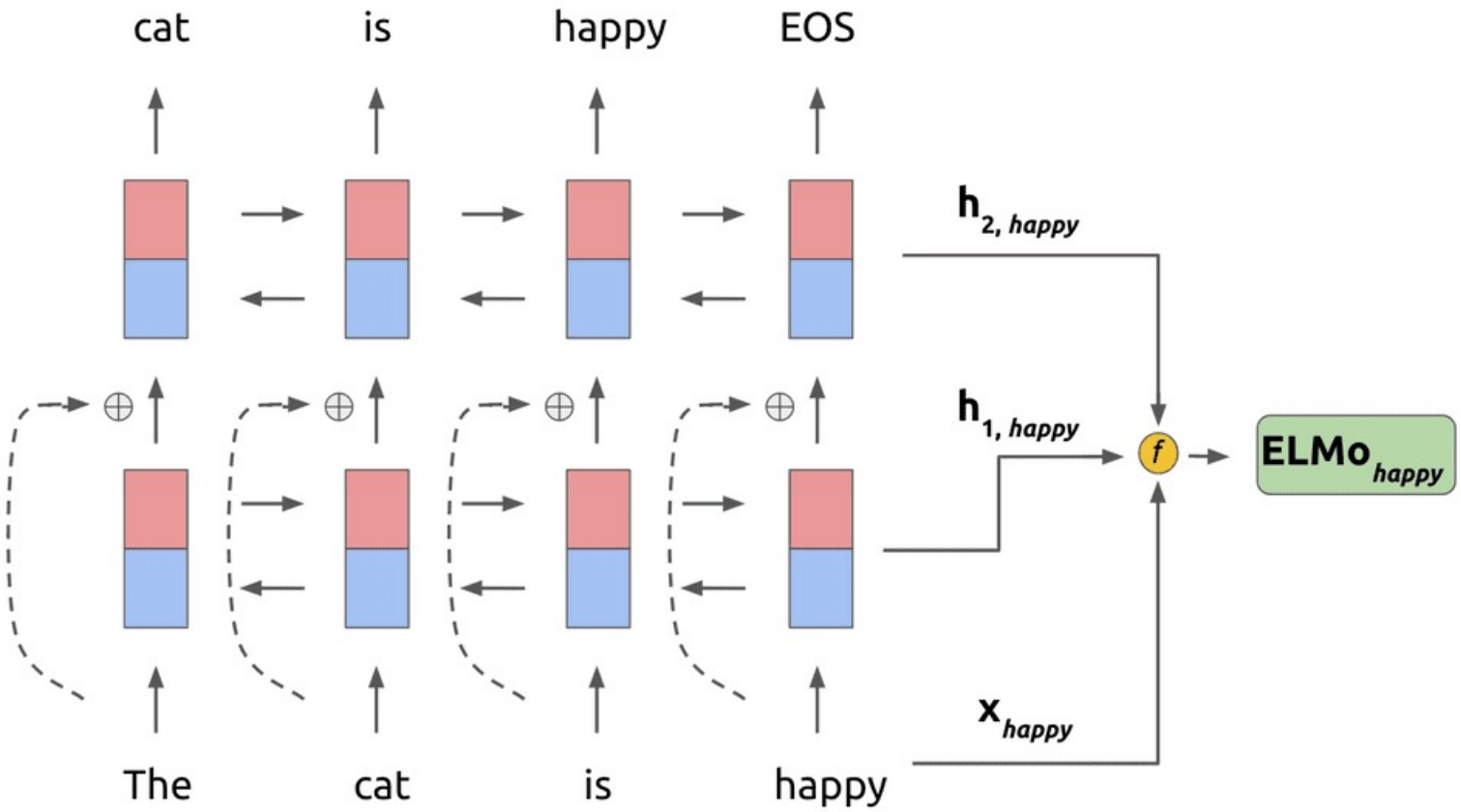
- To use BERT embedding, an open source project called Bert-as-Service is used: https://github.com/hanxiao/bert-as-service

Source: https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/?utm_source=blog&utm_medium=comprehensive-guide-attention-mechanism-deep-learning

ELMo Embedding

- Contextual
- Character based
- Bi-LSTM

ELMo: Example



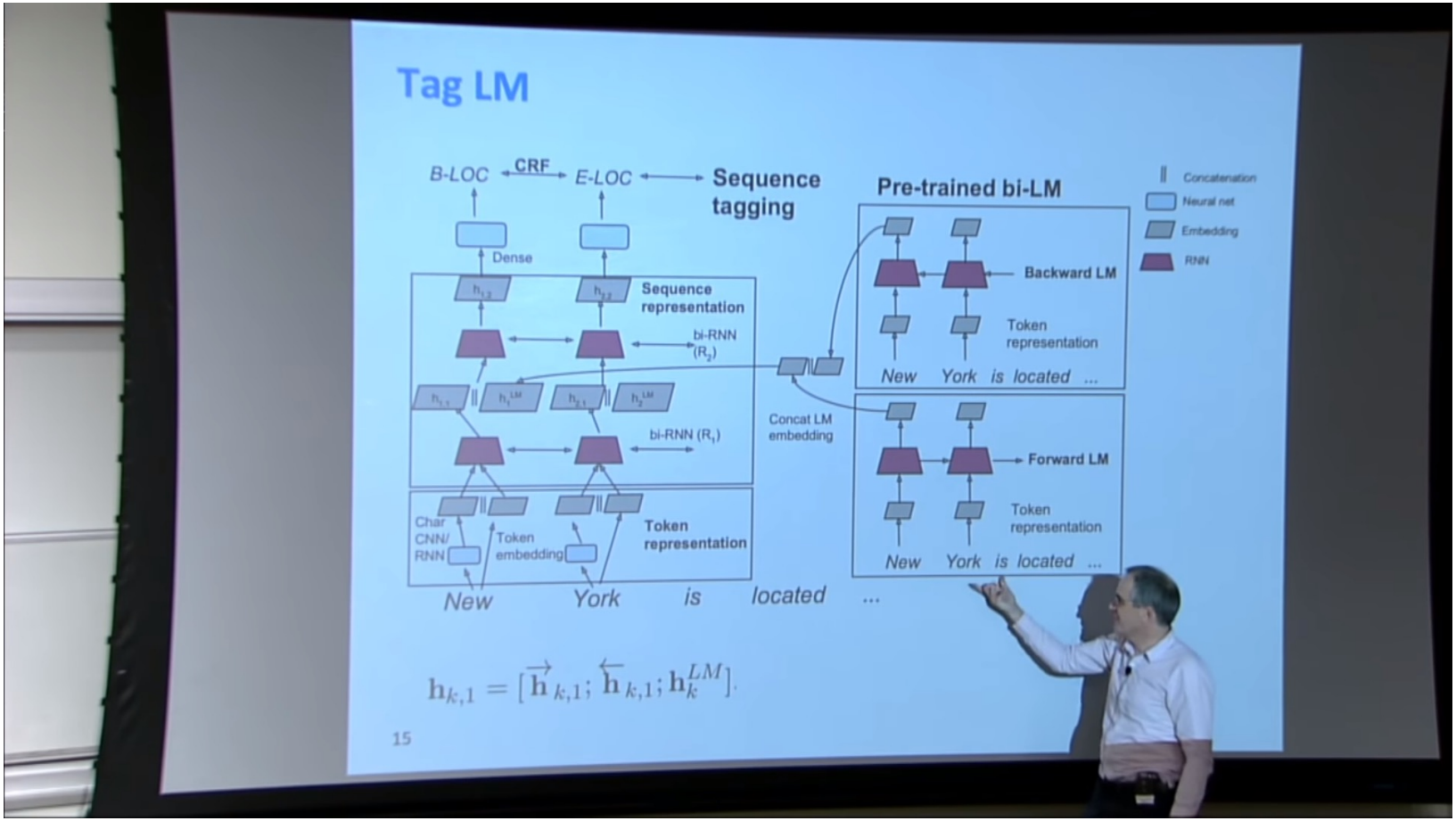
An example of combining the bidirectional hidden representations and word representation for "happy" to get an Elmo-specific representation.

Image source: https://medium.com/saarthi-ai/elmo-for-contextual-word-embedding-for-text-classification-24c9693b0045

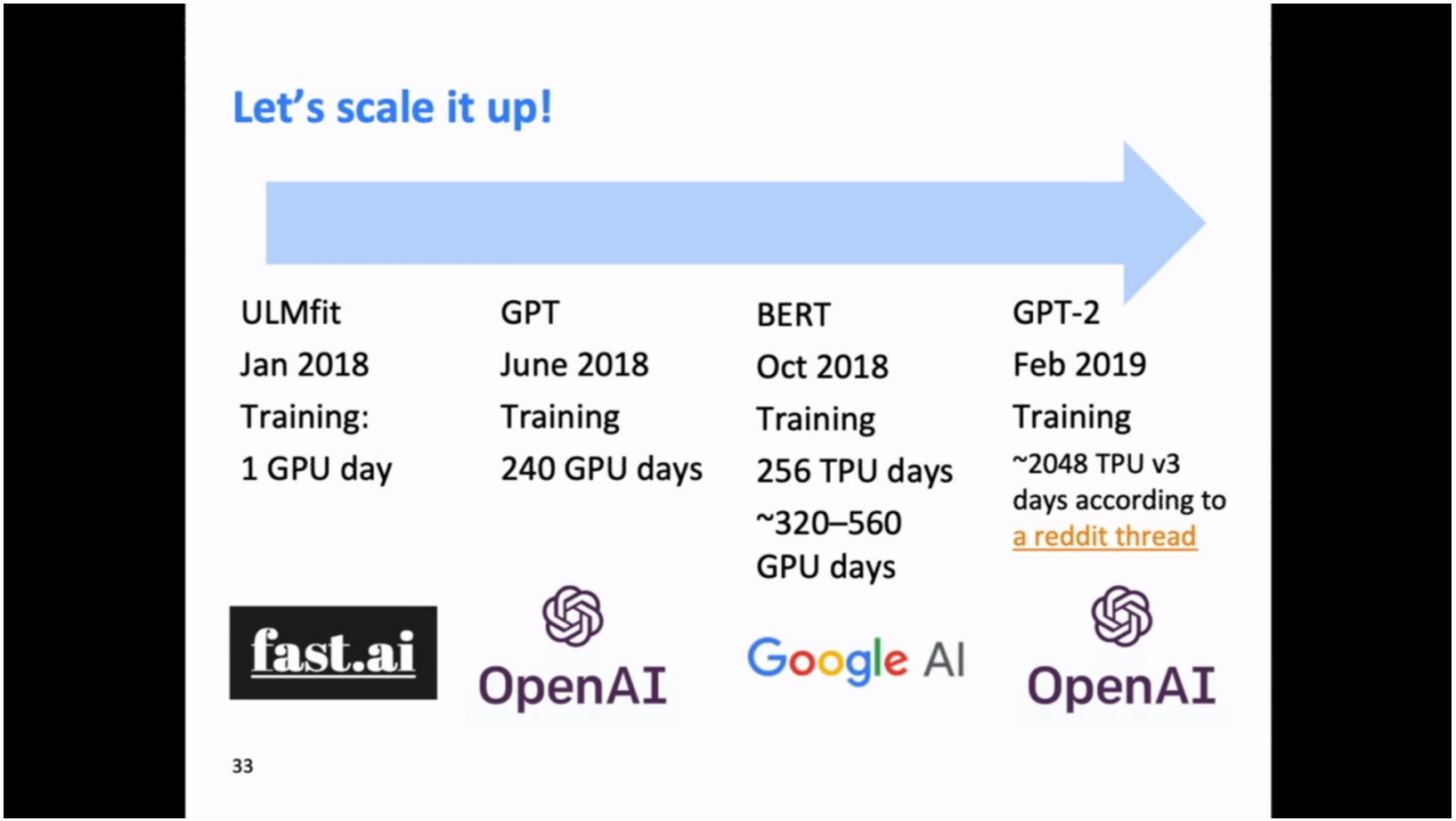
ELMo: Pre-training

- **1 Billion Word Benchmark**, approximately 800M tokens of news crawl data from WMT 2011

- Wikipedia (1.9B) and all of the monolingual news crawl data from WMT 2008-2012 (3.6B)



Screenshot taken from Stanford CS224N: NLP with Deep Learning | Winter 2019 | Lecture 13 – Contextual Word Embeddings
https://www.youtube.com/watch?v=S-CspeZ8FHc



Screenshot taken from Stanford CS224N: NLP with Deep Learning | Winter 2019 | Lecture 13 – Contextual Word Embeddings
https://www.youtube.com/watch?v=S-CspeZ8FHc

XLNet

- Transformer XL
  - Relative positioning
  - The recurrence mechanism
- Permutation language modeling: word prediction in an arbitrary order
  - Example: *"I", "like", "cats", "more", "than", "dogs"*