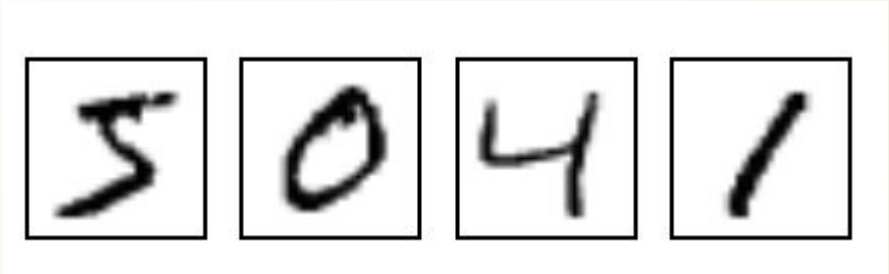
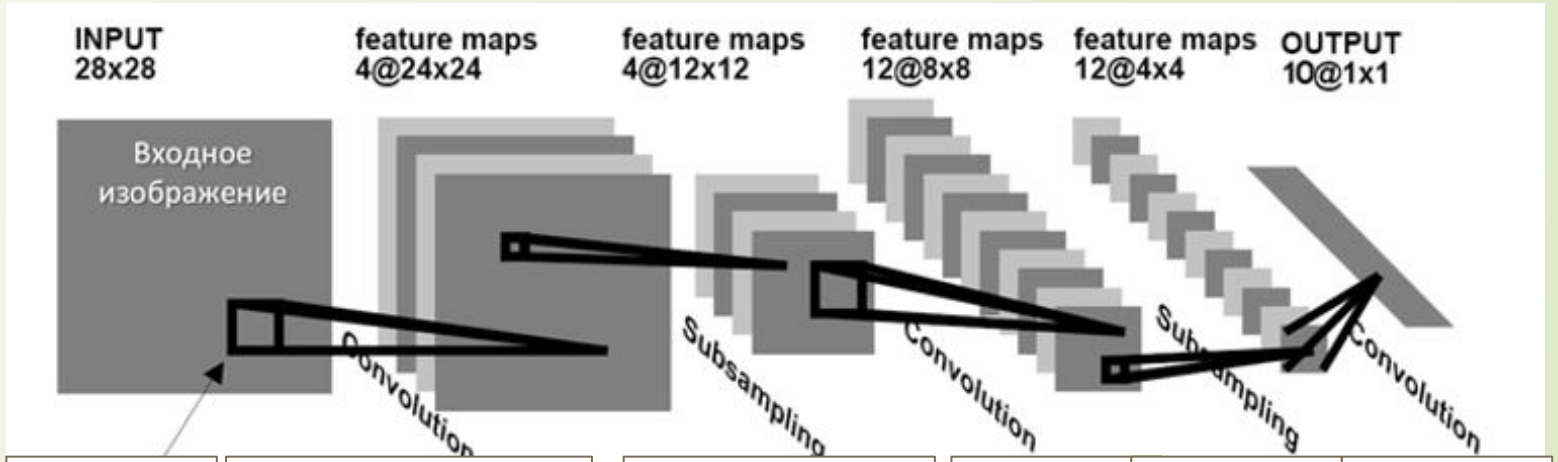
Contextualized Word Embeddings

Ivan Bordarenko (MIPT, Data Monsters, NSU)

email: bond005@Yandex.ru

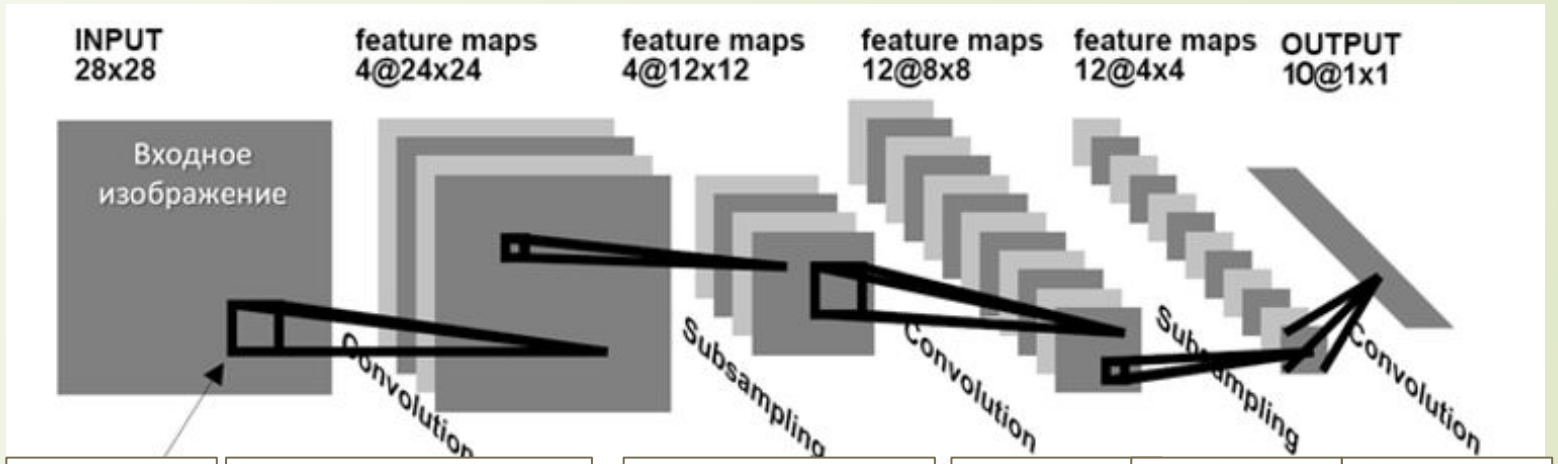
Handwritten digits recognition





Convolution kernel - matrix of weights, for example of shape 5*5	**Convolution - process all neurons of lower layer with kernel to activate neurons of upper layer - get feature maps. For different kernel types get different feature maps.**	**Subsample - operation to decrease dimension. Often - choose from maximum or search of average through neighbor neurons.**	**Convolution**	**Subsample**	**Convolution**

Handwritten text recognition



Convolution kernel - matrix of weights, for example of shape 5*5

Convolution - process all neurons of lower layer with kernel to activate neurons of upper layer - get feature maps. For different kernel types get different feature maps.

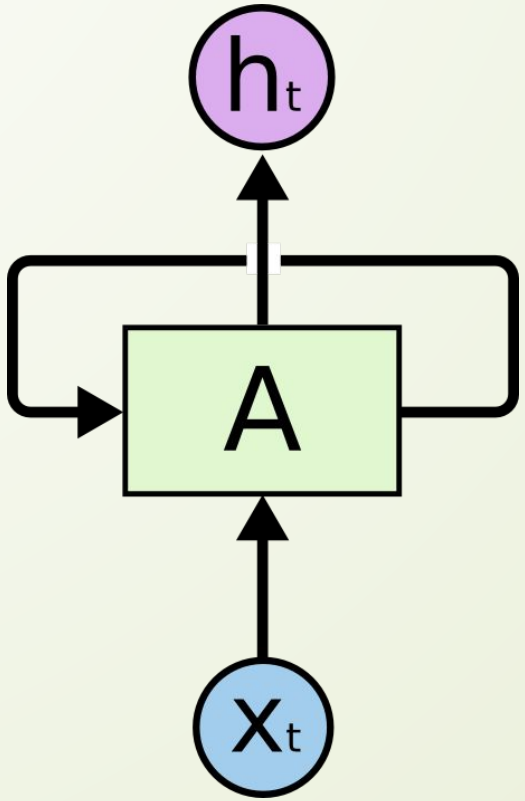
Subsample - operation to decrease dimension. Often - choose from maximum or search of average through neighbor neurons.

Convolution	Subsample	Convolution

Problem

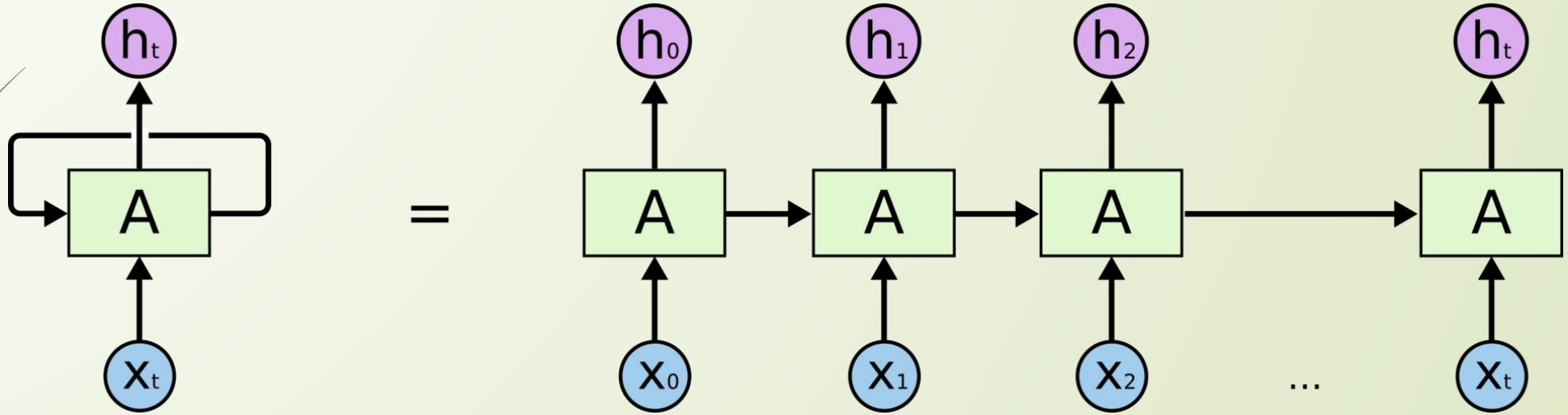
- Neural network with forward signal propagation **has no memory!**

Solution! Recurrent neural networks.

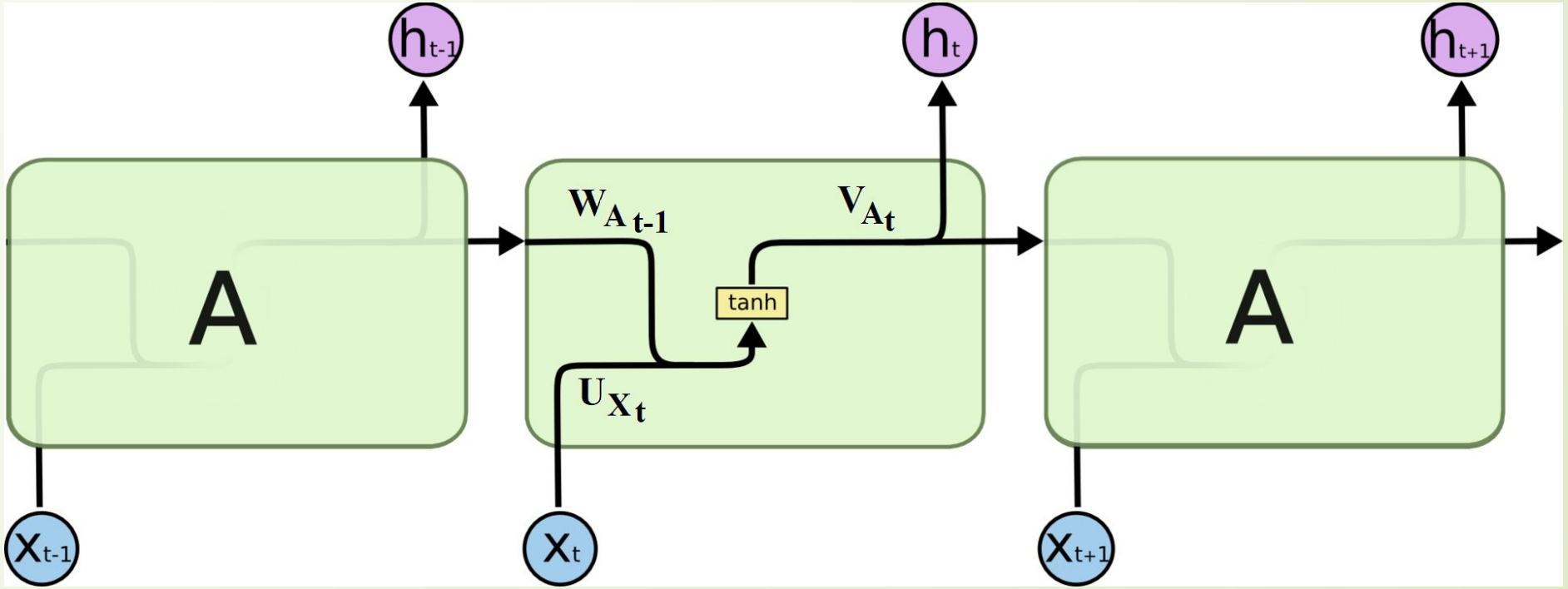


Recurrent neural networks have reverse connections.

Recurrent neural network



Recurrent neural network as multilayer



$$h_t = \text{softmax}(V_{A_t})$$

$$A_t = \tanh(U_{X_t} + W_{A_{t-1}})$$

Training of recurrent neural network. Backpropagation Through Time

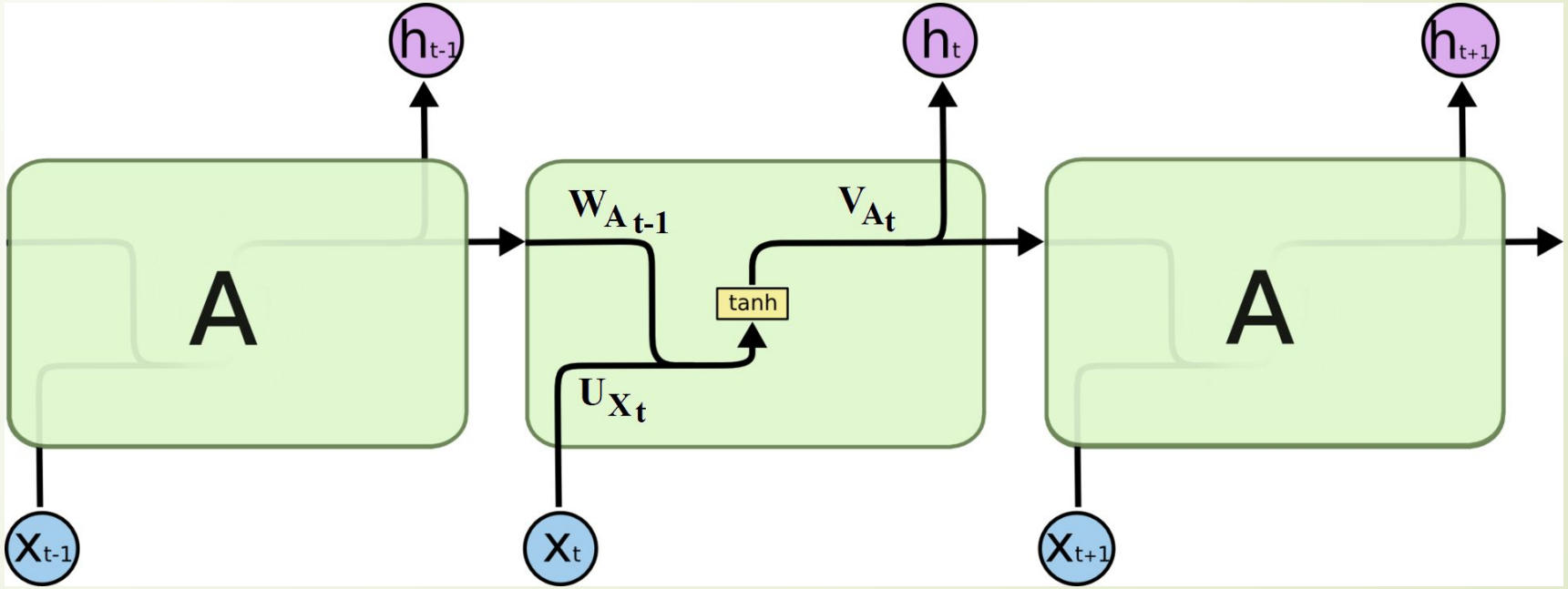
Cross-entropy loss function:

$$E_t\ (y_t, h_t) = -y_t \cdot \log(h_t)$$

$$E = \sum_t E_t(y_t, h_t)$$

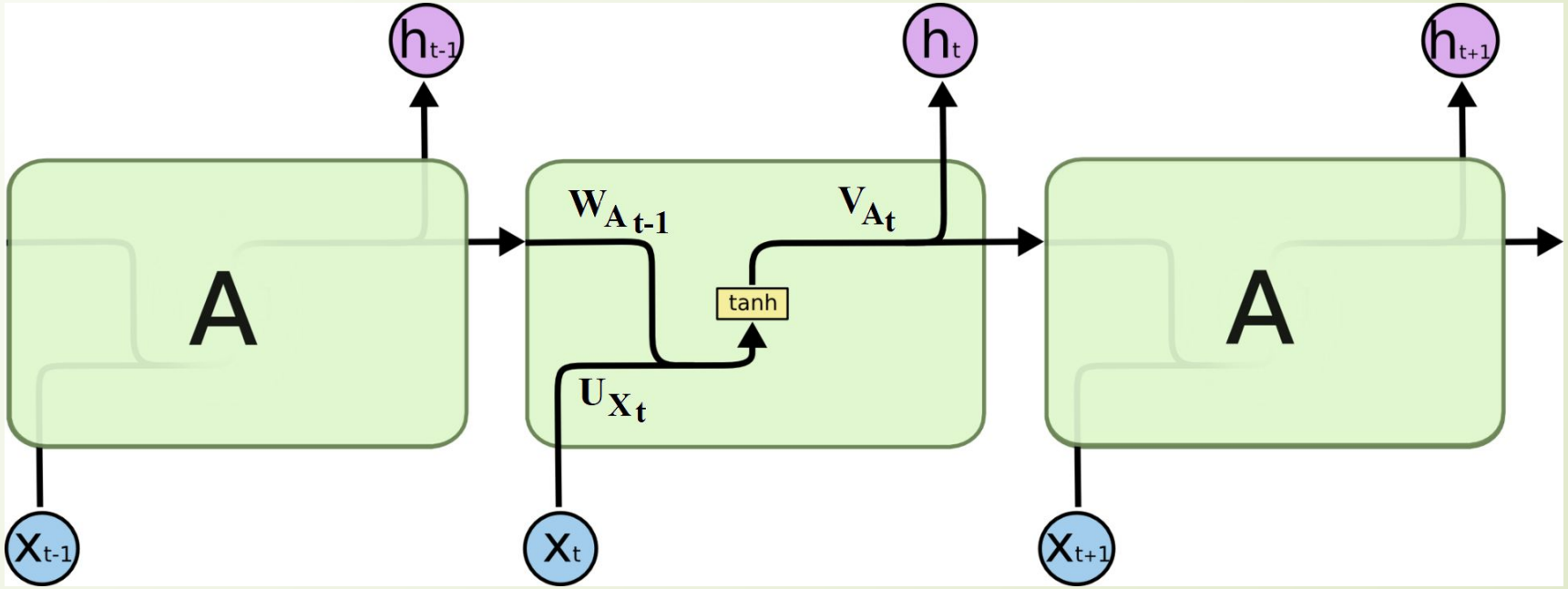
Goal - choose weight *V*, *U*, *W* to minimize *E*!

Backpropagation Through Time. Gradient in *V*



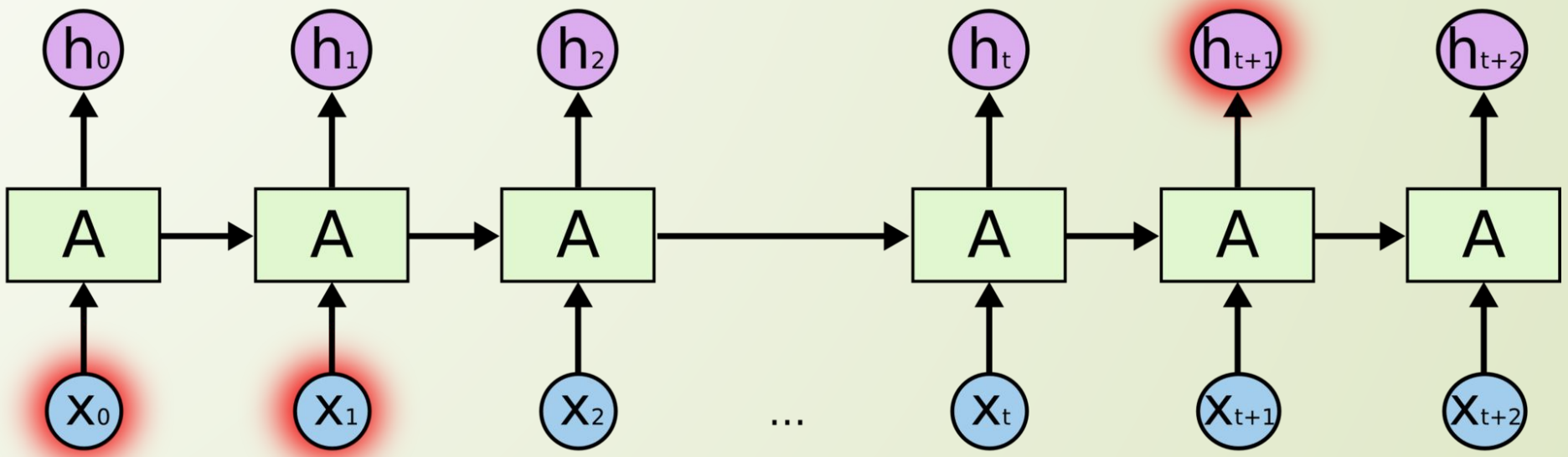
$$\frac{\partial E_t}{\partial V} = \frac{\partial E_t}{\partial E_t h_t} \cdot \frac{\partial E_t}{\partial V} = \frac{\partial E_t}{\partial h_t} \cdot \frac{\partial E_t}{\partial V_{A_t}} \cdot \frac{\partial V_{A_t}}{\partial V} = (h_t - y_t) \otimes A_t$$

Backpropagation Through Time. Gradient in *W*



$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial A_t} \cdot \frac{\partial A_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial A_t} \cdot \frac{\partial A_t}{\partial A_k} \cdot \frac{\partial A_k}{\partial W}$$

Again memory problems



Programmer Vasya like beer. Every night **he** comes to "Jonathan" and drink a couple of bottles.

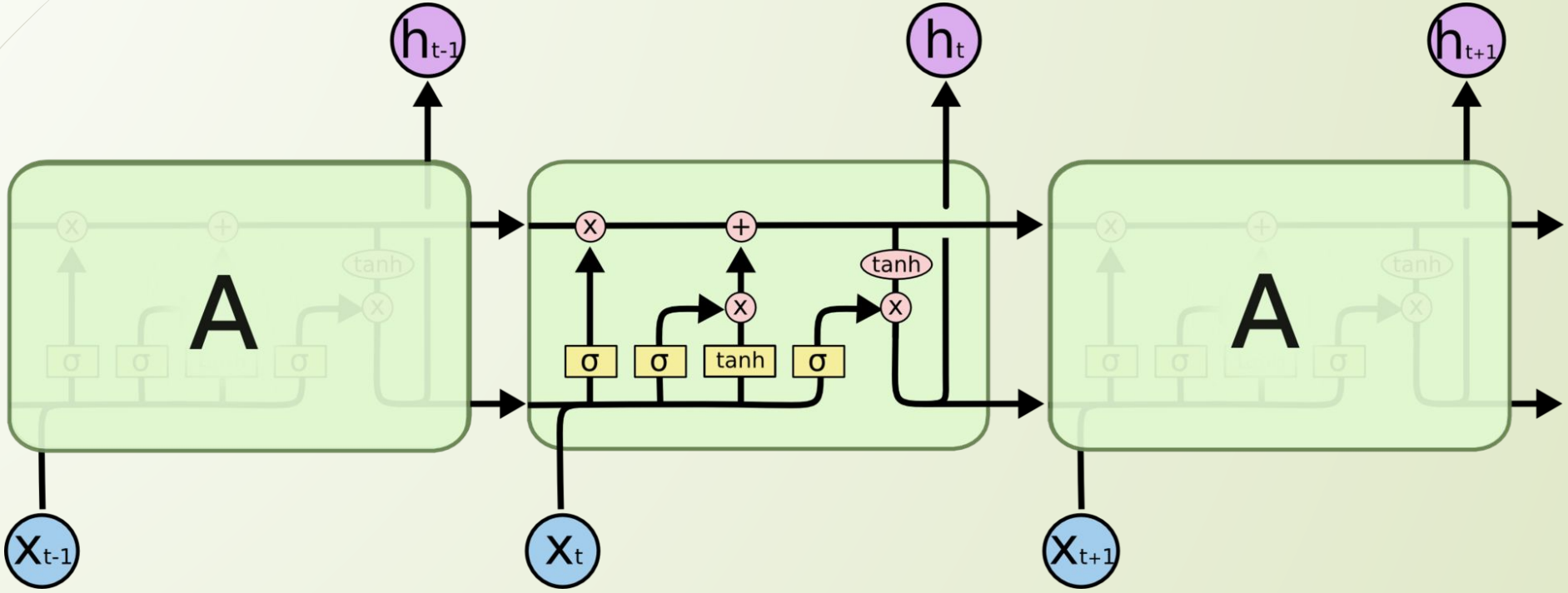
Long and Short memory

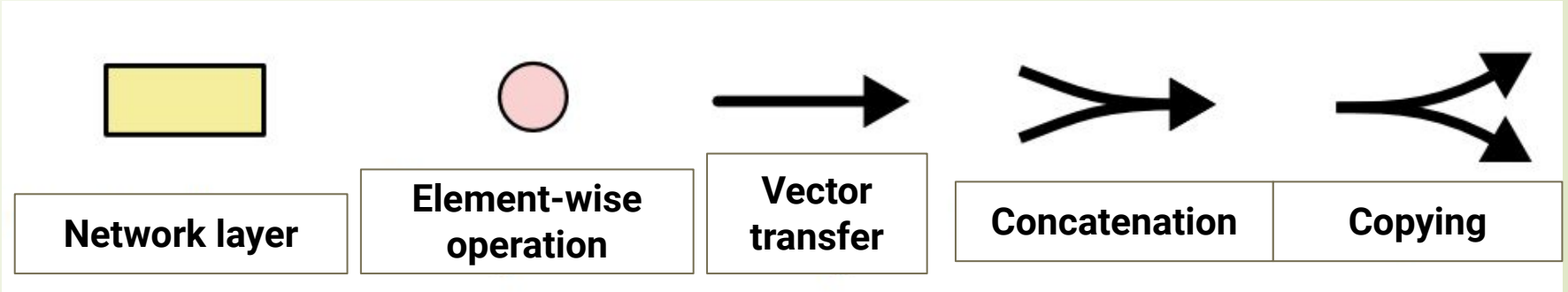
Long Short-Term Memory (LSTM)

1997

Sepp Hochreiter and *Jürgen Schmidhuber*

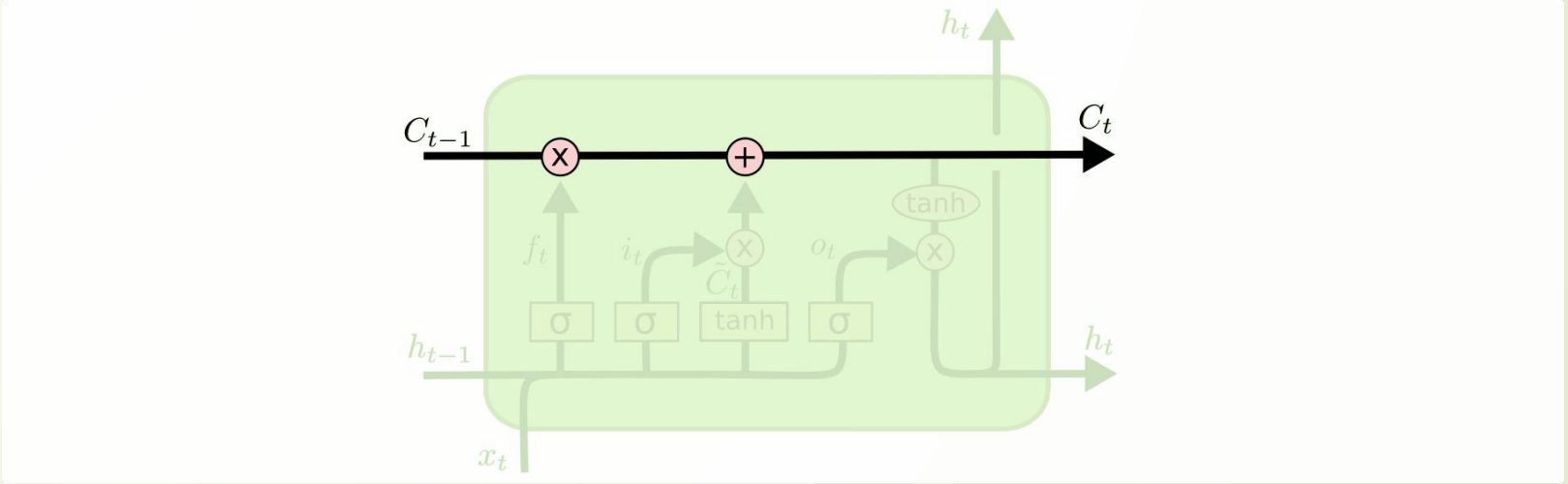
LSTM structure



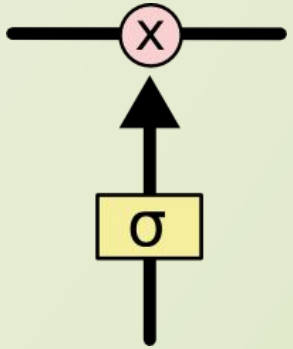


State cell

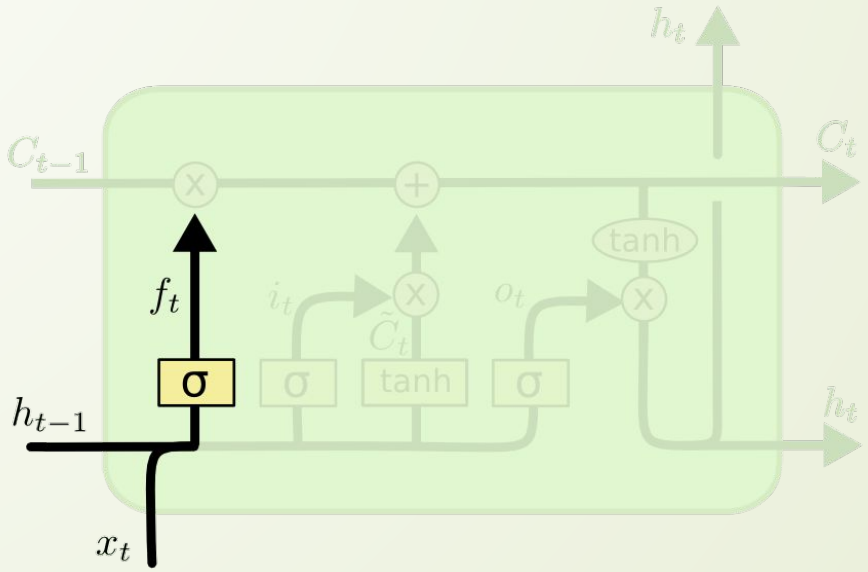
"Conveyor", passing through the whole cell.
Information can process it without changes. Or not.



Gates let us **delete information** according to some conditions.



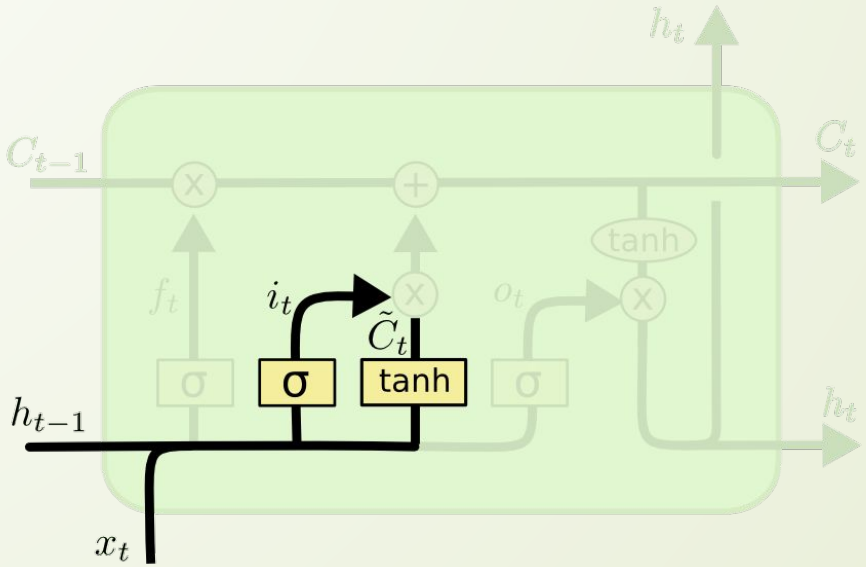
Step 1 in LSTM. Forget gate layer



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

What information to forget (delete from the state)?

Step 2 in LSTM. Input layer gate and tanh-layer

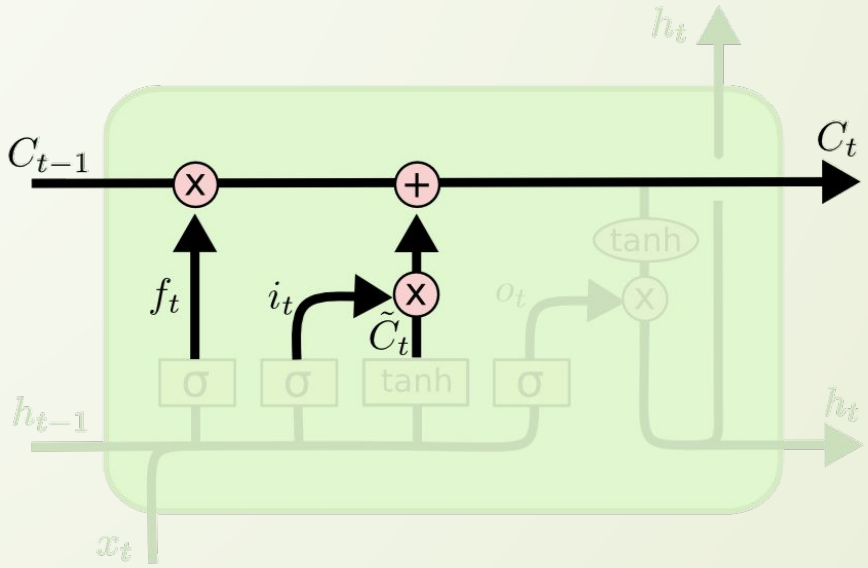


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

What information to add to the state?

Step 3 in LSTM. Final update of the state.



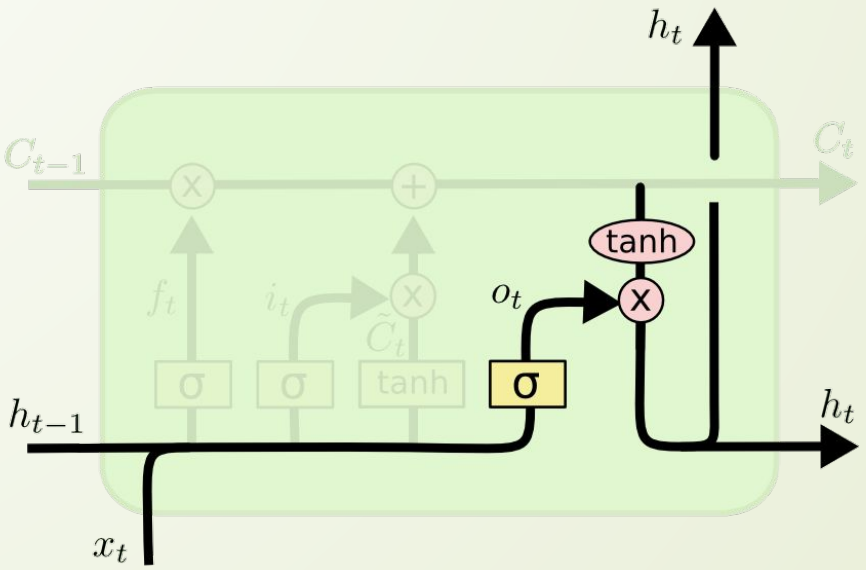
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Forget what forget gate layer decided to forget.

Add what input gate layer and tanh-layer decided to add.

Now state is changed.

Step 4 in LSTM. Output useful information from cell state.



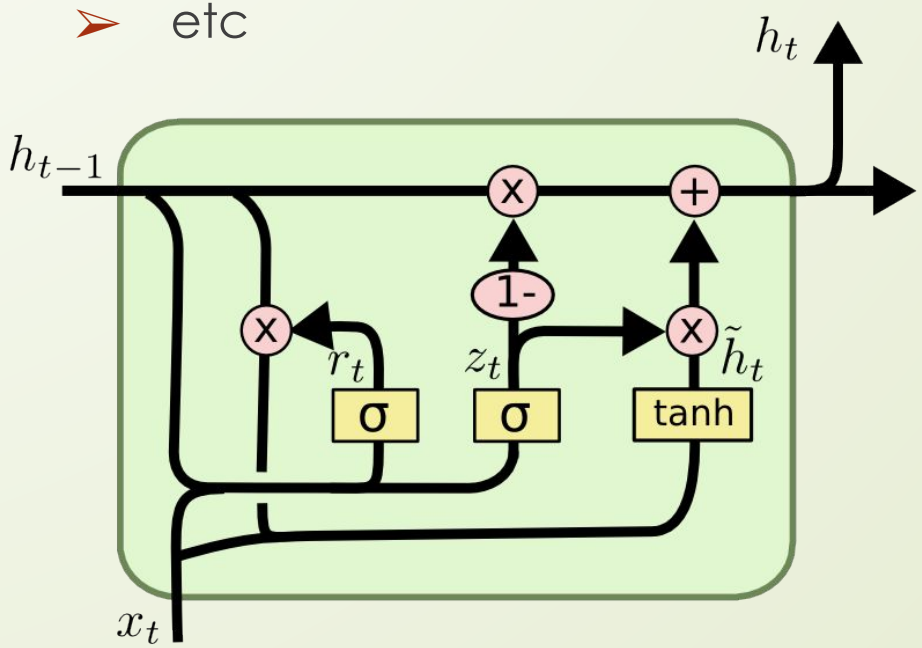
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

LSTM types. Gated recurrent units.

Gated recurrent units, GRU:

- forget gate layer and input gate layer are combined in one forget gate layer;
- Cell state is combined with internal state;
- etc



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

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

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

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

Useful reading

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

- **Understanding recurrent neural networks** (chapter 6.2 of *Francois Chollet's* book "**Deep learning with Python**")
  https://www.manning.com/books/deep-learning-with-python

  https://github.com/rtlatimer/Deep-Learning-with-Python/blob/master/Ch6.2_Understanding_Recurrent_Neural_Networks.ipynb

Example of using: named entity recognition (NER)

U.N. official Ekeus heads for Baghdad.

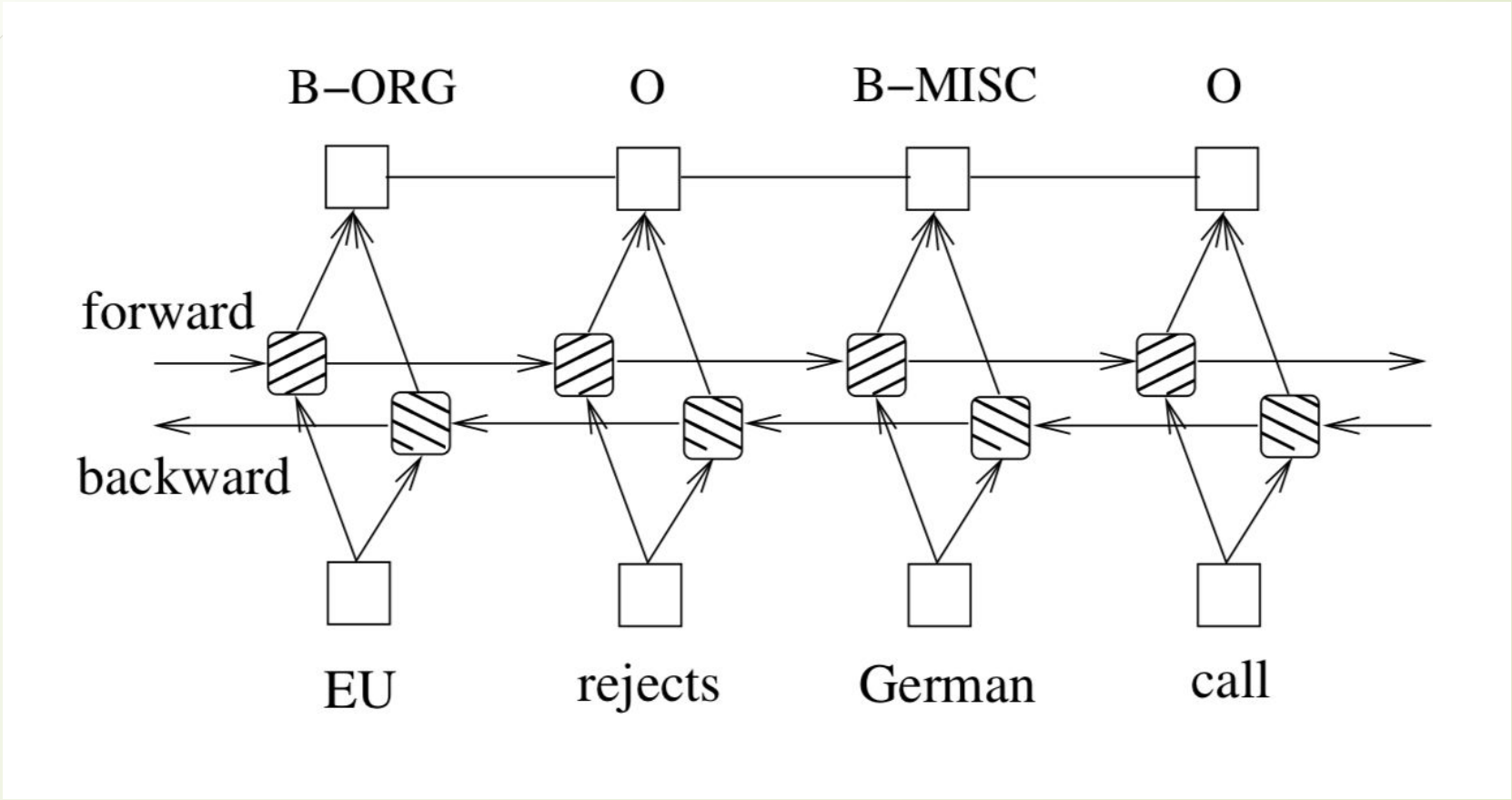
Organization *Person* *Location*

Tokenizing

Transforming to BIO labels

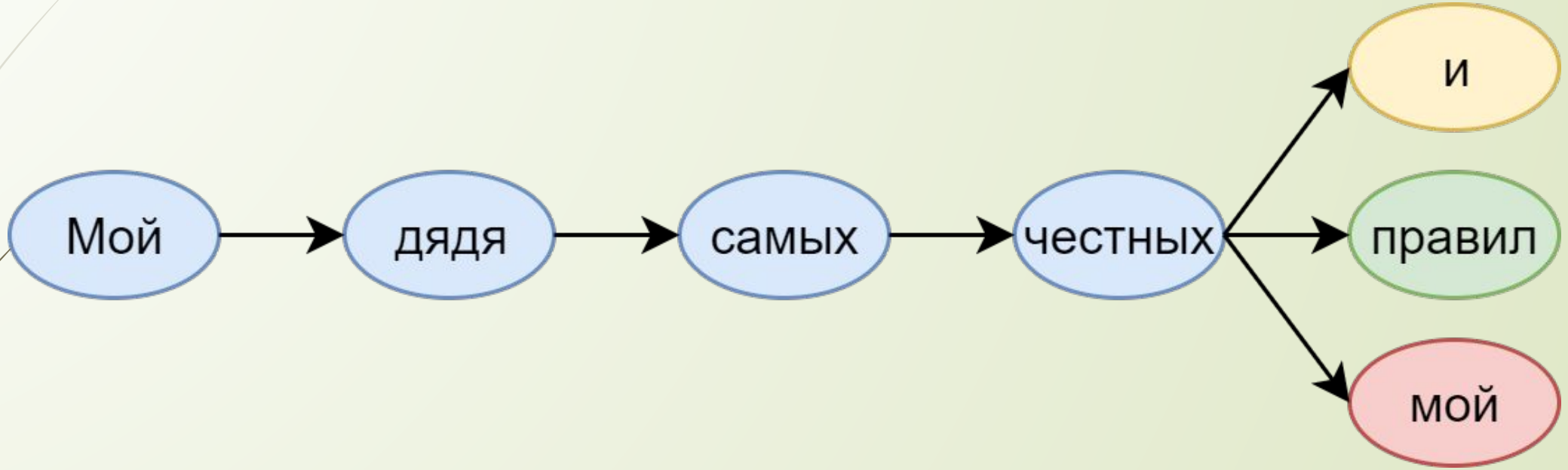
U	B-ORG
.	I-ORG
N	I-ORG
.	I-ORG
official	O
Ekeus	B-PER
heads	O
for	O
Baghdad	B-LOC
.	O

Bidirectional LSTM for NER



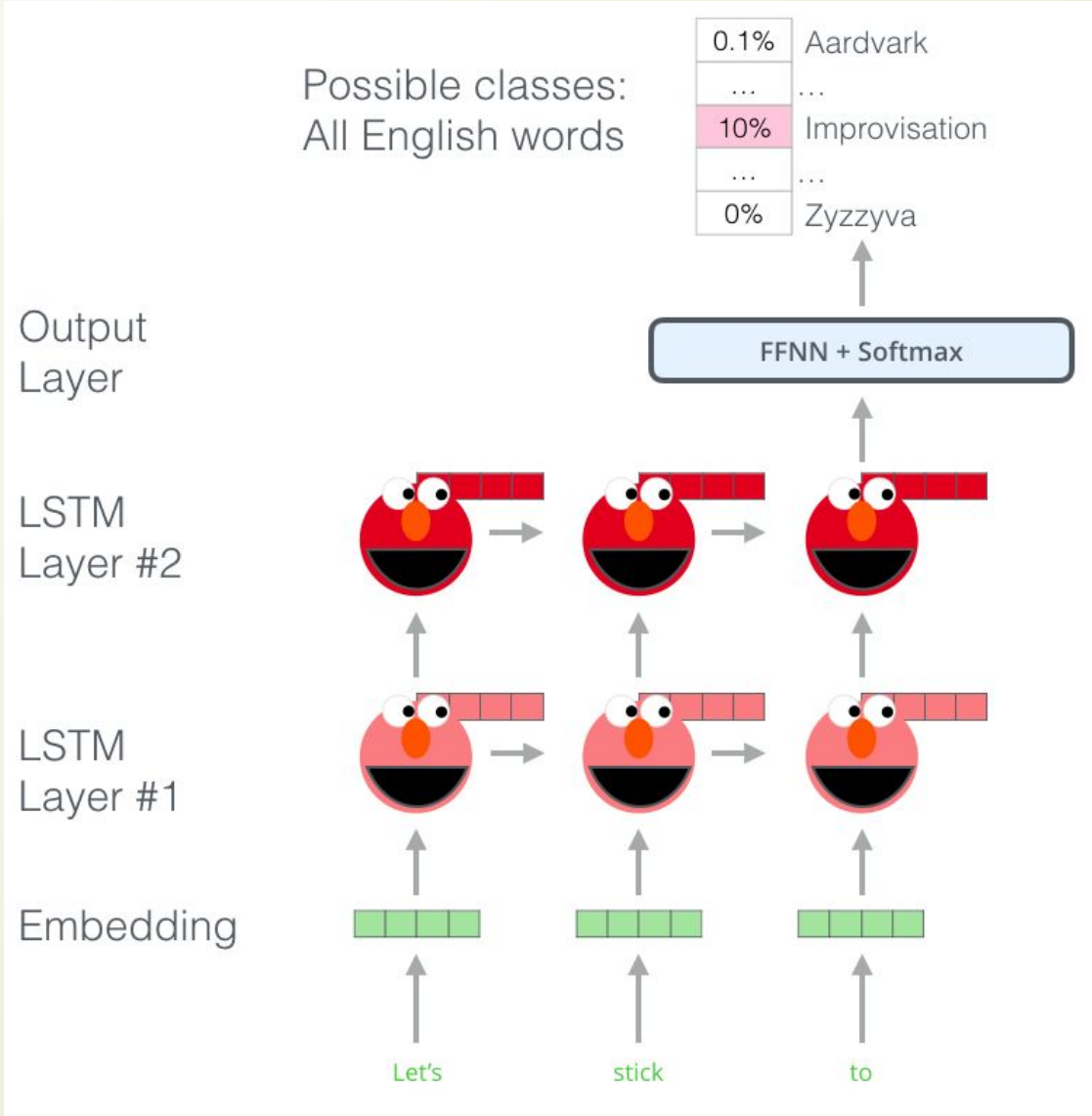
- **Guide To Sequence Tagging With Neural Networks In Python**
  https://www.depends-on-the-definition.com/guide-sequence-tagging-neural-networks-python/
- **Sequence Tagging With A LSTM-CRF**
  https://www.depends-on-the-definition.com/sequence-tagging-lstm-crf/

Another example of using: language modeling

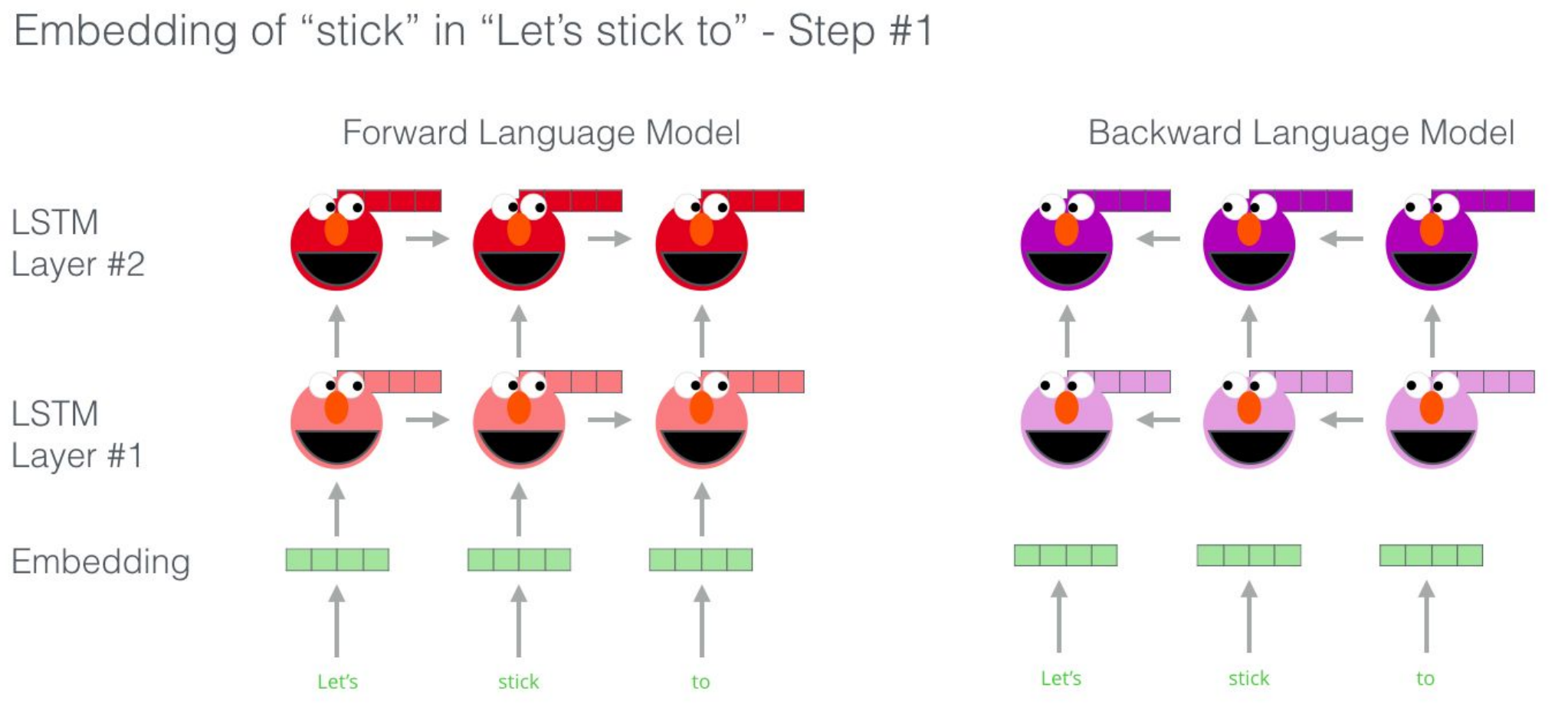


ELMo

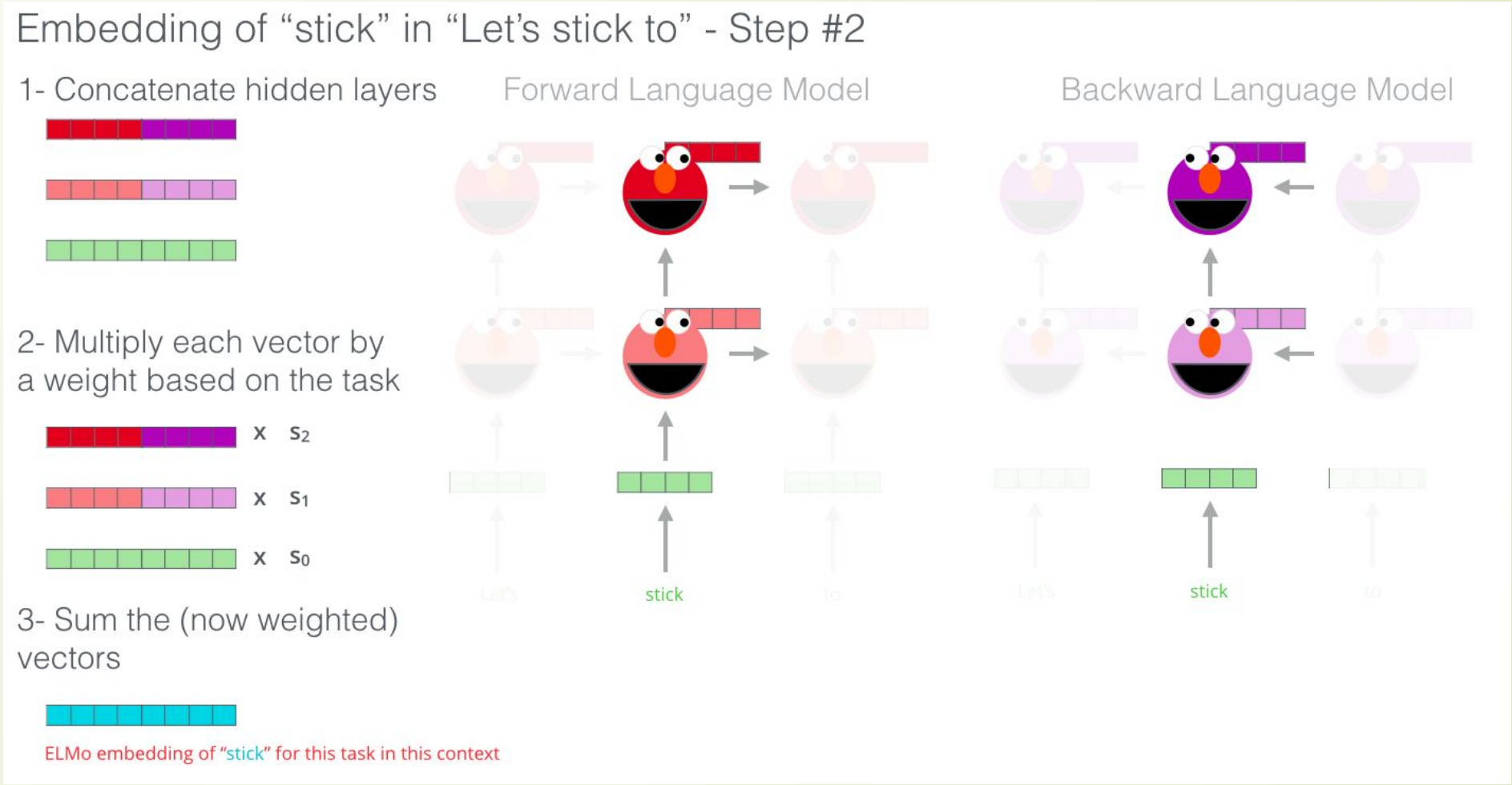
Embeddings from **L**anguage **Mo**dels



Bidirectional LSTM is base for ELMo



Concatenation and weighting of LSTM states



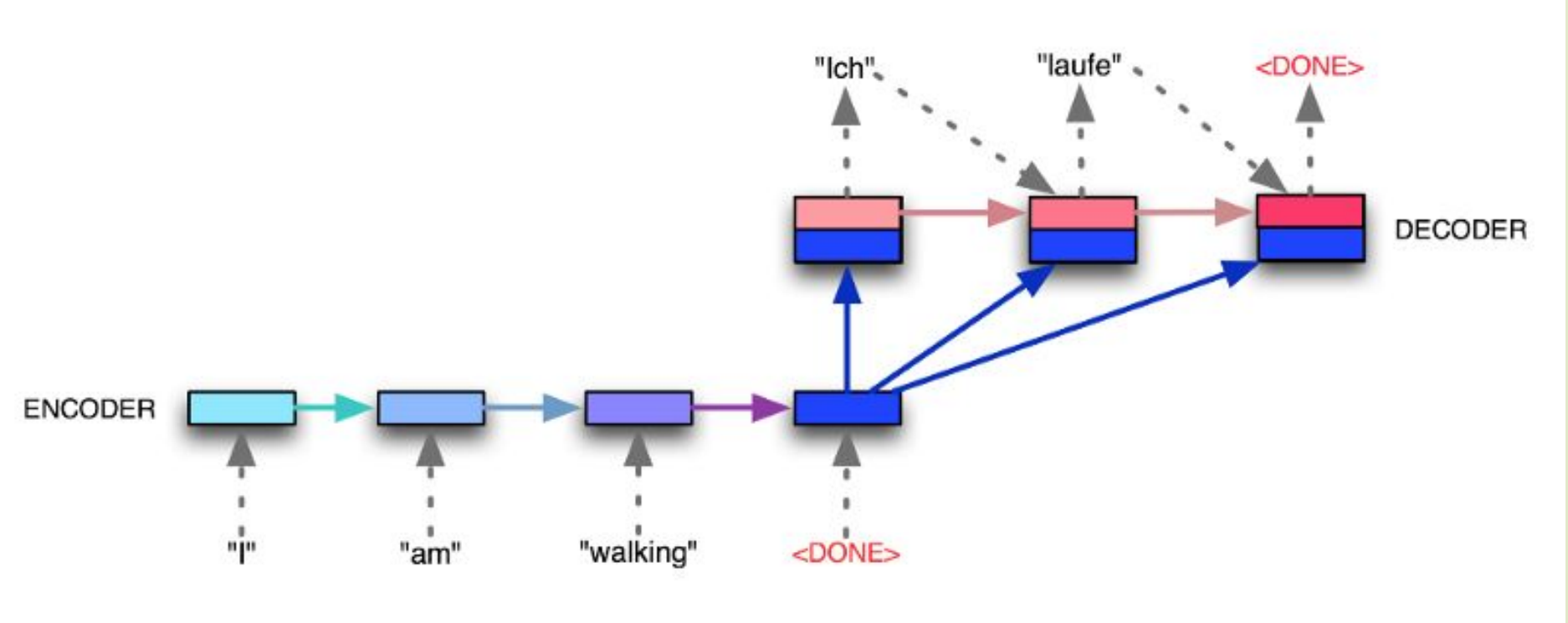
Useful reading

- *Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer*
  Deep contextualized word representations
  https://arxiv.org/abs/1802.05365

- **Example of using with Tensorflow**: https://tfhub.dev/google/elmo/2 (pretrained ELMo for English)

- **ELMo training and fine-tuning with DeepPavlov**: http://docs.deeppavlov.ai/en/latest/apiref/models/elmo.html (with pretrained models for Russian)

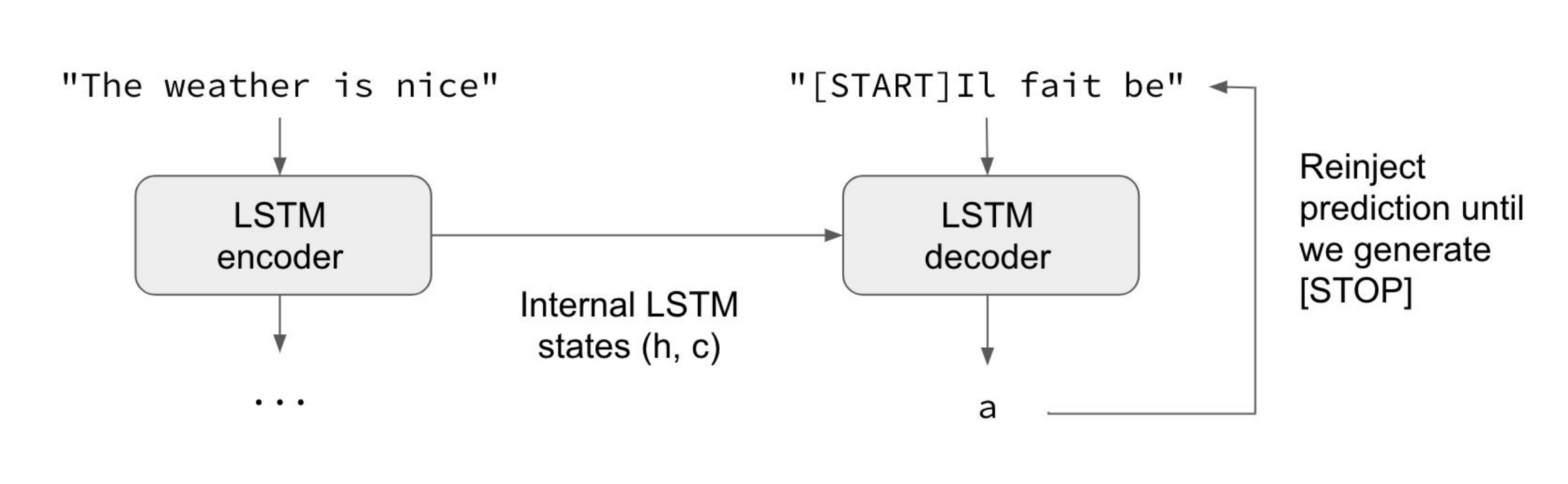
Sequence-to-sequence



Tasks for sequence-to-sequence models:

- Machine translation
- Text summarization
- Speech recognition (spectrums to letters)
- …

Sequence-to-sequence with LSTM

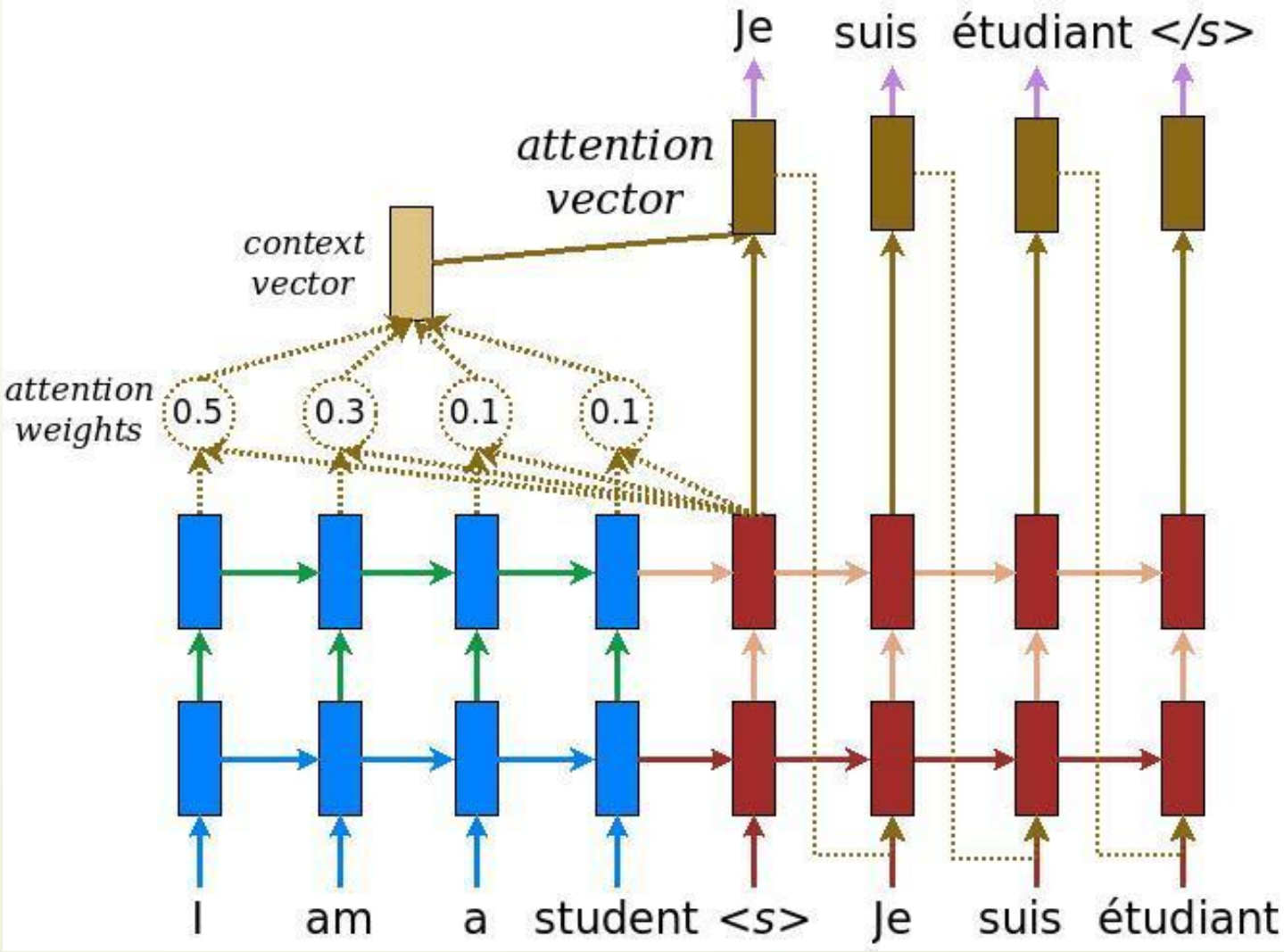


Useful reading

- *Ilya Sutskever, Oriol Vinyals, Quoc V. Le*
  Sequence to Sequence Learning with Neural Networks
  https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

- *Francois Chollet*
  A ten-minute introduction to sequence-to-sequence learning in Keras
  https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html

Sequence-to-sequence with attention

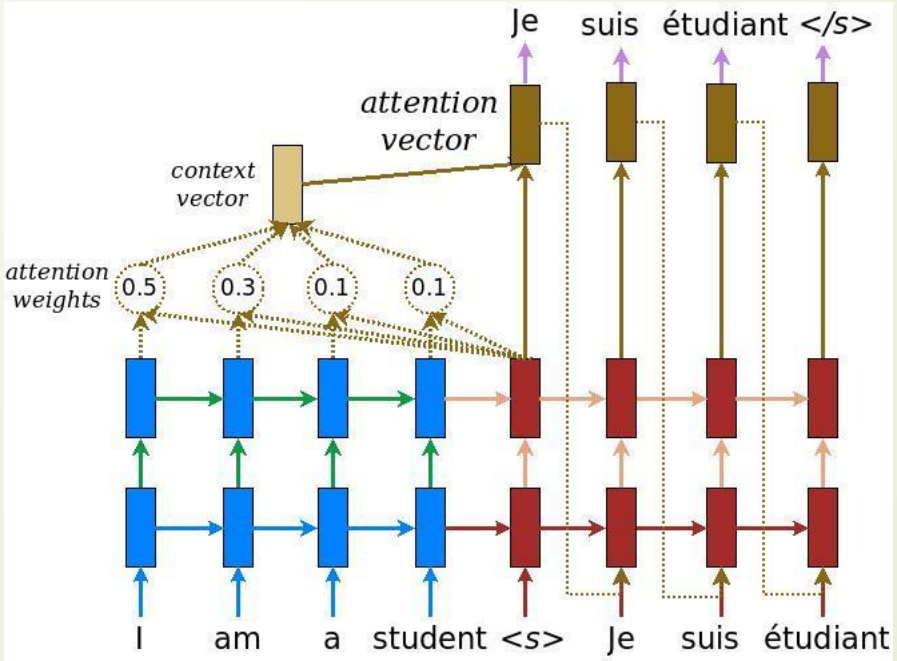


Sequence-to-sequence with attention

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \quad \text{[Attention weights]} \quad (1)$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \quad \text{[Context vector]} \quad (2)$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_{\boldsymbol{c}}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad \text{[Attention vector]} \quad (3)$$



$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \boldsymbol{W} \bar{\boldsymbol{h}}_s \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_1 \boldsymbol{h}_t + \boldsymbol{W}_2 \bar{\boldsymbol{h}}_s\right) \end{cases}$$

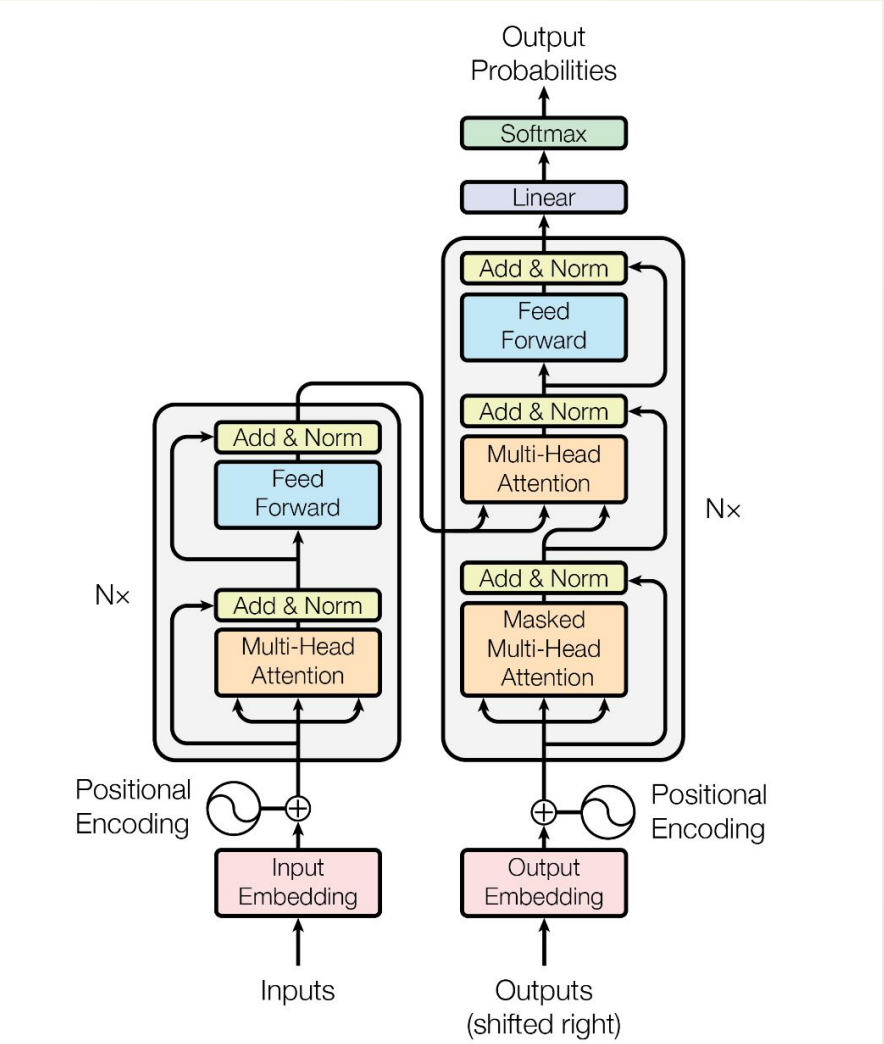
Useful reading

- *Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le* and others
  Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation
  https://arxiv.org/abs/1609.08144

- **Neural Machine Translation with Attention using Tensorflow**
  https://www.tensorflow.org/alpha/tutorials/text/nmt_with_attention

- *Guillaume Genthial*
  Seq2Seq with Attention and Beam Search
  https://guillaumegenthial.github.io/sequence-to-sequence.html

Sequence-to-sequence without LSTM! Attention Is All You Need



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin

Attention Is All You Need

https://arxiv.org/abs/1706.03762

Byte-pair-encoding (BPE) as quasi-morphemes

We start from the original line:

aaabdaaabac

After that we select the most frequency bigrams:

Z*abd****Z****abac* ***Z****=aa*

Then again we find the most frequency bigrams:

*Z****Y****dZ****Y****ac* ***Y****=ab Z=aa*

Over and over again:

X*d****X****ac* ***X****=ZY Y=ab Z=aa*

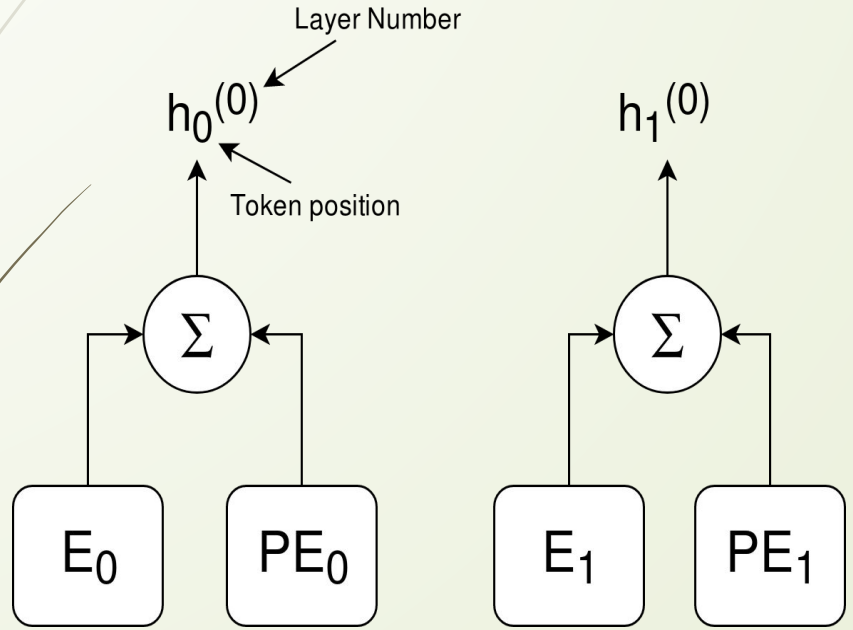
Tokenization example:

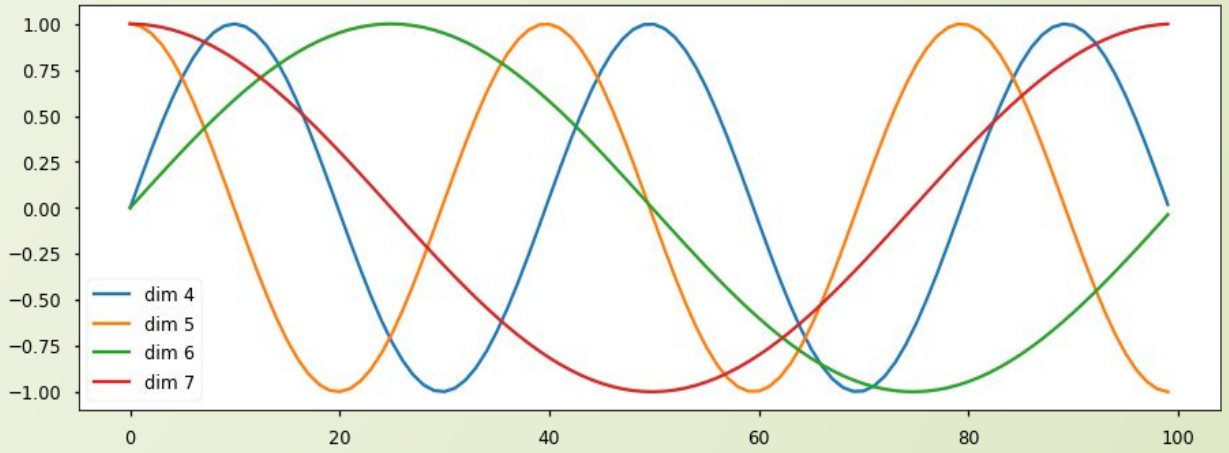
John Johanson 's house

john
johan
##son
'
s
house

Sennrich and others, **Neural Machine Translation of Rare Words with Subword Units**, https://arxiv.org/pdf/1508.07909.pdf

Positional embeddings

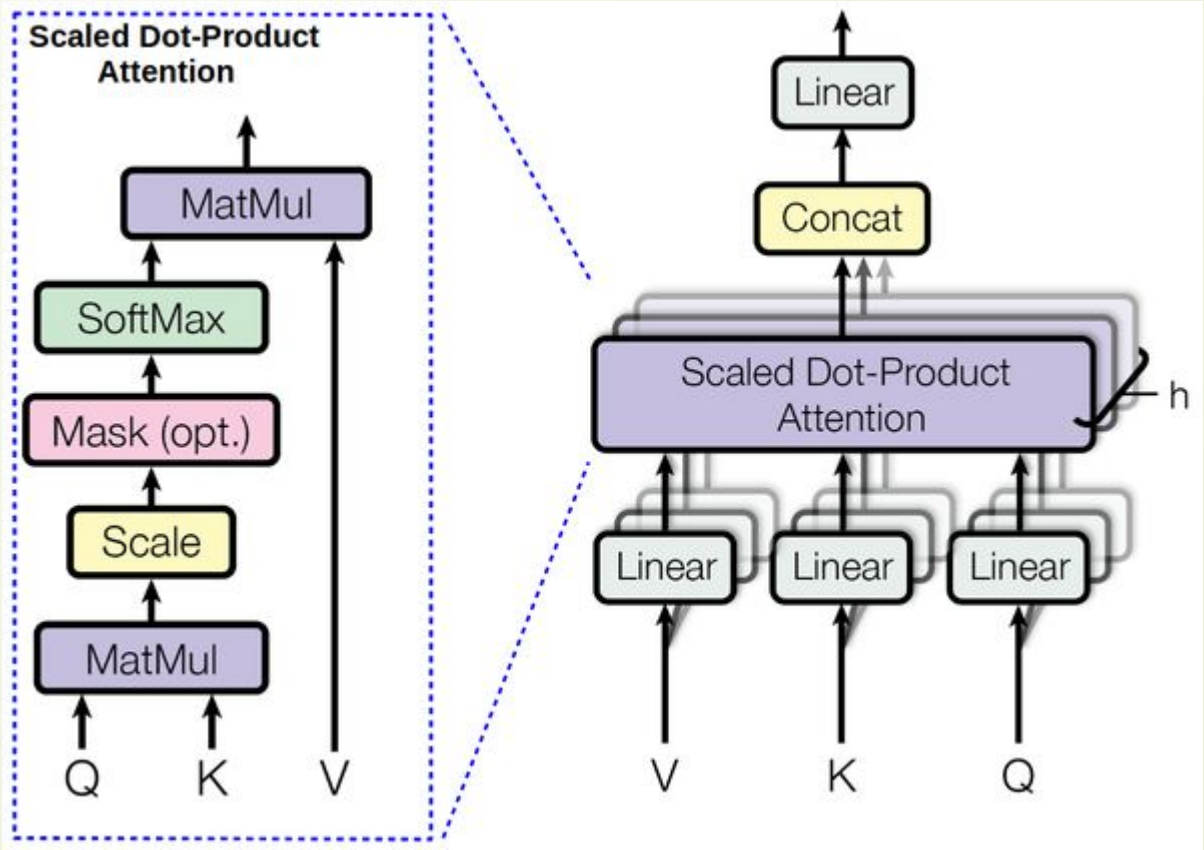




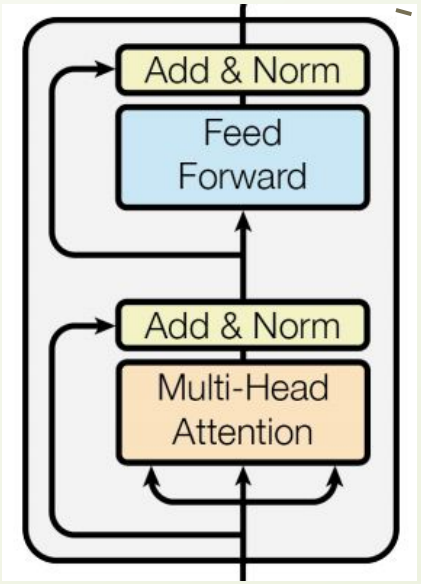
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

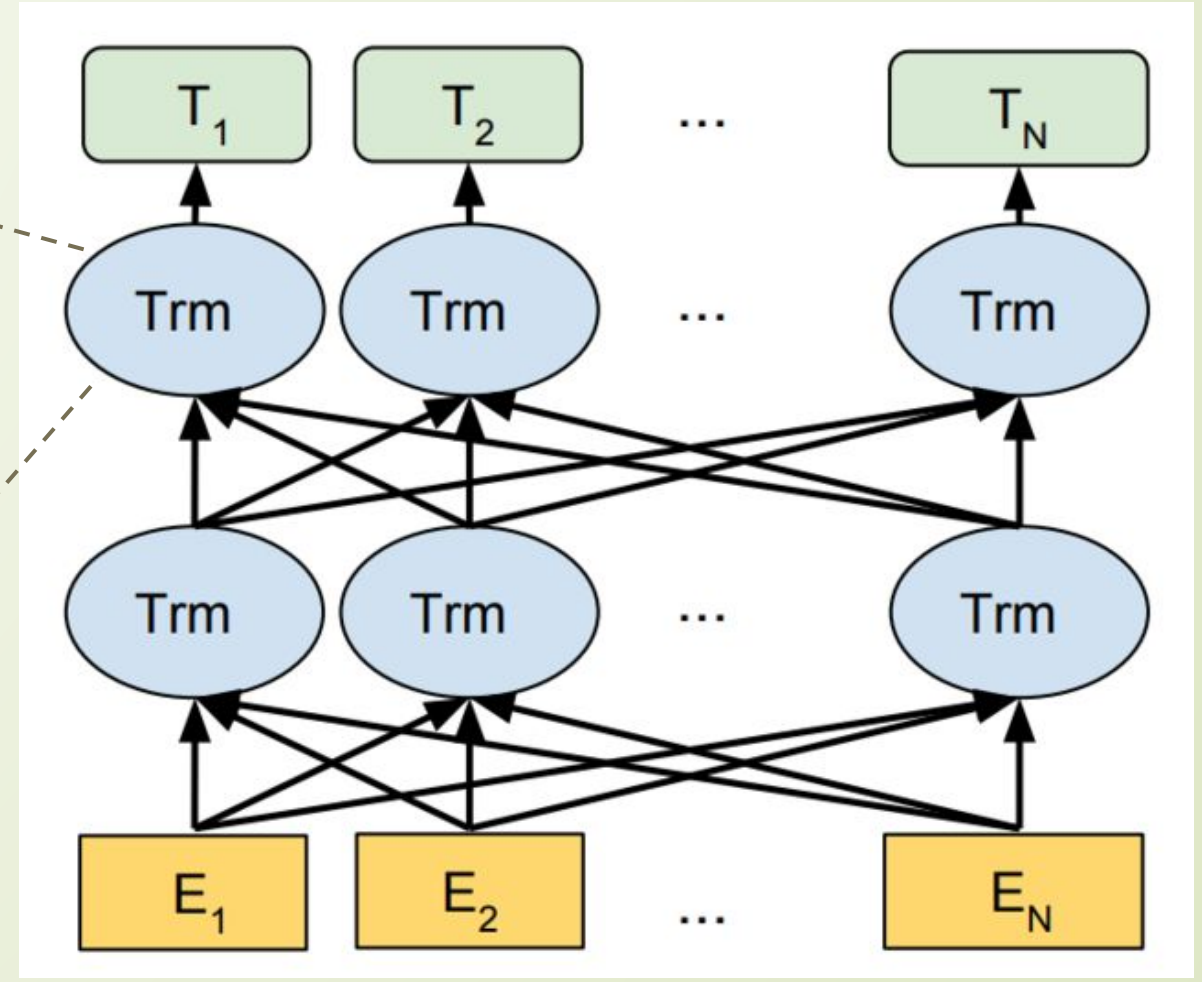
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Multi-Head Attention

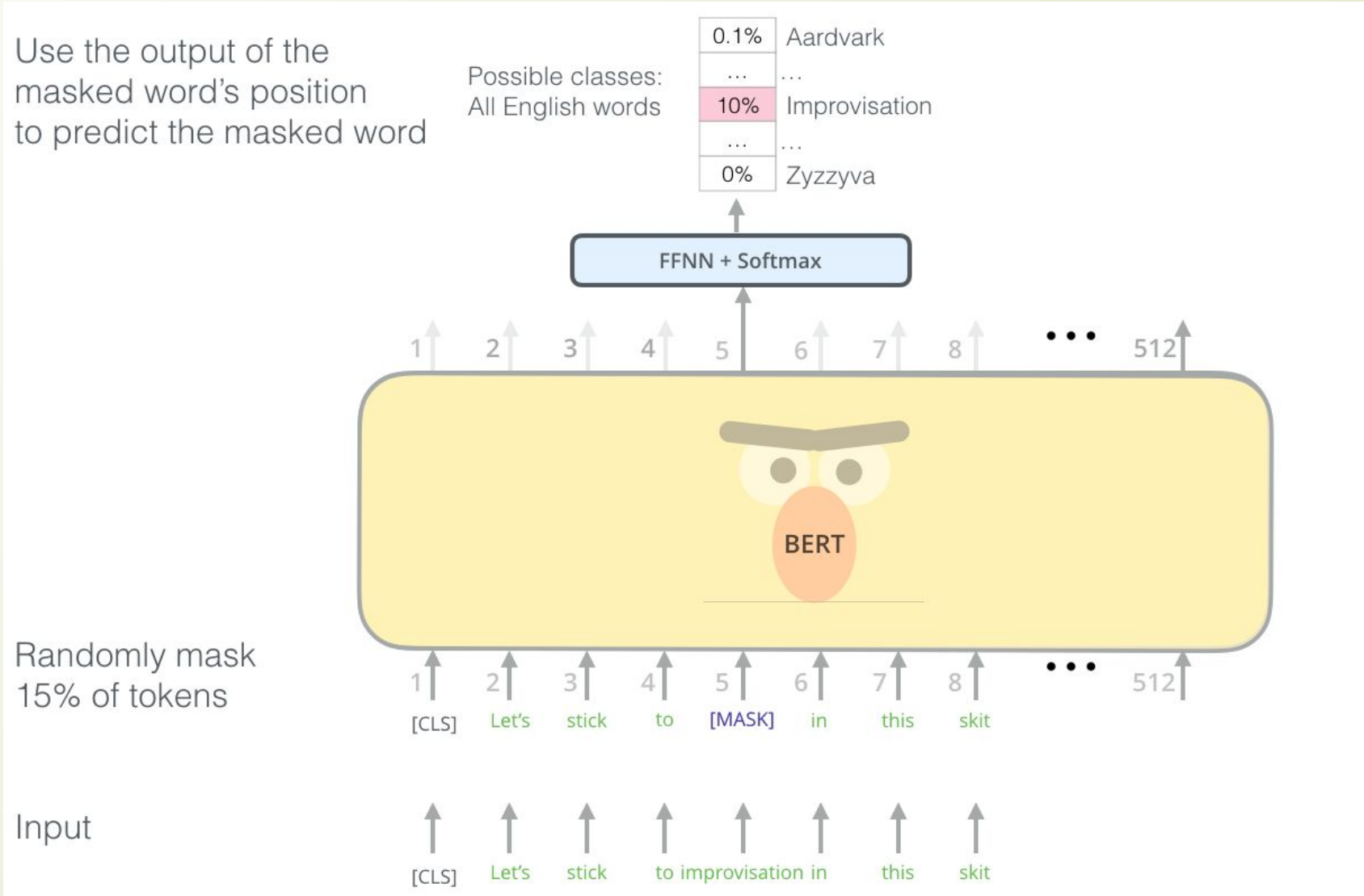


BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers

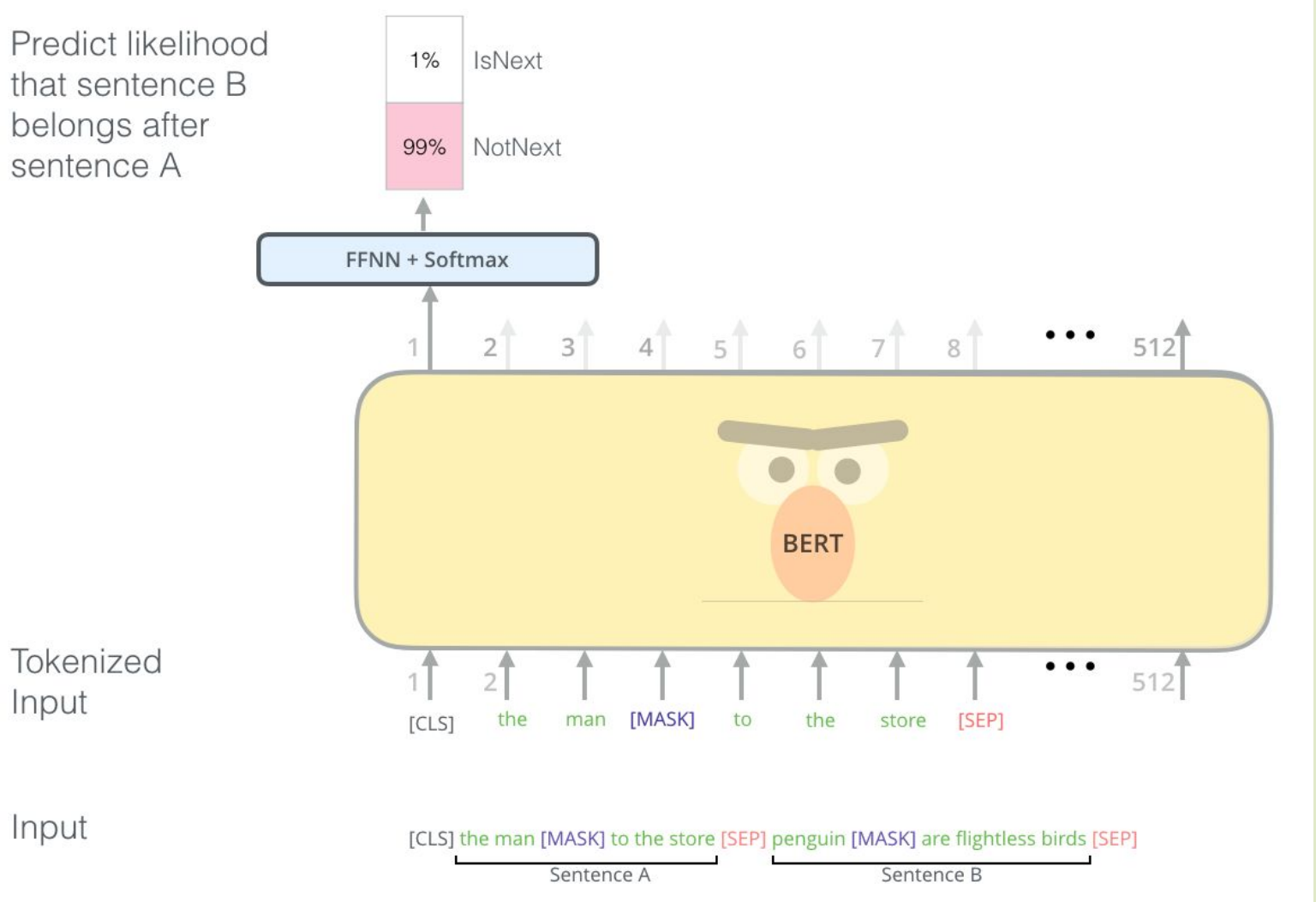




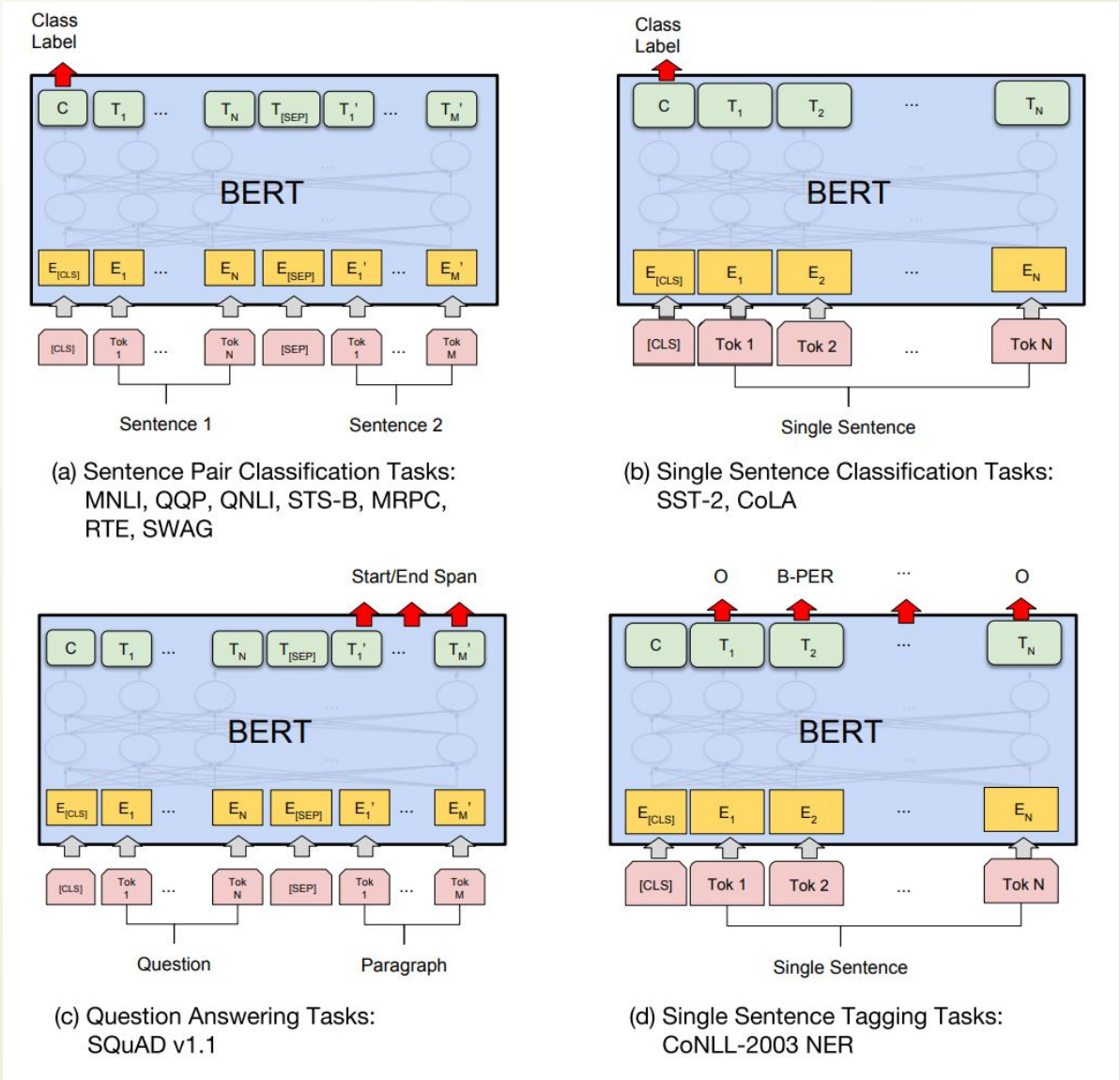
First task for BERT pre-training



Second task for BERT pre-training



Applying of BERT to various NLP tasks



Results for NLP tasks with BERT

Task type	**Type of measure**	**Previous state-of-the-art**	**BERT**
Single Sentence Classification (*SST-2 dataset for sentiment analysis*)	Accuracy	93,2%	94,9%
Sentence Pair Classification (*STS-B dataset for semantic textual similarity*)	Accuracy	81,0%	86,5%
Question Answering (*SQuAD 1.1 dataset*)	F1 score	91,7%	93,2%
Single Sentence Tagging (*CoNLL-2003 dataset for named entity recognition*)	F1 score	92,6%	92,8%

Useful reading

- *Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova*
  BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
  https://arxiv.org/abs/1810.04805

- *Jay Alammar,* **The Illustrated BERT, ELMo, and co.**
  http://jalammar.github.io/illustrated-bert/

- **TensorFlow code and pre-trained models for BERT**
  https://github.com/google-research/bert

- **Various implementations of BERT**:
  - Keras https://pypi.org/project/keras-bert
  - PyTorch https://github.com/huggingface/pytorch-pretrained-BERT

- **Named entity recognizers based on ELMo or BERT:**
  https://github.com/bond005/deep_ner

Thank you for attention!