Recurrent Neural Networks and Trasformers in Natural Language Processing

Università degli Studi di Palermo
dipartimento di ingegneria unipa

Laboratorio di Interazione Uomo-Macchina
CHILAB

Outline

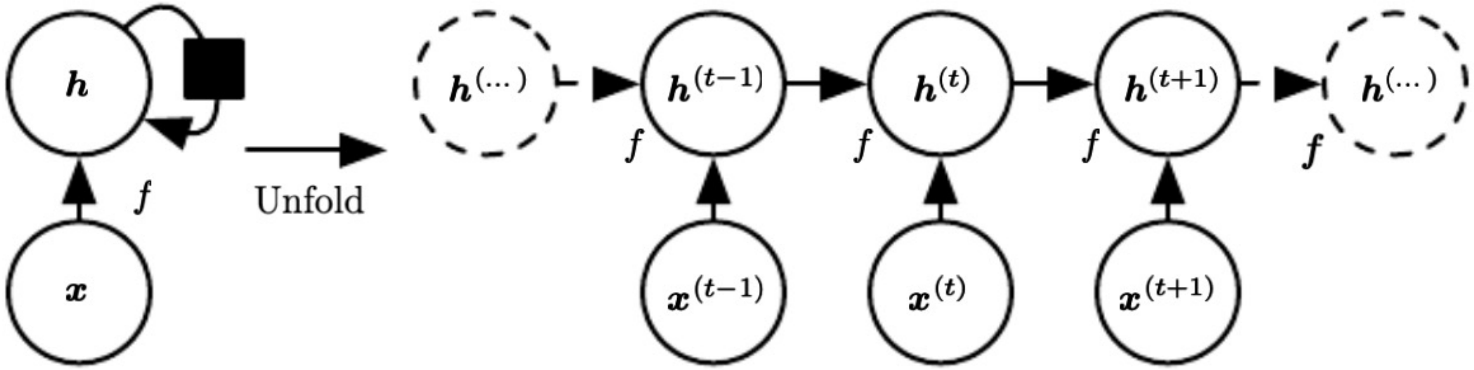
- Recurrent Neural Networks
- Long Short-Term Memory
- Transformers
- Natural Language Processing
- BERT

Recurrent Neural Networks (RNN)

- Neural network that is specialized for processing a sequence of values $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}$
- Each member of the output is a function of the previous members of the output. Each member of the output is produced using the same update rule applied to the previous outputs.

$$\boldsymbol{s}^{(t)} = f(\boldsymbol{s}^{(t-1)}; \boldsymbol{\theta})$$

Recurrent Neural Networks (RNN)



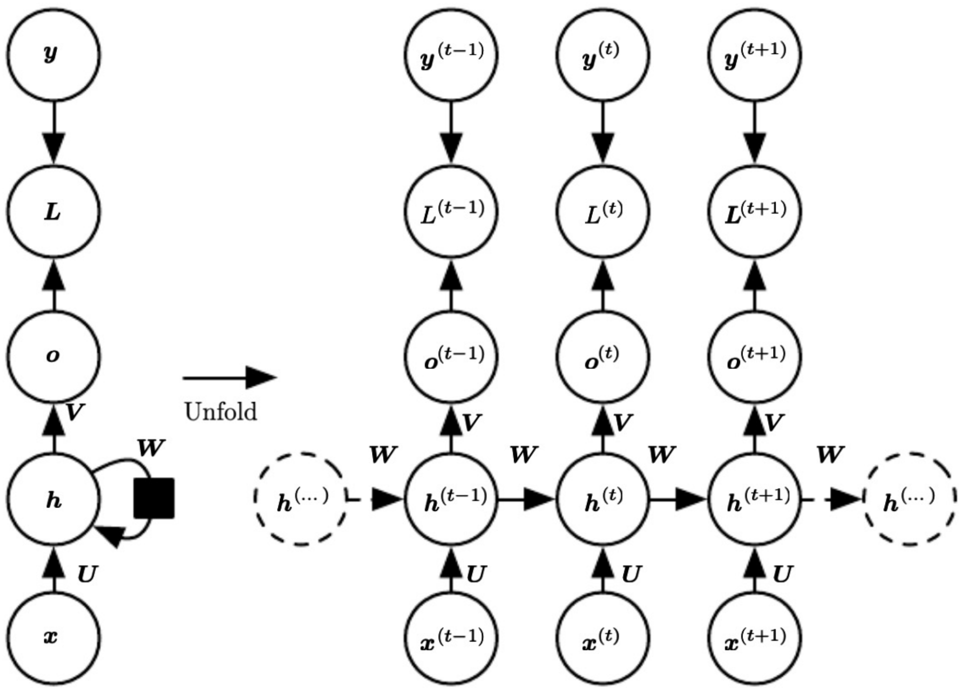
$$\boldsymbol{h}^{(t)} = f\left(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}\right)$$
$$\boldsymbol{h}^{(t-1)} = f(\boldsymbol{h}^{(t-2)}, \boldsymbol{x}^{(t-1)}; \boldsymbol{\theta})$$
$$\ldots$$
$$\boldsymbol{h}^{(1)} = f(\boldsymbol{h}^{(0)}, \boldsymbol{x}^{(1)}; \boldsymbol{\theta})$$

Goodfellow I., Bengio Y. and Courville A., Deep Learning, http://www.deeplearningbook.org

Recurrent Neural Networks (RNN)

Recurrent networks that produce an output at each time step and have recurrent connections between hidden units.

$\boldsymbol{x} \rightarrow$ input sequence
$\boldsymbol{o} \rightarrow$ output values
$\boldsymbol{L} \rightarrow$ loss
$\boldsymbol{h} \rightarrow$ hidden unit
$\boldsymbol{y} \rightarrow$ground truth value

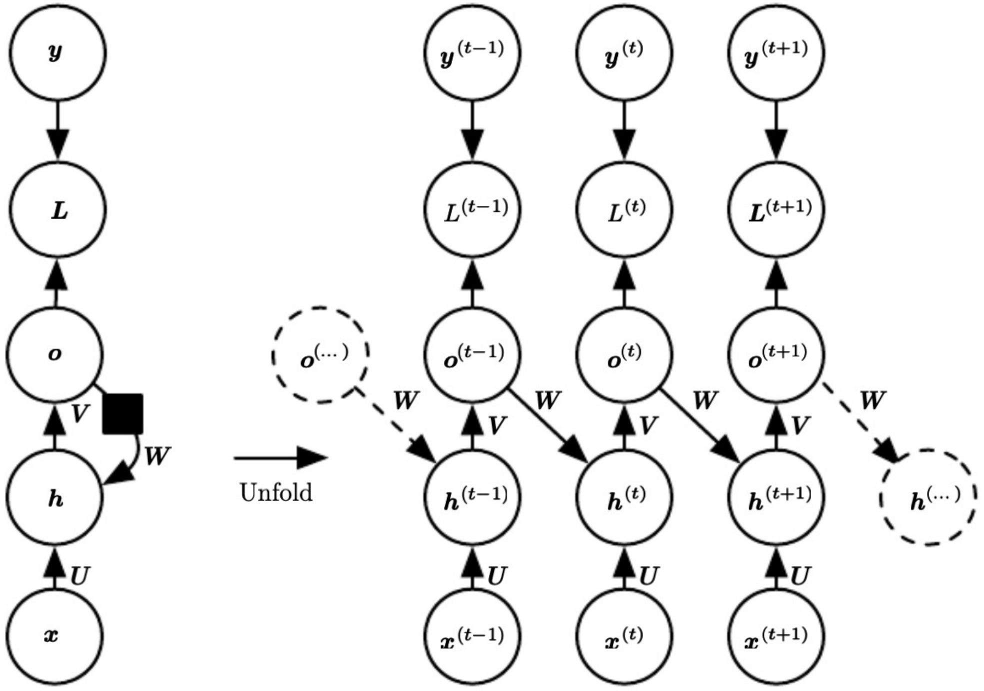


Goodfellow I., Bengio Y. and Courville A., Deep Learning, http://www.deeplearningbook.org

Recurrent Neural Networks (RNN)

Recurrent networks that produce an output at each time step and have recurrent connections only from the output at one time step to the hidden units at the next time step.

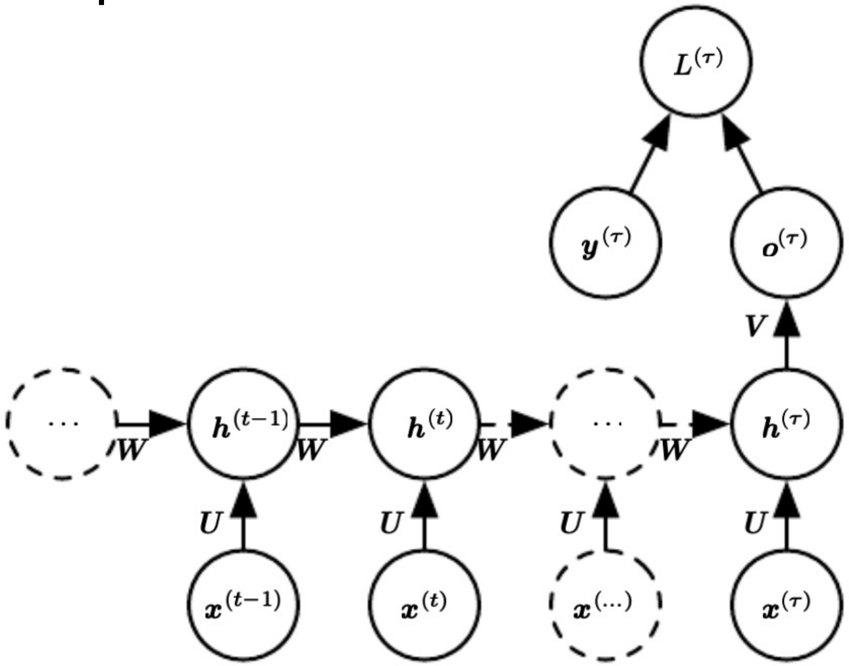
$\boldsymbol{x} \rightarrow$ input sequence
$\boldsymbol{o} \rightarrow$ output values
$\boldsymbol{L} \rightarrow$ loss
$\boldsymbol{h} \rightarrow$ hidden unit
$\boldsymbol{y} \rightarrow$ground truth value



Goodfellow I., Bengio Y. and Courville A., Deep Learning, http://www.deeplearningbook.org

Recurrent Neural Networks (RNN)

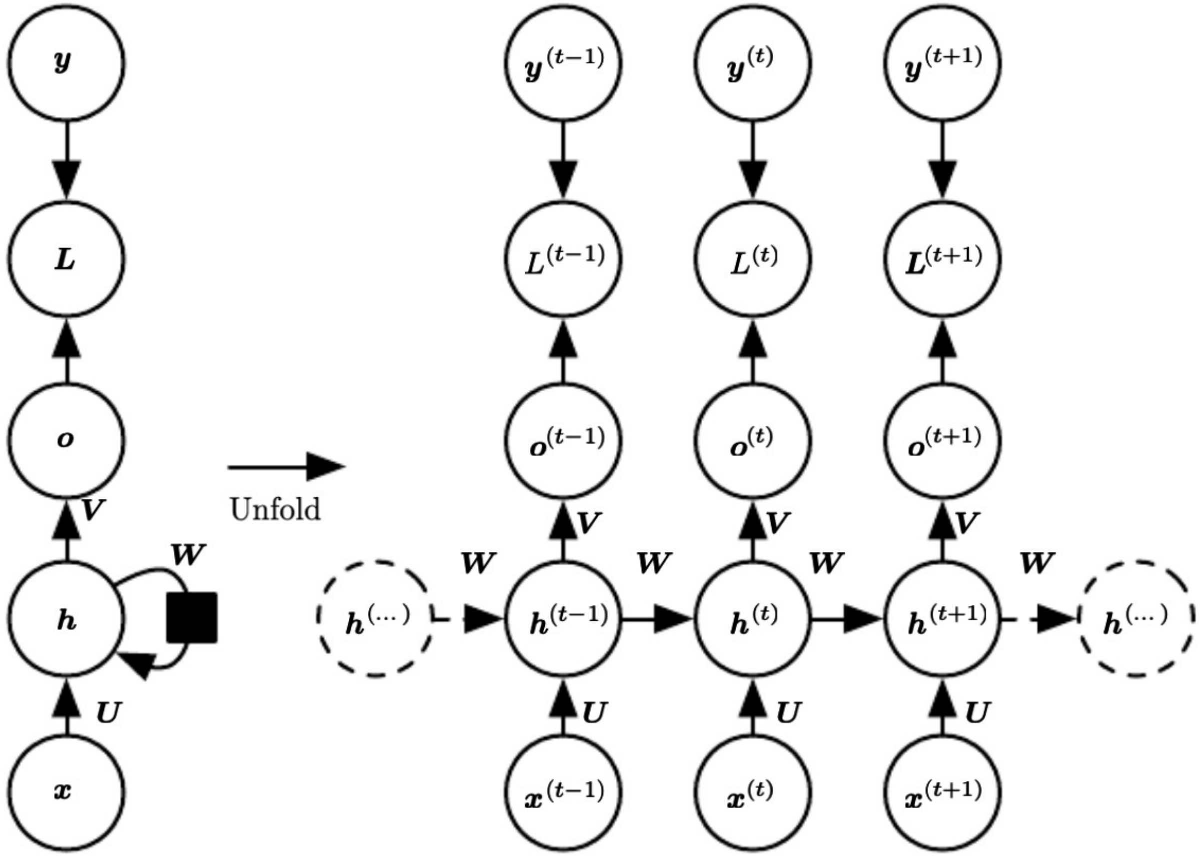
Recurrent networks with recurrent connections between hidden units, that read an entire sequence and then produce a single output.



$\boldsymbol{x} \rightarrow$ input sequence
$\boldsymbol{o} \rightarrow$ output values
$\boldsymbol{L} \rightarrow$ loss
$\boldsymbol{h} \rightarrow$ hidden unit
$\boldsymbol{y} \rightarrow$ground truth value

Goodfellow I., Bengio Y. and Courville A., Deep Learning, http://www.deeplearningbook.org

Recurrent Neural Networks (RNN)



$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}$$

$$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{a}^{(t)})$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}$$

$$\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}(\boldsymbol{o}^{(t)})$$

$$\boldsymbol{L}\left(\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(\tau)}\}\right) =$$
$$= \sum_t \boldsymbol{L}^{(t)} =$$
$$= -\sum_t \log p_{model}\left(y^{(t)} \middle| \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(t)}\}\right)$$

Goodfellow I., Bengio Y. and Courville A., Deep Learning, http://www.deeplearningbook.org

Recurrent Neural Networks (RNN)

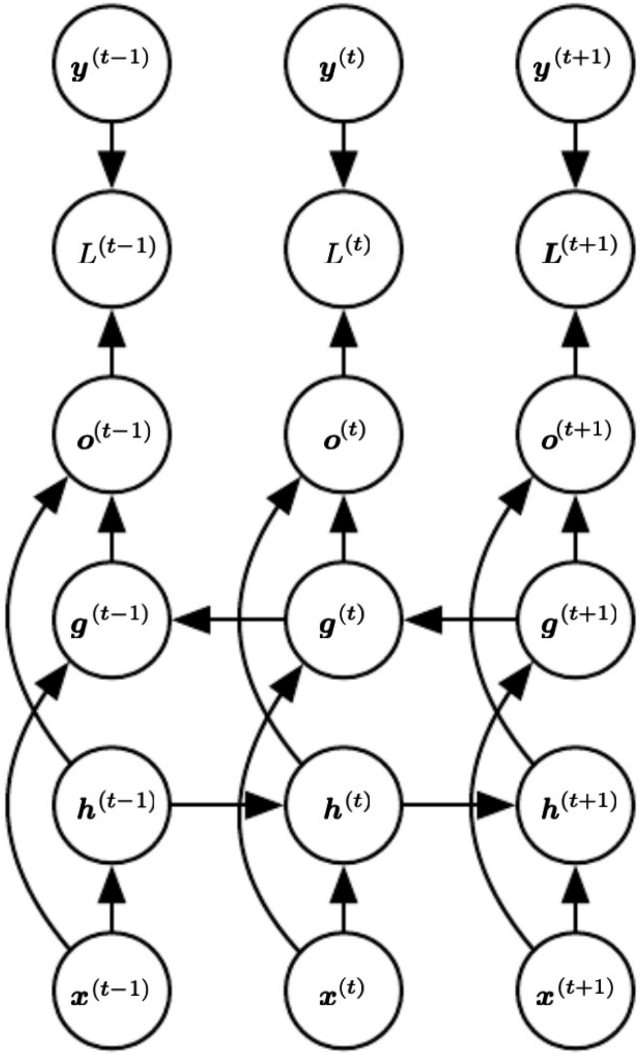
Training:

- Back-propagation through time (BPTT)
- Teacher forcing

Difficult in training

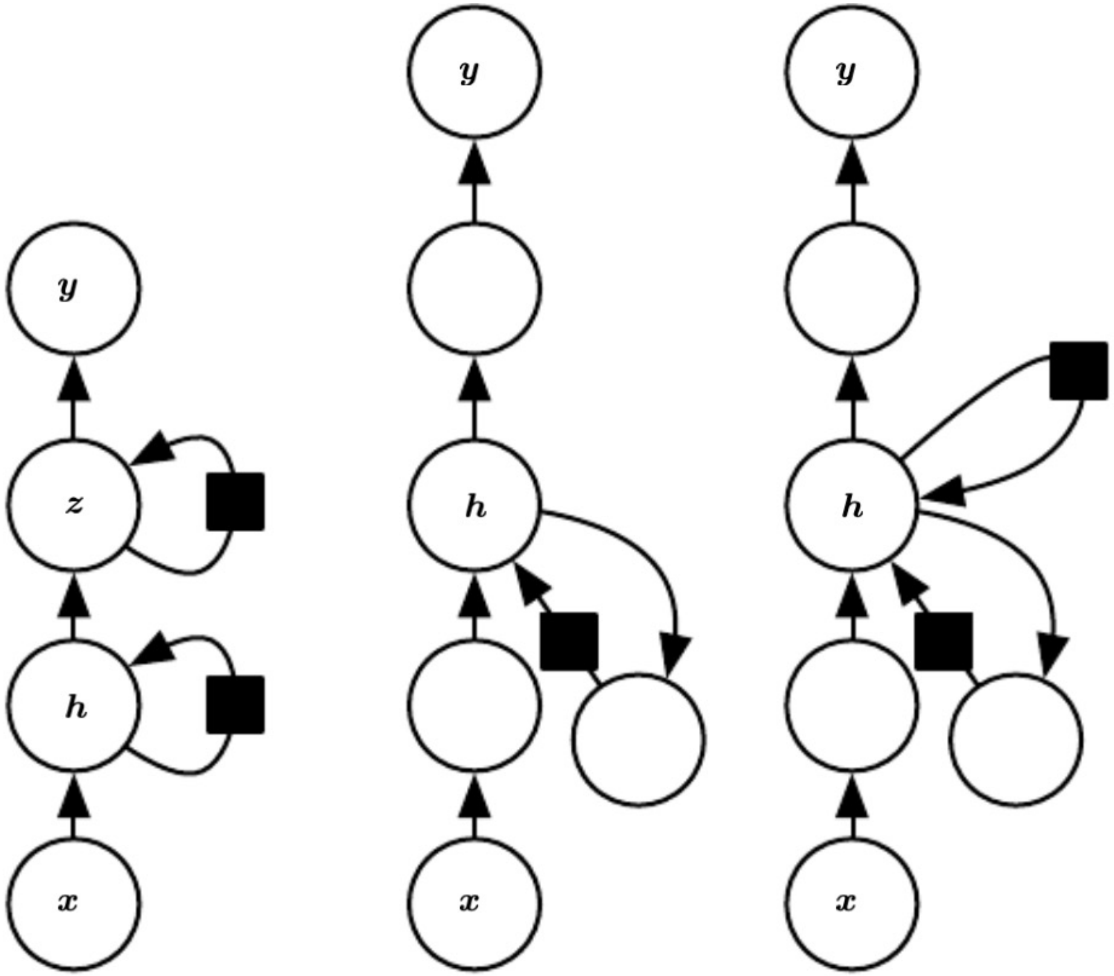
- Attention window
- Reduced number of parameters

BiDirectional RNN



Goodfellow I., Bengio Y. and Courville A., Deep Learning, http://www.deeplearningbook.org

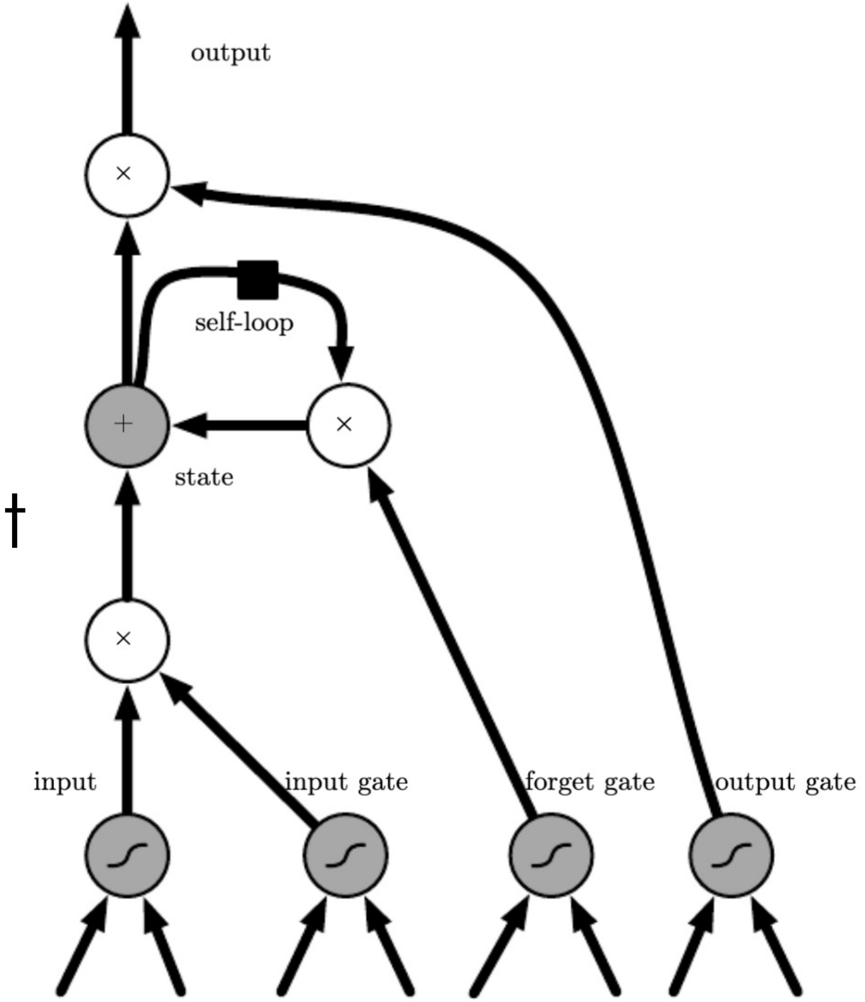
Deep Recurrent Networks



Goodfellow I., Bengio Y. and Courville A., Deep Learning, http://www.deeplearningbook.org

Long Short-Term Memory (LSTM)

LSTM learn long-term dependencies more easily than the simple recurrent architectures.

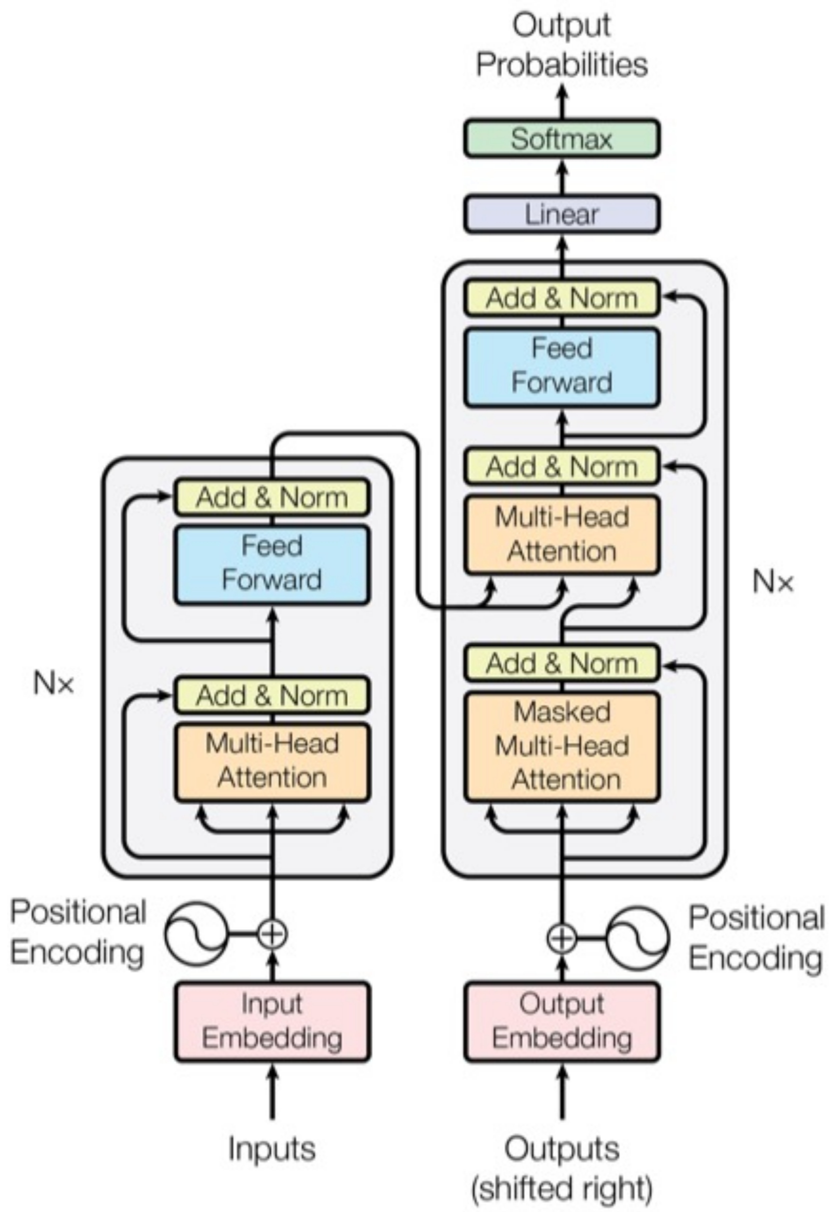


Goodfellow I., Bengio Y. and Courville A., Deep Learning, http://www.deeplearningbook.org

Transformers

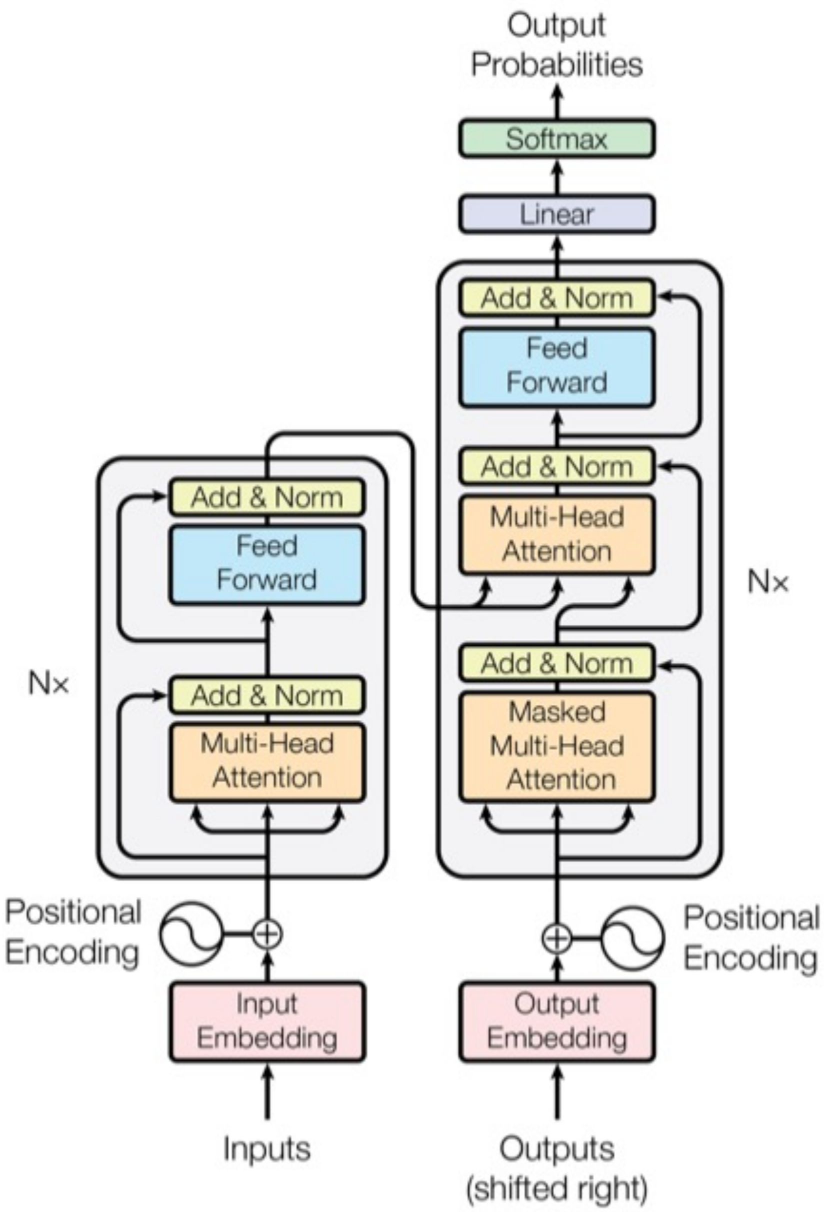
Model architecture delete recurrence and relying entirely on an attention mechanism to draw global dependencies between input and output.

$$(\boldsymbol{z}_1, \dots, \boldsymbol{z}_n) = \text{encoder}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$$
$$(\boldsymbol{y}_1, \dots, \boldsymbol{y}_m) = \text{decoder}(\boldsymbol{z}_1, \dots, \boldsymbol{z}_n)$$



Transformers

- Encoder
  - Multi-head self-attention mechanism + residual connection + normalization layer
  - Feed-forward fully connected layer + residual connection + normalization layer
- Decoder
  - Multi-head attention over the output of the encoder stack + residual connection + normalization layer
  - Multi-head self-attention mechanism + residual connection + normalization layer
  - Feed-forward fully connected layer + residual connection + normalization layer



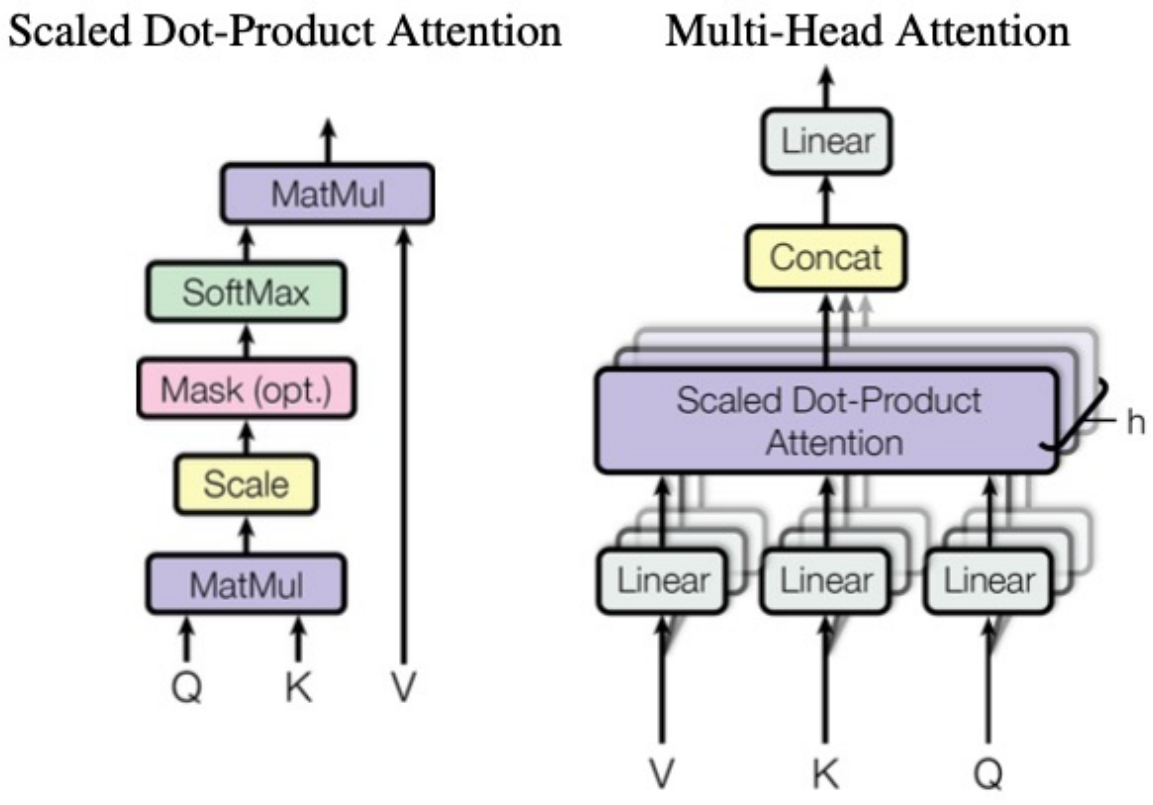
Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N. , Kaiser L., and Polosukhin I., Attention Is All You Need, http://arxiv.org/abs/1706.03762

Transformers

From an input embedding, three roles are played:

- Query → the current focus of attention
- Key → a preceding input
- Value → the value for the computation

$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{(\sqrt{d_k})}\right)\mathrm{V}$$



Jurafsky D. and Martin J. H., Speech and Language Processing (3rd ed. draft), https://web.stanford.edu/~jurafsky/slp3/
Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N. , Kaiser L., and Polosukhin I., Attention Is All You Need, http://arxiv.org/abs/1706.03762

Natural Language Processing (NLP)

- NLP is the use of human languages, such as English or French, by a computer.
- Many NLP applications are based on language models that define a probability distribution over sequences of words, characters, or bytes in a natural language.

Natural Language Processing (NLP)

n-grams is a simple language model where the conditional probability of the *n*-th token is given from the preceding n-1 tokens

$$P(x_1, \dots, x_\tau) = P(x_1, \dots, x_{n-1}) \prod_{t=n}^{\tau} P(x_t | x_{t-n+1}, \dots, x_{t-1})$$

$$P(x_t | x_{t-n+1}, \dots, x_{t-1}) = \frac{P_n(x_{t-n+1}, \dots, x_t)}{P_{n-1}(x_{t-n+1}, \dots, x_{t-1})}$$

$$P(\text{THE DOG RAN AWAY}) = P_3(\text{THE DOG RAN}) \frac{P_3(\text{DOG RAN AWAY})}{P_2(\text{DOG RAN})}$$

Goodfellow I., Bengio Y. and Courville A., Deep Learning, http://www.deeplearningbook.org
Jurafsky D. and Martin J. H., Speech and Language Processing (3rd ed. draft), https://web.stanford.edu/~jurafsky/slp3/

Natural Language Processing (NLP)

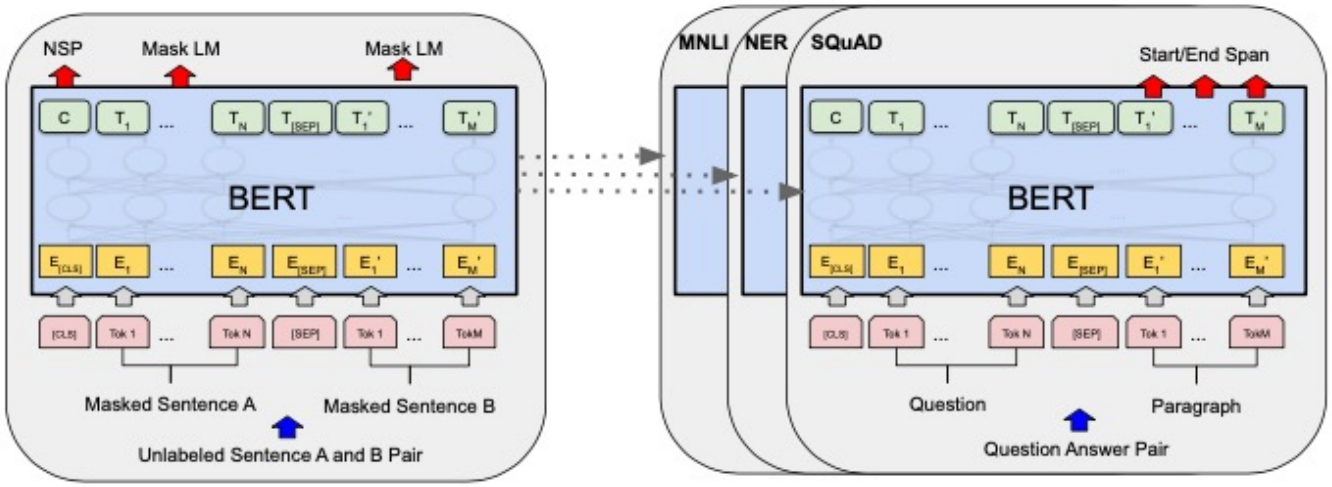
```
A misty ridge uprises from the surge
```

tokenization

```
['a', 'misty', 'ridge', 'up', '##rise', '##s',
'from', 'the', 'surge']
```

Bidirectional **E**ncoder **R**epresentations from **T**ransformers (BERT)

- Pre-Training
  - Masked Language Model
  - Next Sentence Prediction
- Fine-Tuning



Pre-training Fine-Tuning

Bidirectional **E**ncoder **R**epresentations from **T**ransformers (BERT)

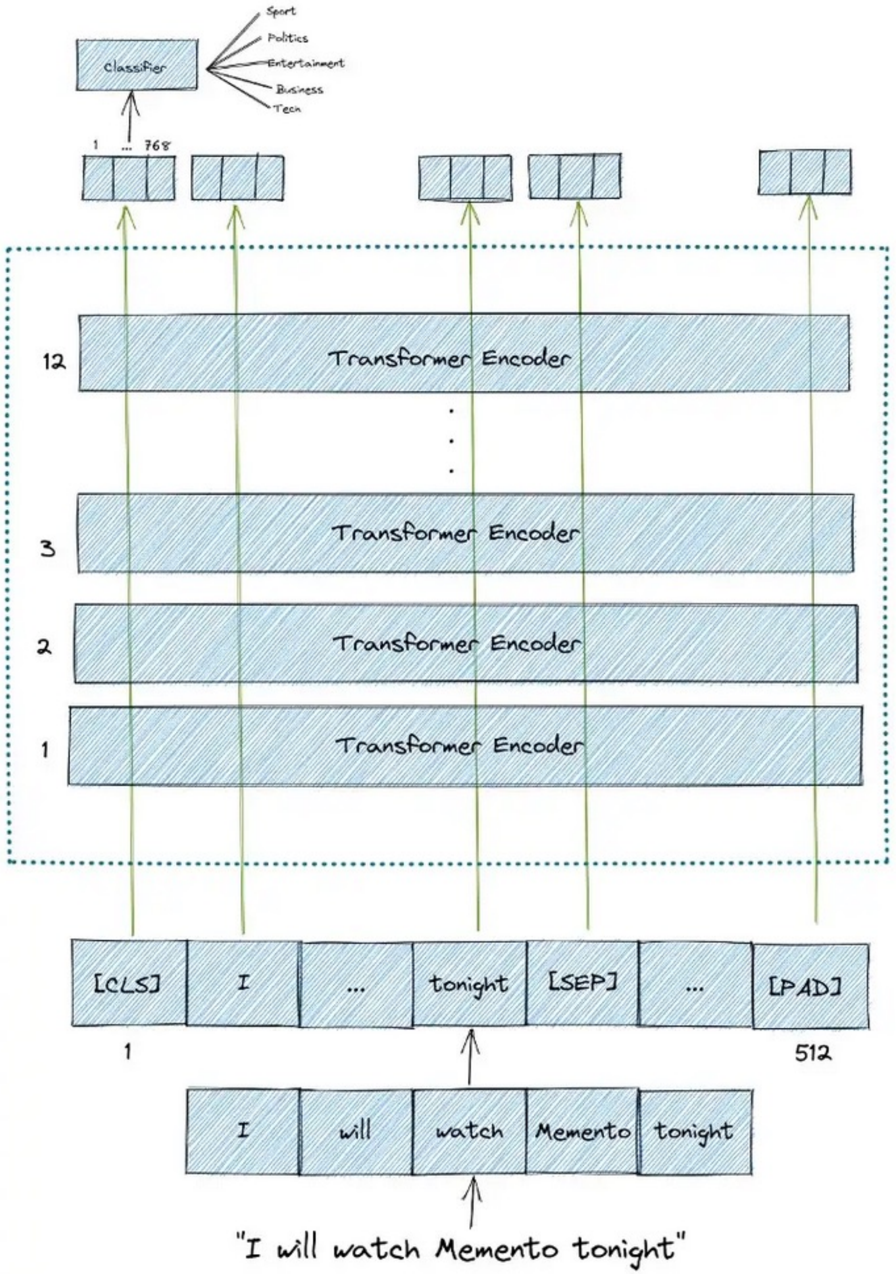
L → Number of layers (Transformer blocks)

H → Hidden size

A → Number of self-attention heads

- $\mathrm{BERT}_{\mathrm{BASE}}$ (L=12, H=768, A=12, Total Parameters=110M)
- $\mathrm{BERT}_{\mathrm{LARGE}}$ (L=24, H=1024, A=16, Total Parameters=340M)

Devlin J., and Chang M., Lee K. and Toutanova K., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, http://arxiv.org/abs/1810.04805



https://towardsdatascience.com/text-classification-with-bert-in-pytorch-887965e5820f