



# Attention in NNs: the advent of Transformers

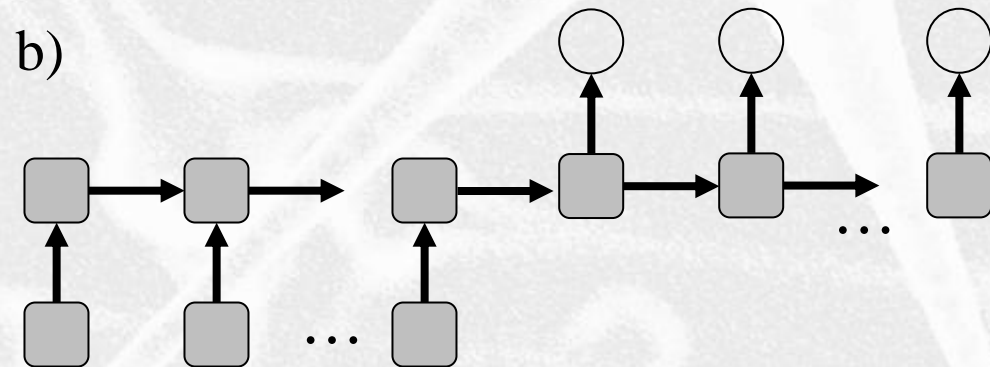
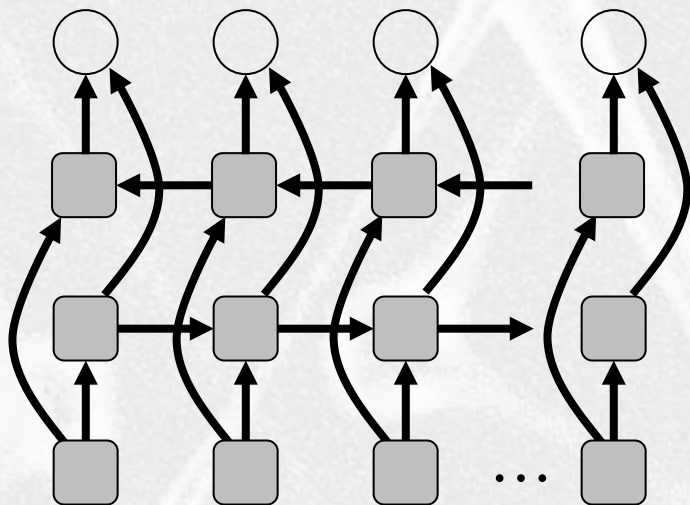
Roberto Basili, Danilo Croce  
Deep Learning, 2024/2025

# Outline

- Attention Mechanisms in Recurrent Networks
- Trasformers
- Applications to Language Processing
- Perspectives

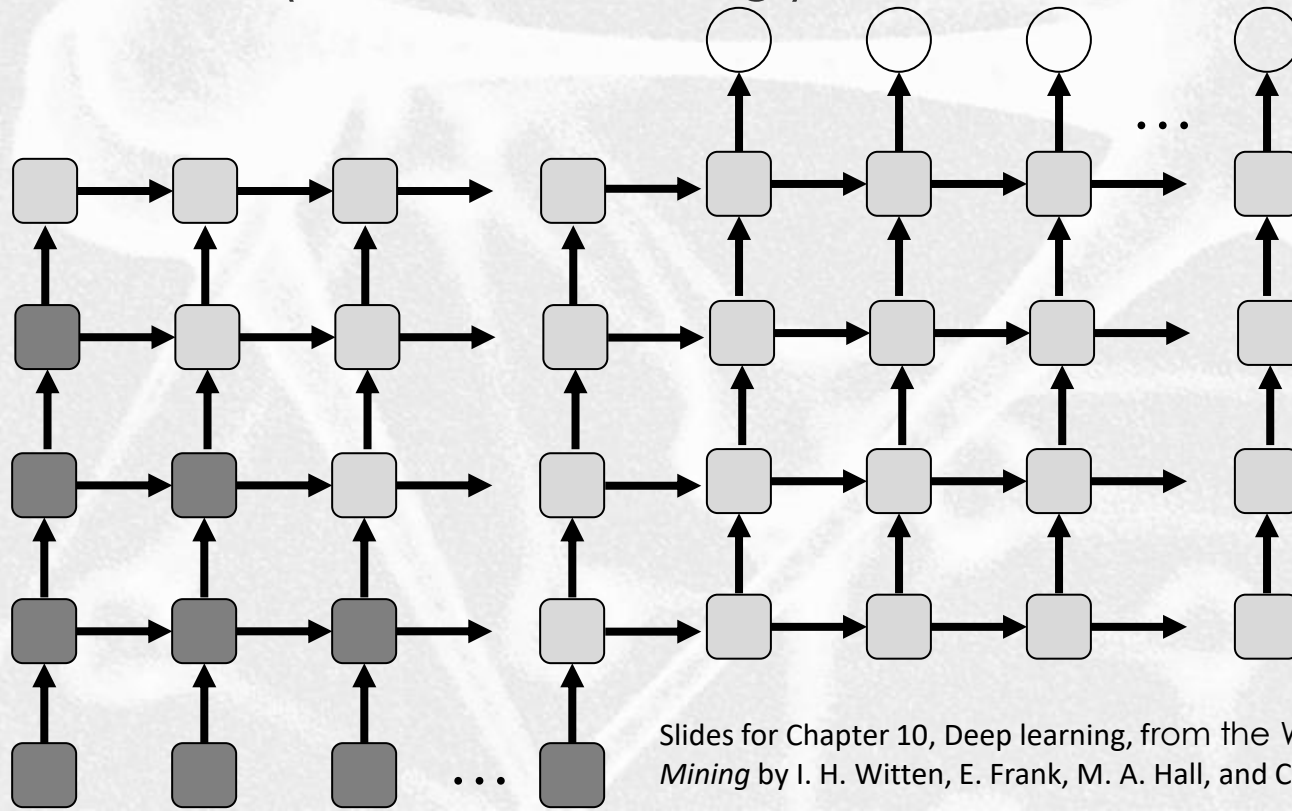
# Other RNN architectures

- a) Recurrent networks can be made bidirectional, propagating information in both directions
  - They have been used for a wide variety of applications, including protein secondary structure prediction and handwriting recognition
- b) An “encoder-decoder” network creates a fixed-length vector representation for variable-length inputs, the encoding can be used to generate a variable-length sequence as the output
  - Particularly useful for machine translation



# Encoder-decoder deep architectures

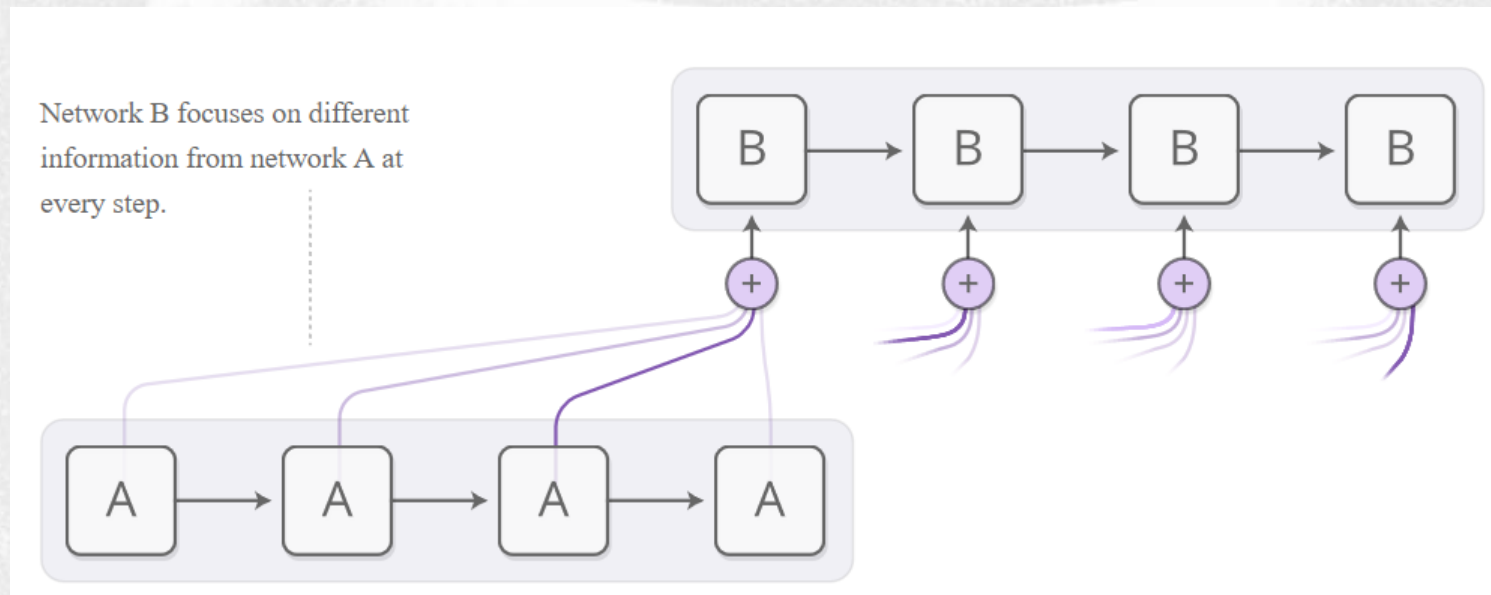
- Given enough data, a deep encoder-decoder architecture (see below) can yield results that compete with hand-engineered translation systems.
- The connectivity structure means that partial computations in the model can flow through the graph in a wave (darker nodes in fig.)





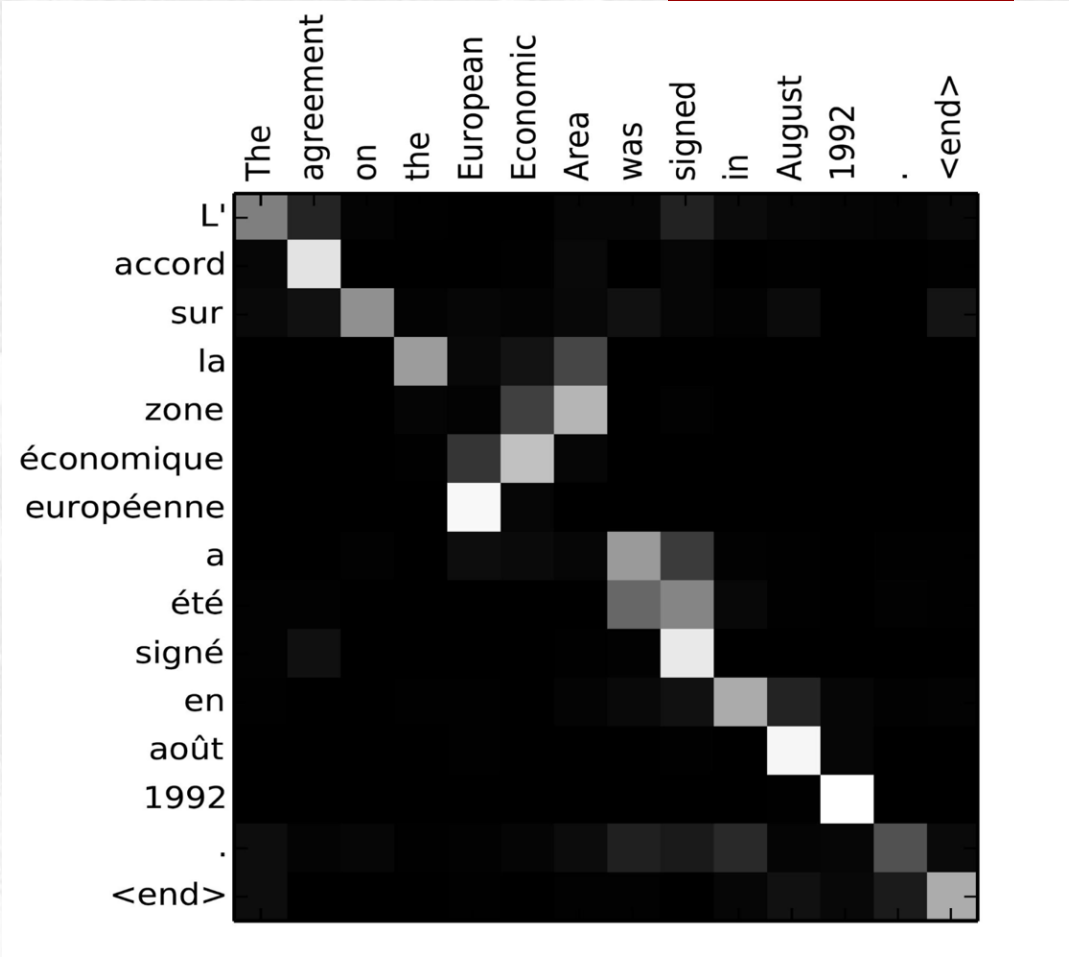
# Attention-based RNNs

- A NN (e.g. B) is used to attend the outcome of a second network A, e.g. (Vaswani et al., 2017)



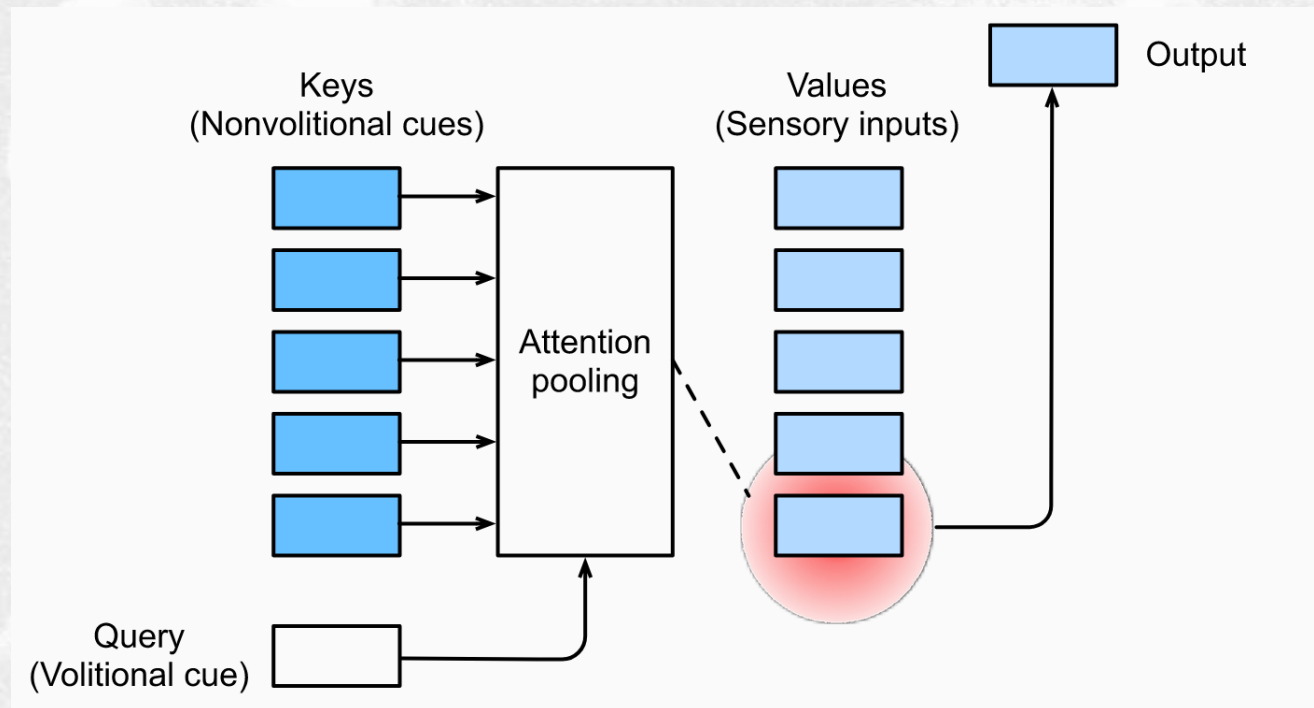
100

- 

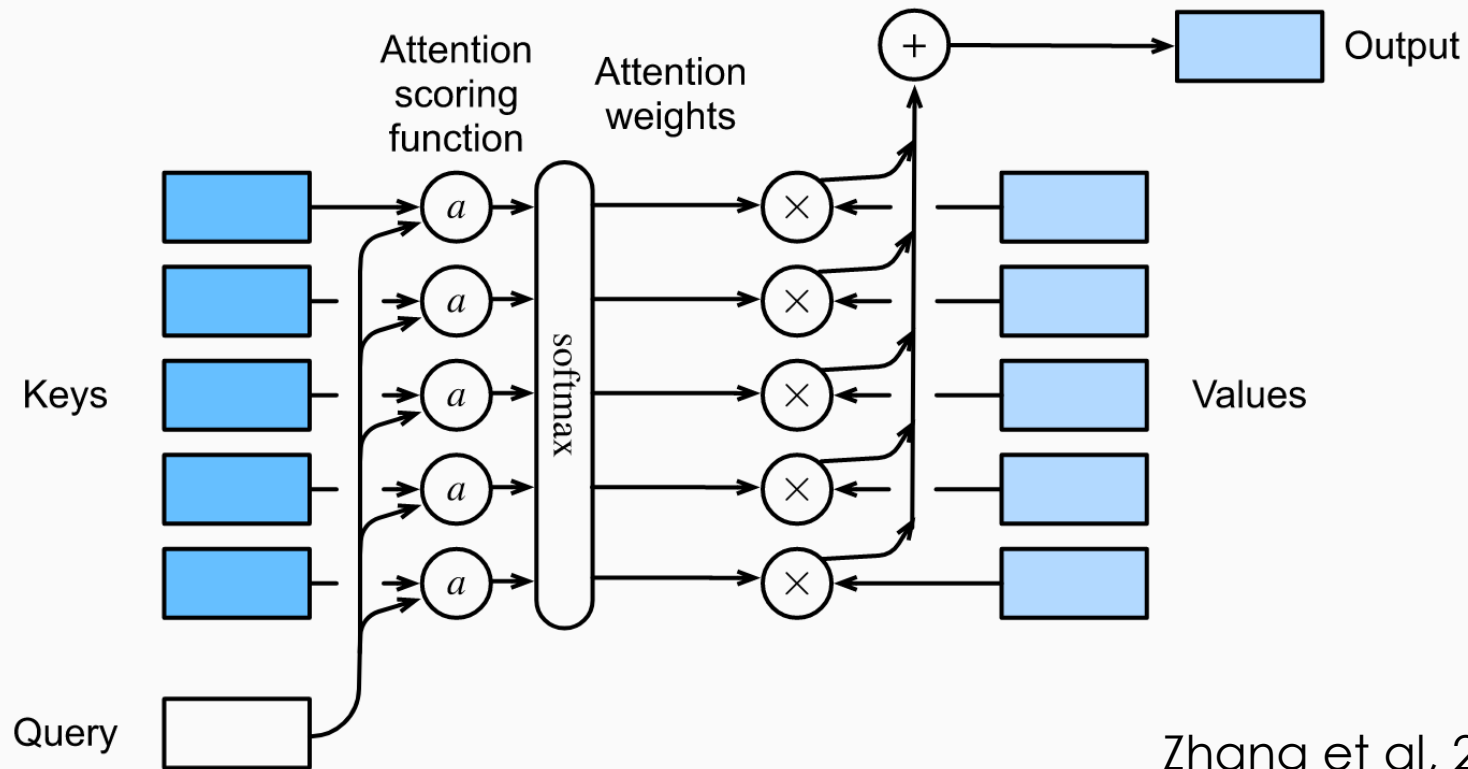


# Attention: motivations

- From (*Dive into Deep Learning*, Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., 2021).



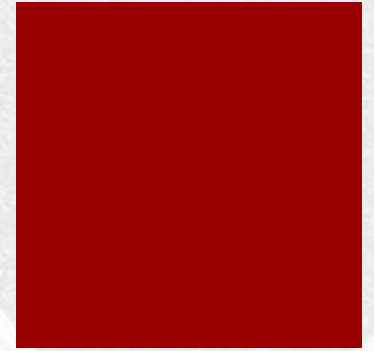
# Attention functions



Zhang et al, 2021



# The Importance of Attention in Neural Learning

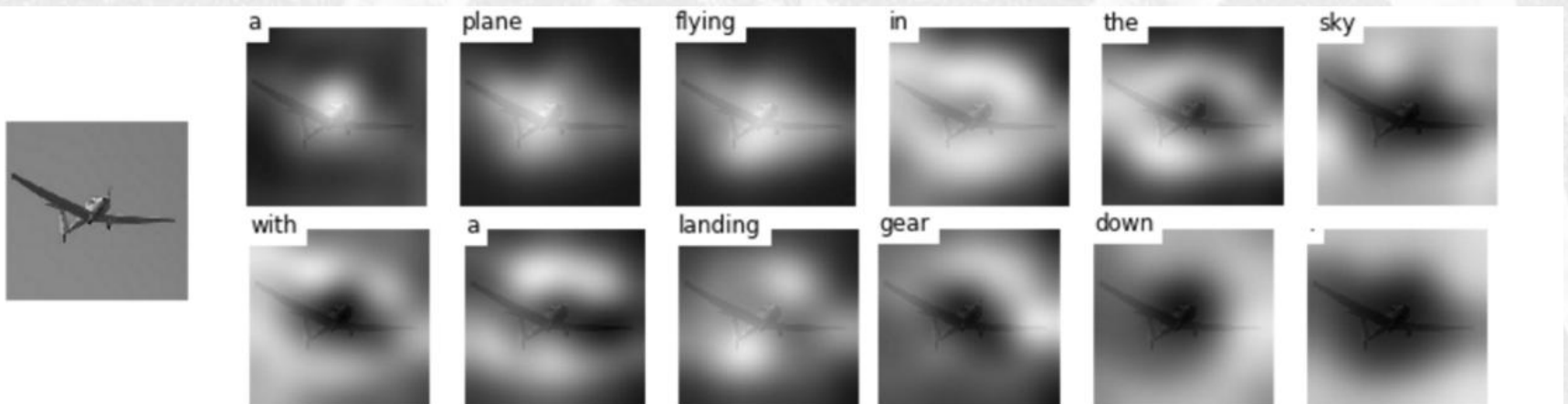


## ■ Revolution in Computer Vision

- It significantly improved **object detection** and **recognition in computer vision**.
- It enables models to focus on relevant parts of an image, improving accuracy and efficiency.
- **An interesting Survey:** <https://github.com/MenghaoGuo/Awesome-Vision-Attentions>

## ■ Breakthrough in tasks such as Image Captioning:

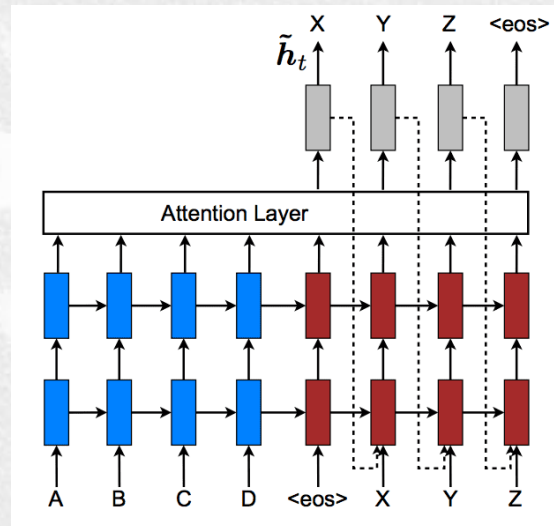
- Attention helps in **identifying key components** within images to **generate accurate and contextually relevant descriptions**.
- **Seminal work:** (Xu et al, 2015) <https://arxiv.org/abs/1502.03044>



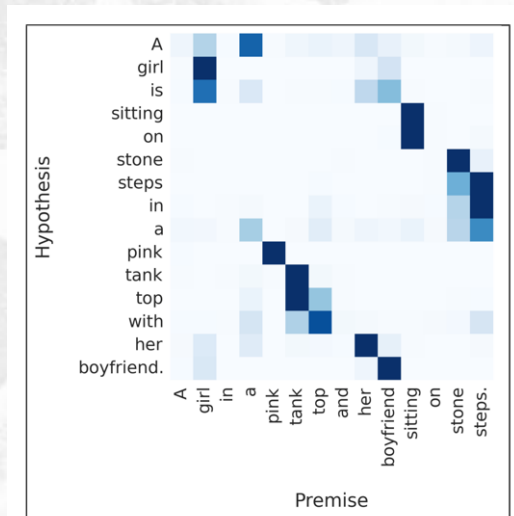


# Enhancement in Recurrent Neural Networks

- For RNNs, **attention mechanisms** were used to address the **challenge of handling long sequences**.
- It allows RNNs to **focus on important parts** of the input sequence
  - improving performance in tasks like language translation and speech recognition.



(Luong et al, 2015)  
Effective Approaches to Attention-based Neural Machine Translation  
<https://arxiv.org/abs/1508.04025>



(Rocktäschel et, al., 2015)  
Reasoning about Entailment with Neural Attention, 2015.  
<https://arxiv.org/abs/1508.04025>

# Going back in time to 2017: the Transformer

(Vaswani et al. 2017)

- **Attention in Transformers:**

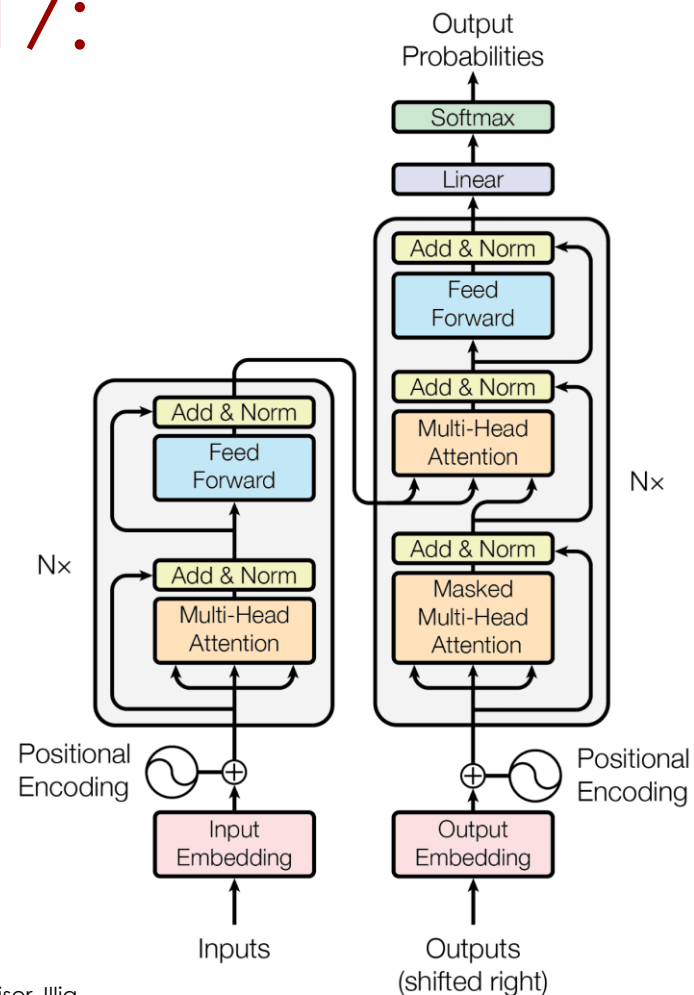
- In 2017, the **attention mechanism became an integral part** of this architecture.
- a **significant evolution in seq2seq modeling**

- **Main advantages:**

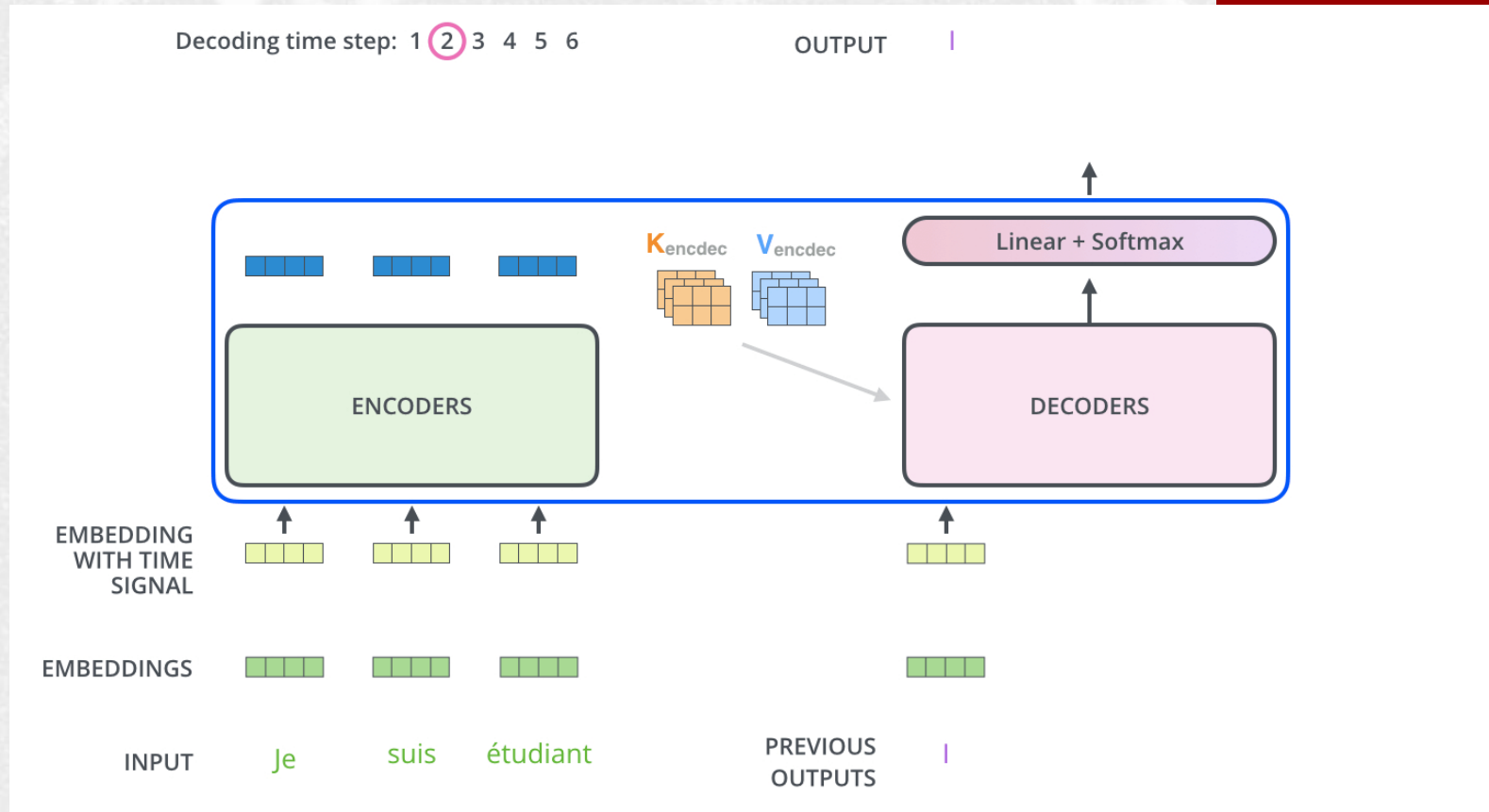
- Better with long range dependencies
- Parallel processing (more scalable than RNNs)
- State-of-the-art performances

- **Originally meant for Automatic machine translation:**

- E.g., French to English



# Seq2Seq: A transformer in action



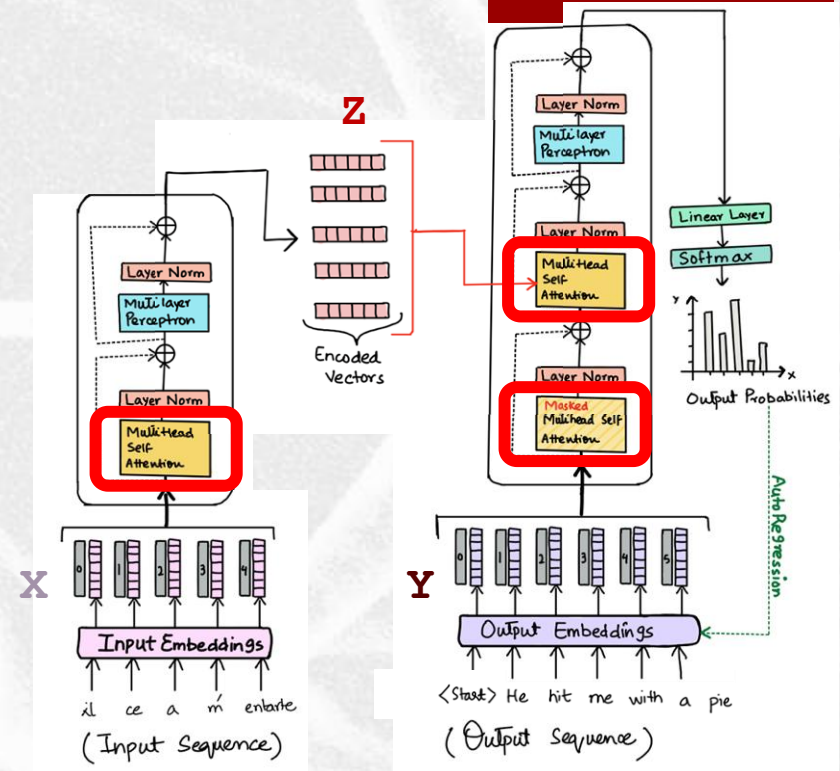
Alammar, J (2018). The Illustrated Transformer [Blog post]. Retrieved from <https://jalammar.github.io/illustrated-transformer/>



# Encoding/Decoding Architecture with Attention Mechanism

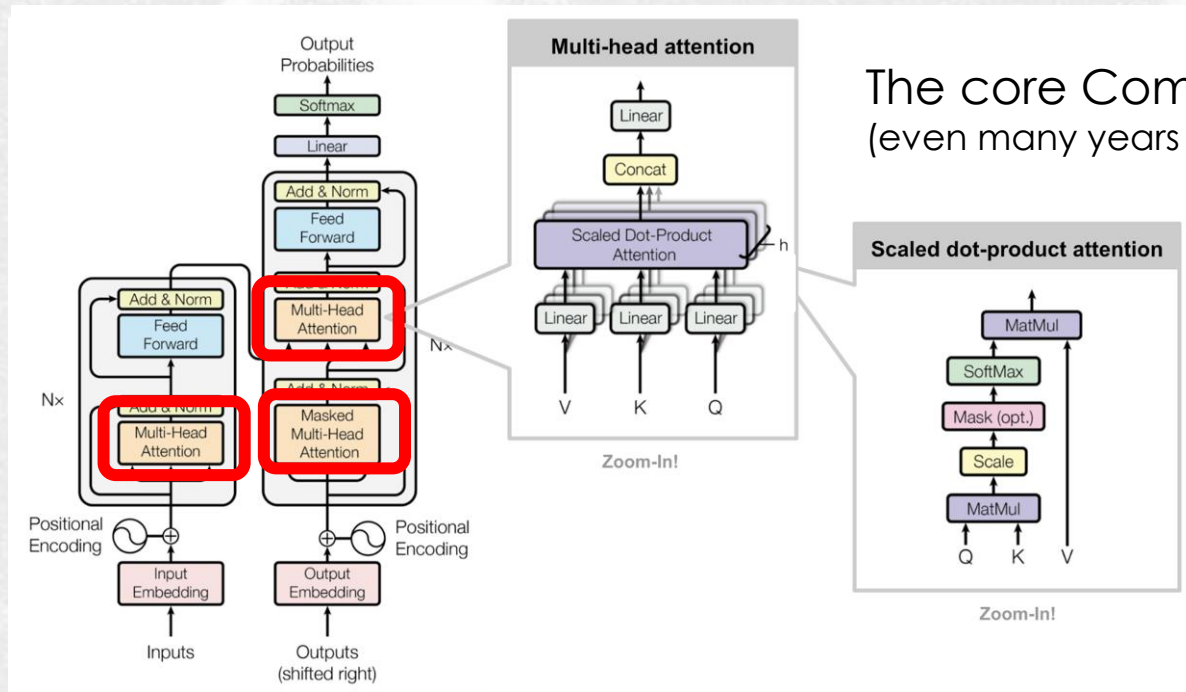
## ■ Two components

- **Encoder:** Maps input sequence  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$  to continuous representations  $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_n)$ .
- **Decoder:** Decoder uses  $\mathbf{Z}$  to generate output sequence  $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_m)$
- Encoder/Decoder process input vectors through **self-attention layer** and feed-forward network.
  - It enables to selectively **concentrate on pertinent parts of the input**
  - It improves **context awareness**
  - It allows to **consider positions** in the that also depends on the output



# How does Self-attention work

It is not magic, it is not a human brain, it is **just matrix multiplication**

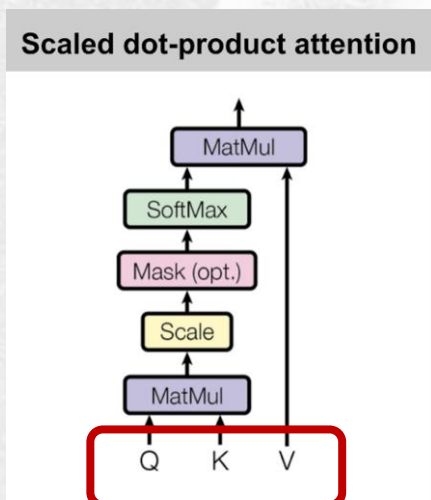


Many thanks to <https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a>

# «Attention in action»

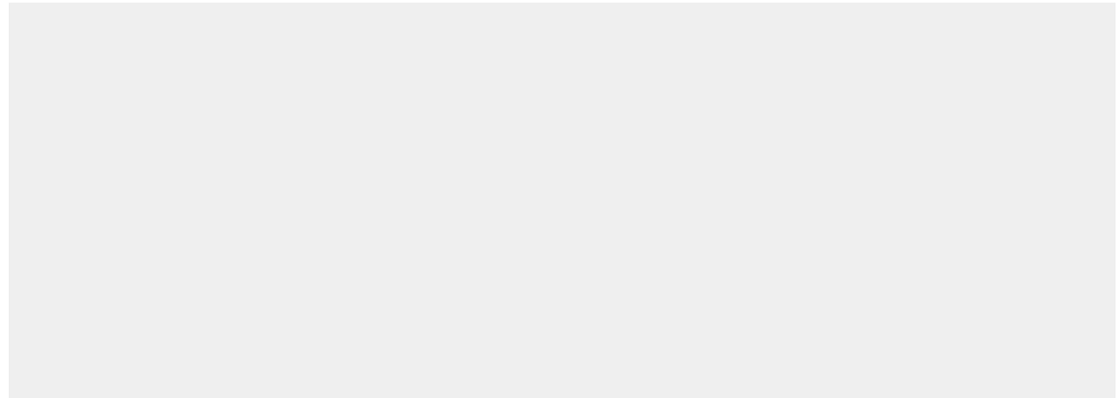
## Prepare inputs

Each word is associated to embeddings



**Positional Encoding:** Part of these vector encodes the tokens' position

Self-attention



input #1  
1 0 1 0

input #2  
0 2 0 2

input #3  
1 1 1 1

I

like

pizza



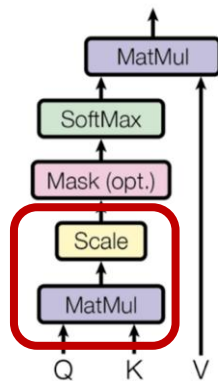
# «Attention in action»

## Compute Query, Key, and Value Vectors

For each word vector, calculate the Query, Key, and Value vectors by multiplying with respective weight matrices  $W^Q$ ,  $W^K$ ,  $W^V$ .

- $\text{key}_i = \text{input}_i W^K$
- $\text{value}_i = \text{input}_i W^V$
- $\text{query}_i = \text{input}_i W^Q$

### Scaled dot-product attention

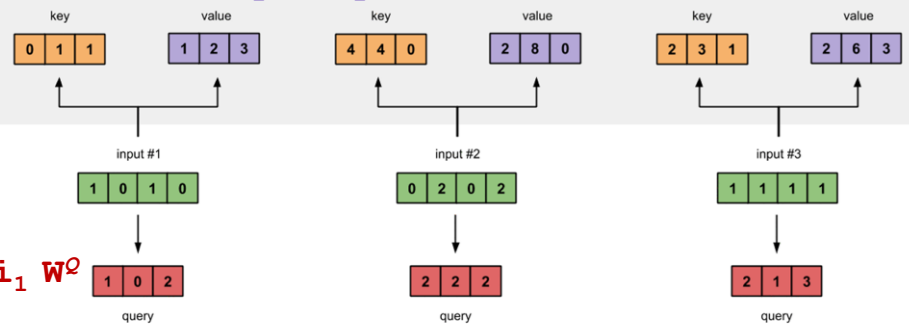


Self-attention

$$k_1 = i_1 W^K$$

$$v_1 = i_1 W^V$$

$$q_1 = i_1 W^Q$$



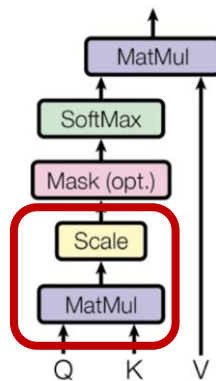


# «Attention in action»

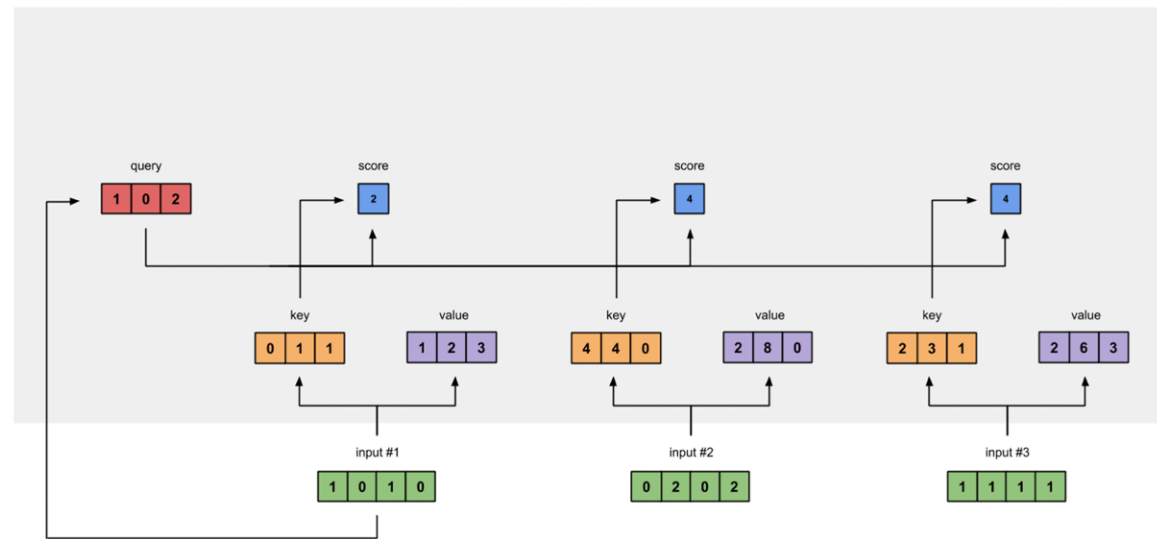
Calculate the Attention scores for **input<sub>1</sub>**.

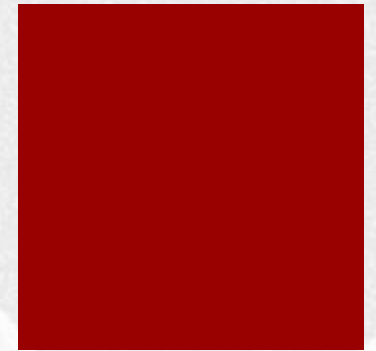
Attention scores are computed to «weight» the contribution of ALL words in the input sequence when representing **input<sub>1</sub>**.

## Scaled dot-product attention



Self-attention

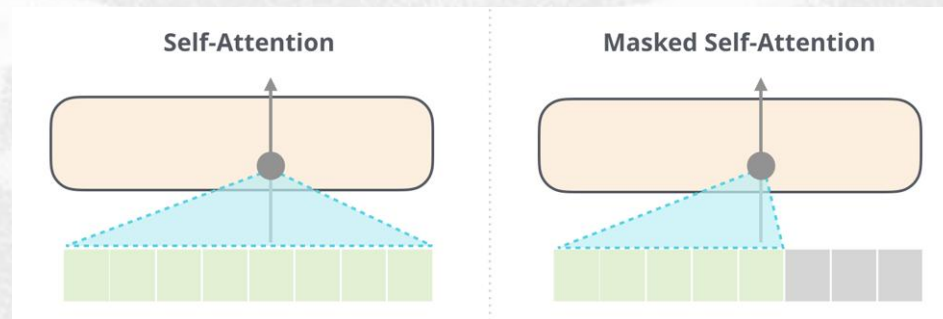
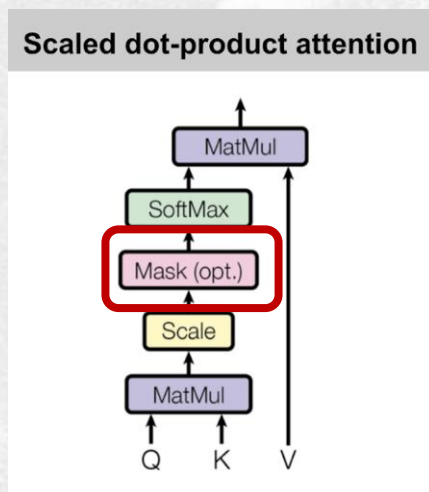




# «Attention in action»

## Role of Masked Attention

Use masked attention to handle sequences of different lengths



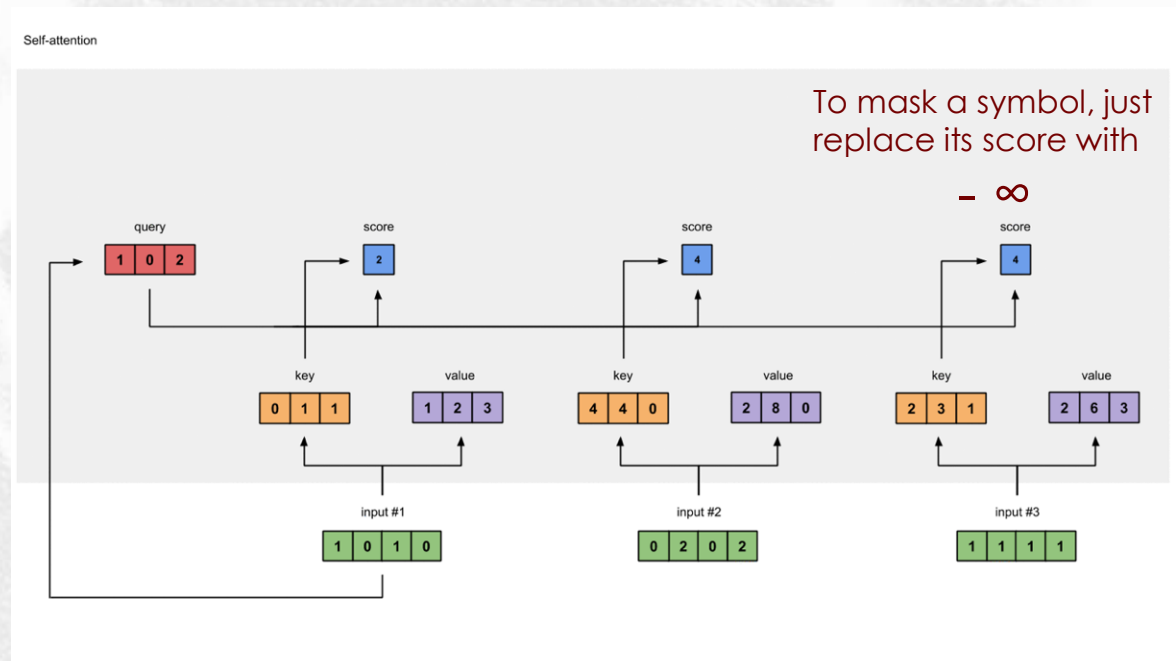
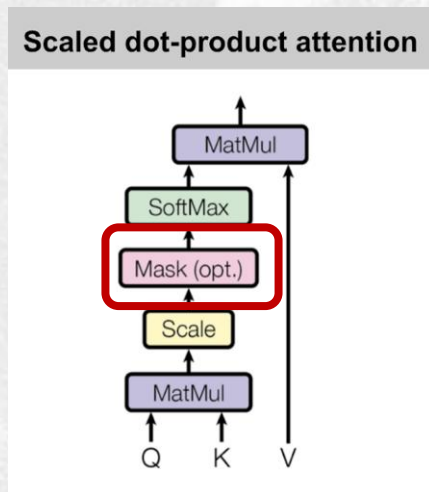
<https://jalammar.github.io/illustrated-gpt2/>



# «Attention in action»

## Role of Masked Attention (2)

Use masked attention to handle sequences of different lengths

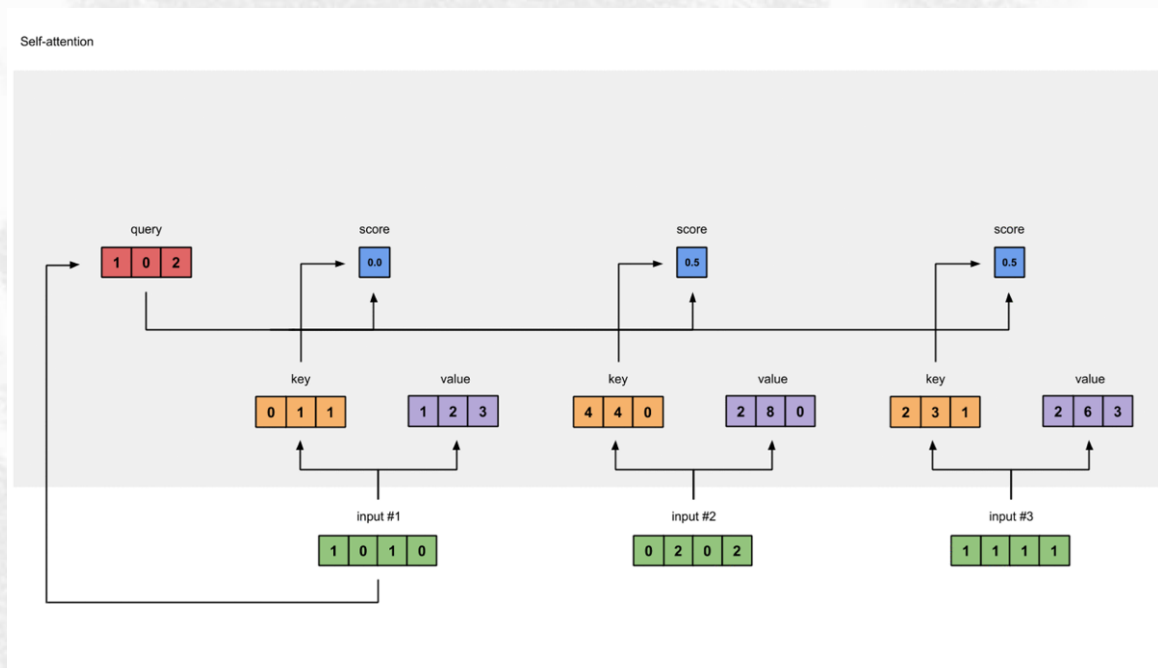
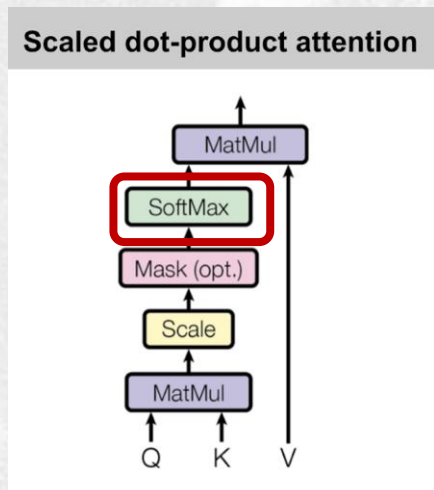


# «Attention in action»

## Calculate softmax

Softmax «simply» maps **attention scores** to a «probability» in  $[0, 1]$

- The masked elements (i.e., with  $-\infty$  gets a score near to 0)





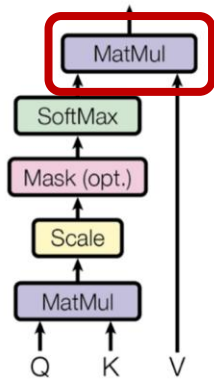
# «Attention in action»

## Multiply scores with values

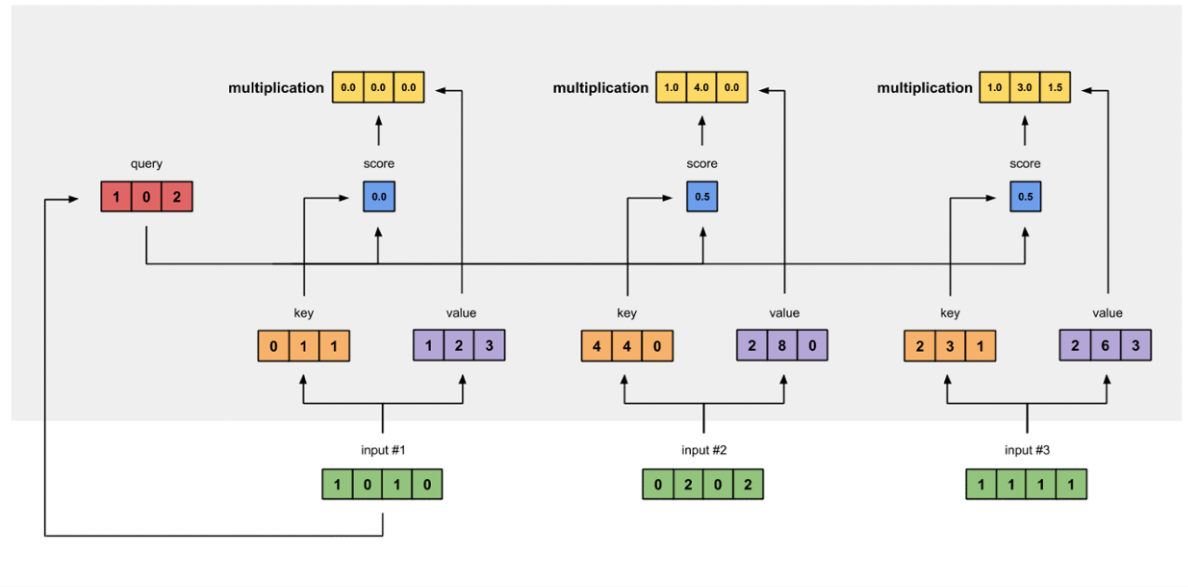


Each  $\text{input}_n$  (through each  $\text{value}_n$ ) is weighted based of its importance in representing  $\text{input}_1$

### Scaled dot-product attention



Self-attention

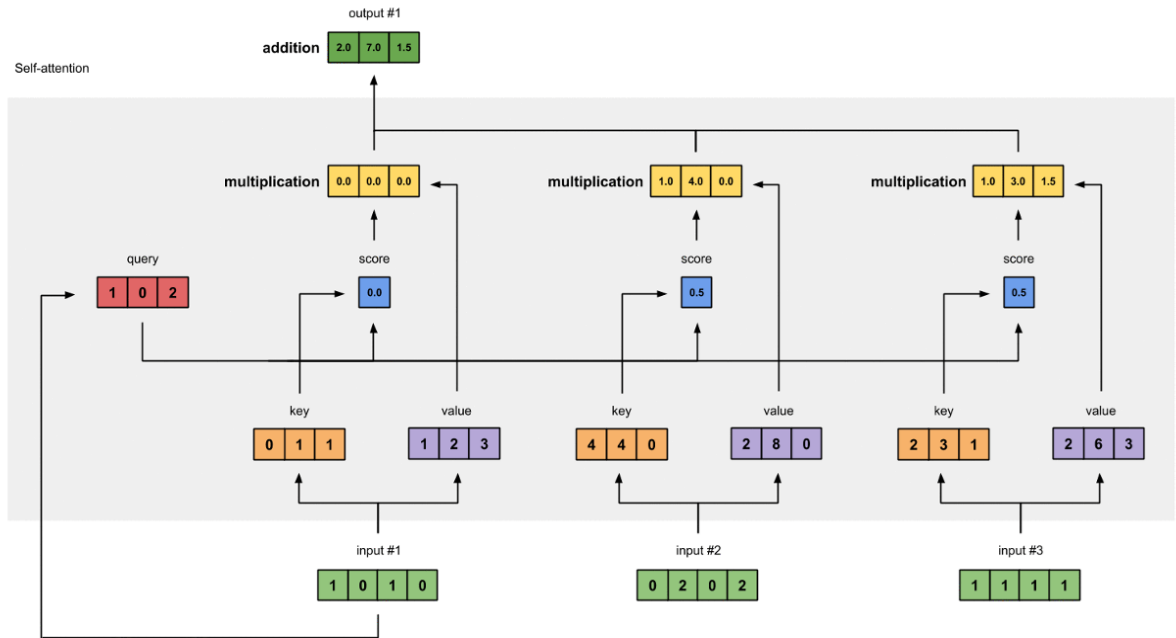
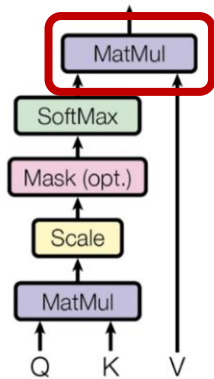


# «Attention in action»

Complete the linear combination

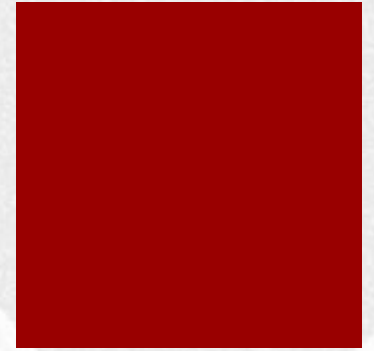
Sum weighted values to get **output<sub>1</sub>** that is the linear combination of all input elements (represented as **values**) weighted through the **attention scores**

## Scaled dot-product attention

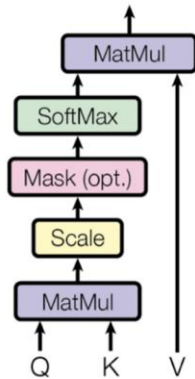


# «Attention in action»

Repeat for  $\text{input}_2$  and  $\text{input}_3$



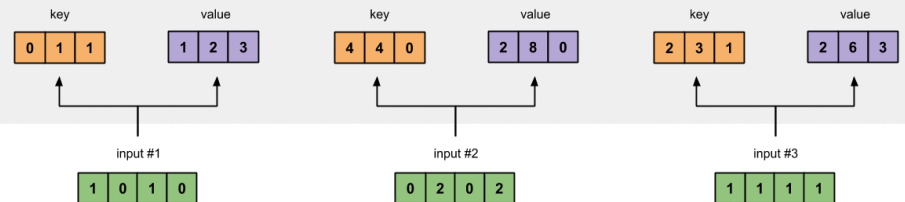
## Scaled dot-product attention



Self-attention

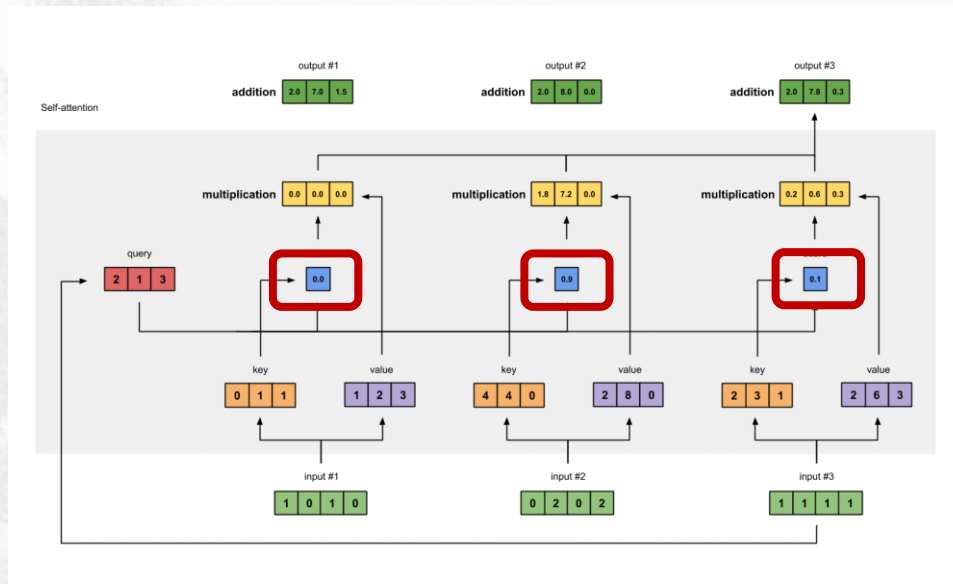
output #1  
addition 

2.0	7.0	1.5
-----	-----	-----



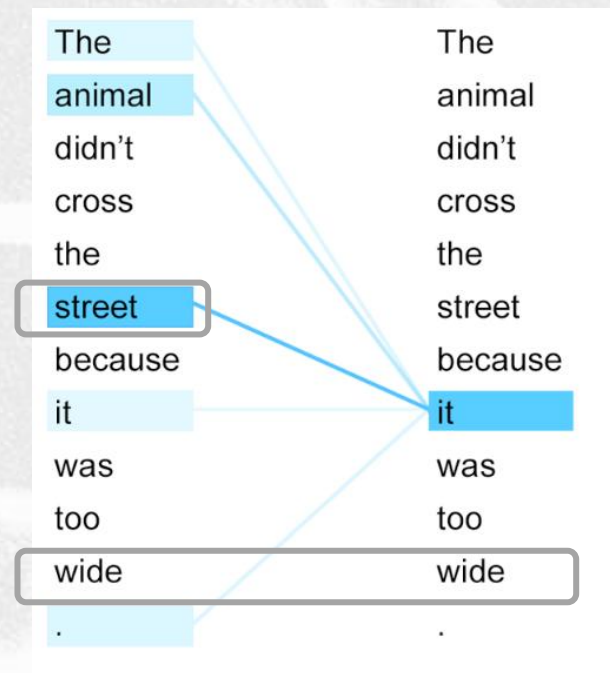
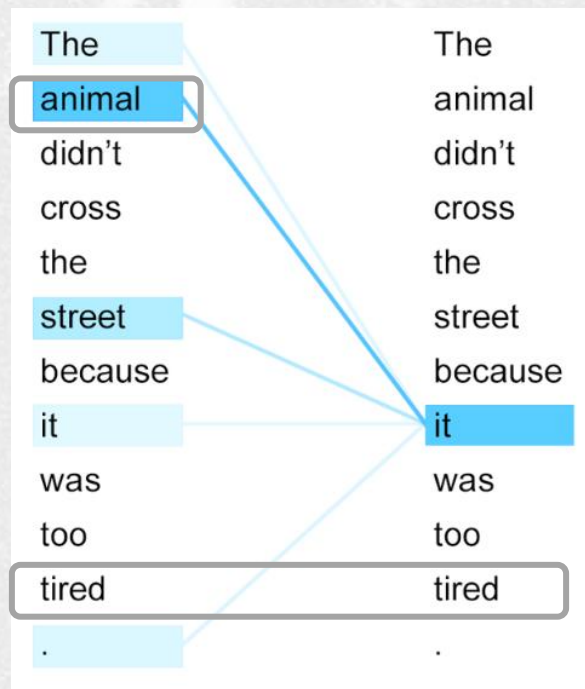
# So... what is self-attention?

- It is not just a number, but a «probability distribution» for each symbol in input
- And it allows weighting how **all words** are combined to generate the (hidden) representation of each word



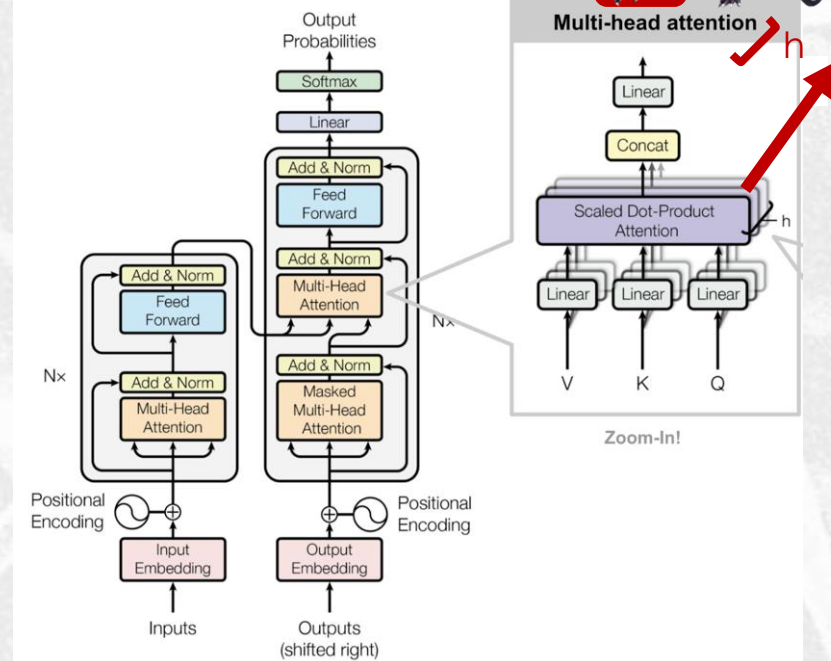


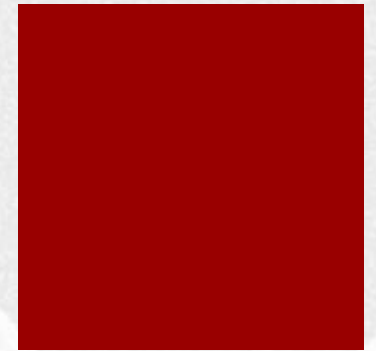
# Self-Attention



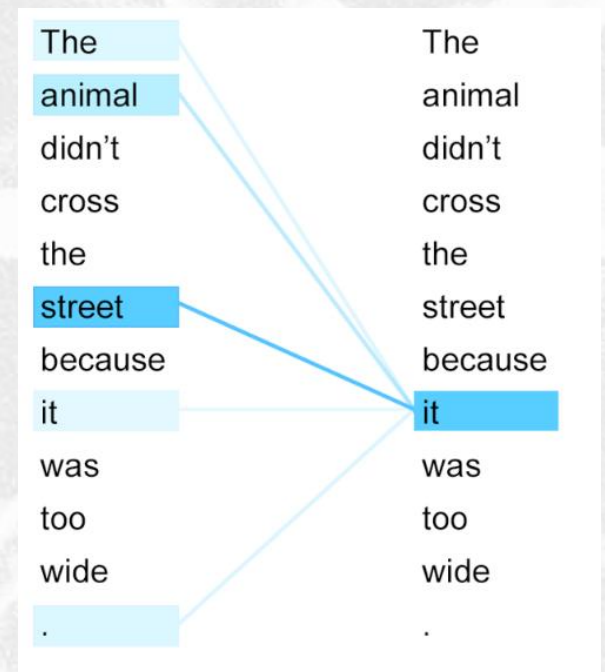
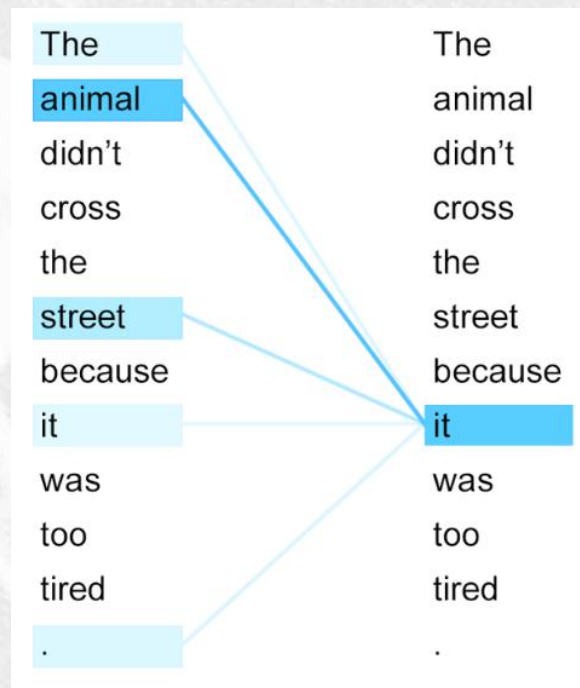
# The Multi-Headed «Beast»

- Humans can attend to many things simultaneously.
- Can we extend attention to achieve the same?
- **Idea:** apply **Redundancy**, i.e., Scaled Dot-Product Attention **multiple times**
  - For each input, just generate  $h$  output
    - using  $h$  different different ( $W^Q, W^K, W^V$ )
  - Concatenate the  $h$  output vectors of each input
  - Use a linear layer to “restore” the initial dimensionality
    - **But combining all multiple evidences**





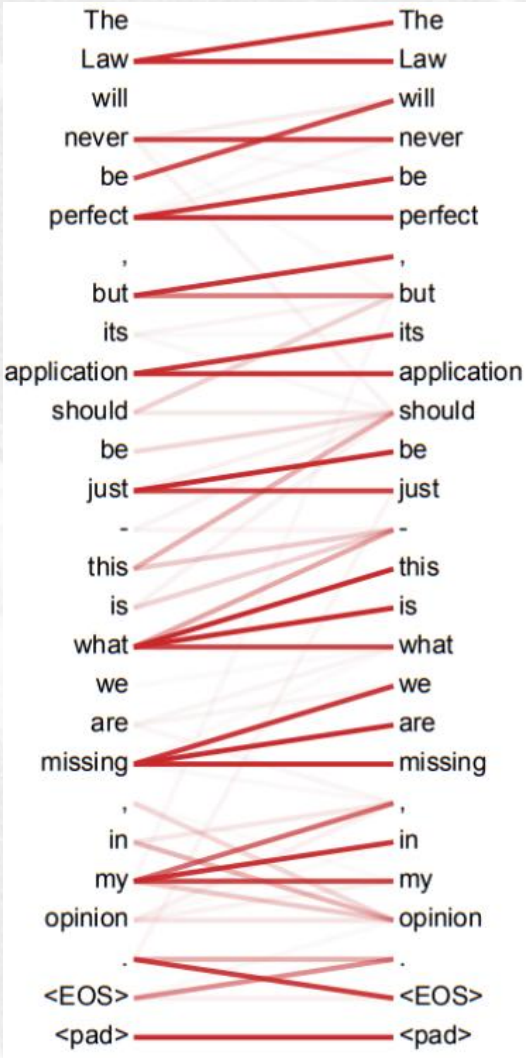
# From «simple» attention...







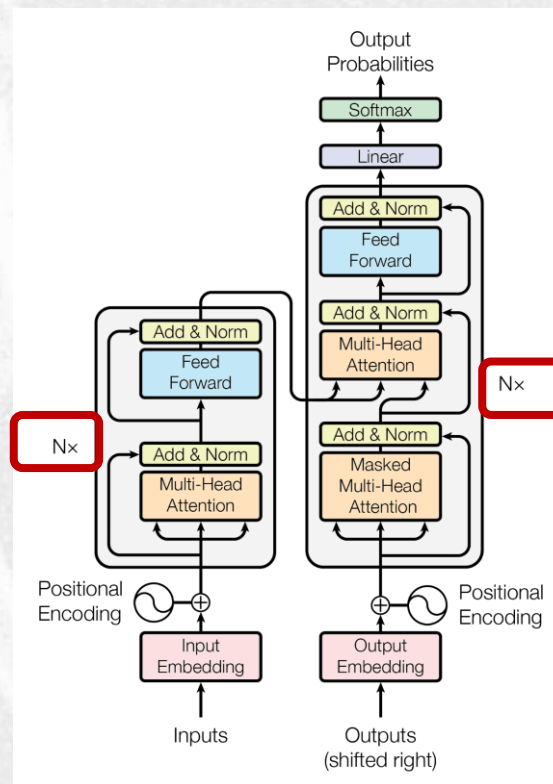
... to  
Multi-head  
Attention



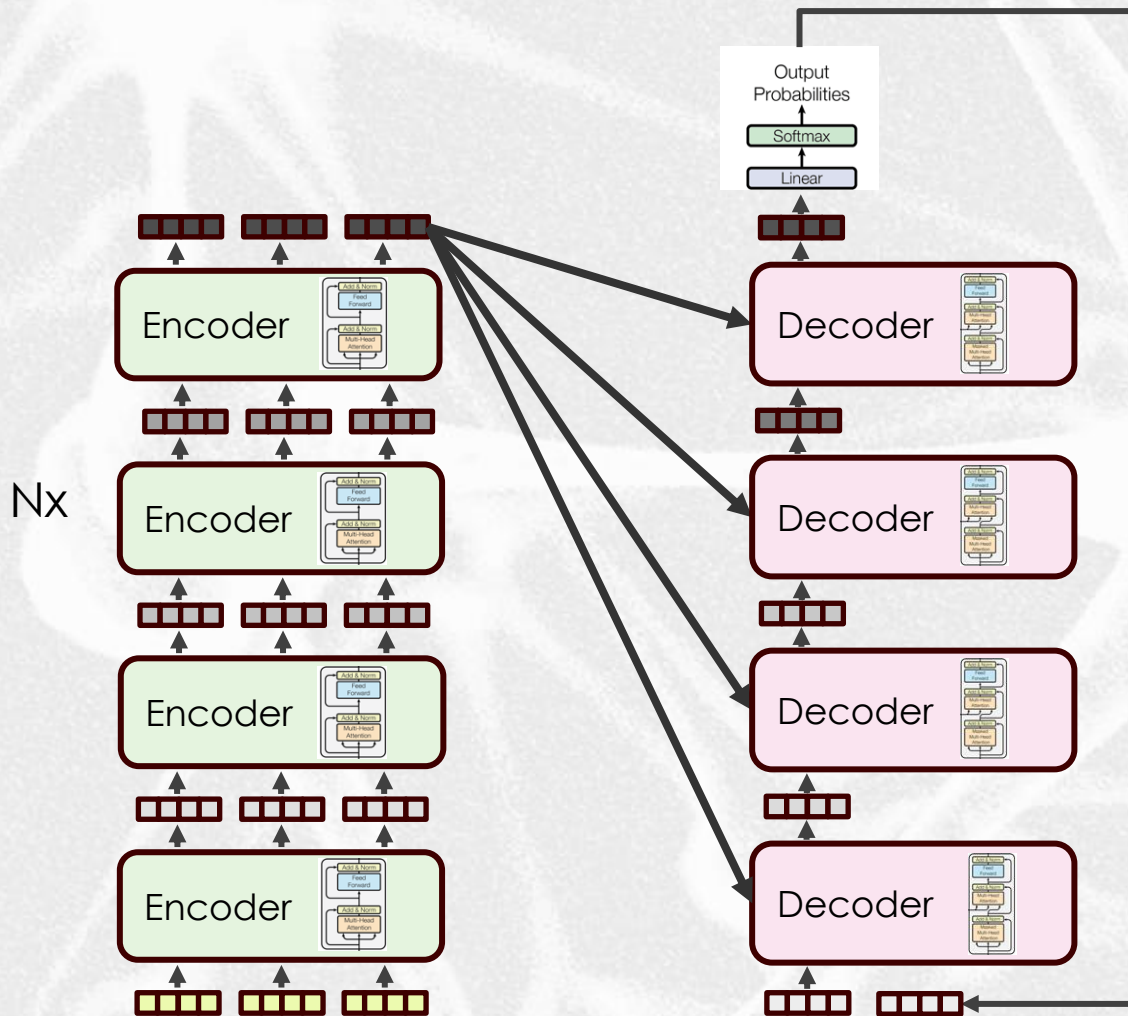


# Where is the «Deep Learning»?

Encoders and decoders are repeated  $N$  times



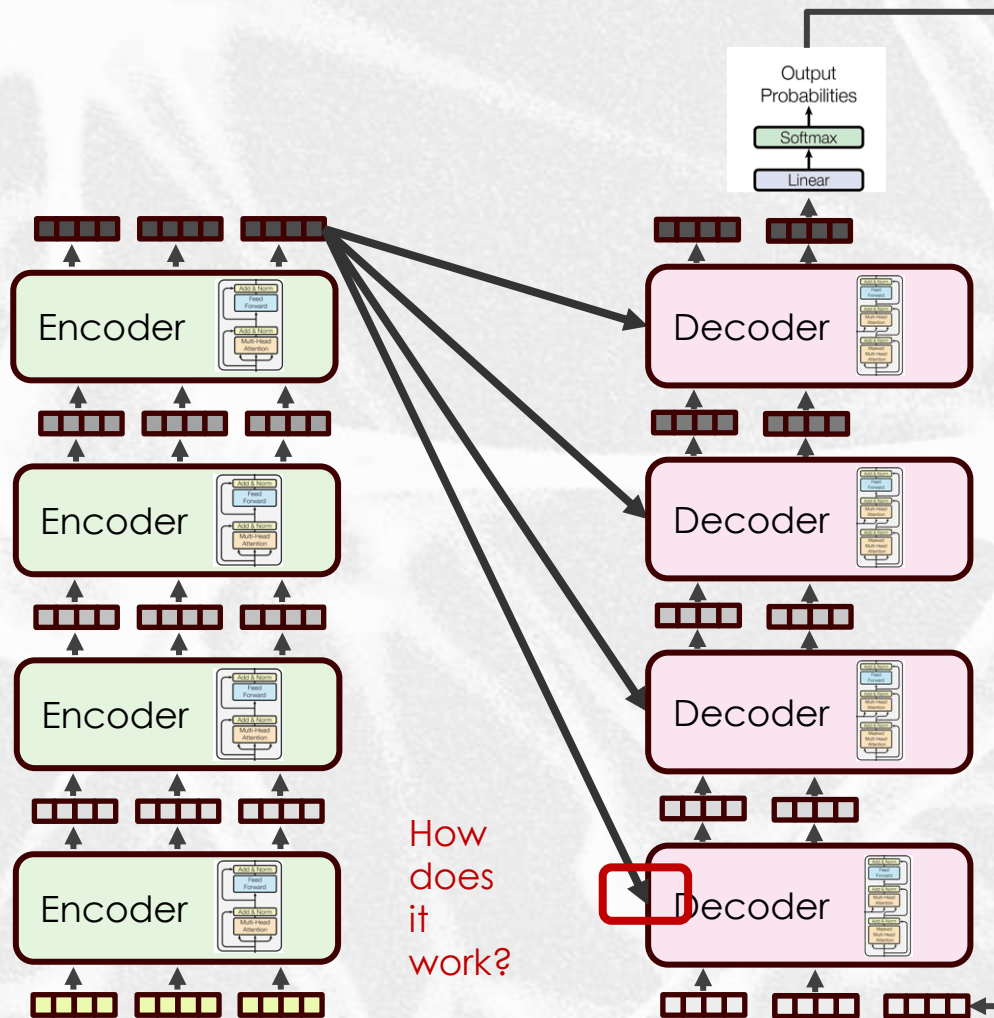
# Again: the Transformer in Action



Generated hidden representations for each symbol initially rely on the first token, called `<start>`.

These representations are influenced by all hidden representations from the encoder.

# Again: the Transformer in Action



After the generation begins...

the **decoder's hidden representations simultaneously depend on all input tokens** attended to in the encoder...

but on the decoder's own previously generated hidden representations up to that point.

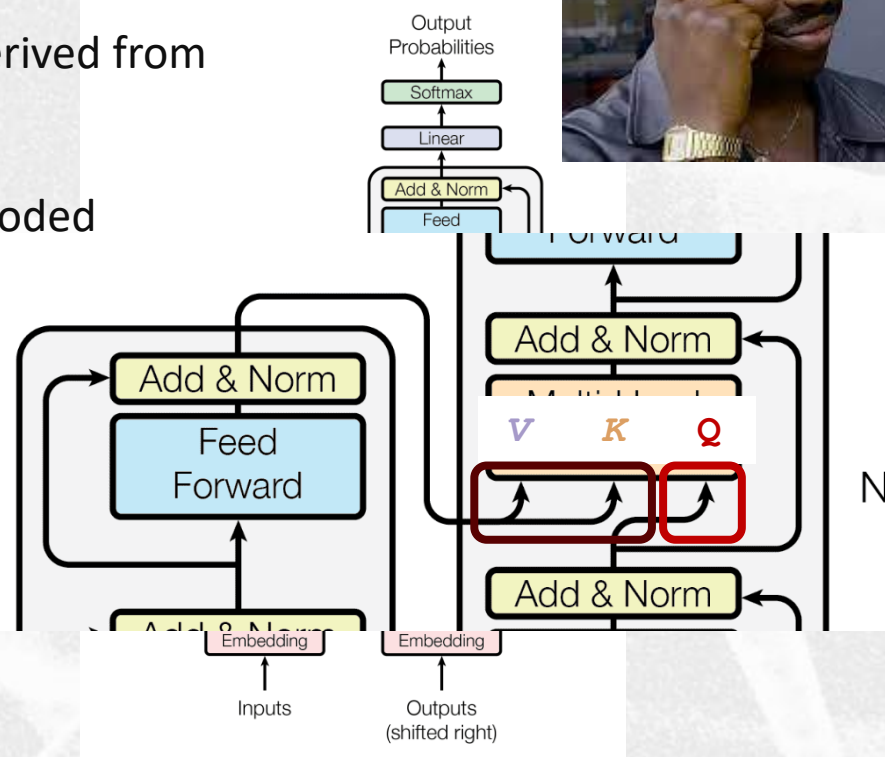
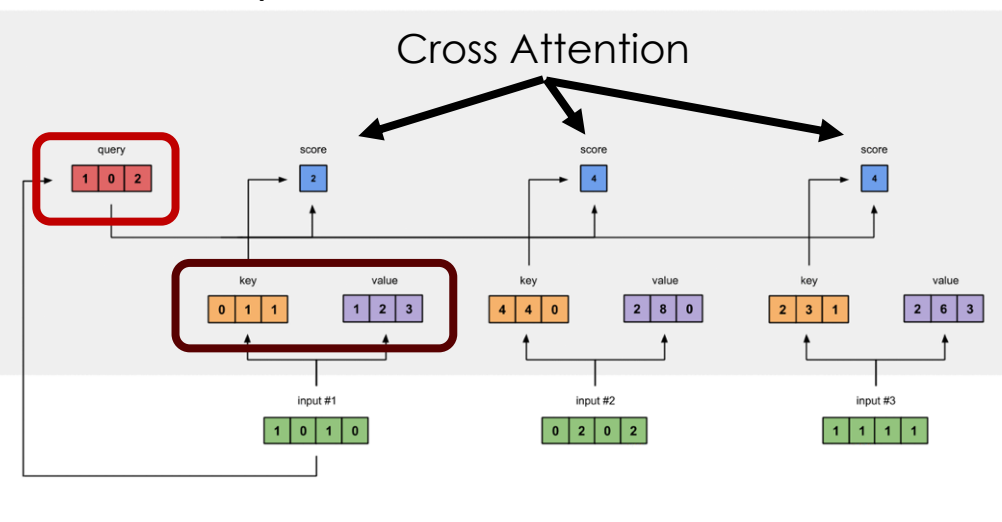




# How to combine Encoders and Decoders?

- In the decoder, **key** and **value vector** are derived from the input.
- The **query**, in contrast, depends on the decoded sequence.

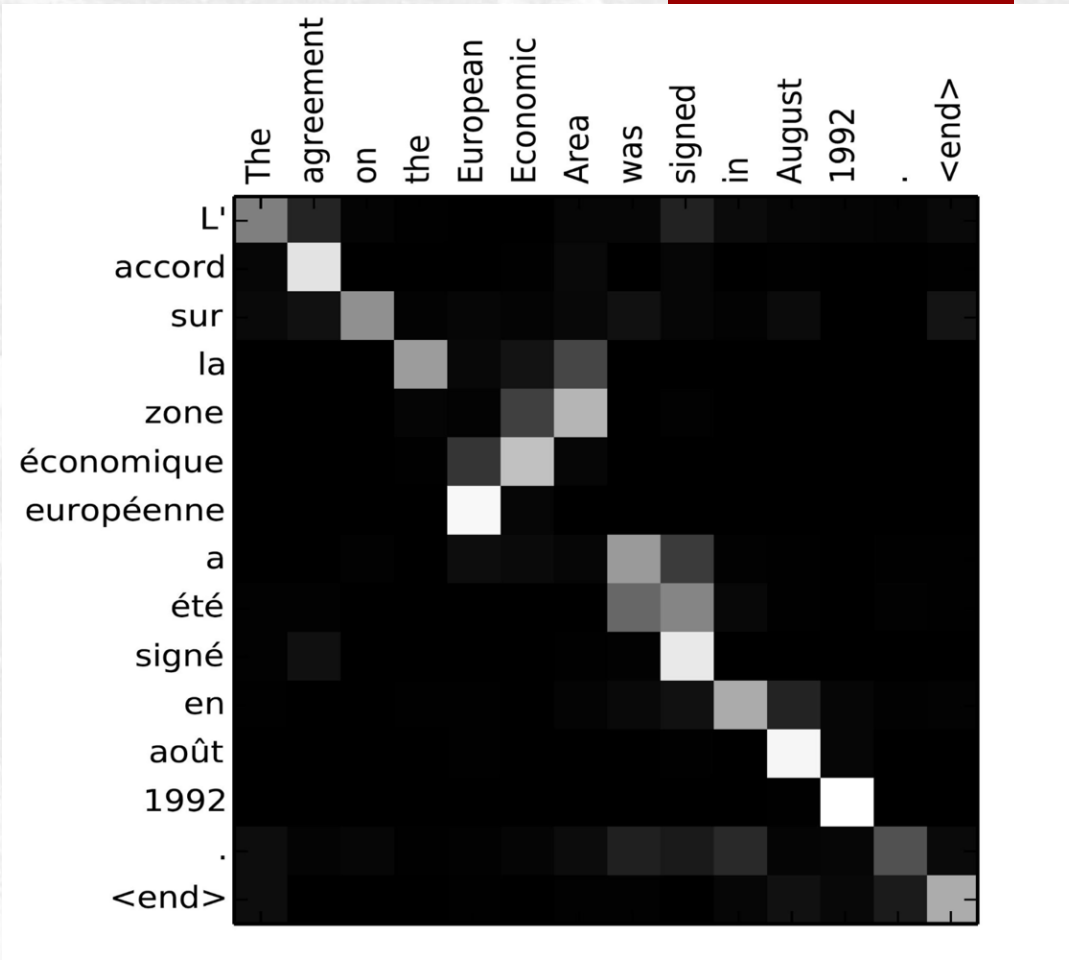
Self-attention





# Cross-Attention

- Attention scores between input and output words
- White equals higher score
- The diagonal is highly correlated
- The scores reveal the grammatical difference for adjectives for the two languages (*zone économique européenne*)





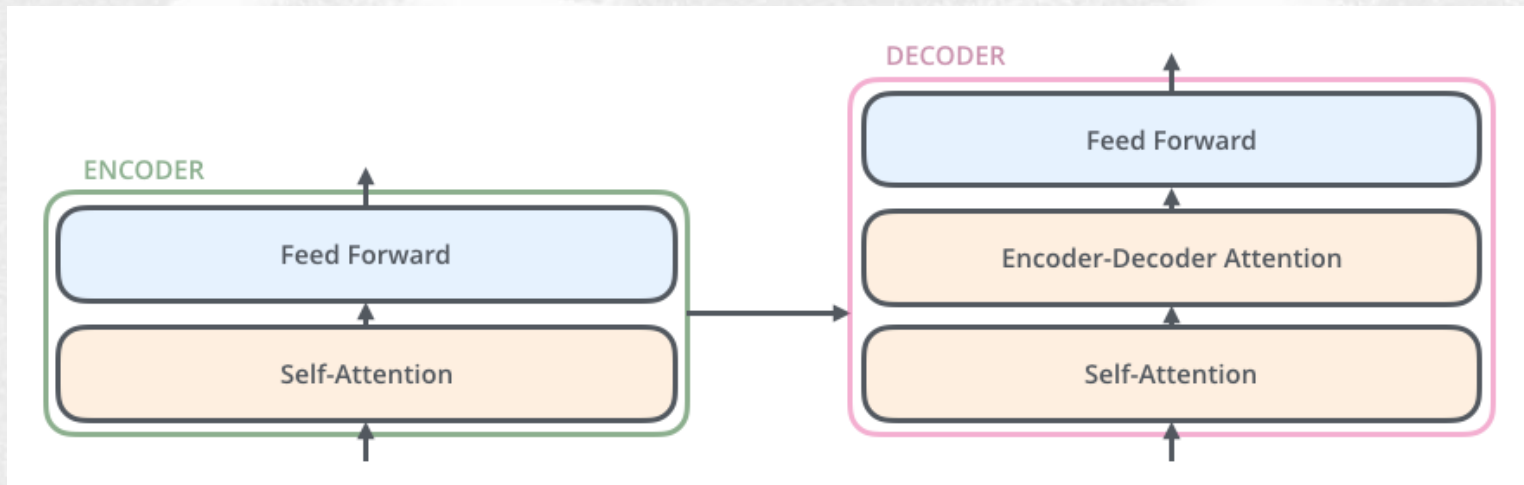
# Advantages of Attention

- **Targeted Focus in Decoding:**
  - The decoder, with attention, can **strategically concentrate on relevant segments** of the source text
  - leading to more coherent and **contextually accurate** translations.
- **Addressing Vanishing Gradient Problem:** The mechanism offers a solution to the vanishing gradients issue
  - creating shortcuts between distant states in the sequence, facilitating smoother gradient flow during backpropagation.
- **Enhancing Model Interpretability:** we gain insights into what the model focuses on at each step

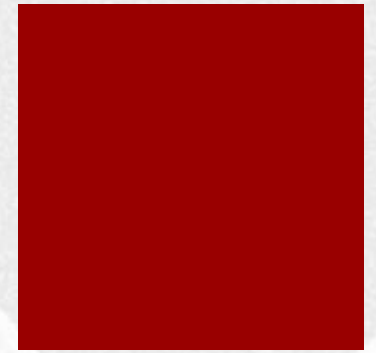
# The Transformer was only the beginning

A transformer is made of two components

- Encoder
- Decoder

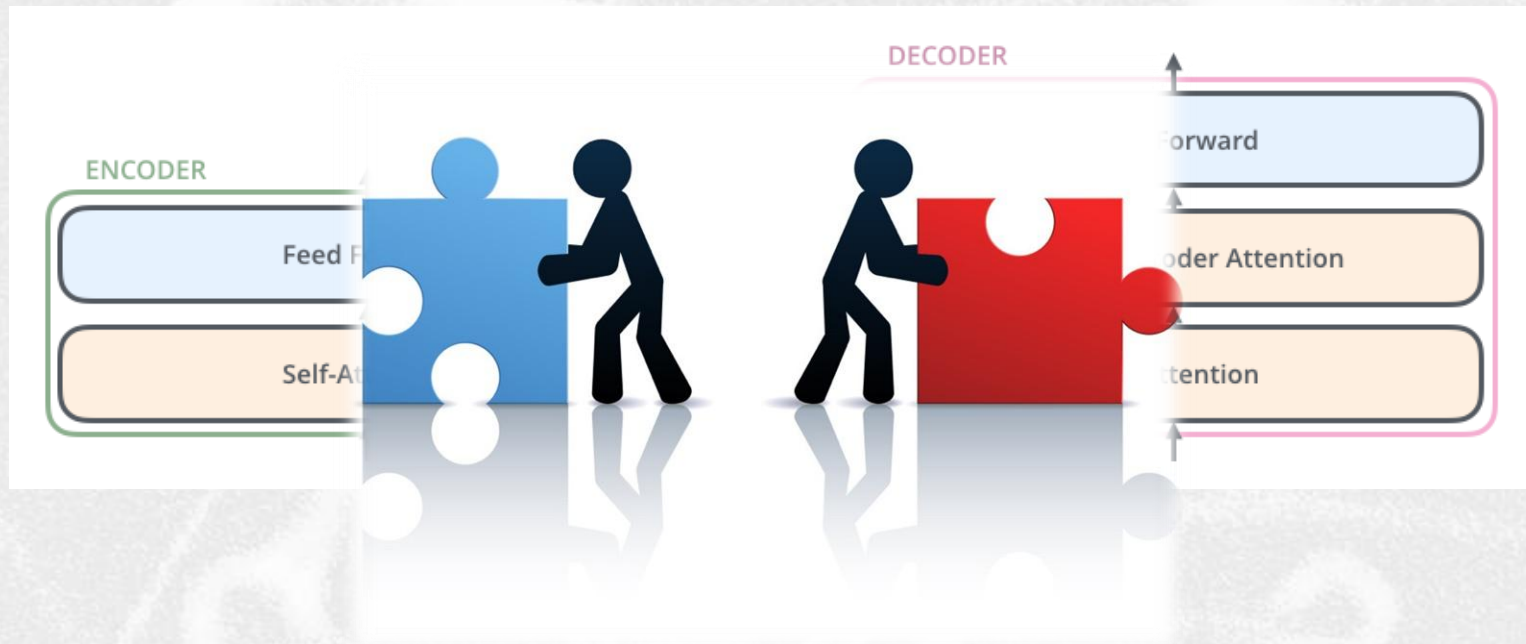


# The Transformer was only the beginning



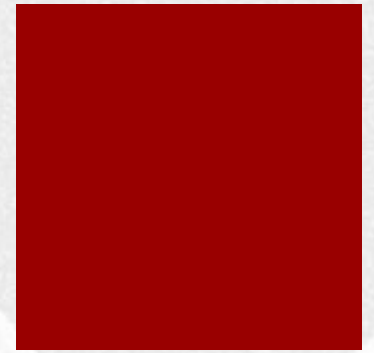
A transformer is made of two components

- Encoder
- Decoder

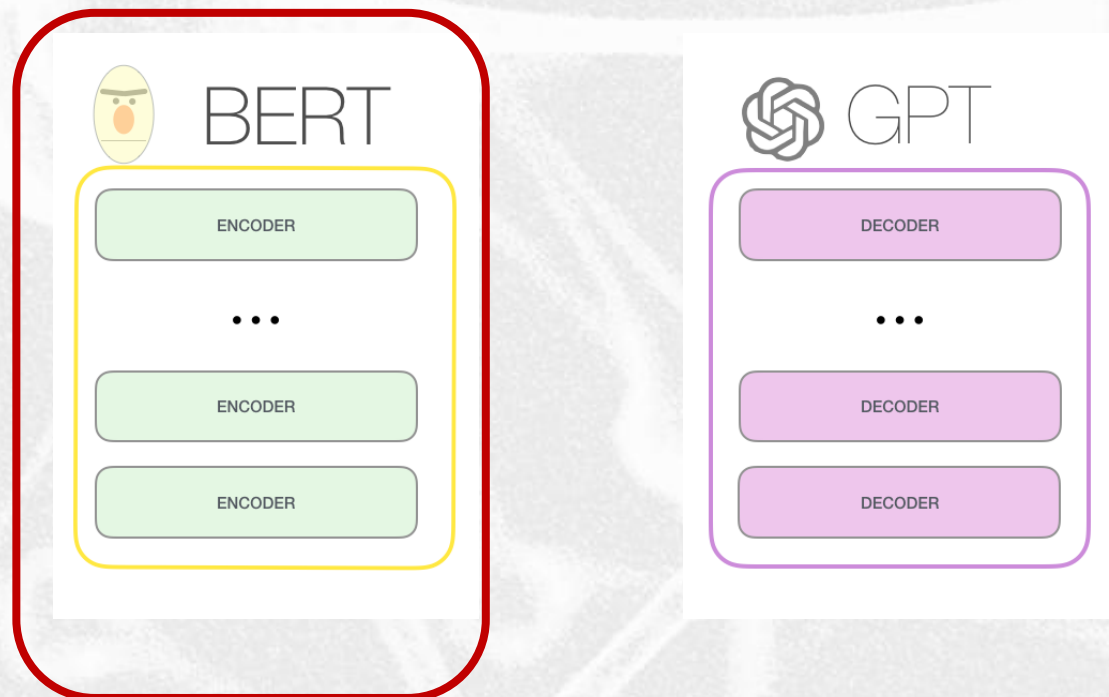




# The transformer was only the beginning



- This separation led to two «classes» of methods
  - **«Encoder-only» models:** the most famous one is BERT
  - **«Decoder-only» models:** the most famous one is GPT

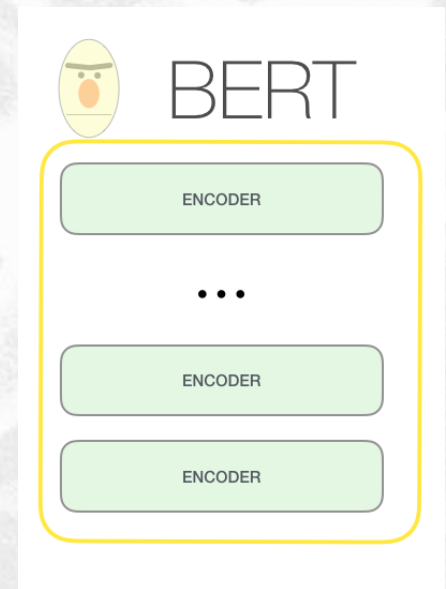




# BERT (Devlin et al, 2018)

## Bidirectional Encoder Representations from Transformers

- Only the encoder is used
- Designed to generate **contextual meaningful representation** of input words
  - Representations are **context sensitive**, thanks to self-attention
  - Understand the context of a word in a sentence from **both left and right sides** (bidirectionally).
- Representations are embeddings
  - not suitable for text generation
  - ... but for many other tasks



Images from <https://jalammar.github.io/illustrated-bert/>

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805.

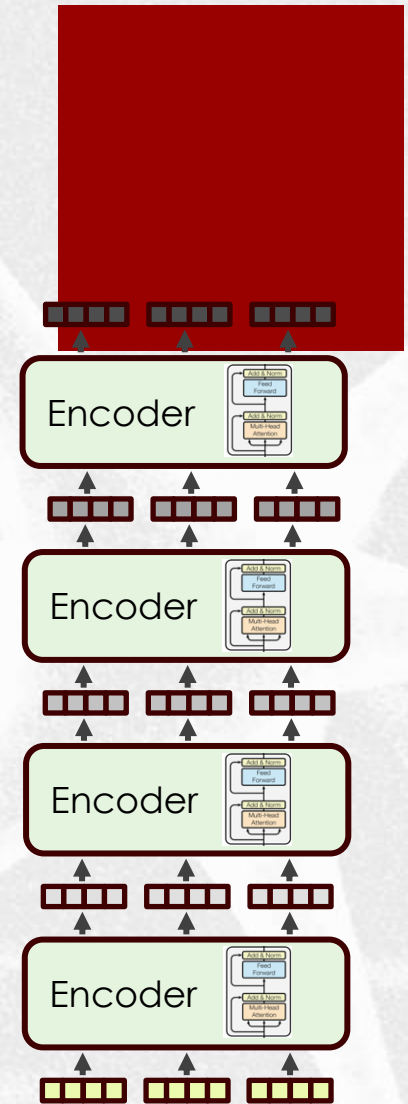
# BERT (Devlin et al, 2018)

## 🤖 Why should it work?

- It is just a piece of the Transformer architecture seen a few slides ago.

## 💡 The GREAT IDEA: Pre-Training the encoder

- Pre-trained on a large corpus of text and then fine-tuned for specific tasks like question answering, sentiment analysis, etc.



Images from <https://jalammar.github.io/illustrated-bert/>

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805.



# No pre-training no party!

## The Revolution of Pre-Training in NLP



- **Simple idea:** train a (possibly large) model on a different task and re-use it on your task
  - circumventing the need for training from scratch
  - facilitating “quicker”, more effective deployment of the model
- **Precedent in Computer Vision:**
  - This strategy mirrors developments in computer vision
  - Architectures pre-trained on classification tasks using datasets like ImageNet
  - When applied on related task, these “starting point” achieve very good results
- **Addressing Overfitting in Large Models:**
  - With **increasing model sizes** and parameter counts, the **risk of overfitting grows**
  - Pre-training on vast datasets mitigates this by providing a broad learning base.



# Pretraining BERT

- BERT takes a sequence of tokens as input
- Utilizes **self-attention across layers** to **generate context-aware** representations of each token in the sequence.
- In each layer,  $h=12$   
 $\mathbf{w}^Q, \mathbf{w}^K, \mathbf{w}^V$  matrices
- **Pre-training tasks:**
  - **Masked-language modeling**



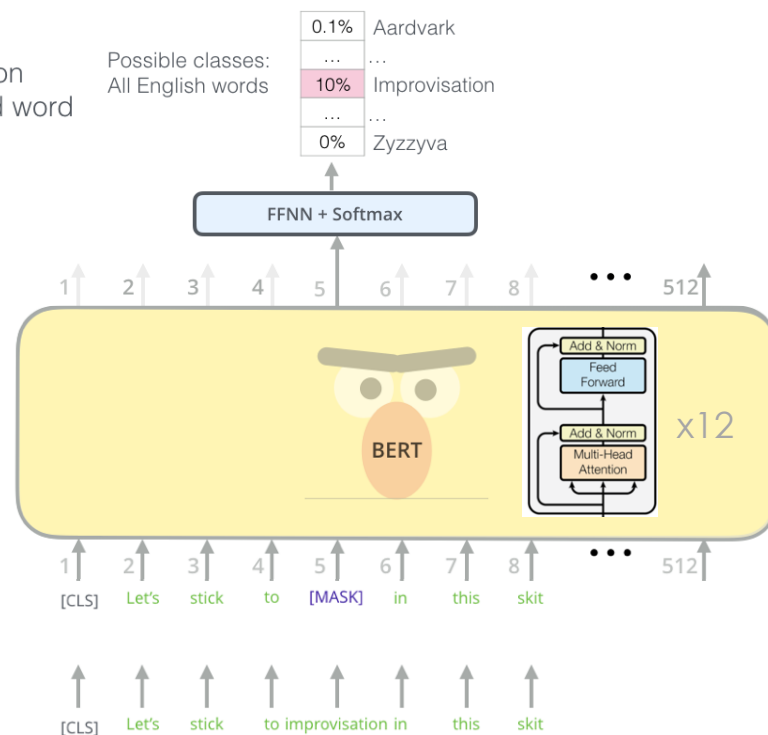
Use the output of the masked word's position to predict the masked word

Possible classes:  
All English words

0.1%	Aardvark
...	...
10%	Improvisation
...	...
0%	Zyzyva

Randomly mask 15% of tokens

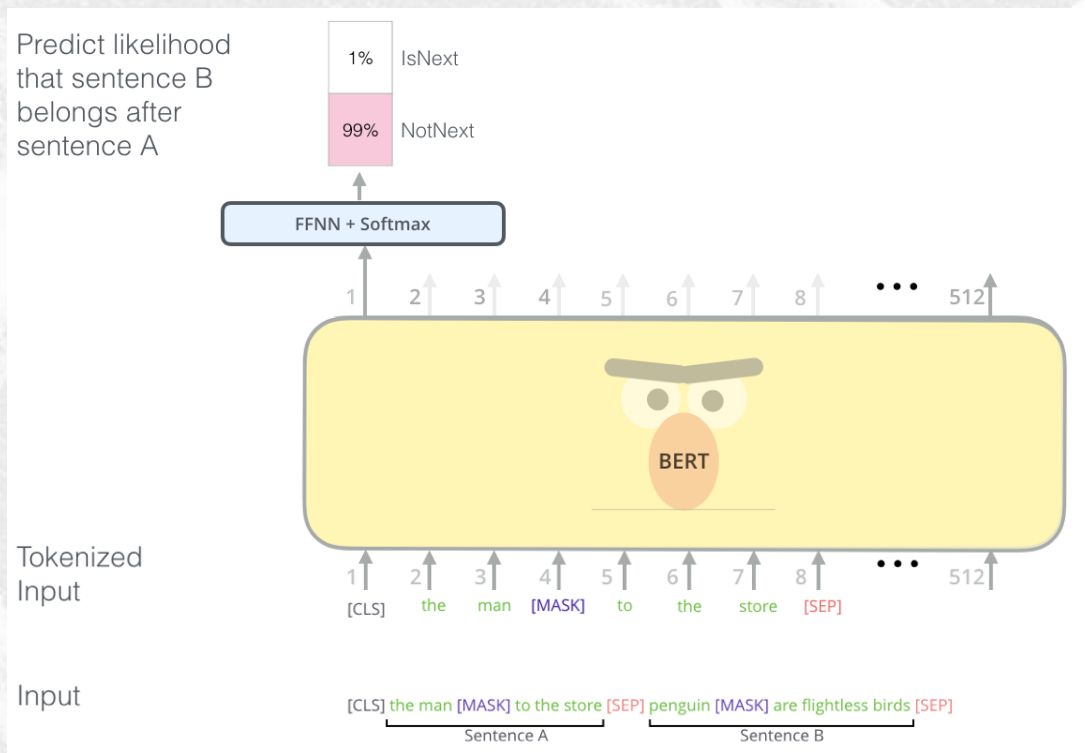
Input

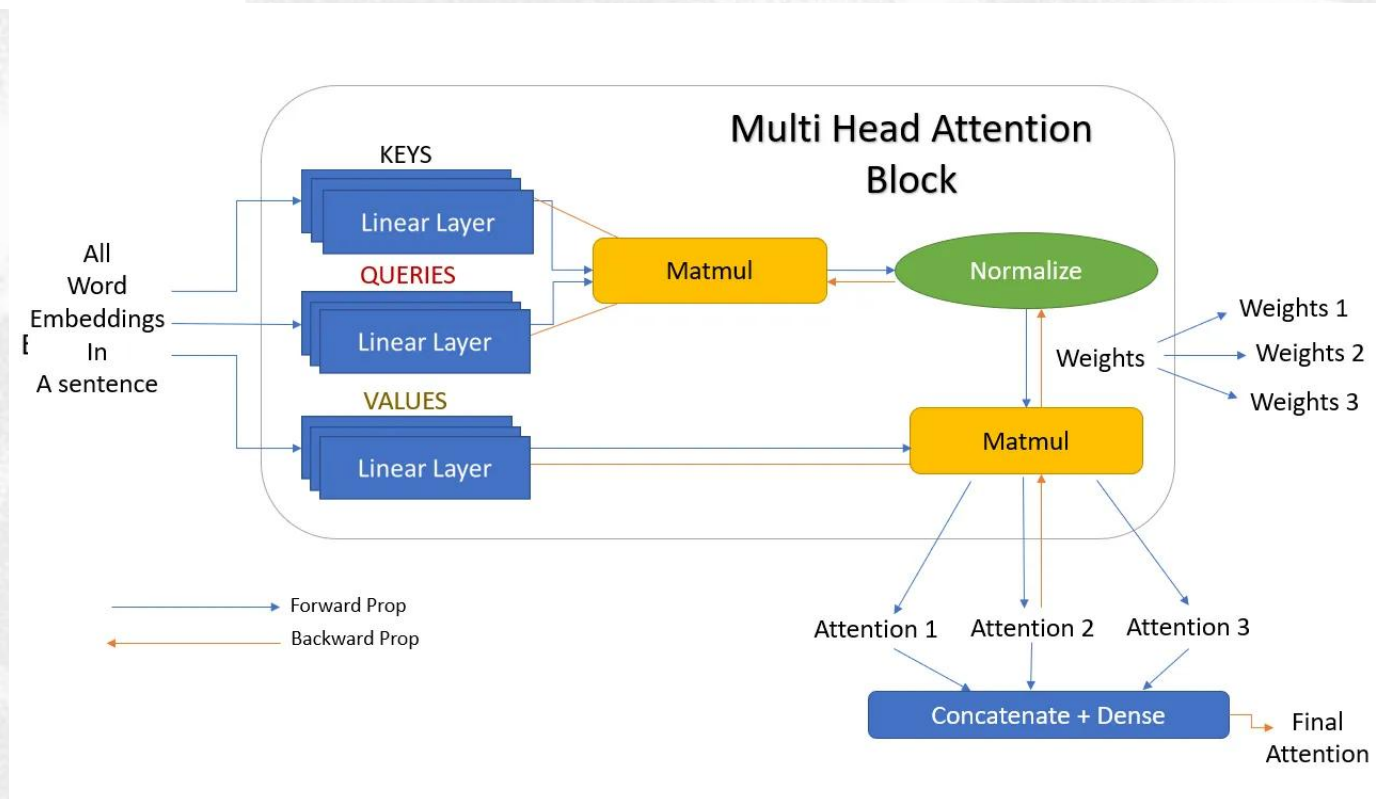
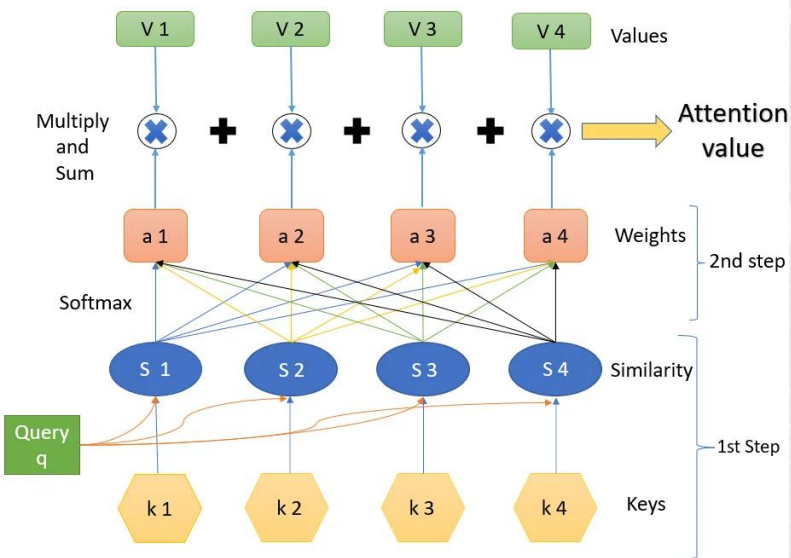


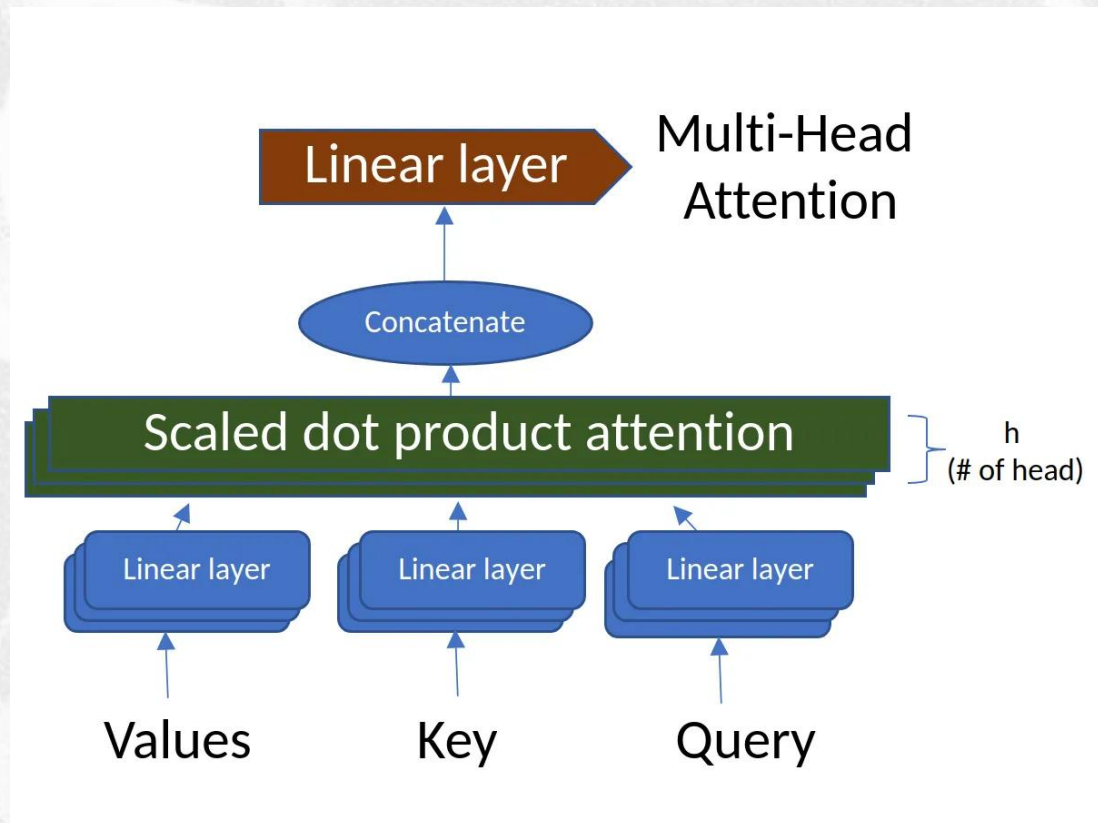
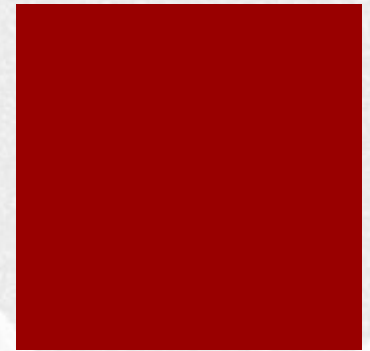
# Pretraining BERT (2)

- BERT takes a sequence of tokens as input
  - Utilizes **self-attention across layers** to **generate context-aware** representations of each token in the sequence.
  - In each layer,  $h=12$   $W^Q, W^K, W^V$  matrices
- **Pre-training tasks:**
  - Masked-language modeling
  - **Next sentence prediction**

Pretrained using the Toronto BookCorpus (800M words) and English Wikipedia (2,500M words)

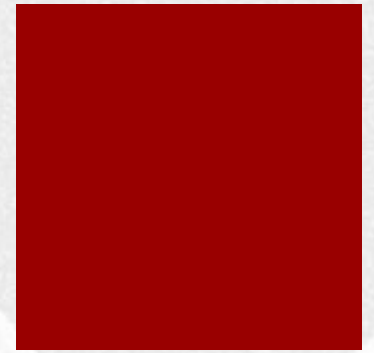








# Attention functions: examples (1)

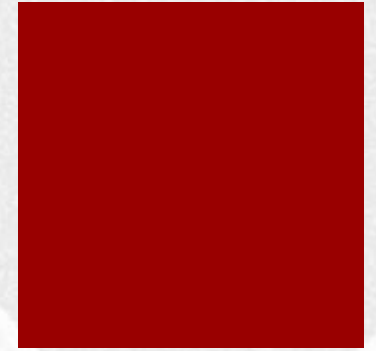


- In general, when queries and keys are vectors of different lengths, we can use additive attention as the scoring function. Given a query  $\mathbf{q} \in \mathbb{R}^q$  and a key  $\mathbf{k} \in \mathbb{R}^k$ , the *additive attention* scoring function

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{w}_v^\top \tanh(\mathbf{W}_q \mathbf{q} + \mathbf{W}_k \mathbf{k}) \in \mathbb{R},$$

- where learnable parameters  $\mathbf{W}_q \in \mathbb{R}^{h \times q}$ ,  $\mathbf{W}_k \in \mathbb{R}^{h \times k}$  and  $\mathbf{w}_v \in \mathbb{R}^h$ .
- In a learnable setting, the query and the key are concatenated and fed into an MLP with a single hidden layer whose number of hidden units is  $h$ , a hyperparameter. By using  $\tanh()$  as the activation function and disabling bias terms, we implement additive attention in the following

# Attention functions: scaled dot-product (2)



- When  $q$  and  $k$  are  $d$ -dimensional vectors whose independent dimensions have mean=0 and variance=1, their dot product has mean = 0 and a variance =  $d$ . To ensure that the variance of the dot product still remains one regardless of vector length, the *scaled dot-product attention* scoring function is adopted

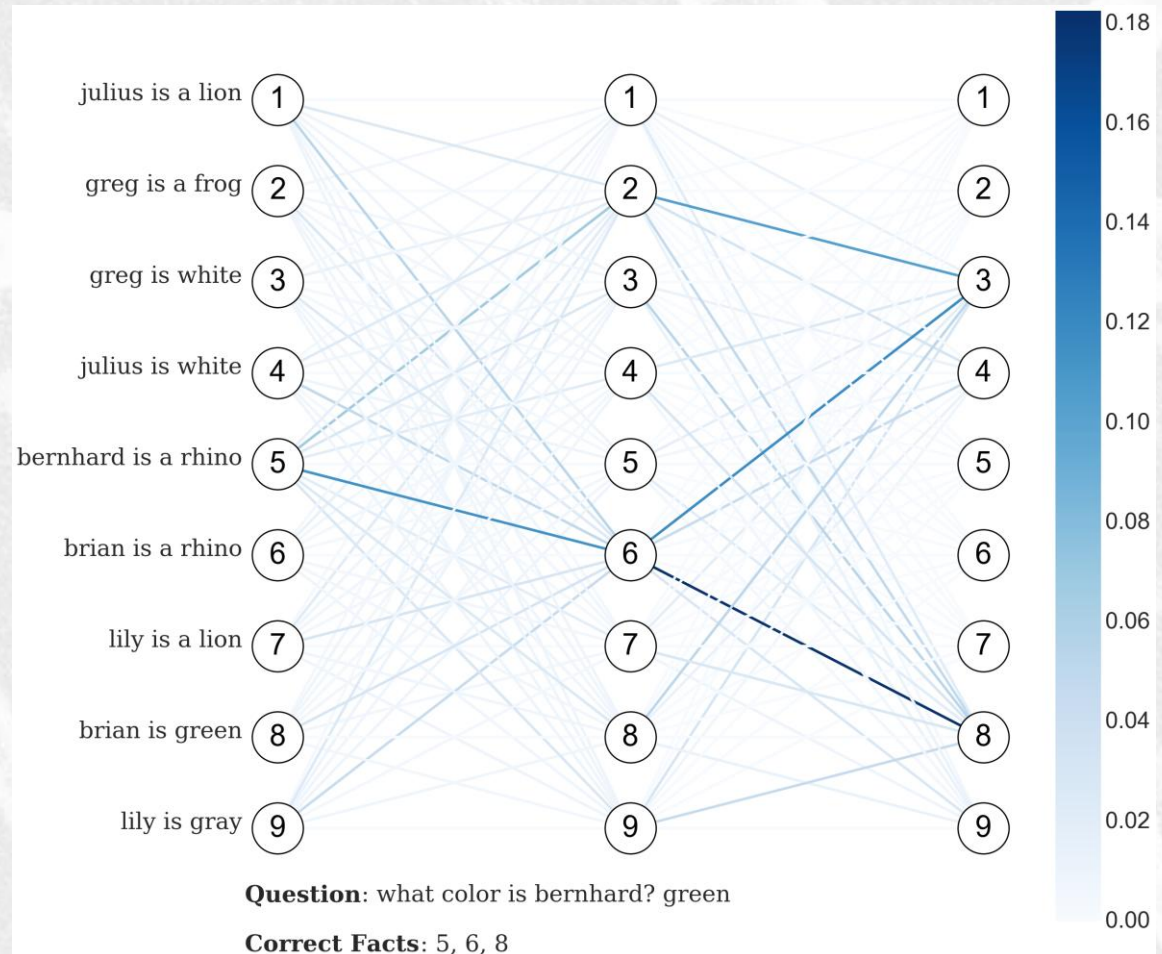
$$a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^\top \mathbf{k} / \sqrt{d}$$

- It divides the dot product by  $\sqrt{d}$ . In practice, we often think in minibatches for efficiency, such as computing attention for  $n$  queries and  $m$  key-value pairs, where queries and keys are of length  $d$  and values are of length  $v$ . The scaled dot-product attention of queries  $\mathbf{Q} \in \mathbb{R}^{n \times d}$ , keys  $\mathbf{K} \in \mathbb{R}^{m \times d}$ , and values  $\mathbf{V} \in \mathbb{R}^{m \times v}$  is

$$\text{softmax} \left( \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}} \right) \mathbf{V} \in \mathbb{R}^{n \times v}.$$

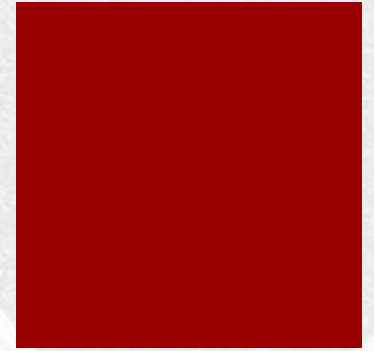
# Visualization of the attention distribution in QA

- Supporting fact sequences for an example question
- On the right the attentions over facts for individual sequences
  - Each sequence is mapped into a Markov process





# Attention & encoding

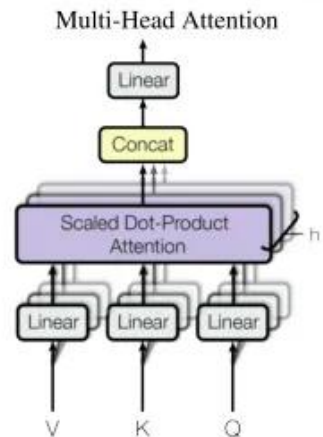
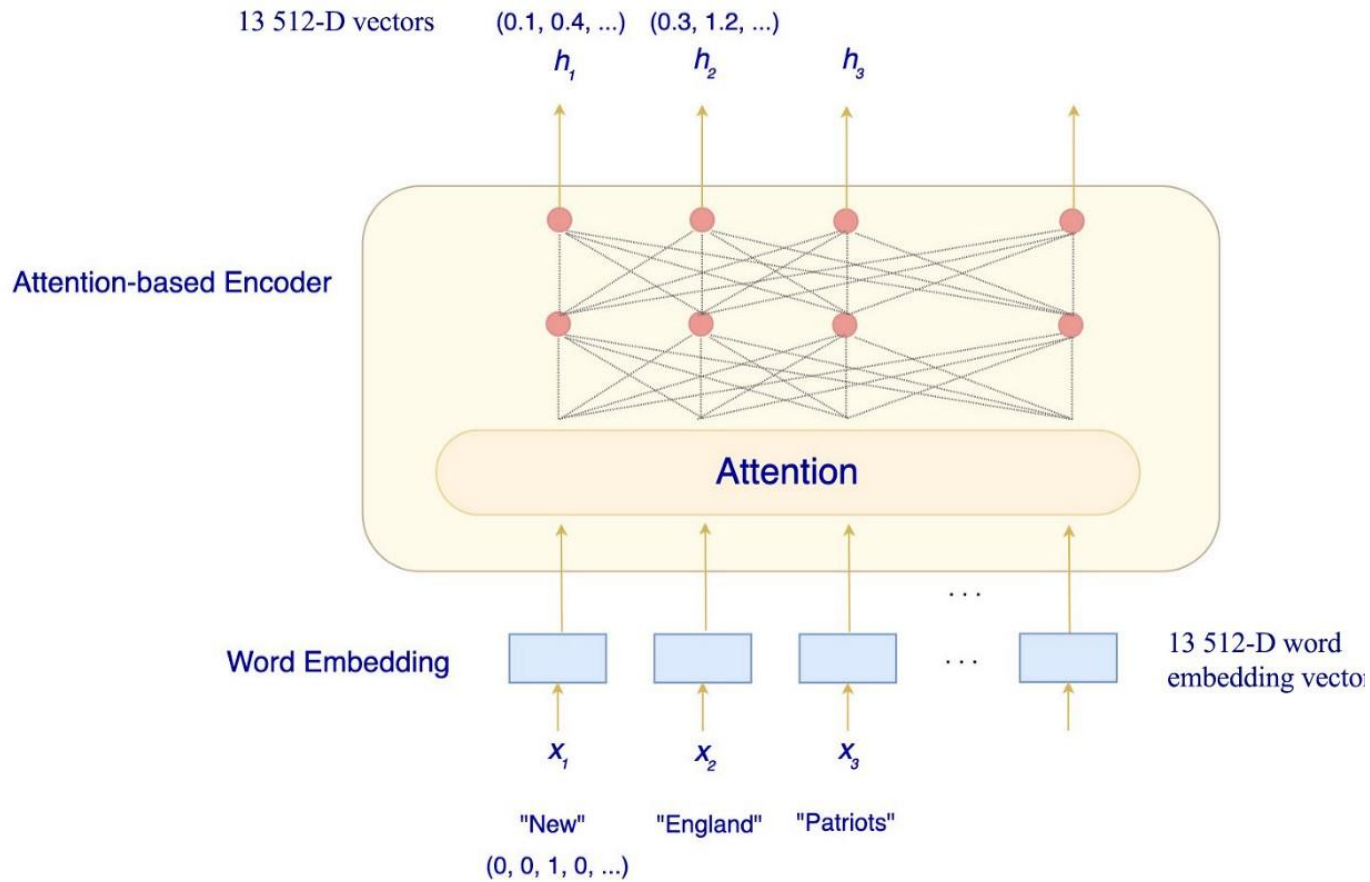


- IN a decoding process (e.g. machine translation) there are **three** kinds of dependencies for neural architectures
- Dependencies can establish between
- (1) the ***input and output*** tokens
- (2) the ***input tokens themselves***
- (3) the ***output tokens themselves***
- Examples:
  - MT
  - QA where the query the answer paragraph is the input and the matched answer is the output



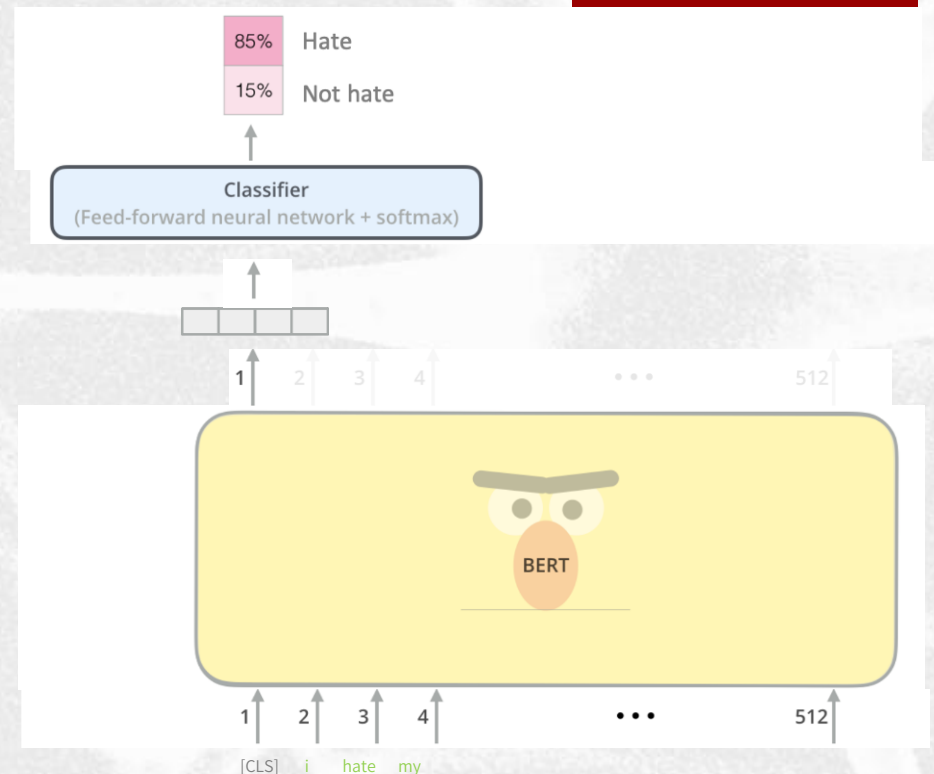
# BERT & NLP

## Encoder

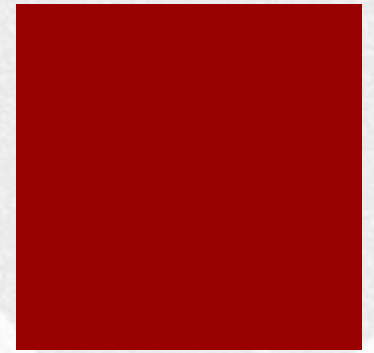


# BERT and fine-tuning

- Once pretrained, we can **apply it to new sentences**
- BERT will **produce encoded representations** for each input symbol
- And it can be used in **different classification tasks**, just adding a new (linear) classifier...
- ... through fine-tuning of the entire architecture
- not trivial to forget what learned during the pre-training



# Towards Foundation Models

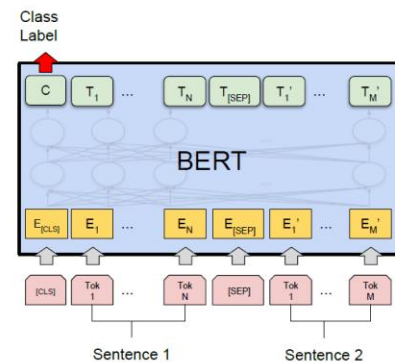


- **Emergence of Foundation Models in NLP:**
  - Large-scale models trained on linguistic tasks, forming a versatile base that can be fine-tuned for various specific applications.
- **Everybody worked on customizing Foundation Models:**
  - Leverage the extensive knowledge encapsulated in Foundation Models by fine-tuning them for particular NLP tasks.
- If you are interested in foundation models
  - [Zhou et al, 2023] A Comprehensive Survey on Pretrained Foundation Models: A History from BERT to ChatGPT
  - <https://arxiv.org/abs/2302.09419>

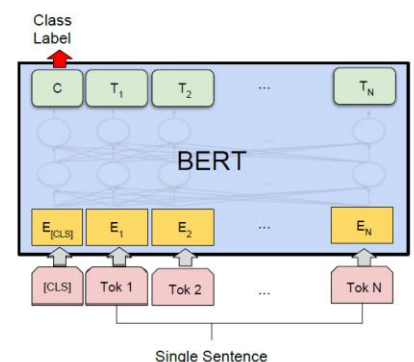


# BERT in action

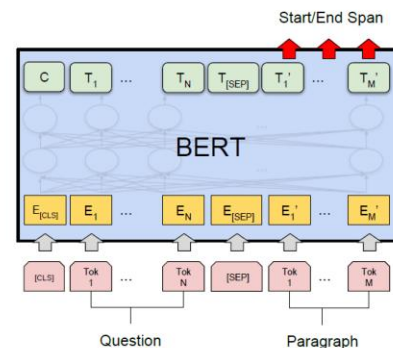
- The final layer outputs hidden representations, which can be utilized with a simple linear classifier
- to address a broad spectrum of NLP tasks efficiently.



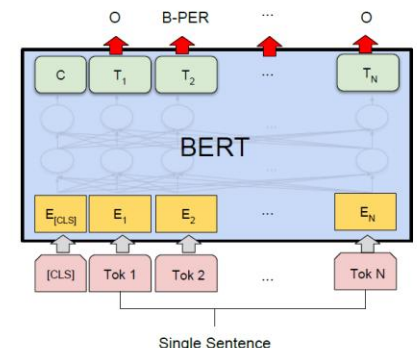
(a) Sentence Pair Classification Tasks:  
MNLI, QQP, QNLI, STS-B, MRPC,  
RTE, SWAG



(b) Single Sentence Classification Tasks:  
SST-2, CoLA



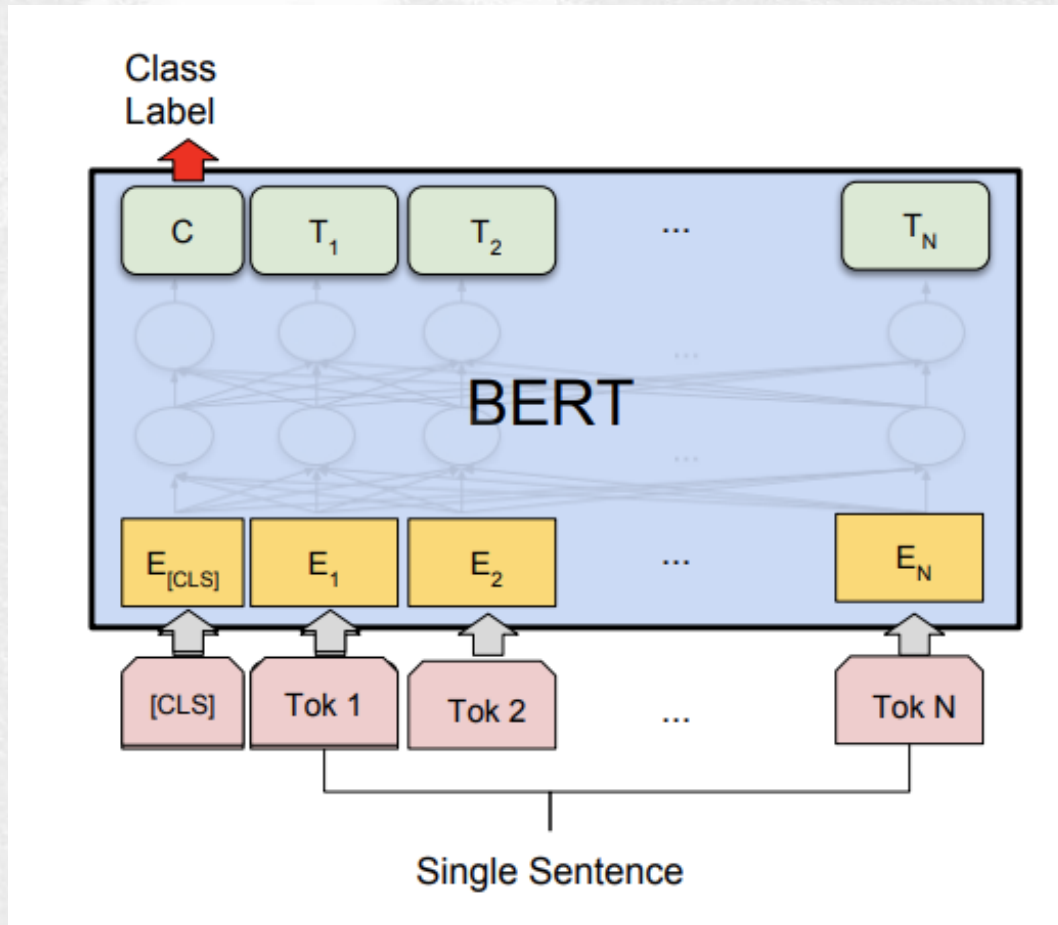
(c) Question Answering Tasks:  
SQuAD v1.1



(d) Single Sentence Tagging Tasks:  
CoNLL-2003 NER

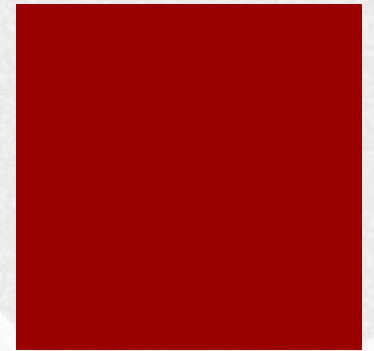
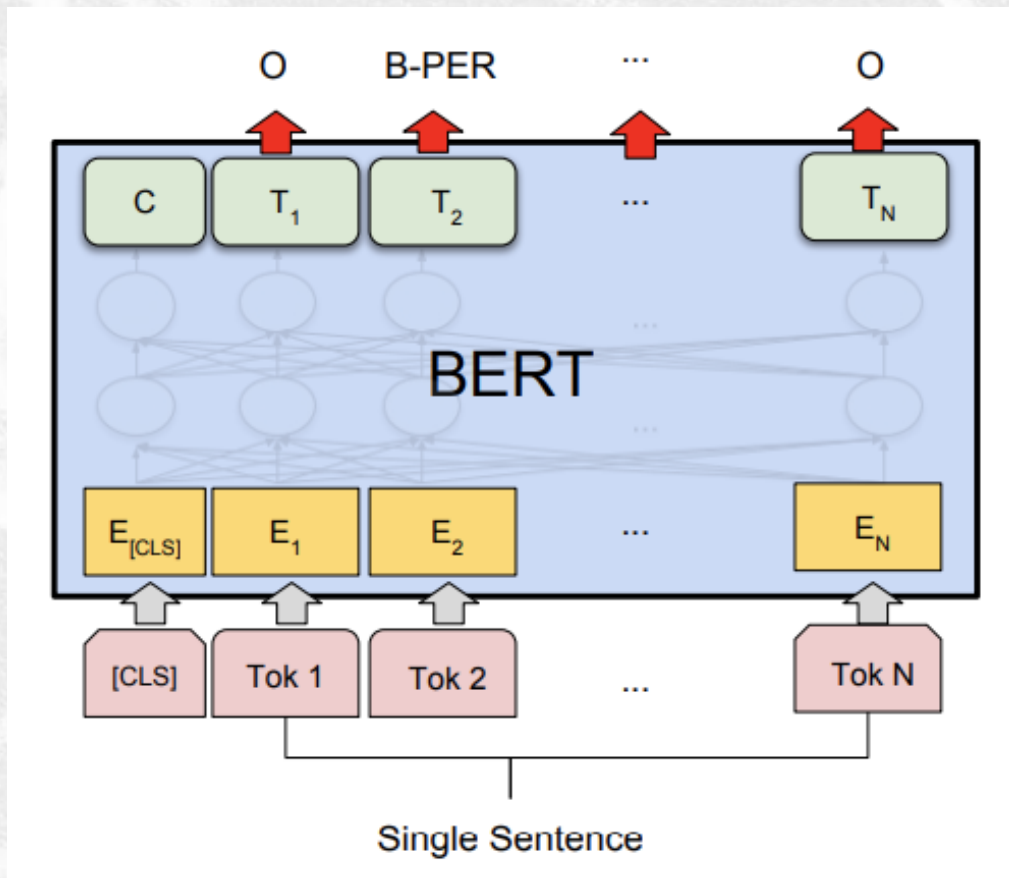


# BERT (Devlin et al. '18)



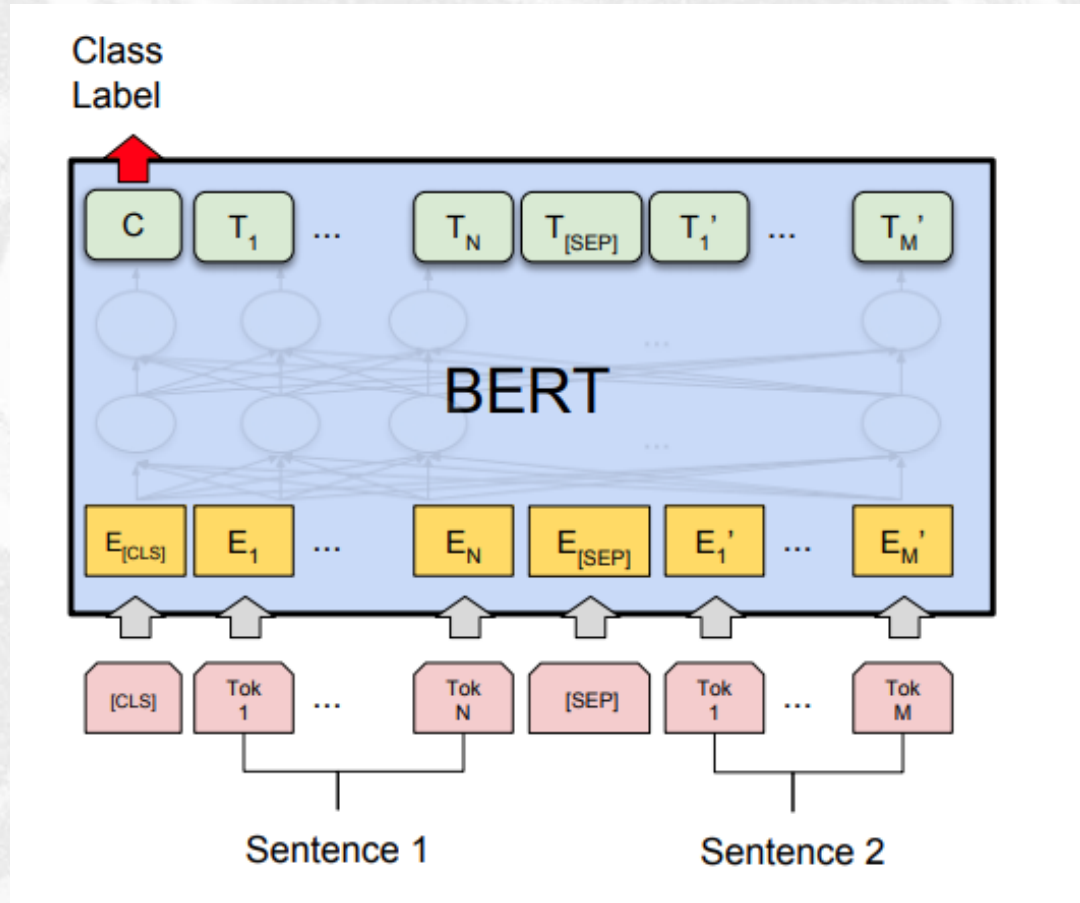
BERT for single sentence classification (Sentiment analysis, Intent Classification, etc.)

# BERT (Devlin et al. '18)



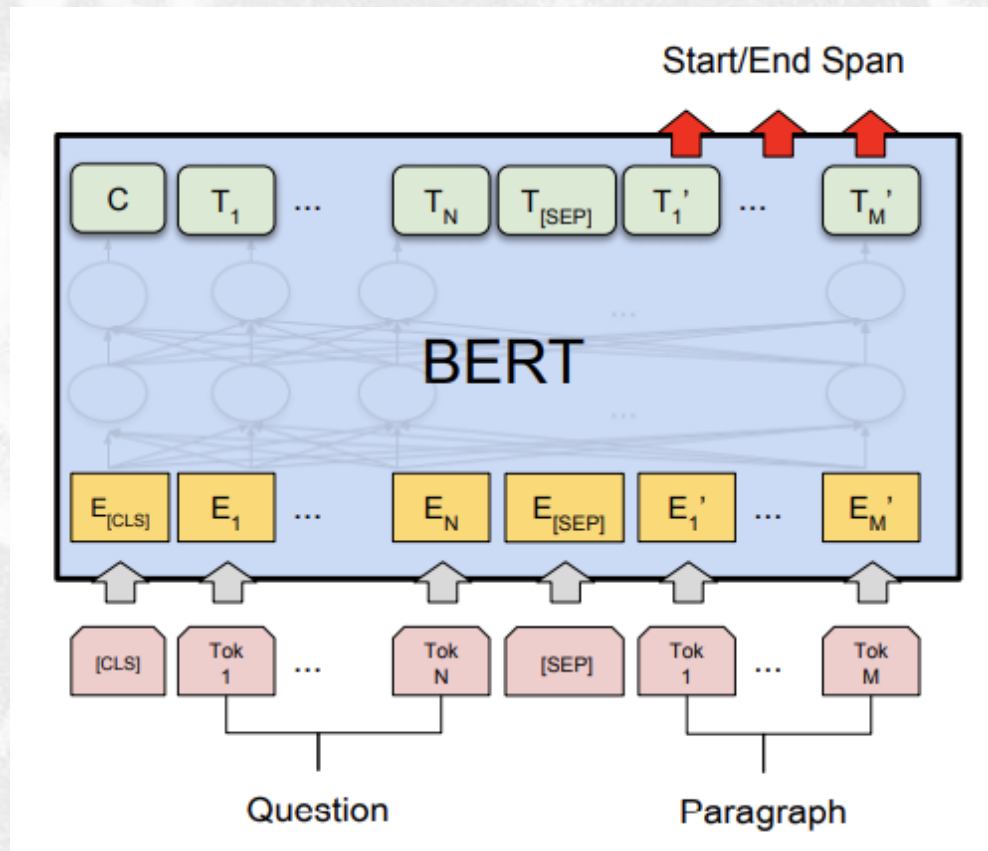
BERT for Sequence Tagging Tasks (e.g., POS tagging, Named Entity Recognition, etc.)

# BERT (Devlin et al. '18)



BERT for sentence pairs classification (Paraphrase Identification, answer selection in QA, Recognizing Textual Entailment)

# BERT (Devlin et al. '18)

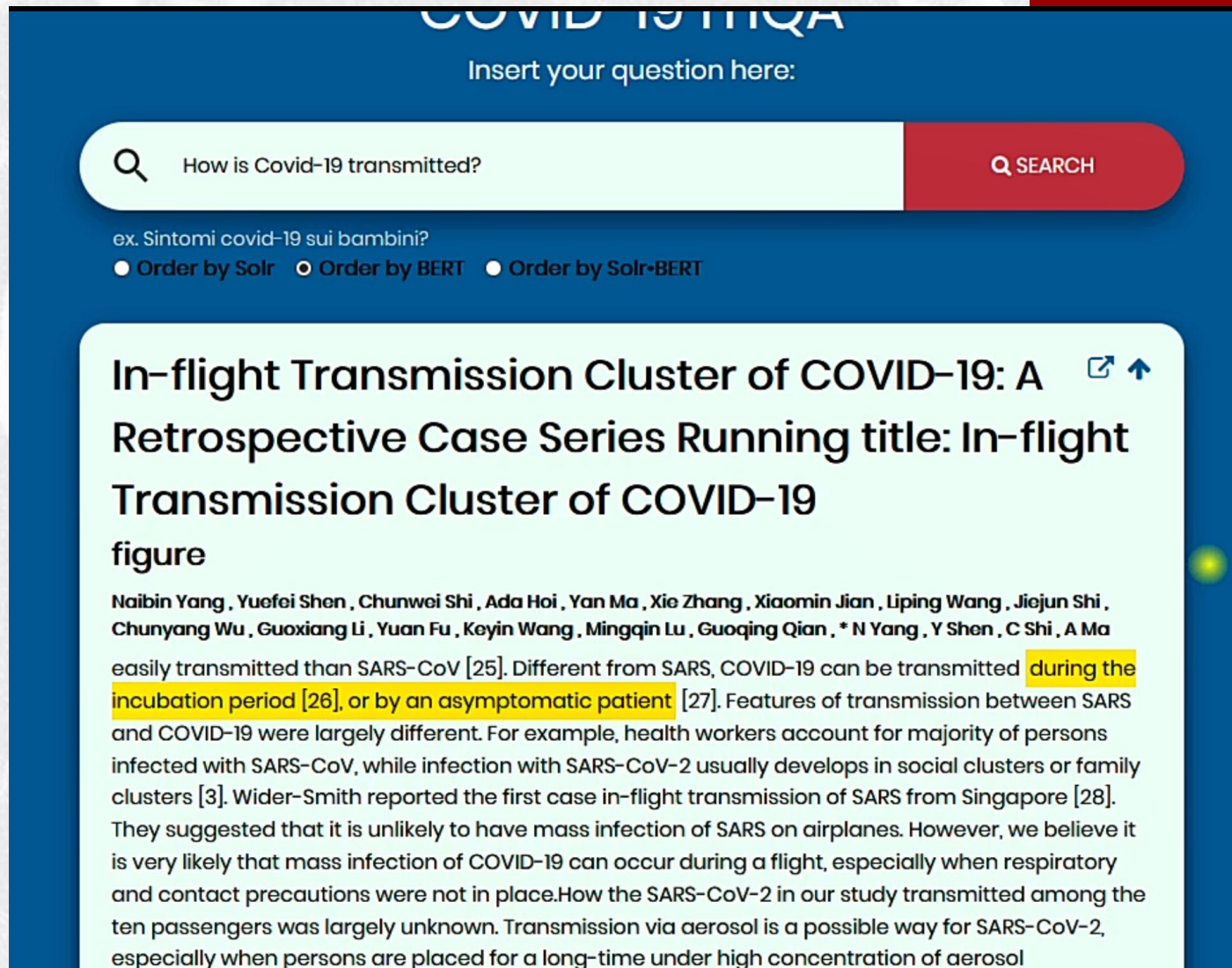


BERT for Answer Span Selection in Question Answering



# A QA example on SquAD

## ■ Cross-lingual Question Answering





COVID-19 FAQ

Insert your question here:

Q How is Covid-19 transmitted? SEARCH

ex. Sintomi covid-19 sui bambini?

● Order by Solr ● Order by BERT ● Order by Solr•BERT

**In-flight Transmission Cluster of COVID-19: A Retrospective Case Series**  

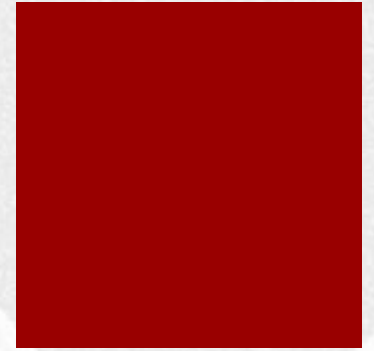
**Running title: In-flight Transmission Cluster of COVID-19**

**figure**

Naibin Yang , Yuefei Shen , Chunwei Shi , Ada Hoi , Yan Ma , Xie Zhang , Xiaomin Jian , Liping Wang , Jiejun Shi , Chunyang Wu , Guoxiang Li , Yuan Fu , Keyin Wang , Mingqin Lu , Guoqing Qian , \* N Yang , Y Shen , C Shi , A Ma

easily transmitted than SARS-CoV [25]. Different from SARS, COVID-19 can be transmitted during the incubation period [26], or by an asymptomatic patient [27]. Features of transmission between SARS and COVID-19 were largely different. For example, health workers account for majority of persons infected with SARS-CoV, while infection with SARS-CoV-2 usually develops in social clusters or family clusters [3]. Wider-Smith reported the first case in-flight transmission of SARS from Singapore [28]. They suggested that it is unlikely to have mass infection of SARS on airplanes. However, we believe it is very likely that mass infection of COVID-19 can occur during a flight, especially when respiratory and contact precautions were not in place. How the SARS-CoV-2 in our study transmitted among the ten passengers was largely unknown. Transmission via aerosol is a possible way for SARS-CoV-2, especially when persons are placed for a long-time under high concentration of aerosol

# BERT (Devlin et al. '18)



**Pretraining** on two unsupervised prediction tasks:

- **Masked Language Model:** given a sentence  $s$  with missing words, reconstruct  $s$ 
  - Example: Amazon <MASK> amazing  $\rightarrow$  Amazon is amazing
  - In BERT the language modeling is deeply Bidirectional, while in ELMo the forward and backward LMs were two independent branches of the NN
- **Next Sentence Prediction:** given two sentences  $s_1$  and  $s_2$ , the task is to understand whether  $s_2$  is the actual sentence that follows  $s_1$ 
  - 50% of the training data are positive examples:  $s_1$  and  $s_2$  are actually consecutive sentences
  - 50% of the training data are negative examples:  $s_1$  and  $s_2$  are randomly chosen from the corpus

# BERT pretraining: Input representations

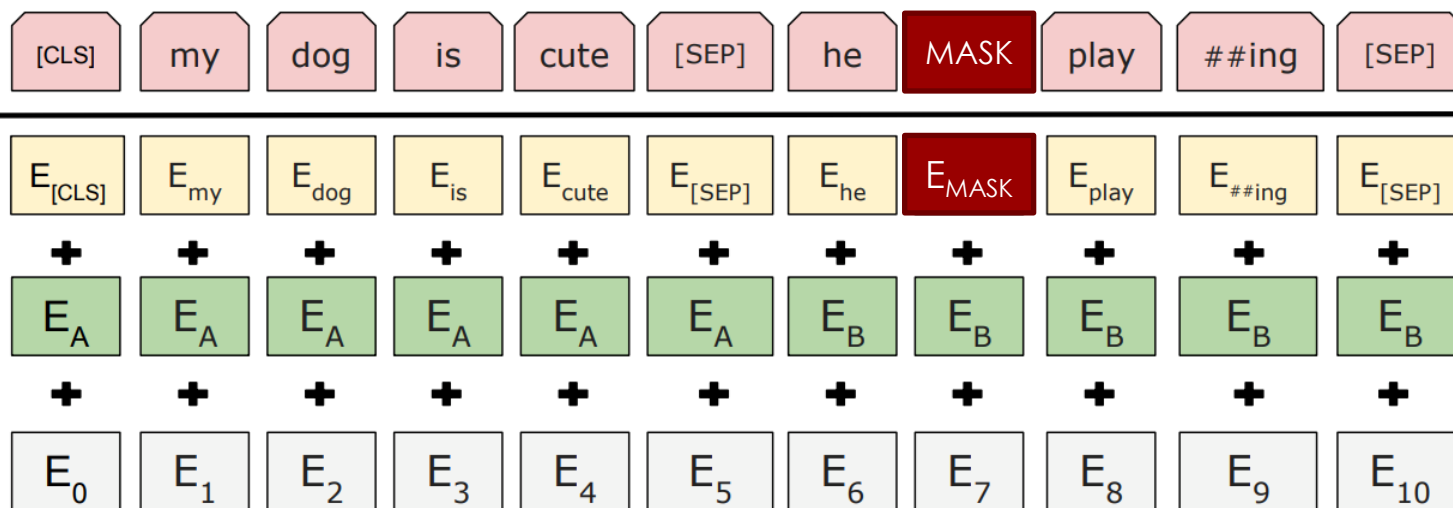


INPUT

WordPieces  
Embeddings

Sentence  
Embeddings

Position  
Embeddings



All these embeddings  
are learned during the  
(pre)training process

In pre-training 15% of the input tokens  
are masked for the masked LM task



# A complex application of LSTM (and recently Transformers): Image captioning



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



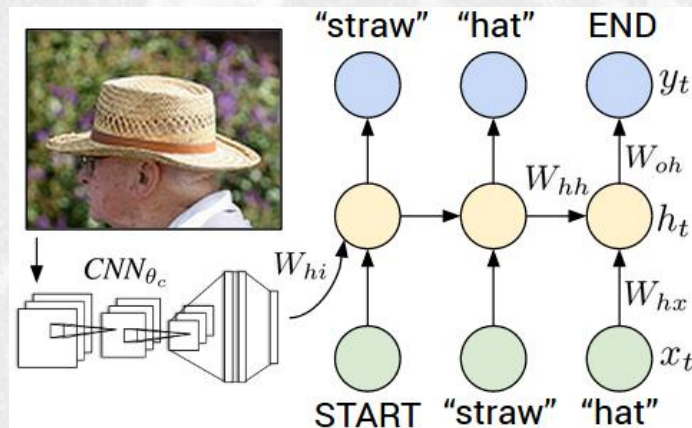
A stop sign is on a road with a mountain in the background.





# Image Captioning

- Image to captions
  - Convolutional Neural Network to learn a representation of the image
  - (Bi-directional) Recurrent Neural Network to generate a caption describing the image
    - its input is the representation computed from the CNN
    - its output is a sequence of words, i.e. the caption

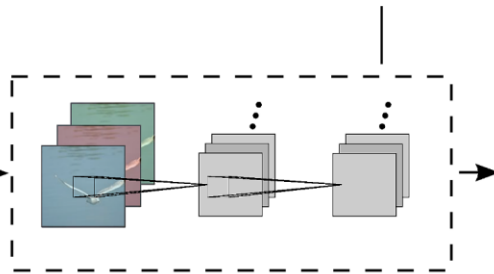


"baseball player is throwing ball in game."

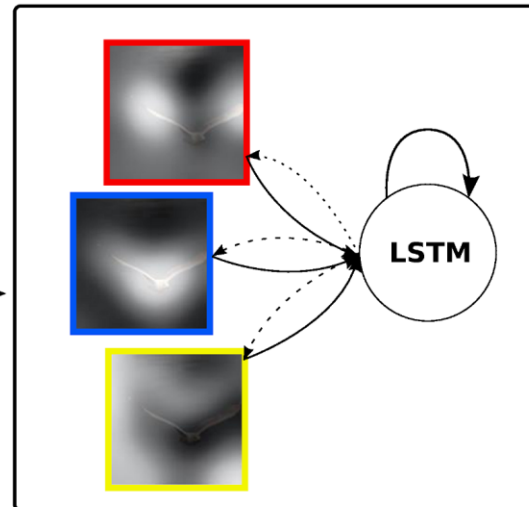


1. Input Image

14x14 Feature Map



2. Convolutional Feature Extraction



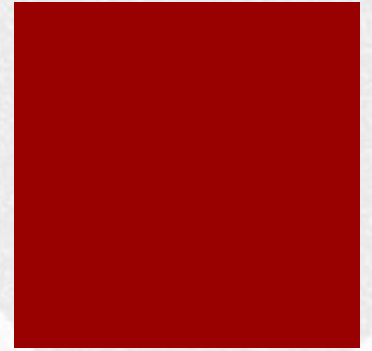
3. RNN with attention over the image



**A**  
**bird**  
flying  
over  
a  
body  
of  
**water**

4. Word by word generation

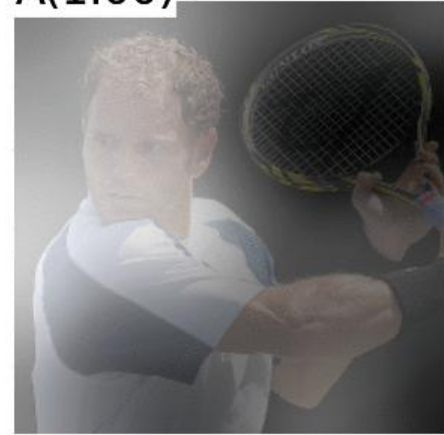
# Attention: a dynamic rendering



A(0.99)



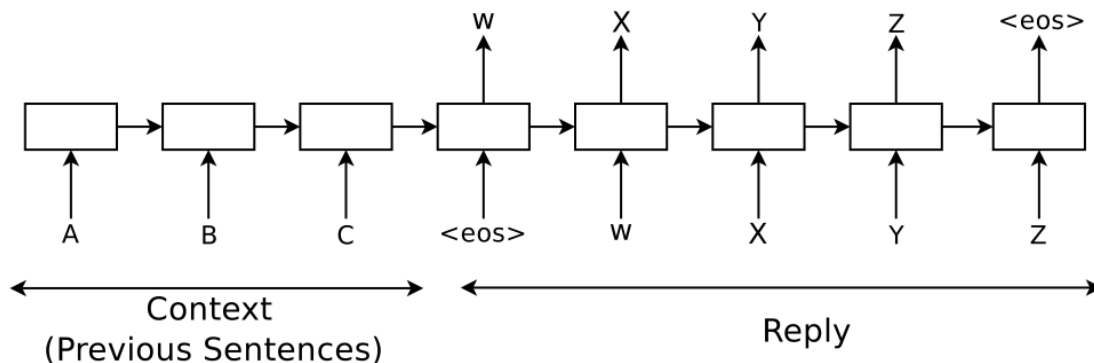
A(1.00)





# Neural Conversation Models

- Learn to answer question in a conversation
- For example, an IT help desk
  - model the input sequence
  - produce a output sequence



**Machine:** *hi*

**Human:** *hello*

**Machine:** *could you please let me know what are the operating systems you are using ?*

**Human:** *i am using linux*

**Machine:** *ok it was in the ssl vpn on which you stated it was broken*

**Machine:** *please check if you are logged in to corp network*

**Machine:** *what is the error that you are running please*

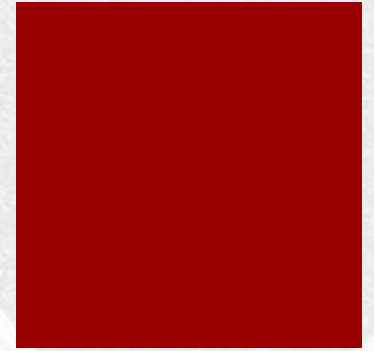
**Human:** *i am seeing an error related to vpn*

**Machine:** *what is the error message that you are getting when connecting to vpn using network connect ?*

**Human:** *connection refused or something like that*



# Transformers



- (Vaswani 2017), Attention is all you need, <https://arxiv.org/abs/1706.03762>
- (Devlin et al 2018), BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, <https://arxiv.org/abs/1810.04805>
- An interesting introduction to the attention mechanism:
  - **All you need to know about 'Attention' and 'Transformers' — In-depth Understanding — Part 1**, A. Sarkar, URL: <https://towardsdatascience.com/all-you-need-to-know-about-attention-and-transformers-in-depth-understanding-part-1-552f0b41d021#4c16>
- Other Task specific works:
  - Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. CoRR, abs/1409.0473, 2014.
  - Effective Approaches to Attention-based Neural Machine Translation, Minh-Thang Luong Hieu Pham Christopher D. Manning, 2015, <https://arxiv.org/abs/1508.04025v5>
  - Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In International Conference on Learning Representations, 2017.