人工智能之NLP

Bert

主讲人：GerryLiu

课程要求

- 课上课下“九字”真言
  - 认真听，**善摘录，勤思考**
  - **多温故，乐实践**，再发散
- 四不原则
  - 不懒散惰性，不迟到早退
  - 不请假旷课，不拖延作业
- 一点注意事项
  - 违反“四不原则”，不推荐就业

课程内容

- Seq2Seq结构和Attention结构回顾
- Transformer结构讲解
- Bert结构讲解

Seq2Seq结构回顾

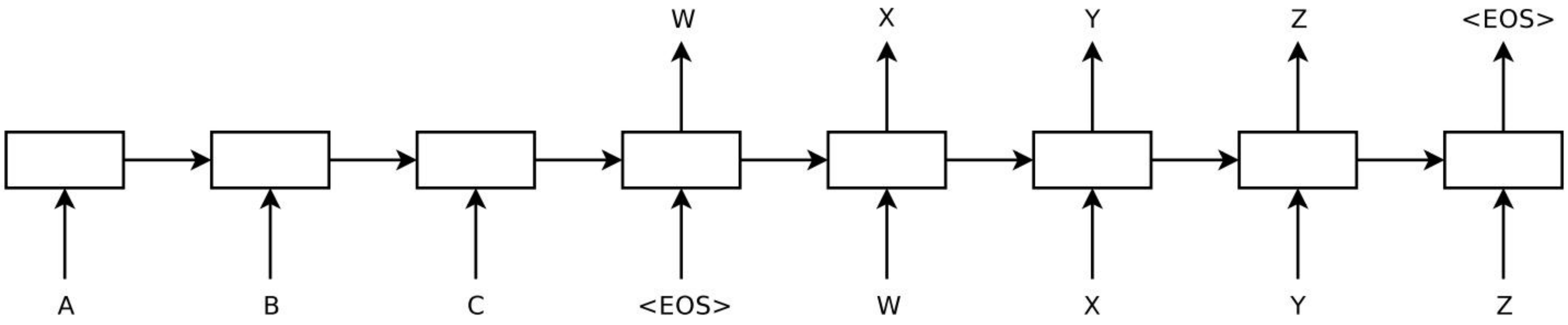
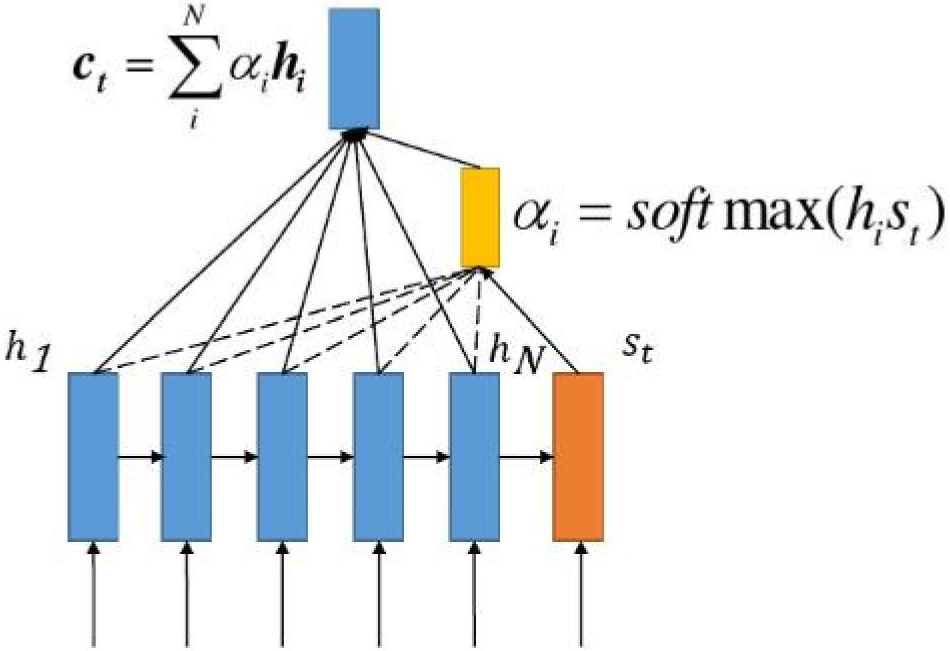
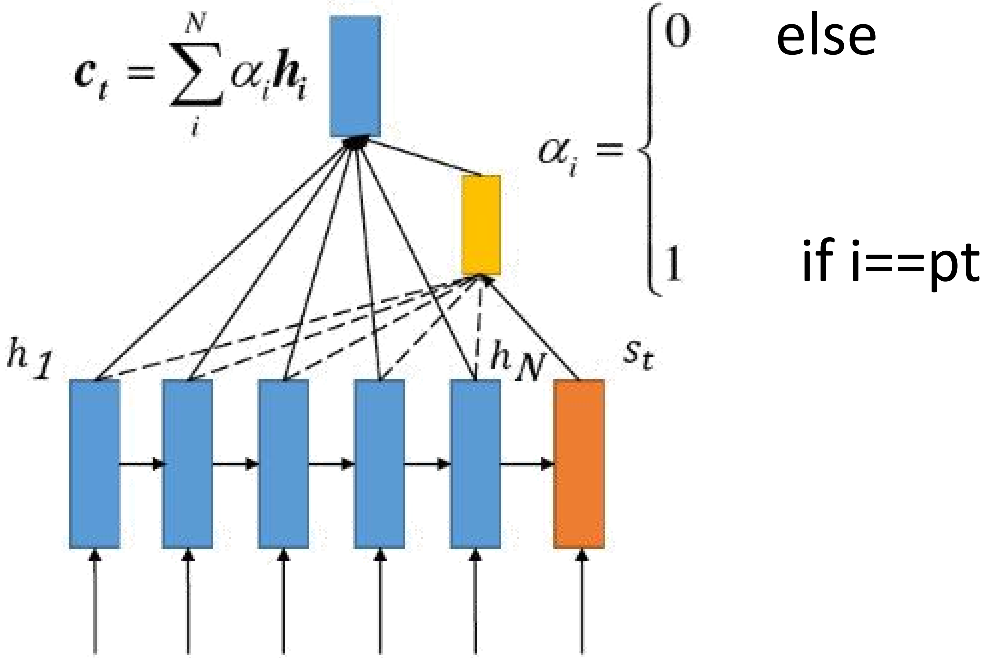


Figure 1: Our model reads an input sentence “ABC” and produces “WXYZ” as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

Attention结构回顾



Soft Attention



Hard Attention

Attention结构回顾

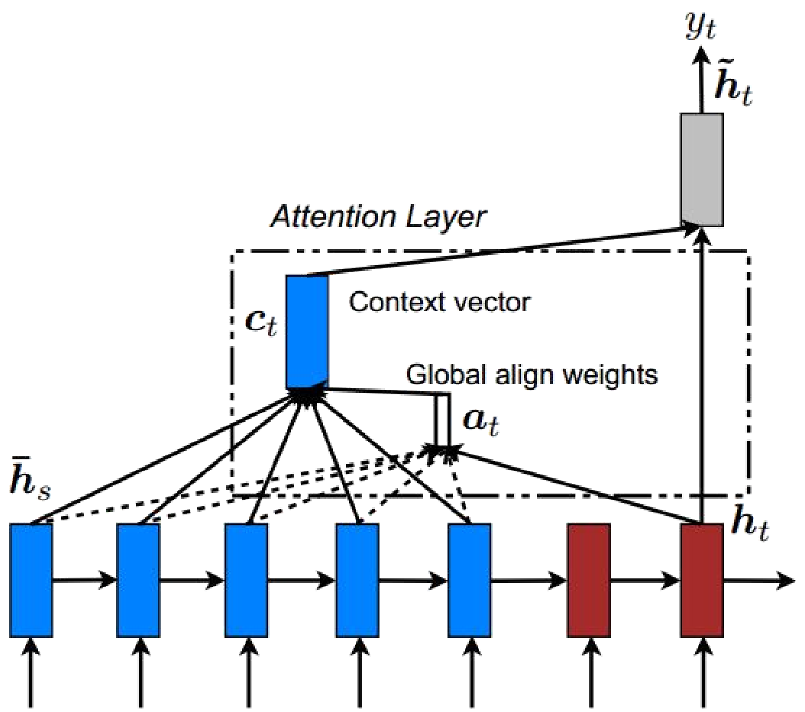


Figure 2: **Global attentional model** – at each time step t, the model infers a *variable-length* alignment weight vector $\boldsymbol{a}_t$ based on the current target state $\boldsymbol{h}_t$ and all source states $\bar{\boldsymbol{h}}_s$. A global context vector $\boldsymbol{c}_t$ is then computed as the weighted average, according to $\boldsymbol{a}_t$, over all the source states.

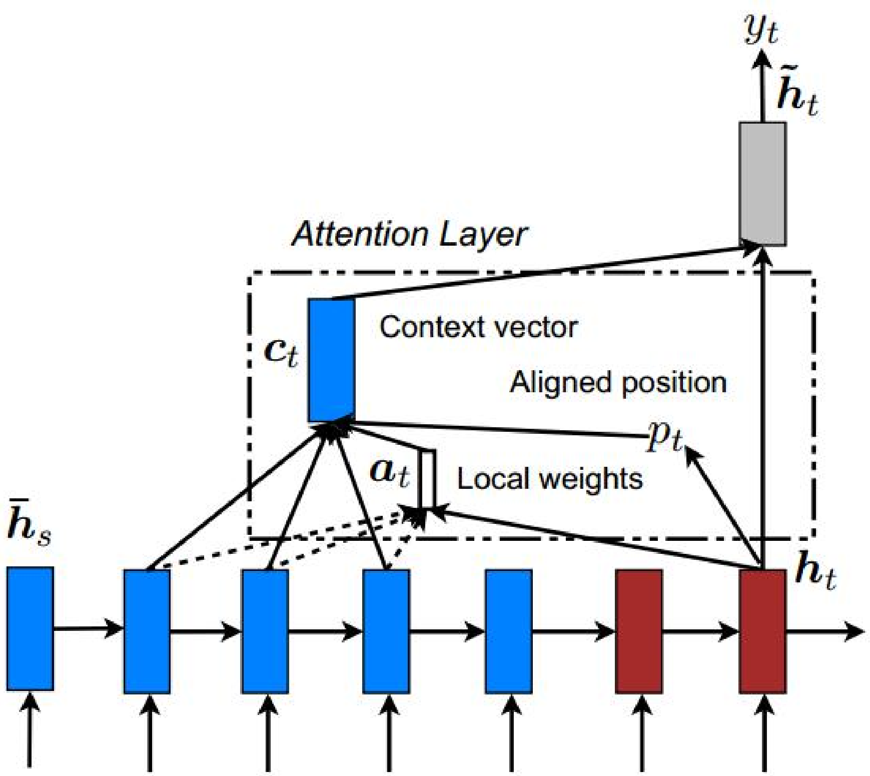
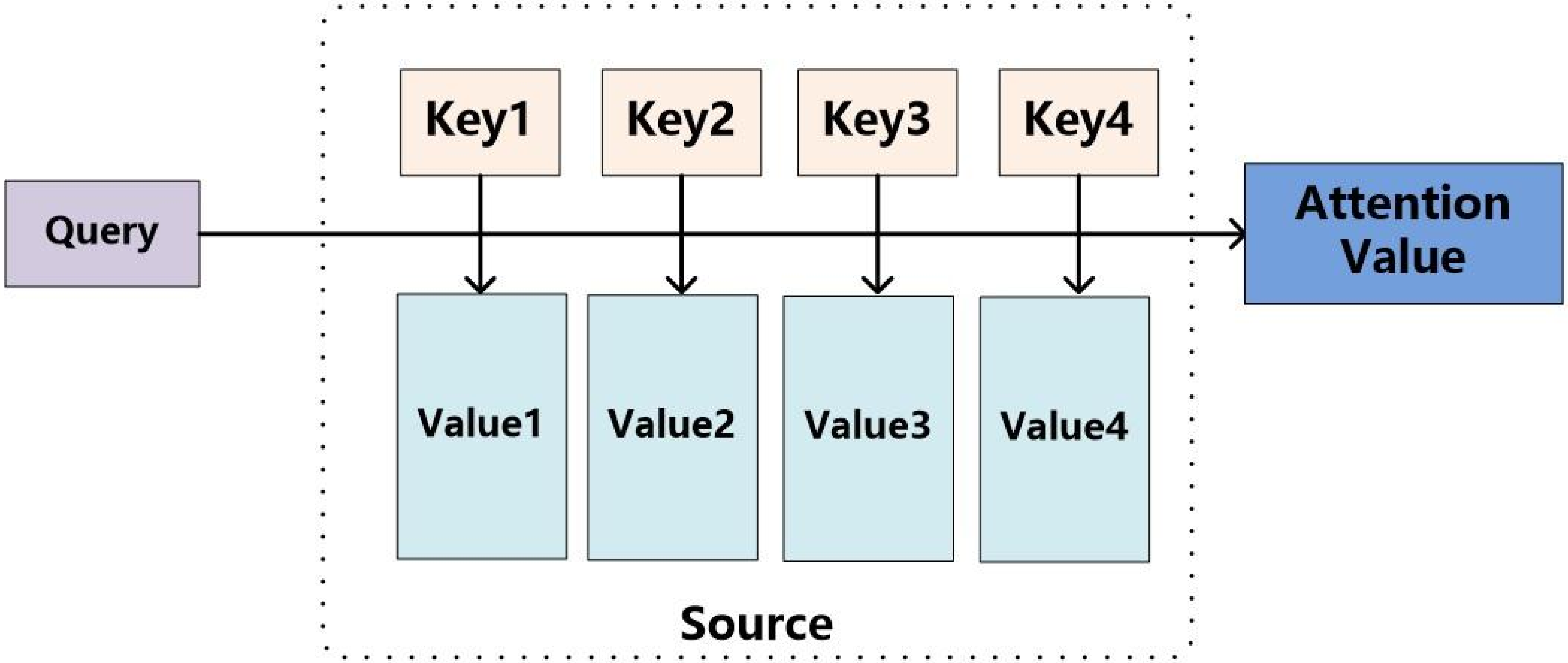


Figure 3: **Local attention model** – the model first predicts a single aligned position p_t for the current target word. A window centered around the source position p_t is then used to compute a context vector $\boldsymbol{c}_t$, a weighted average of the source hidden states in the window. The weights $\boldsymbol{a}_t$ are inferred from the current target state $\boldsymbol{h}_t$ and those source states $\bar{\boldsymbol{h}}_s$ in the window.

Attention结构回顾



Attention结构回顾

$$e_{t,i} = s_{t-1}^T h_i$$

$$e_{t,i} = u^T \tanh\left(W_1 h_i + W_2 s_{t-1}\right)$$

$$e_{t,i} = s_{t-1}^T h_i / \sqrt{d}$$

$$e_{t,i} = W_1 h_i + W_2 s_{t-1}$$

$$e_{t,i} = s_{t-1}^T W h_i$$

$$e_{t,i} = W h_i$$

Self Attention

- 在17年被提出于《Attention Is All You Need, Ashish Vaswani》，也称为Transformer结构；内部包含Multi-Head Attention以及Rest残差结构。
- Transformer是Bert网络结构的基础。
- https://arxiv.org/pdf/1706.03762.pdf

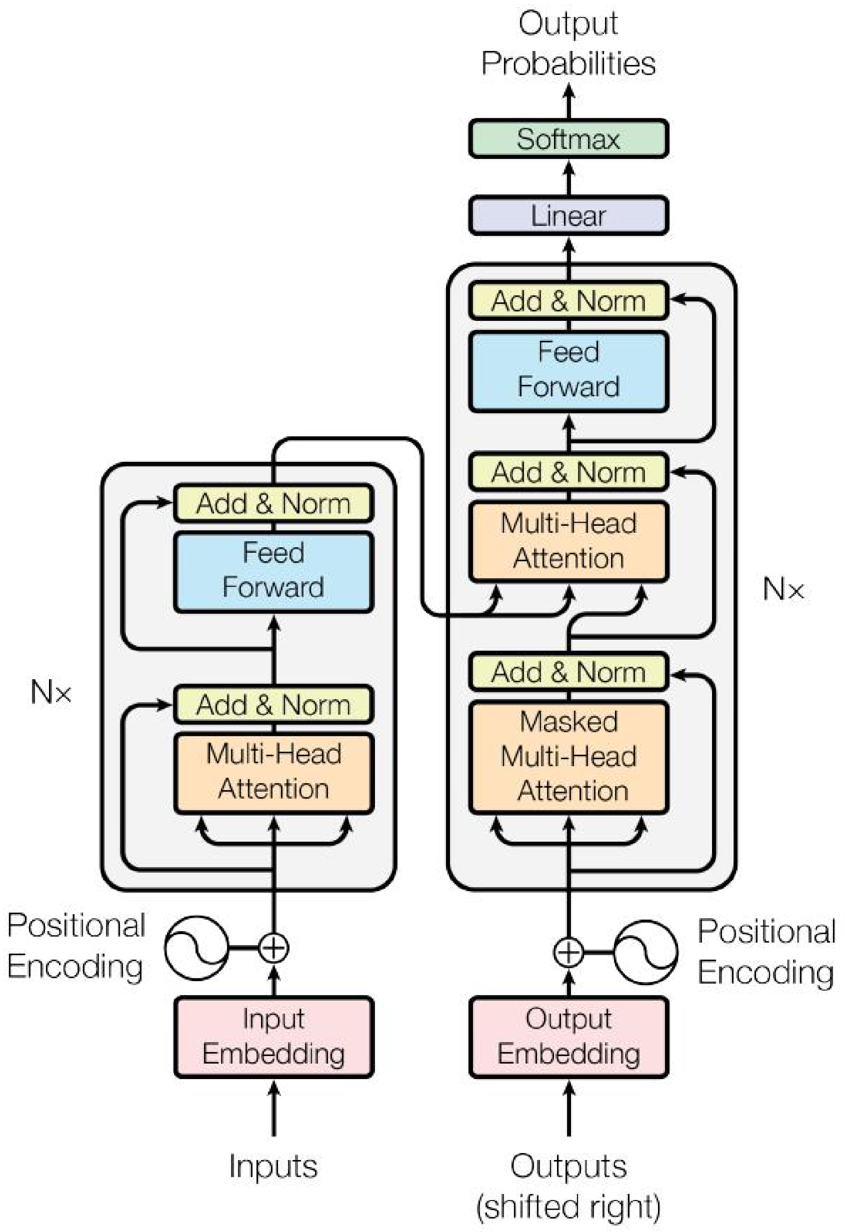


Figure 1: The Transformer - model architecture.

Transformer

- 传统缺点：seq2seq使用循环网络固有的顺序特性阻碍样本训练的并行化，这在更长的序列长度上变得至关重要，因为有限的内存限制样本的批次大小。
- 新结构：Transformer，这种模型架构避免循环并完全依赖于attention机制来绘制输入和输出之间的全局依赖关系。 Transformer允许进行更多的并行化。
- Self-attention：有时称为intra-attention，是一种attention机制，它关联单个序列的不同位置以计算序列的表示。 Self-attention已成功用于各种任务，包括阅读理解、摘要概括、文本蕴涵和学习与任务无关的句子表征。

Transformer

- Transformer: Attention Is All You Need
  - 2017, Google, https://arxiv.org/pdf/1706.03762.pdf
  - New Features:
    - **Self-Attention**
    - **Multi-Headed-Attention**
    - **Positional Encoding**
    - **Residuals**
    - **Layer Norm**
    - **Masked**

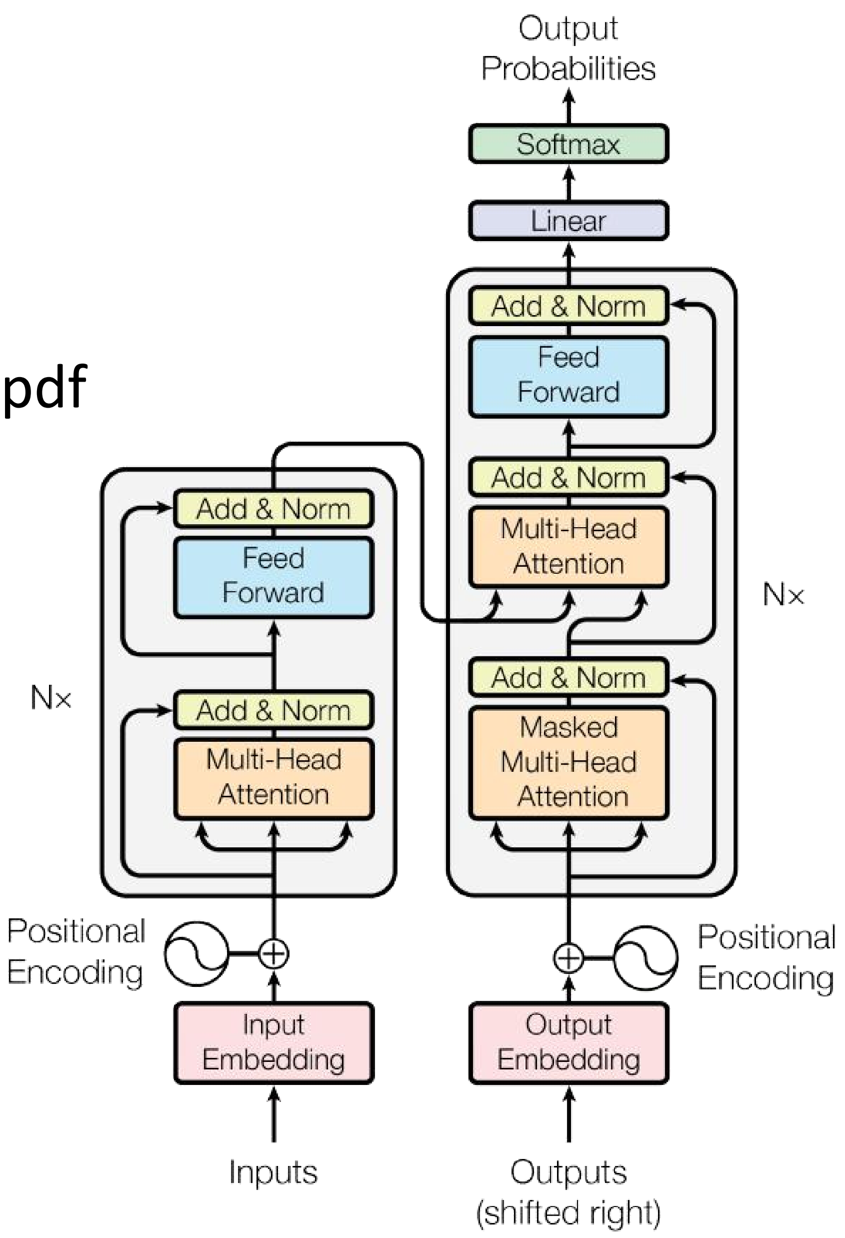
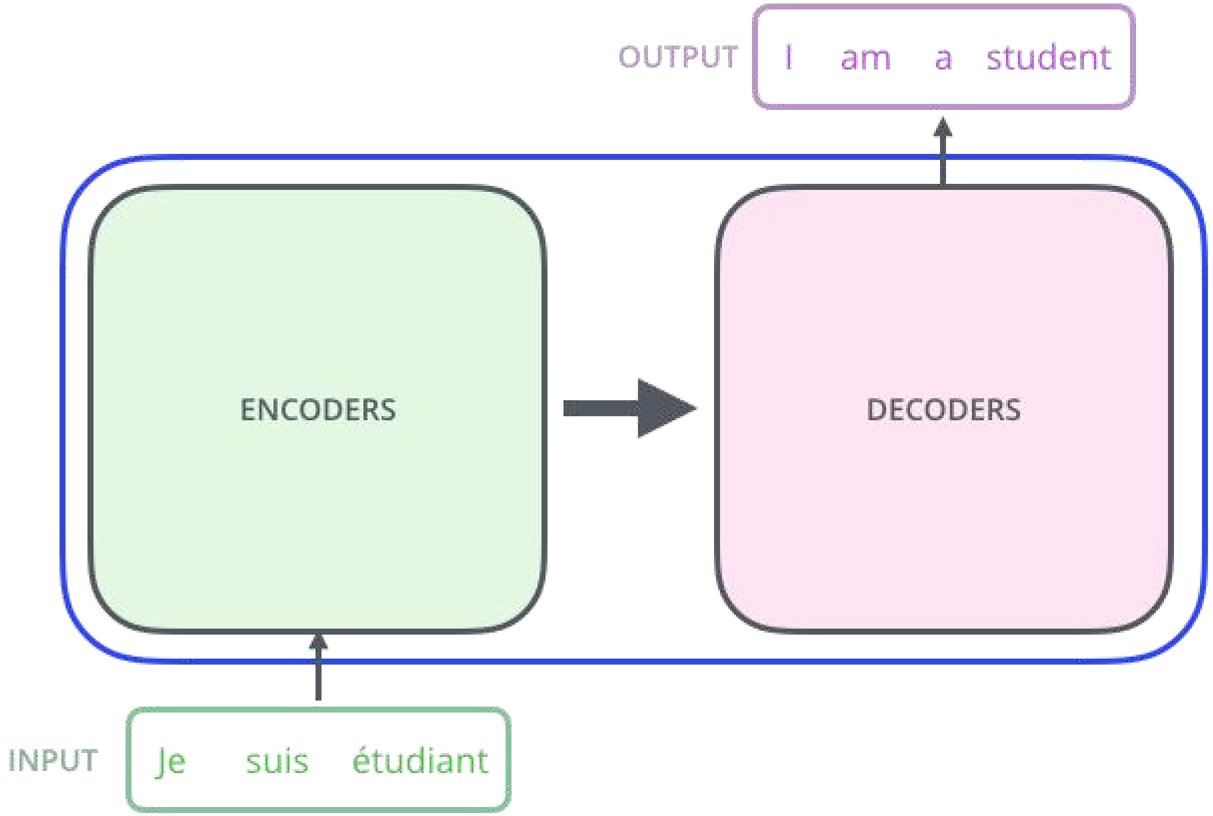


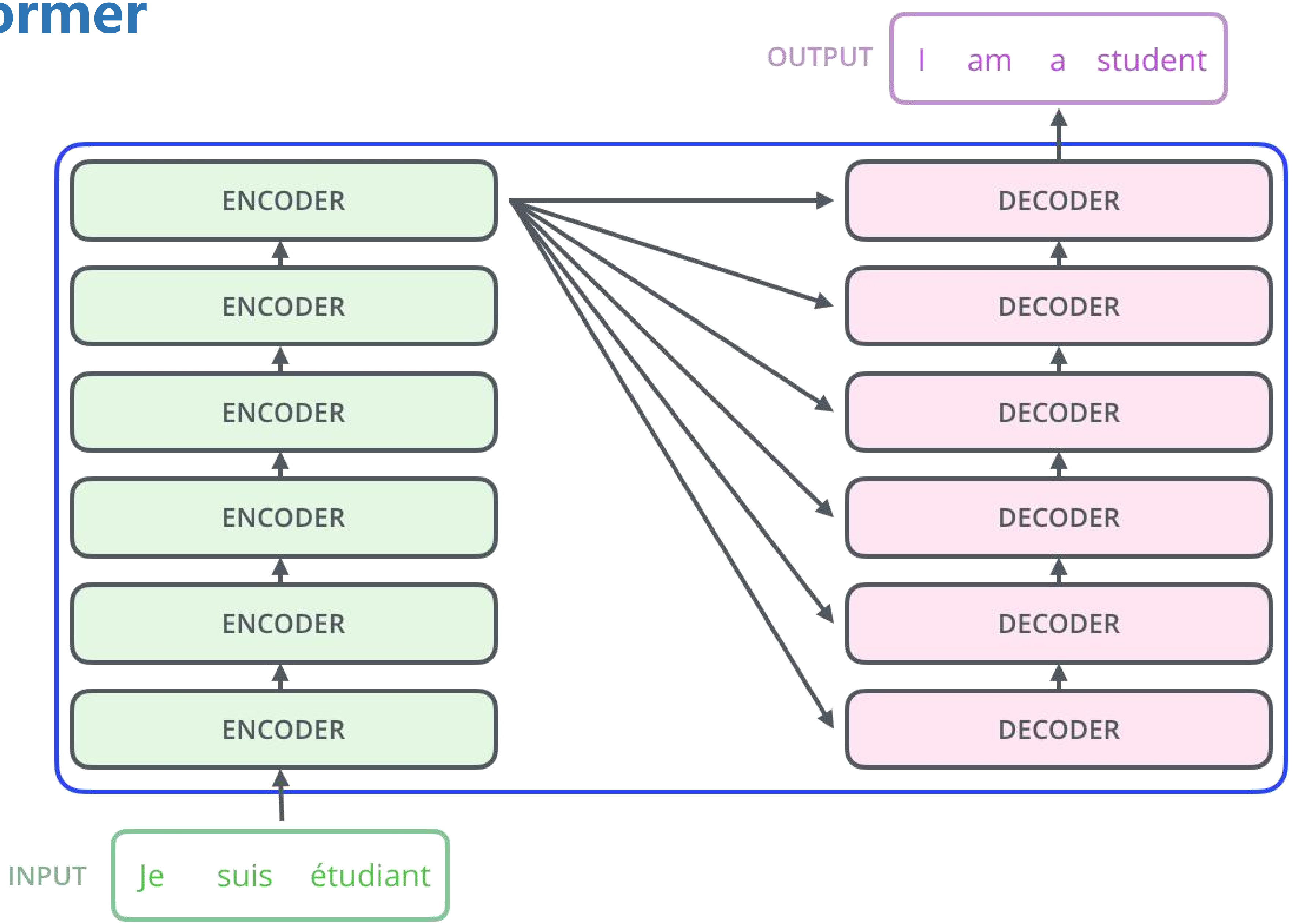
Figure 1: The Transformer - model architecture.

Transformer

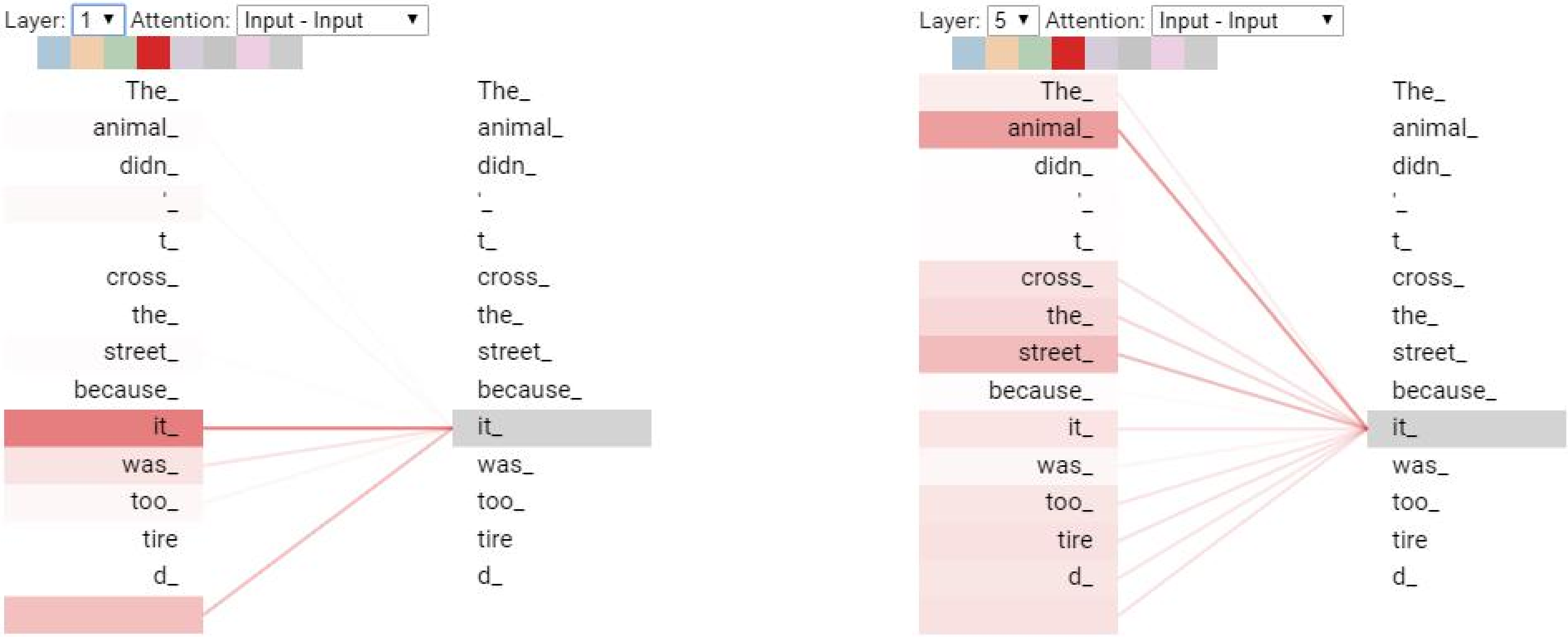


Transformer

六层Encoder和Decoder结构；各个Encoder和Decoder之间不共享权重Weights；



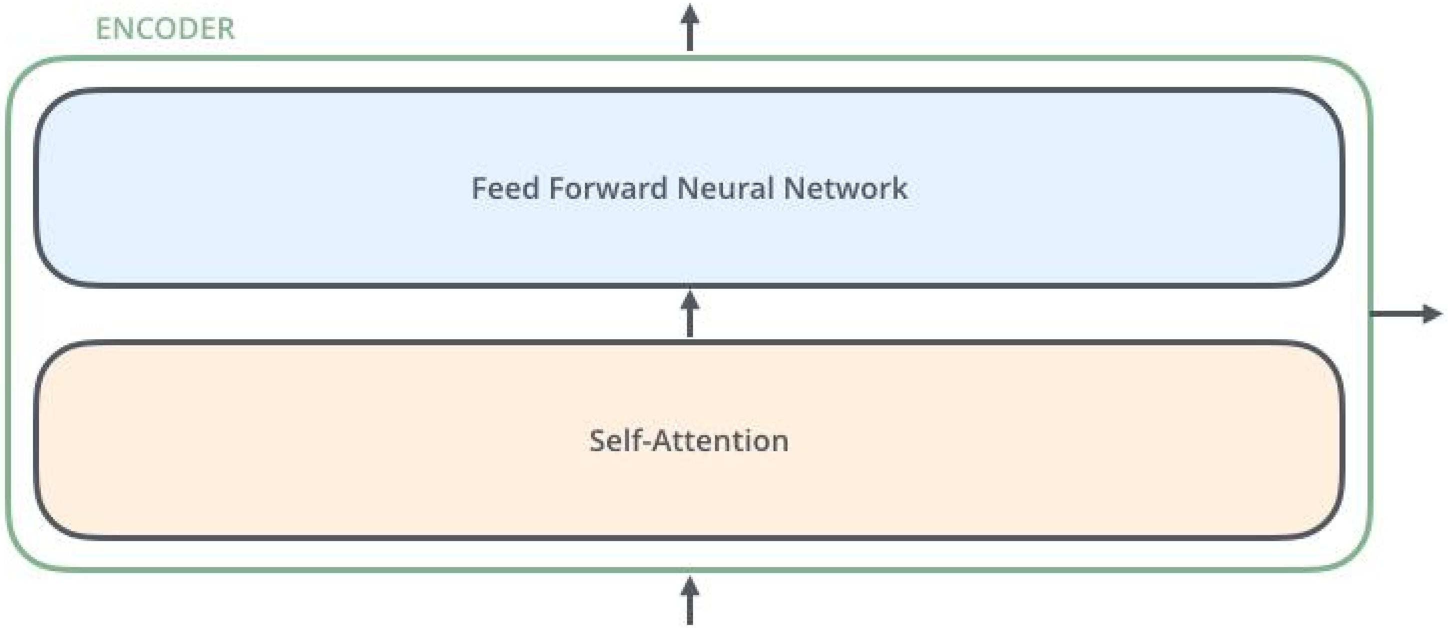
Transformer



The animal didn't cross the street because **it** was too tired

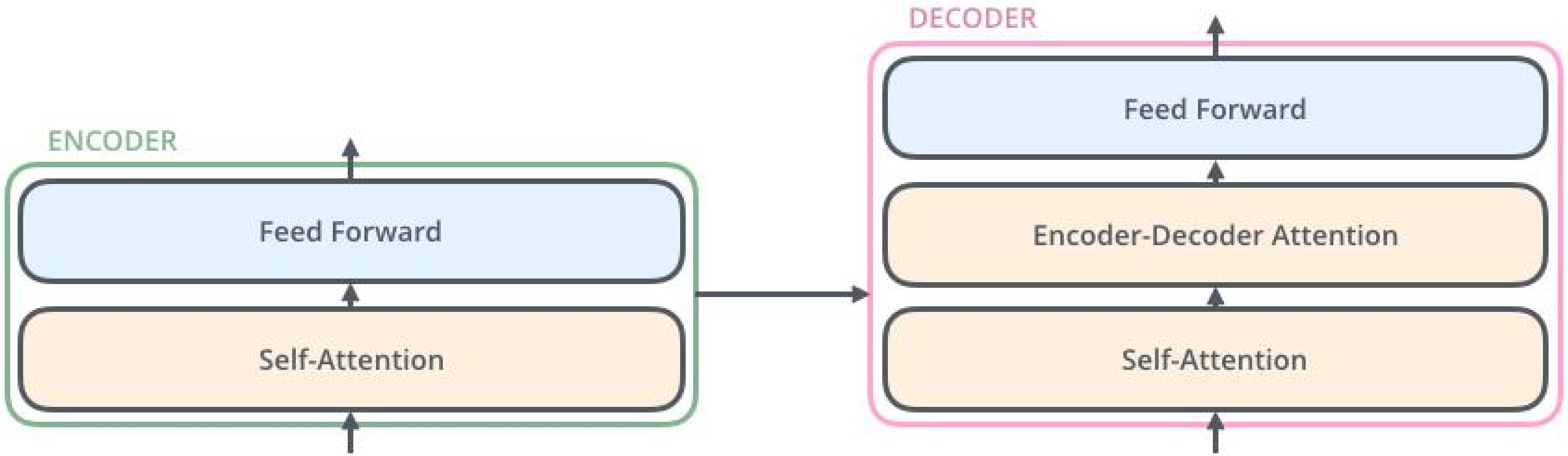
Transformer

每个Encoder包含两层结构: Self-Attention以及feed-forward NN构成。

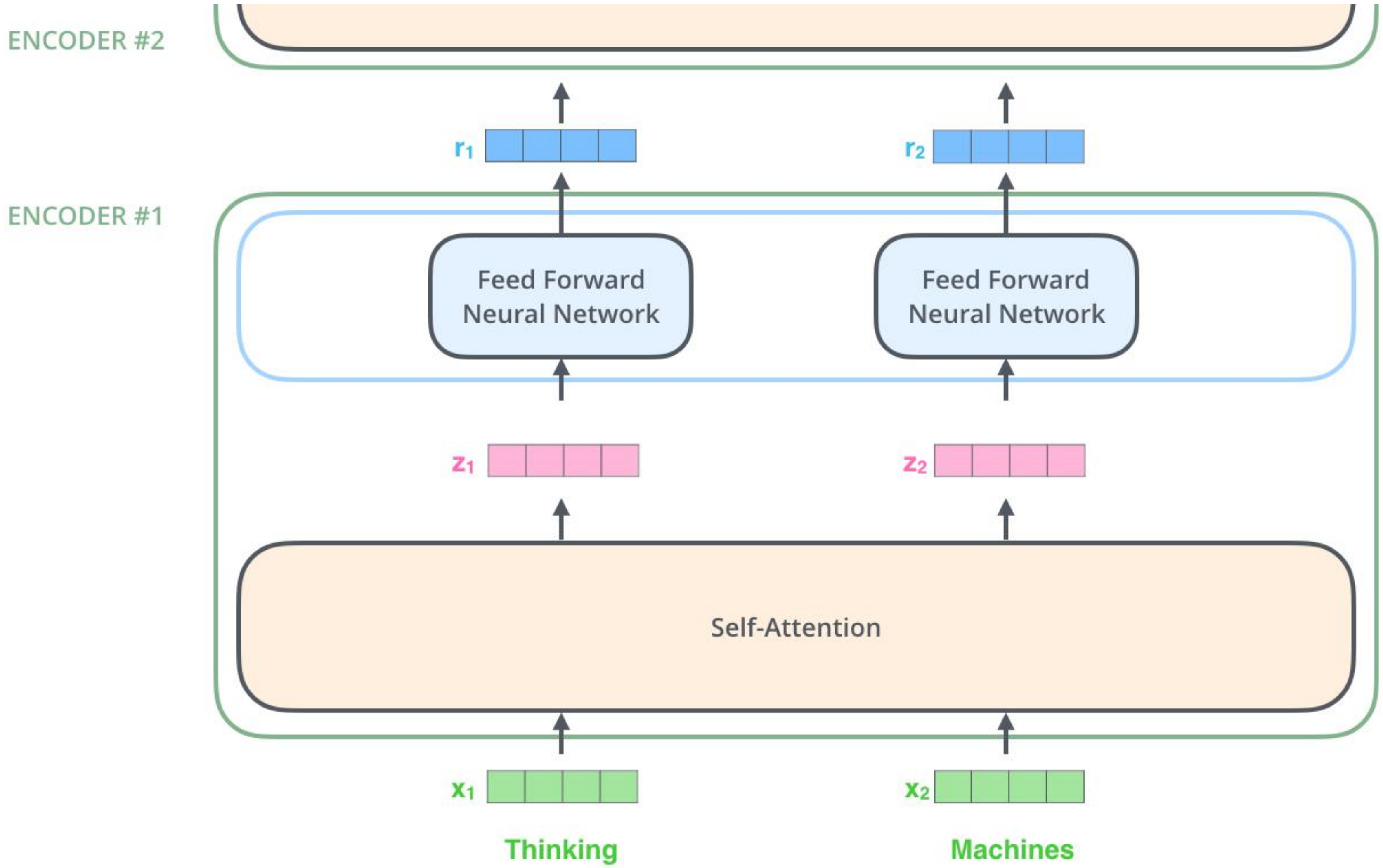


Transformer

每个Decoder在Encoder的基础上，在Self-Attention和feed-forward NN之间增加了一个Encoder-Decoder Attention

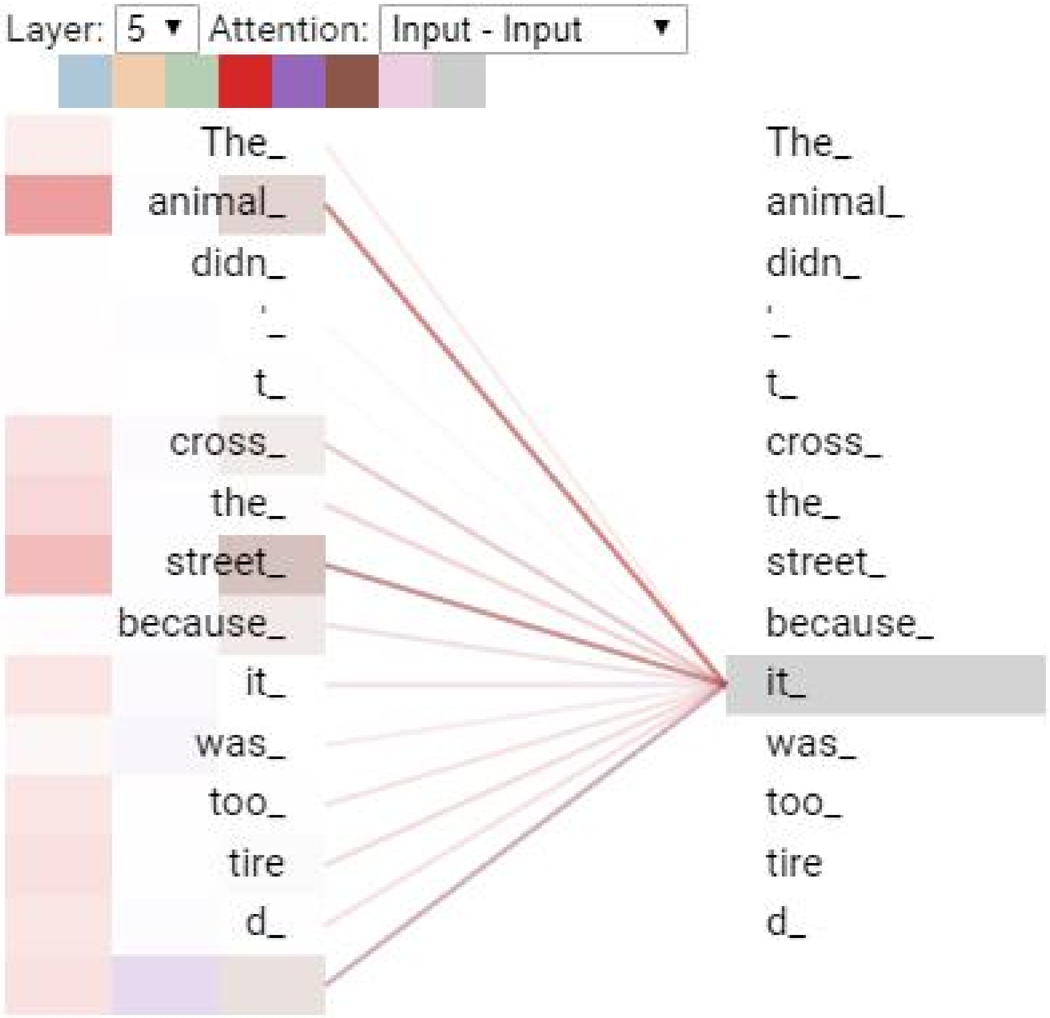


Transformer
Encoder



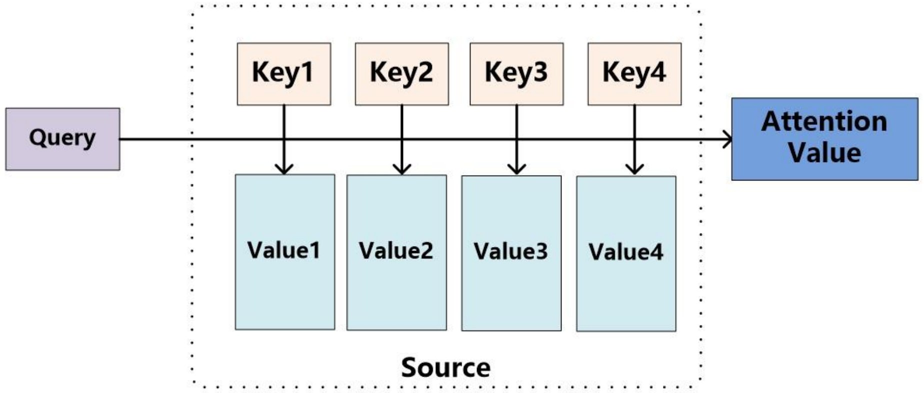
Transformer

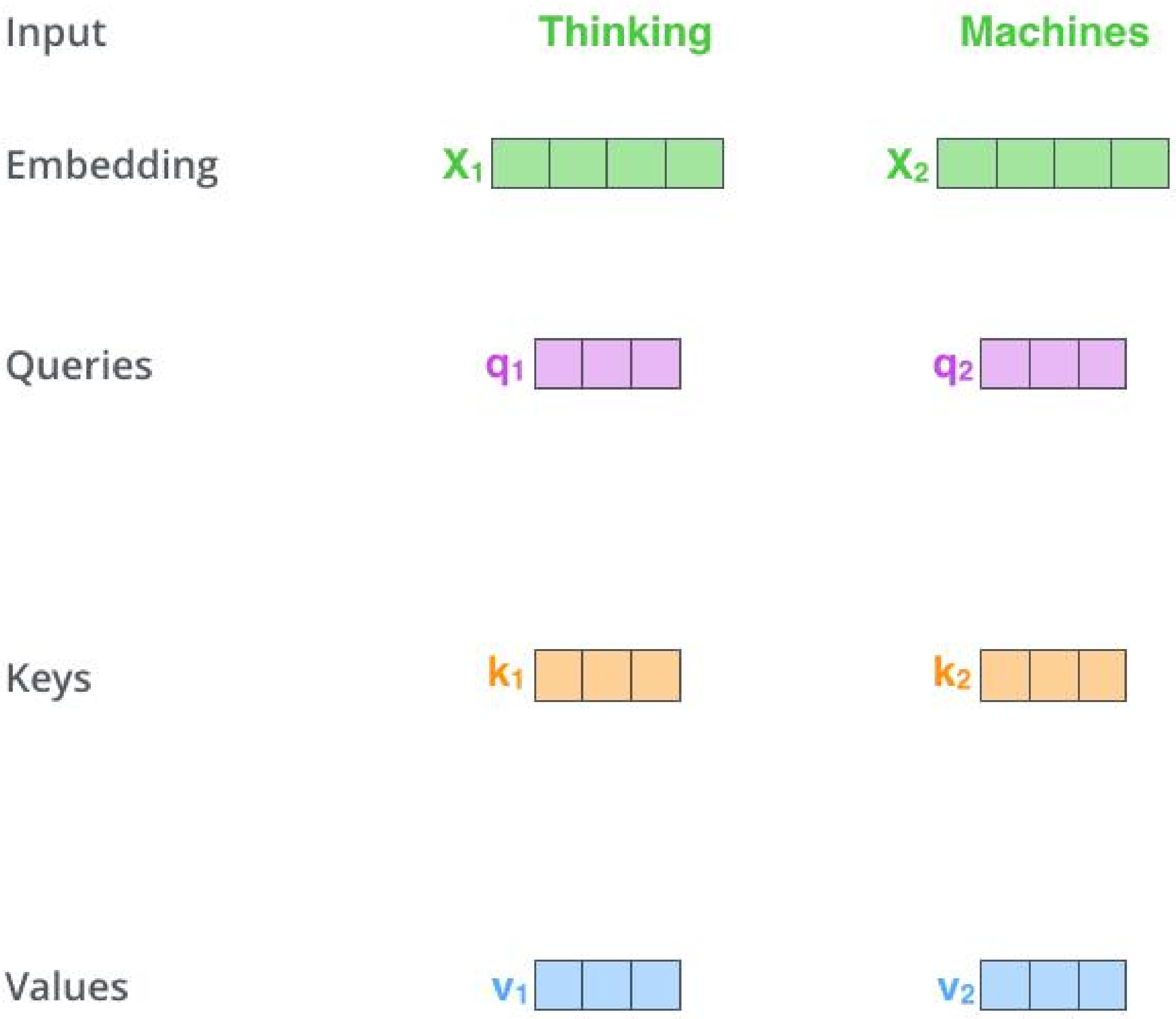
Encoder Self-Attention

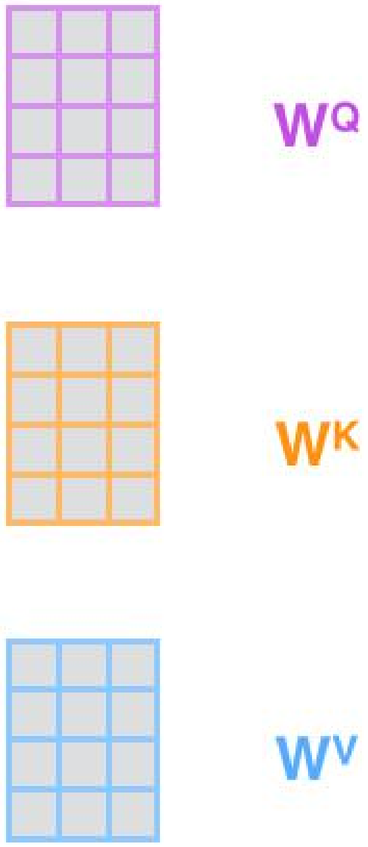


The animal didn't cross the street because **it** was too tired

Transformer
Encoder Self-Attention





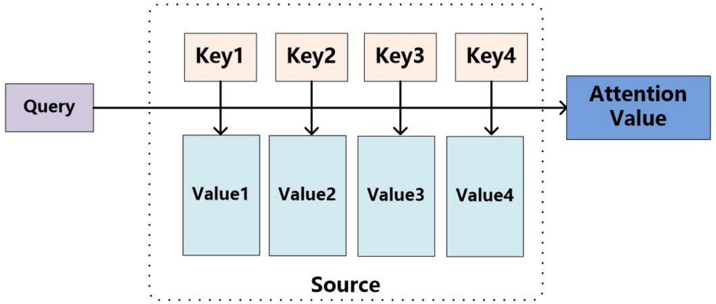


$$q = XW^Q$$

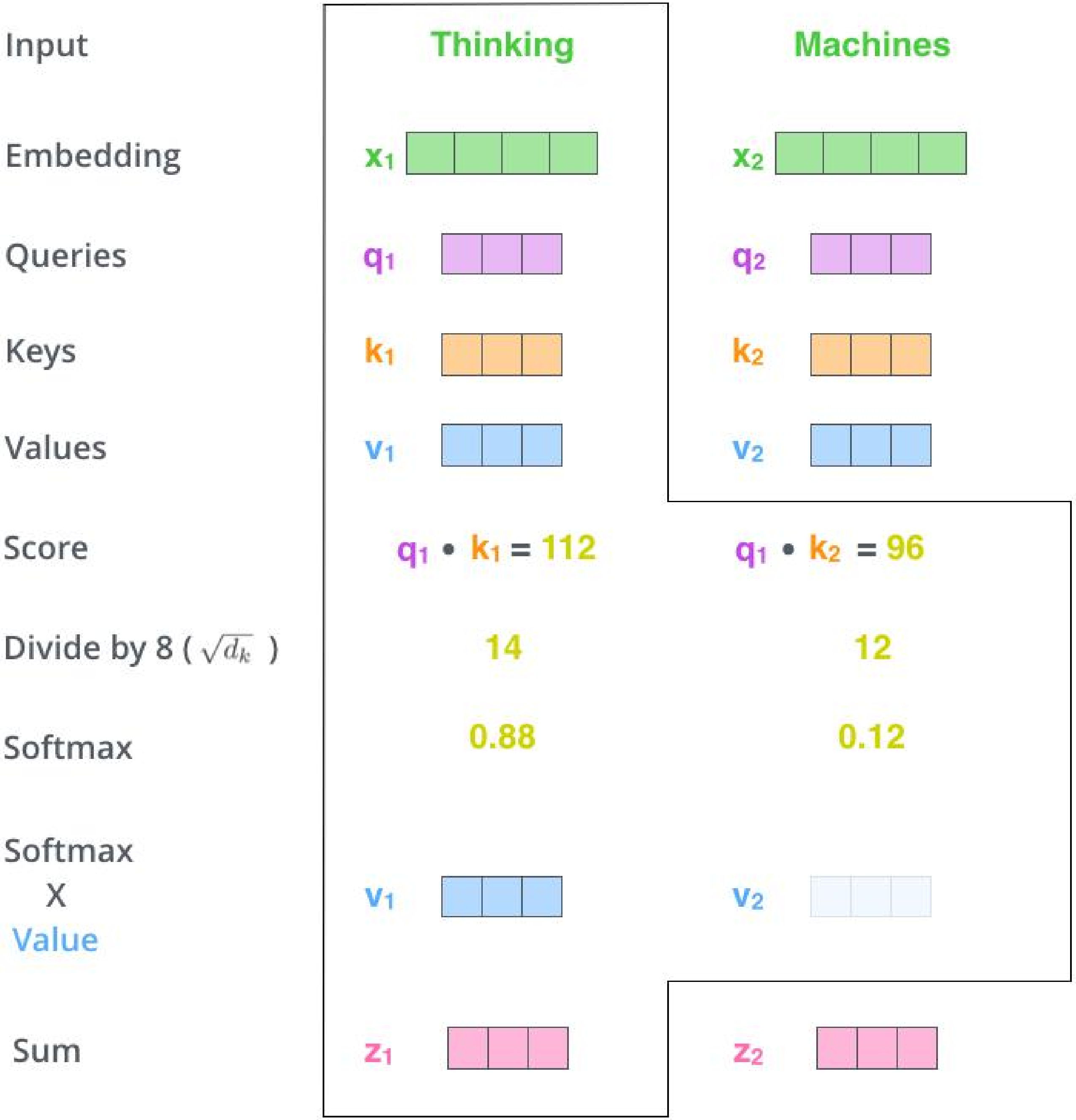
$$k = XW^K$$

$$v = XW^V$$

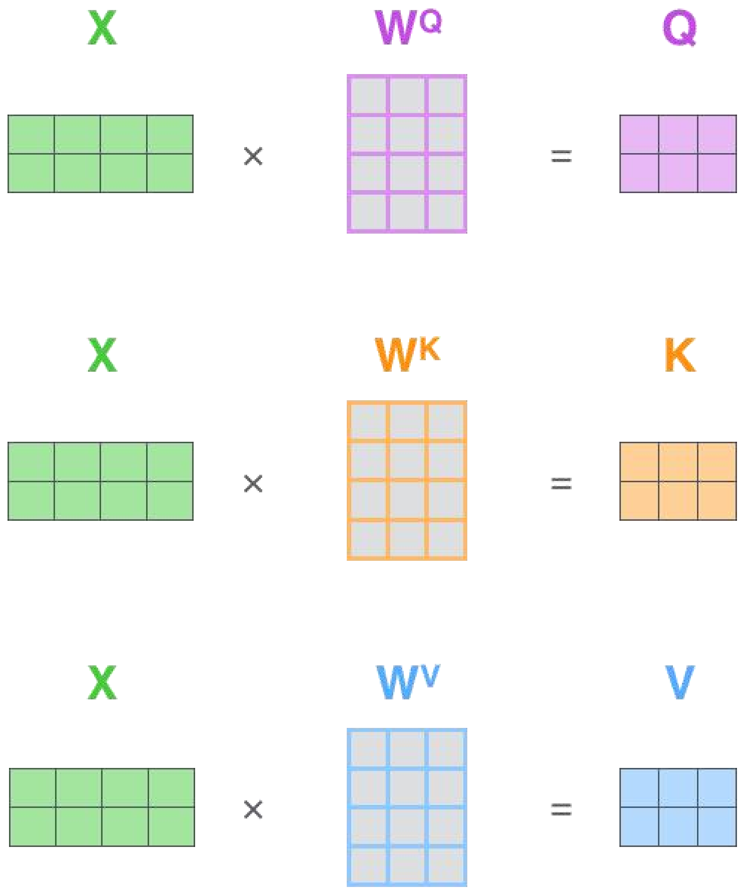
Transformer
Encoder Self-Attention

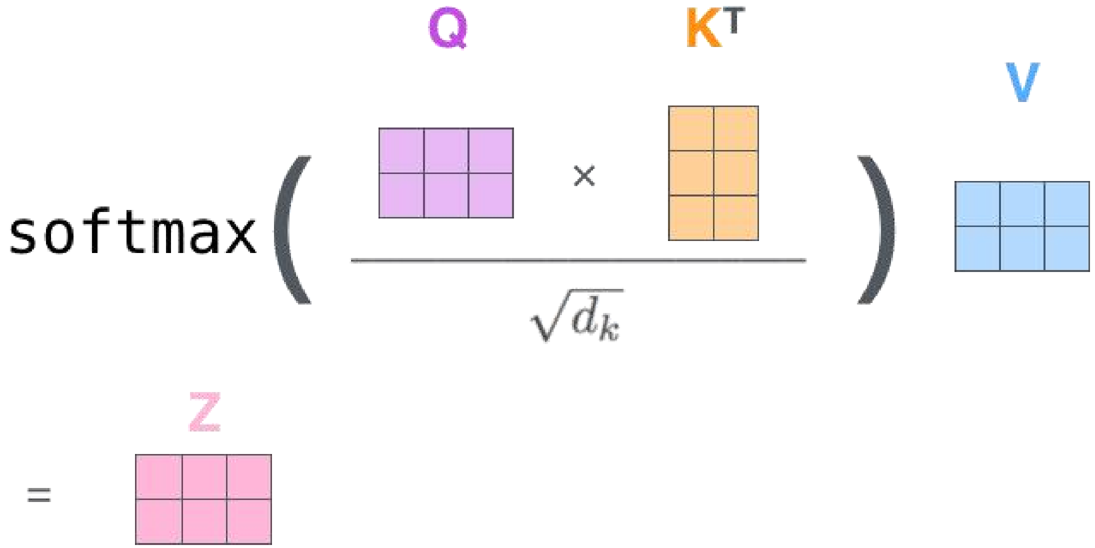


$$e_{t,i} = s_{t-1}^{T} h_i / \sqrt{d}$$

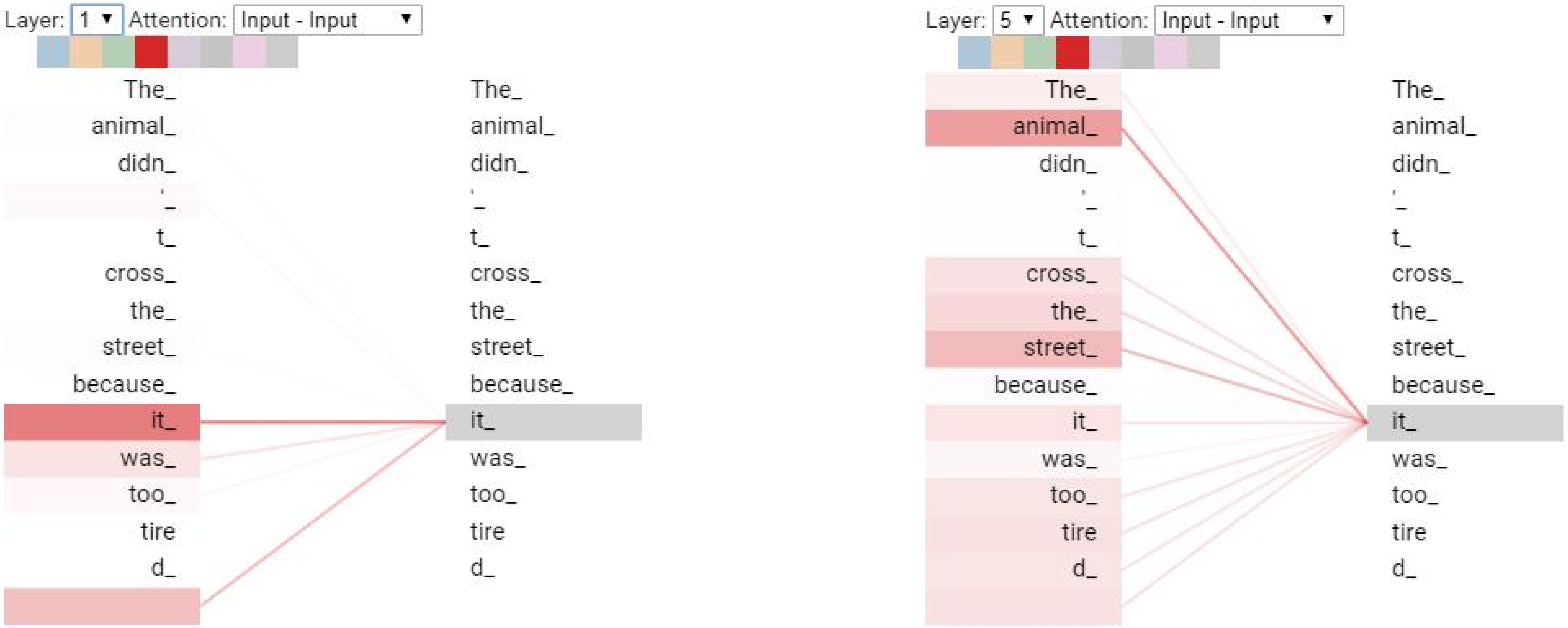


Transformer
Encoder Self-Attention

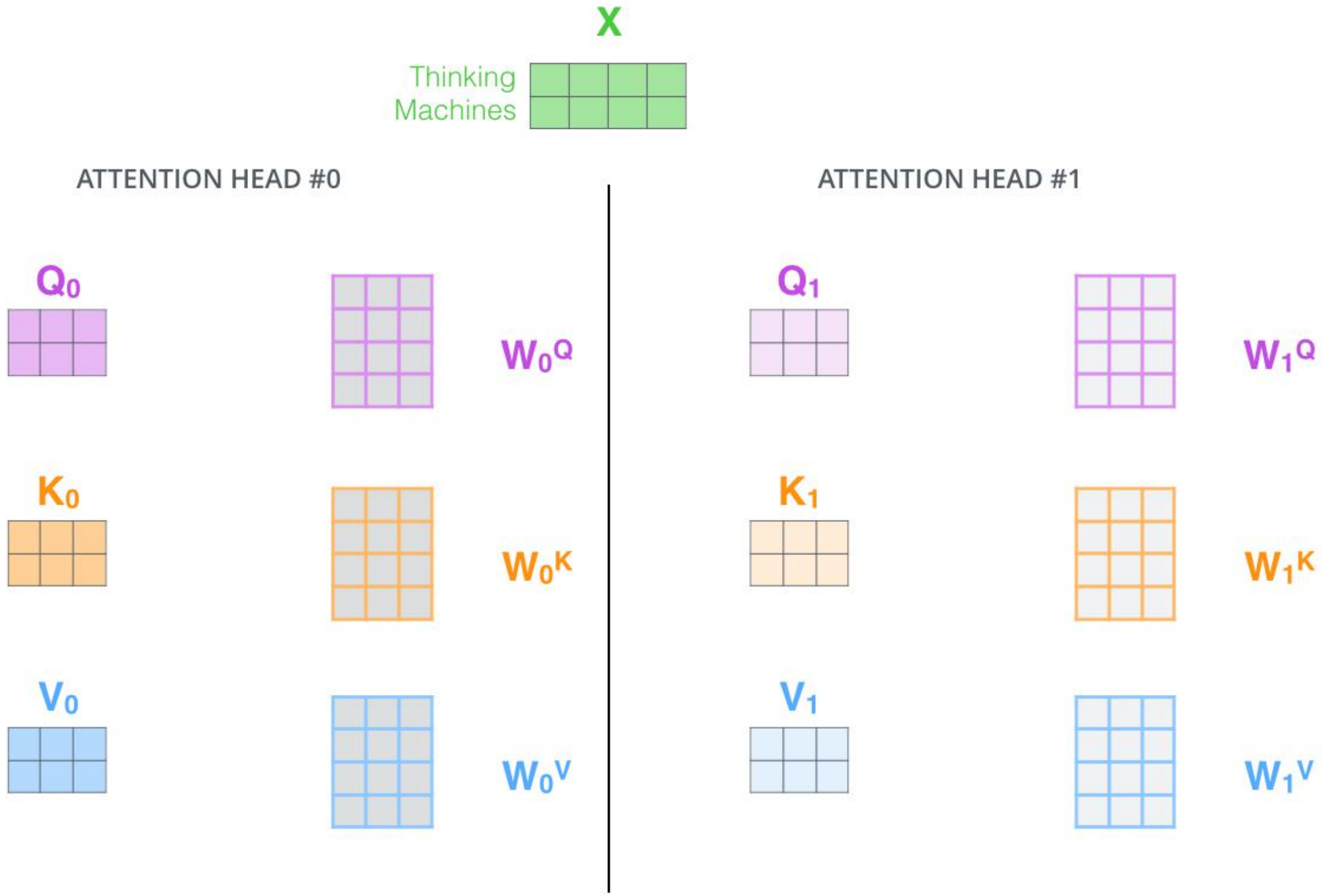




Transformer
Encoder Self-Attention

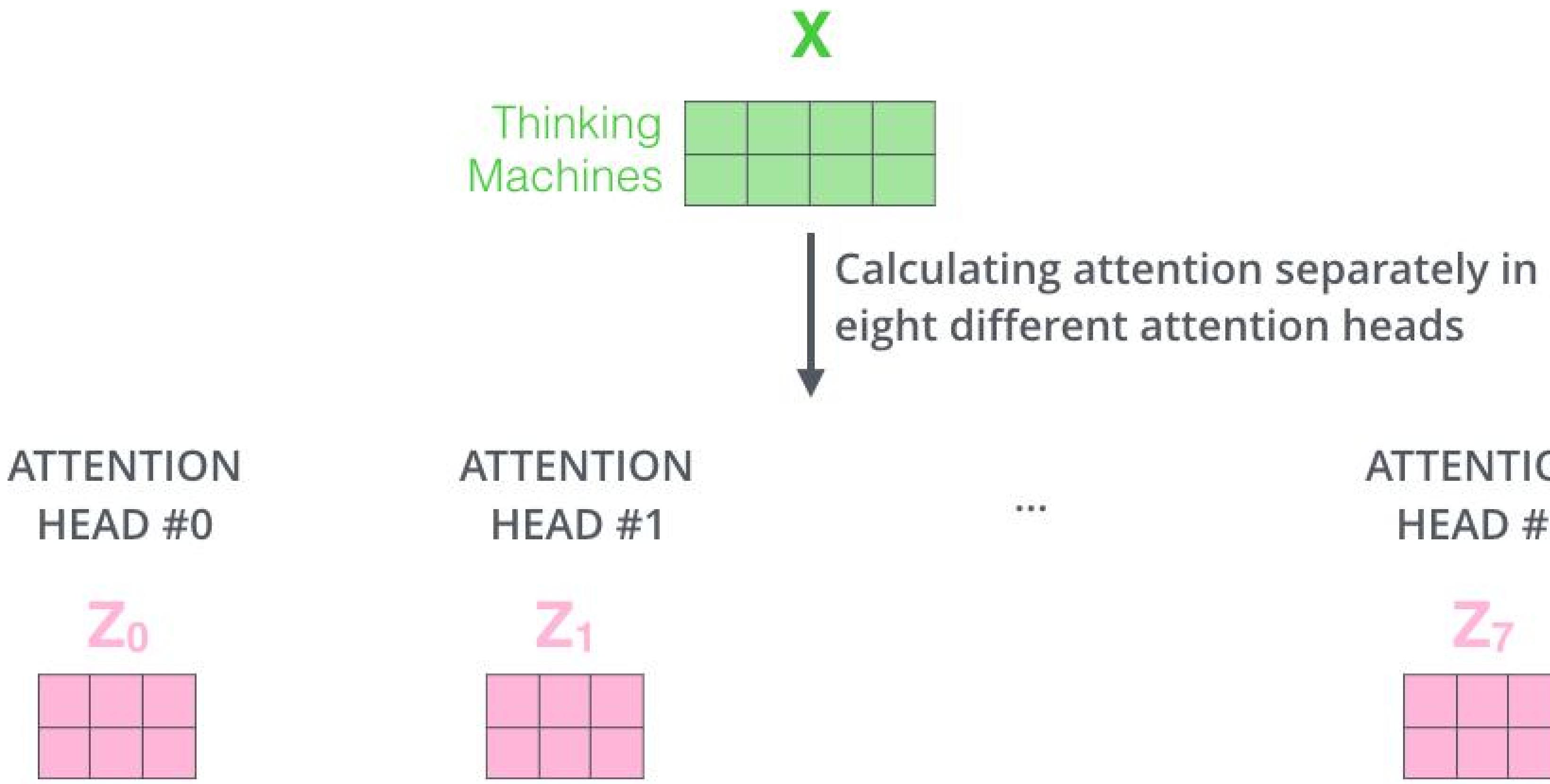


Transformer
Encoder Multi-Headed-Attention



Transformer

Encoder Multi-Headed-Attention



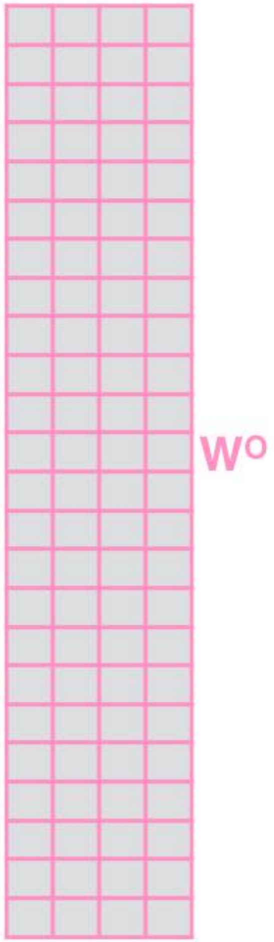
Transformer

Encoder Multi-Headed-Attention

1) Concatenate all the attention heads

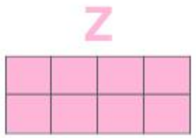
2) Multiply with a weight matrix W^O that was trained jointly with the model

X

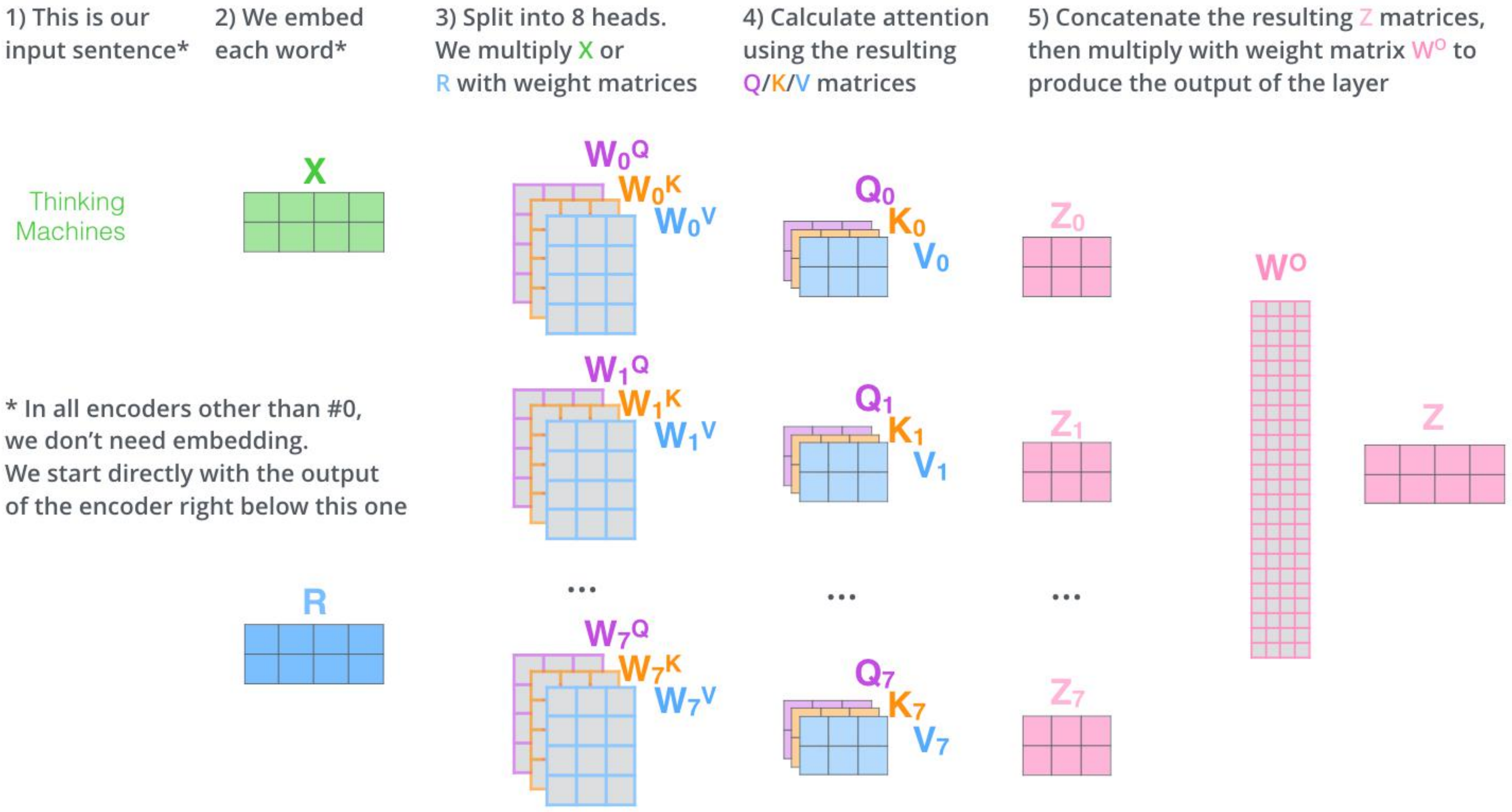


3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

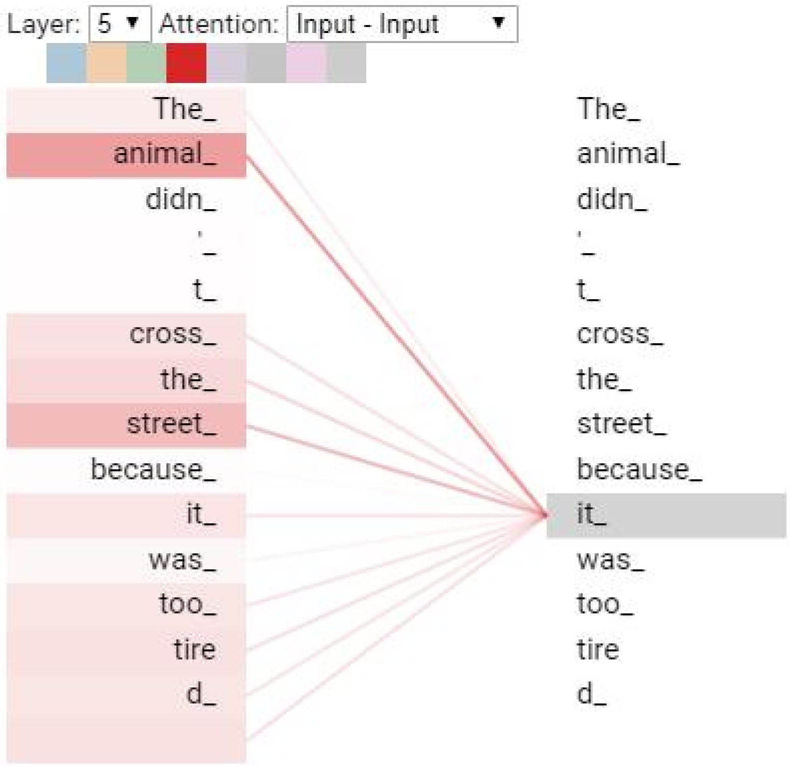
=

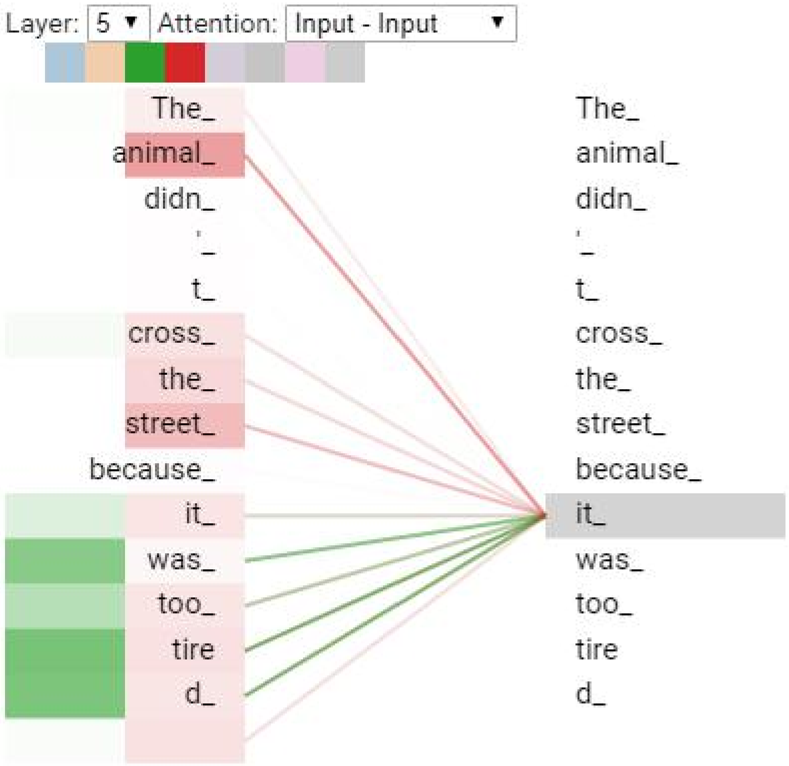


Transformer
Encoder Multi-Headed-Attention



Transformer
Encoder Multi-Headed-Attention





Transformer

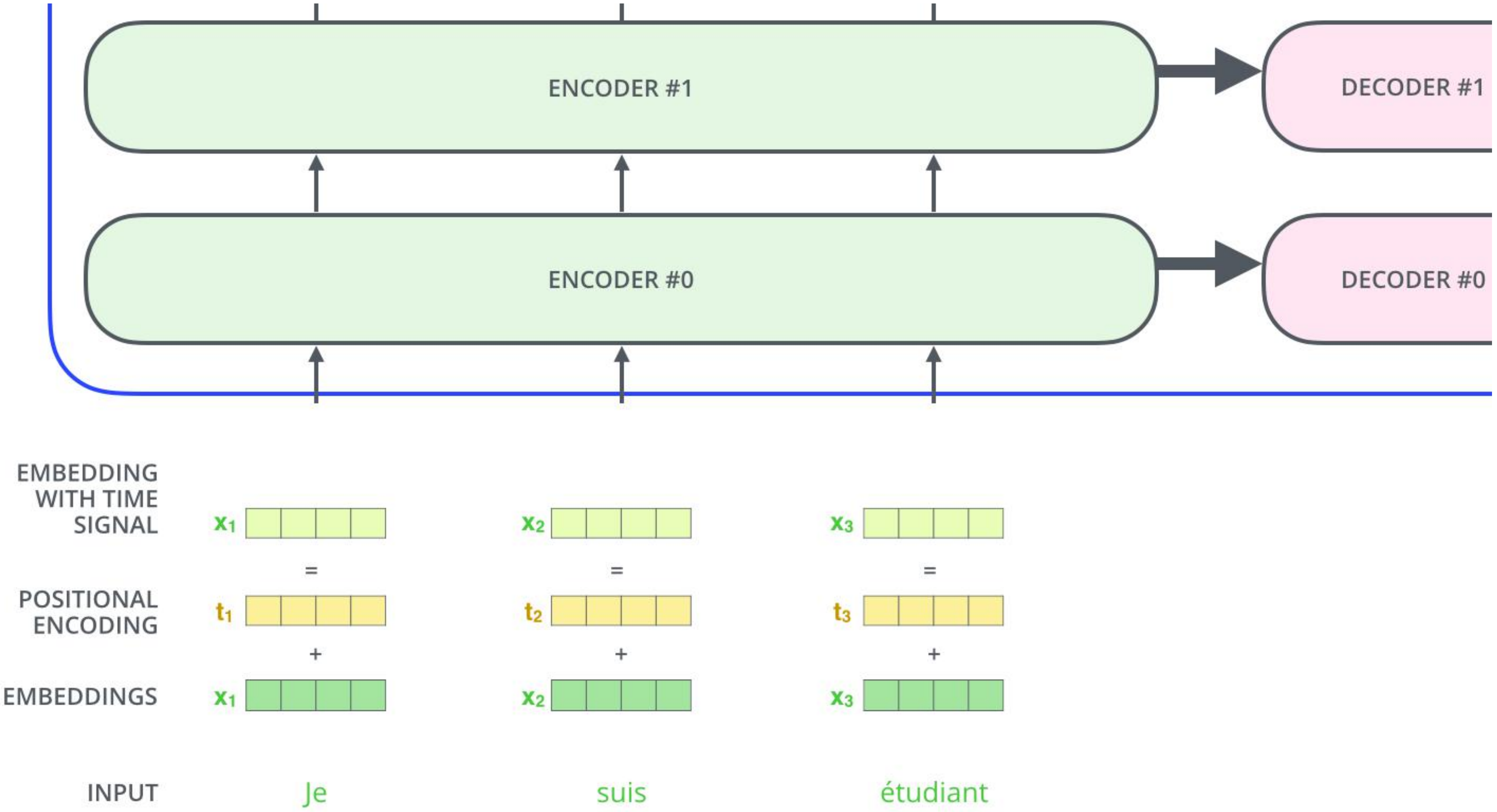
Positional Encoding

- 模型还没有描述词之间的顺序关系，也就是如果将一个句子打乱其中的位置，也应该获得相同的注意力，为了解决这个问题，论文加入了自定义位置编码，位置编码和word embedding长度相同的特征向量，然后和word embedding进行求和操作。

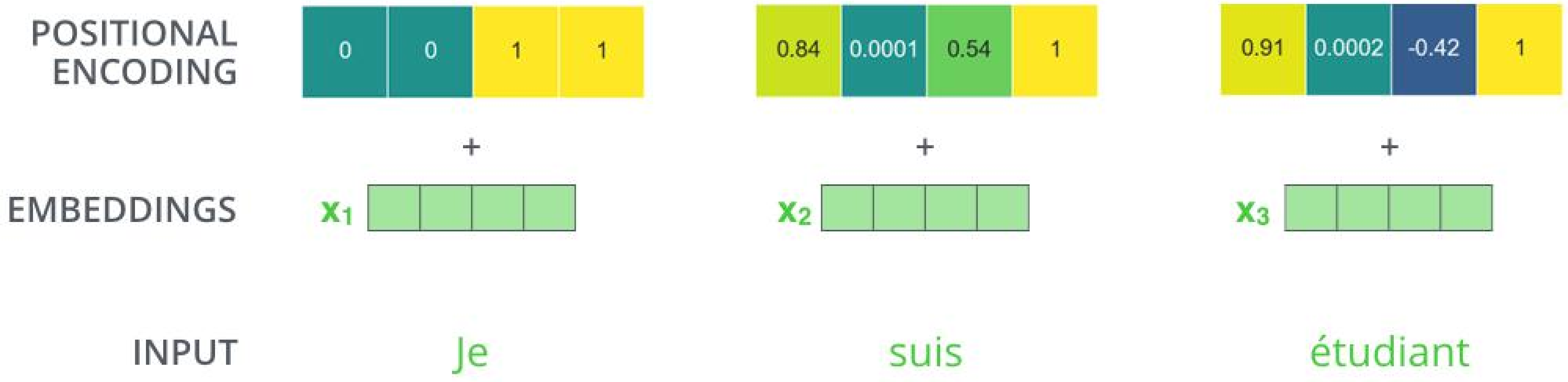
$$PE(pos, 2i) = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$

$$PE(pos, 2i+1) = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$

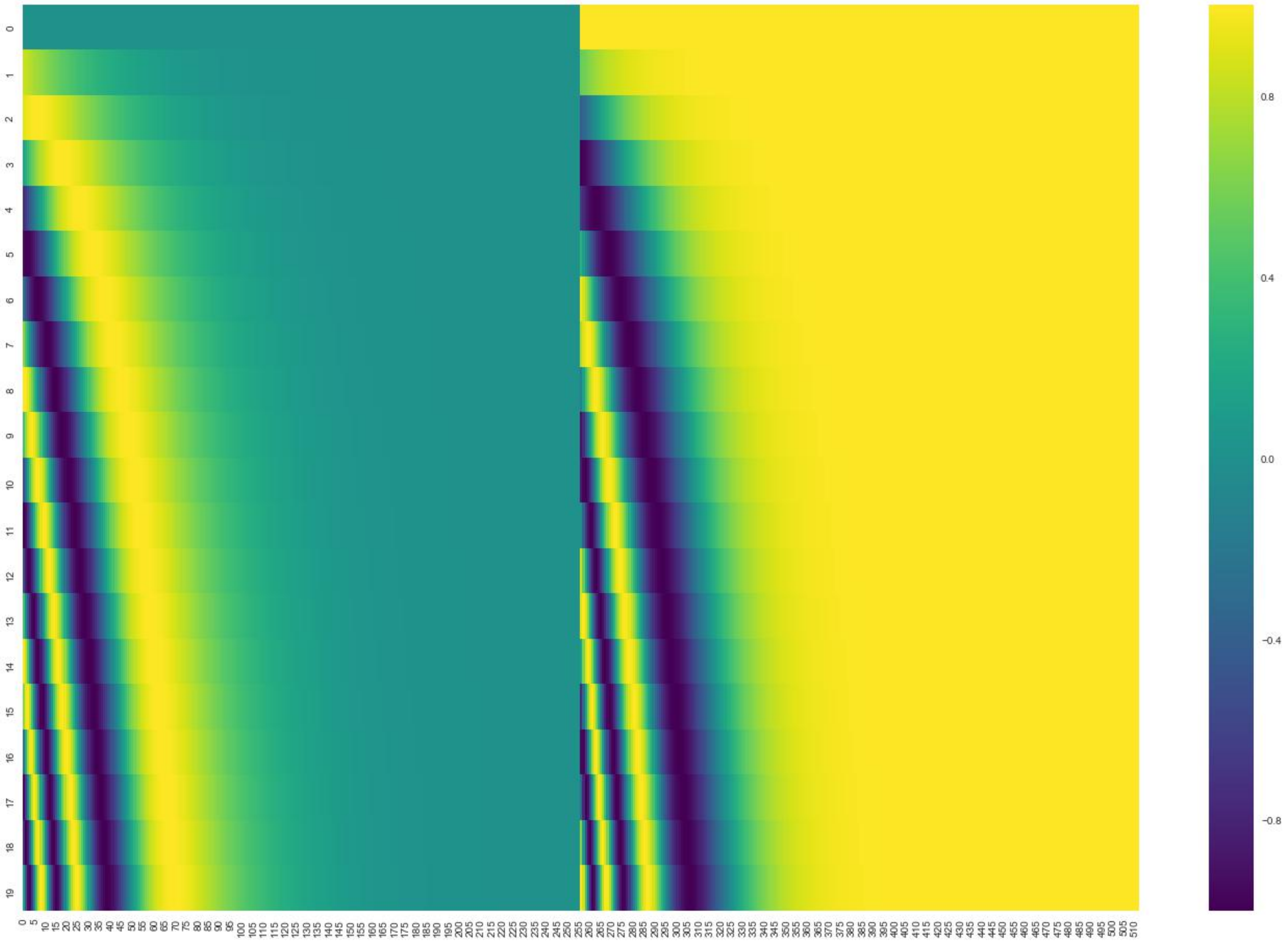
Transformer
Positional Encoding



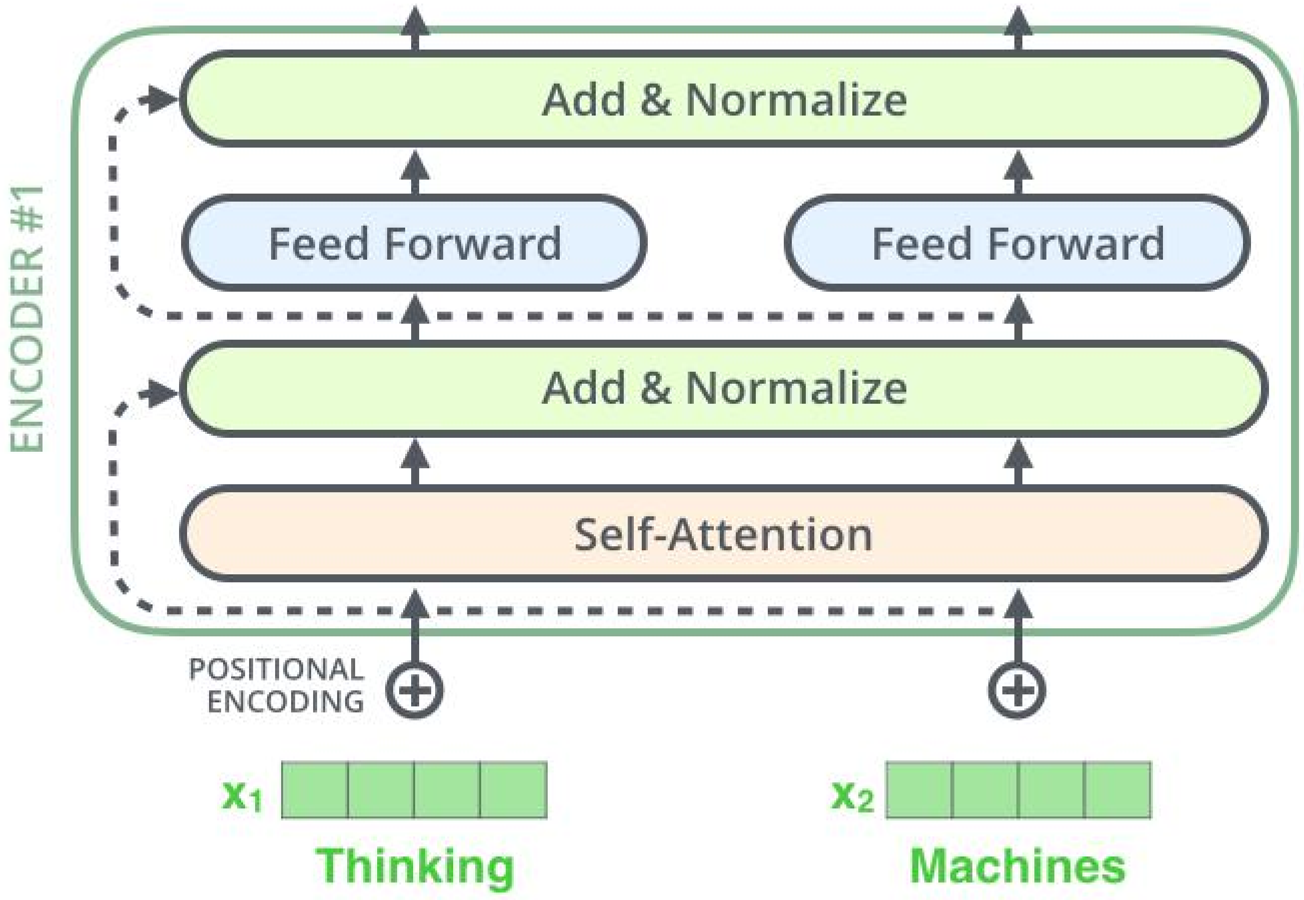
Transformer
Positional Encoding

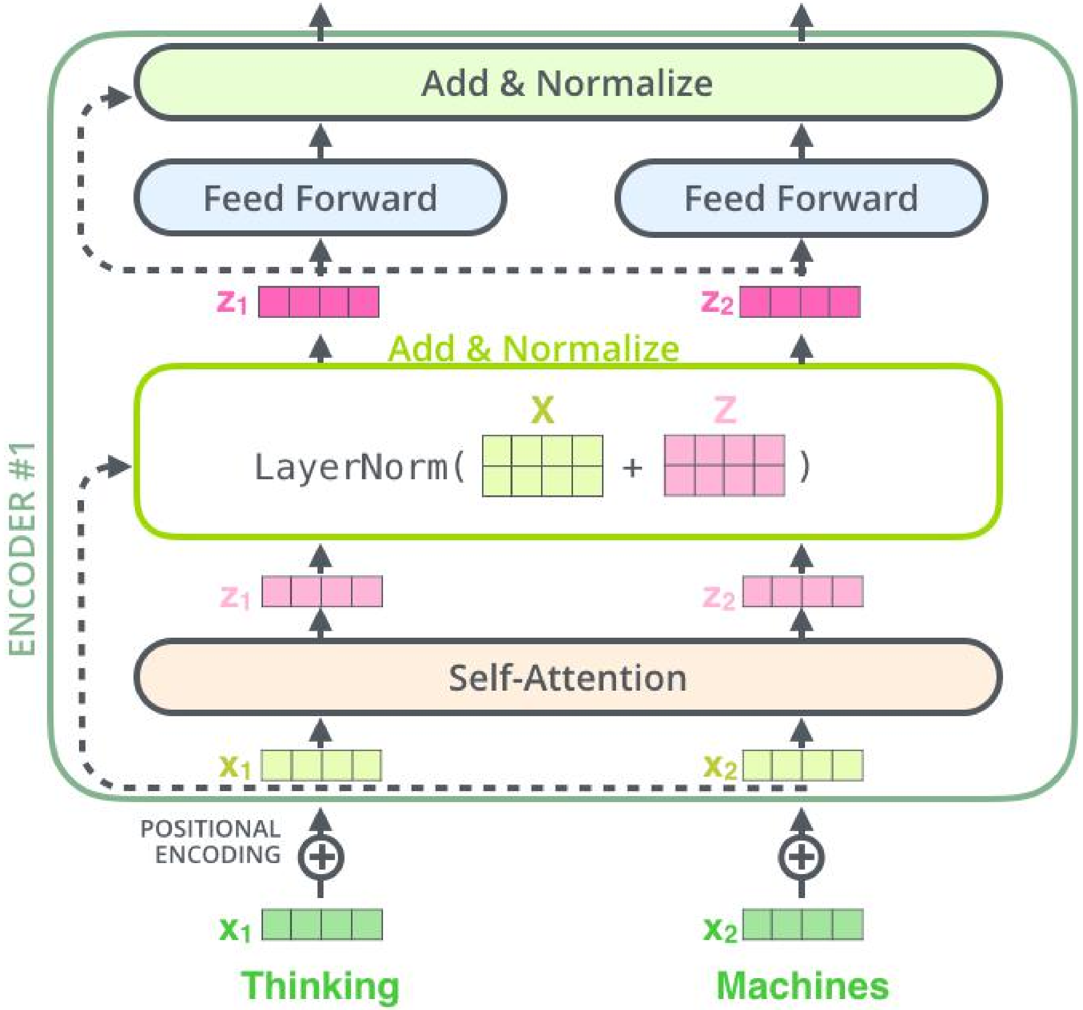


Transformer
Positional Encoding

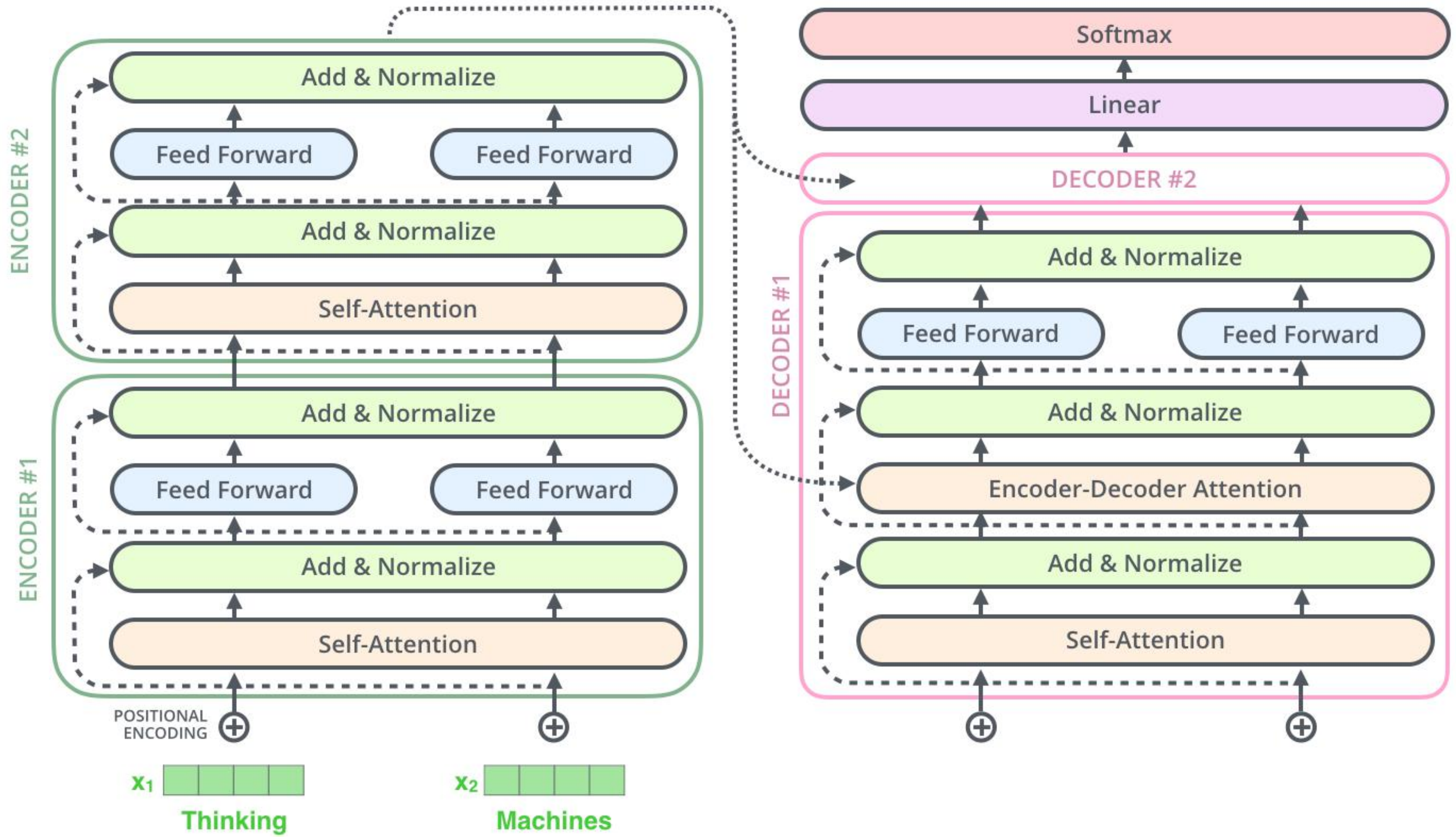


Transformer
LayerNorm&Residuals





Transformer
Decoder



Transformer
Decoder Masked Multi-Headed-Attention

- 和编码部分的multi-head attention类似，但是多了一 次masked，因为在解码部分，解码的时候时从左到右依次解码的，当解出第一个字的时候，第一个字只能与第一个字计算相关性，当解出第二个字的时候，只能计算出第二个字与第一个字和第二个字的相关性。；
- 因此这里引入Mask的概念进行掩码操作。

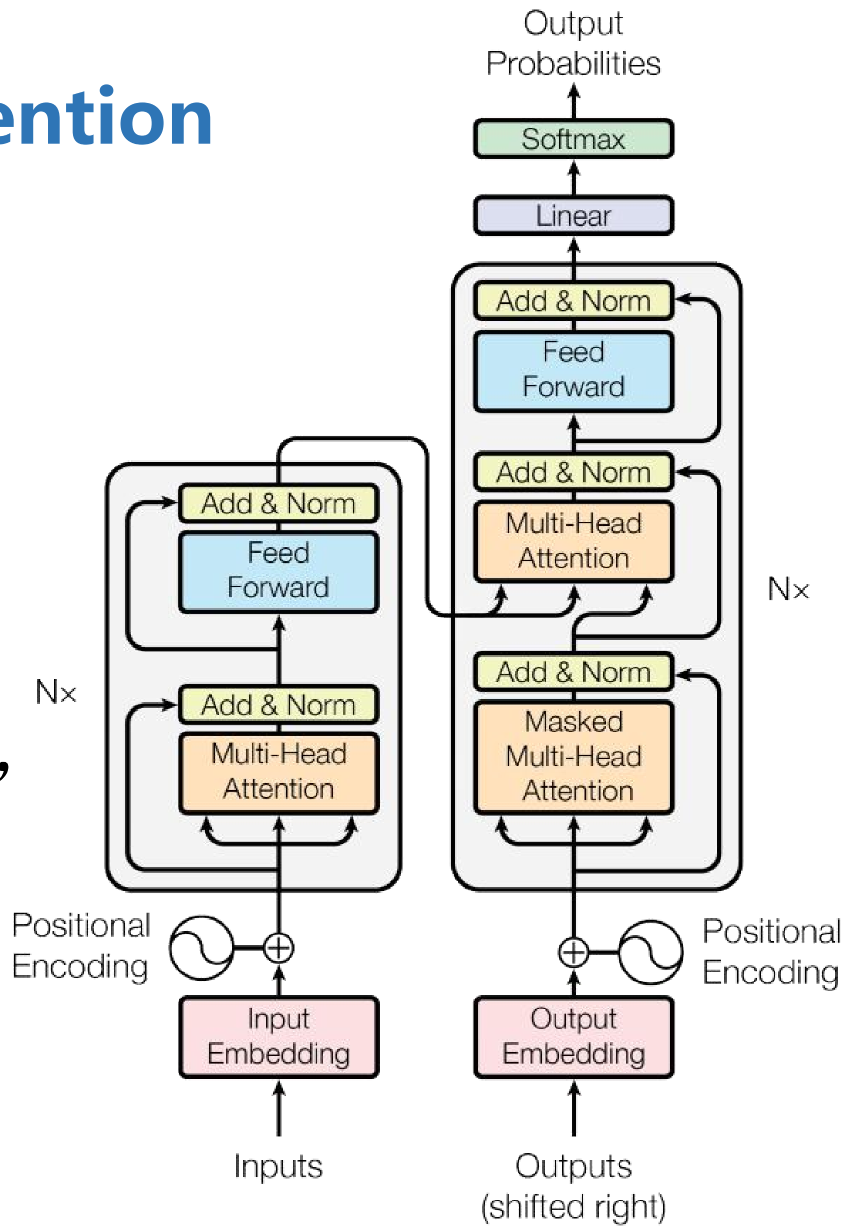
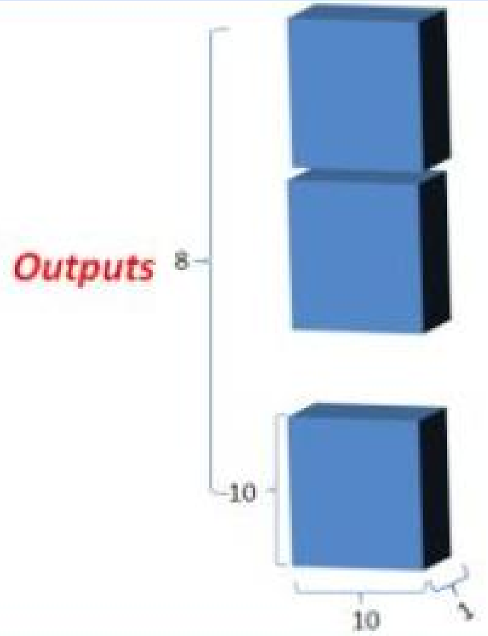


Figure 1: The Transformer - model architecture.

Transformer
Decoder Masked Multi-Headed-Attention



Mask矩阵

1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1

10

10

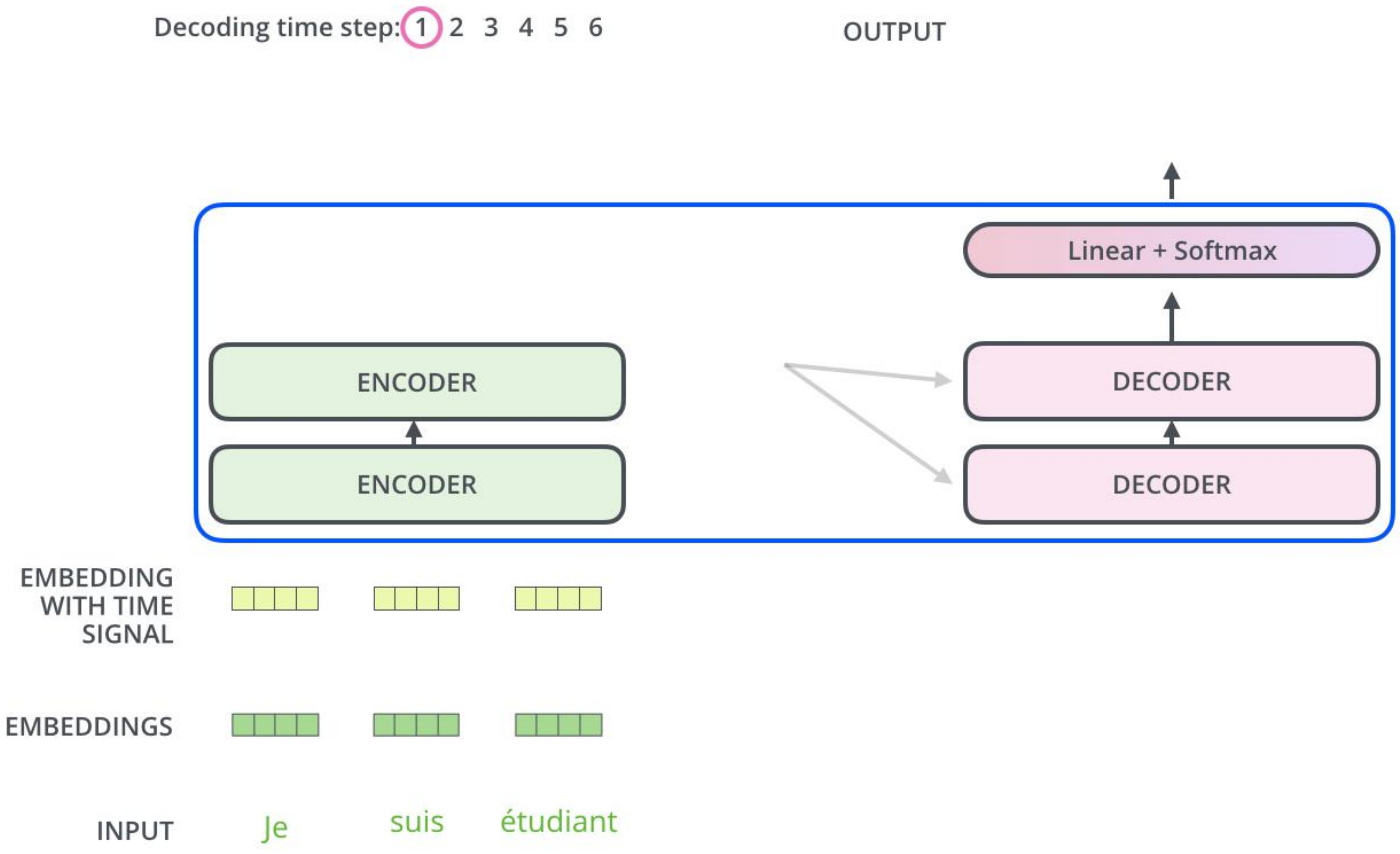
用Mask矩阵作用于Outputs中的每一个10X10单元矩阵：mask矩阵中元素‘1’对应的Outputs单元元素保留原值，‘0’对应的Outputs单元元素替换为负极大值；

原值	负极大	负极大	负极大	负极大	负极大	负极大	负极大	负极大	负极大
原值	原值	负极大	负极大	负极大	负极大	负极大	负极大	负极大	负极大
原值	原值	原值	负极大	负极大	负极大	负极大	负极大	负极大	负极大
原值	原值	原值	原值	负极大	负极大	负极大	负极大	负极大	负极大
原值	原值	原值	原值	原值	负极大	负极大	负极大	负极大	负极大
原值	原值	原值	原值	原值	原值	负极大	负极大	负极大	负极大
原值	原值	原值	原值	原值	原值	原值	负极大	负极大	负极大
原值	原值	原值	原值	原值	原值	原值	原值	负极大	负极大
原值	原值	原值	原值	原值	原值	原值	原值	原值	负极大
原值	原值	原值	原值	原值	原值	原值	原值	原值	原值

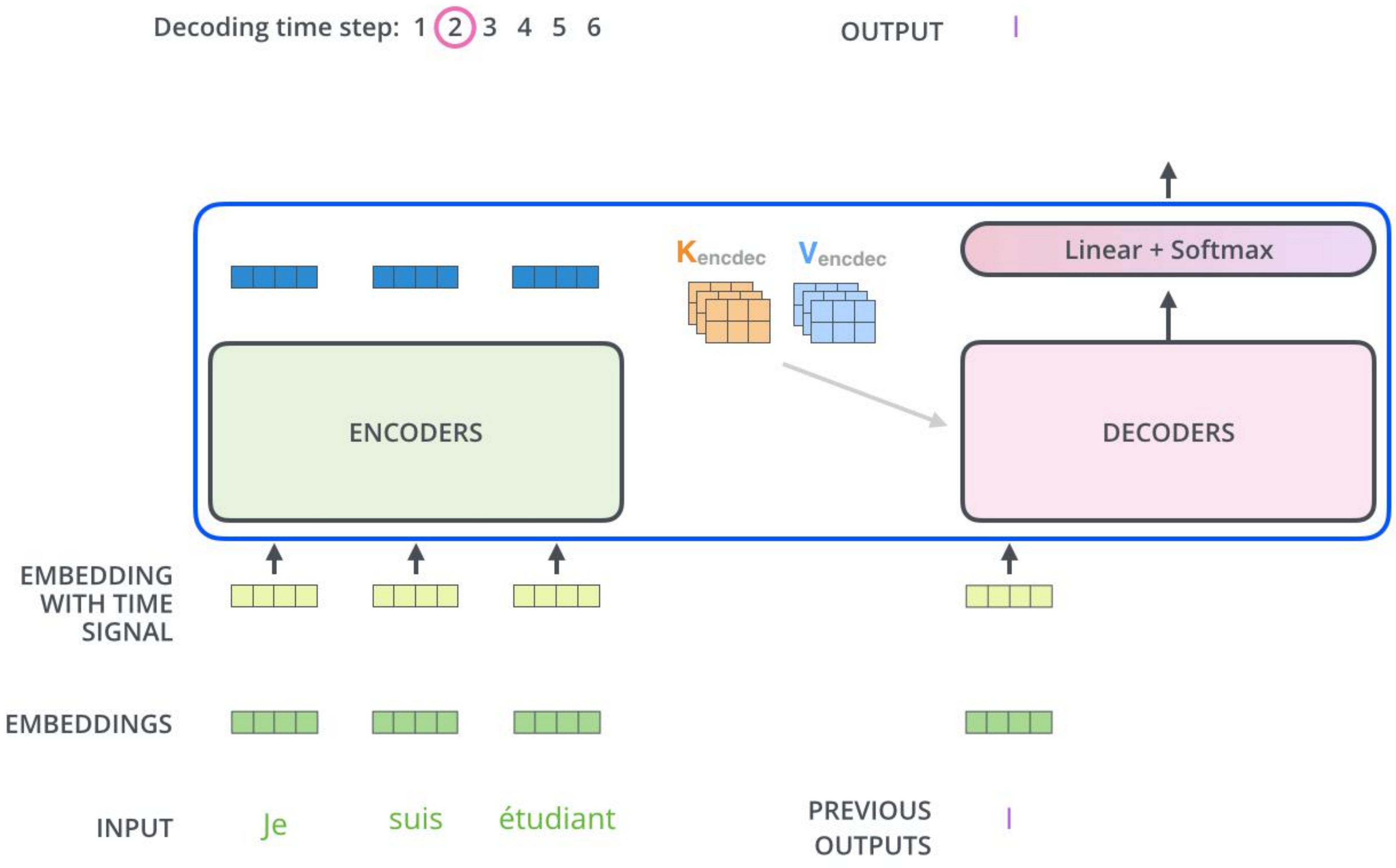
10

10

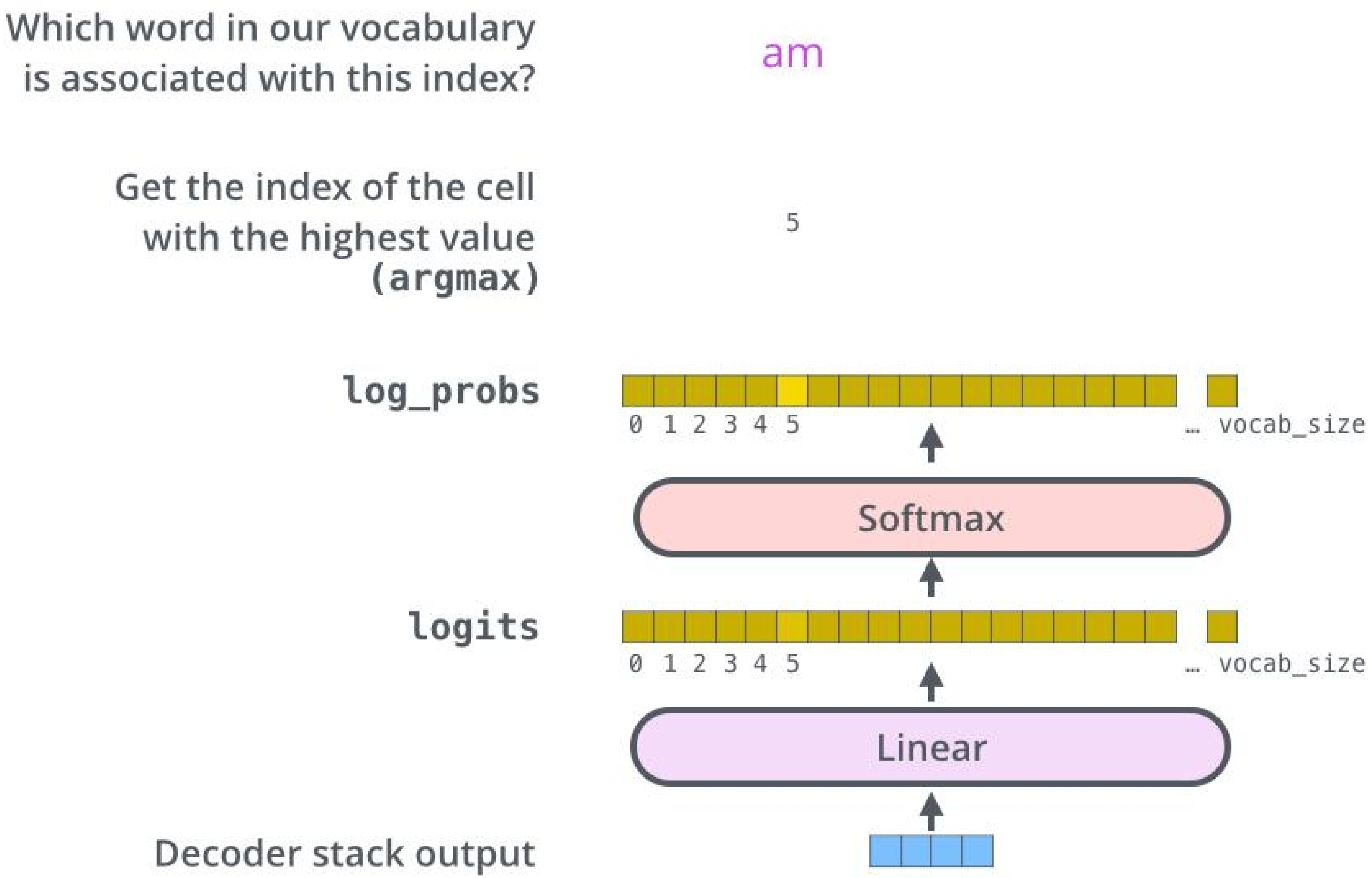
Transformer
Decoder



Transformer
Decoder



Transformer
Final Linear and Softmax Layer



Transformer
Training

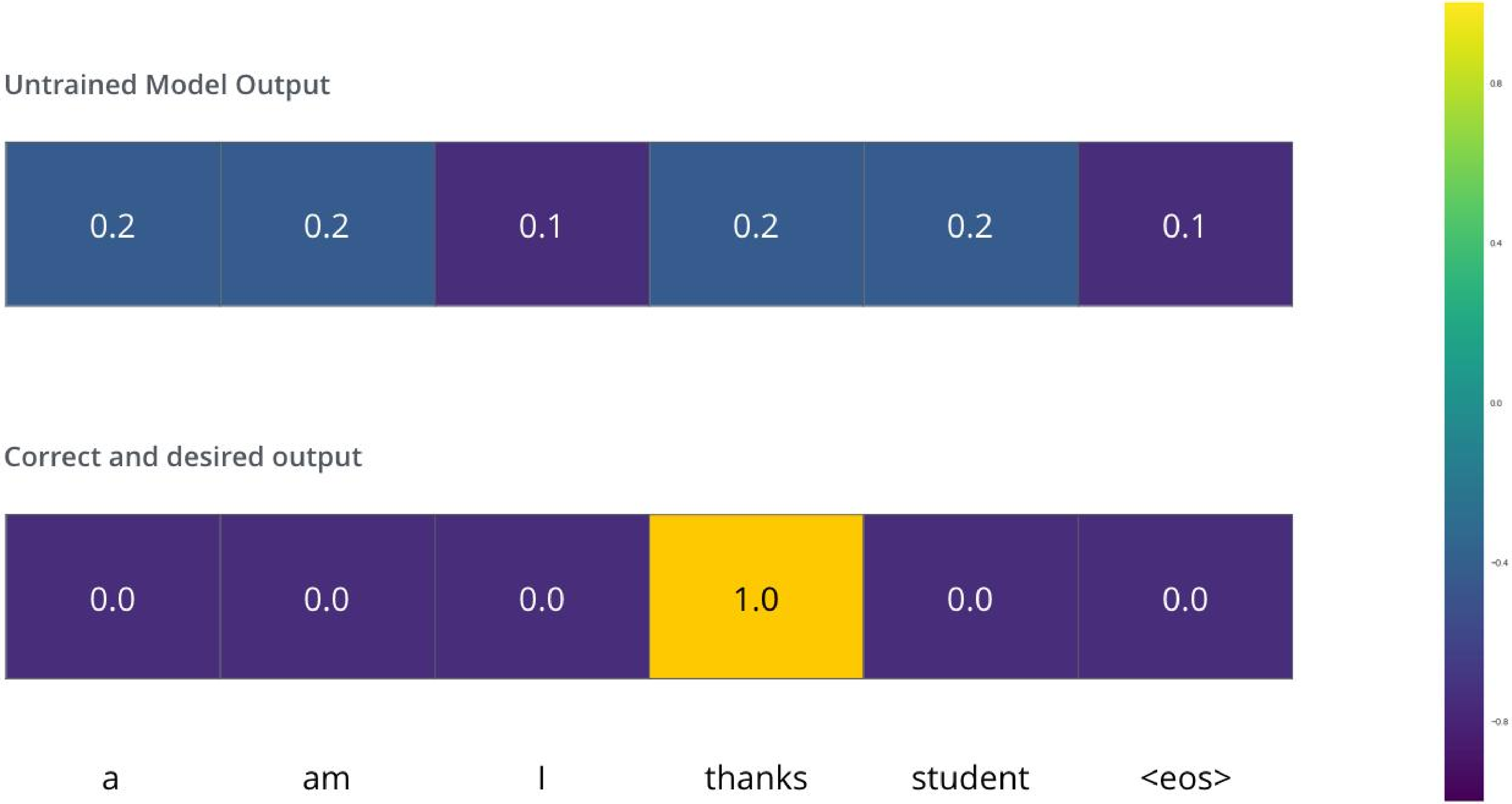
Output Vocabulary

WORD	a	am	I	thanks	student	<eos>
INDEX	0	1	2	3	4	5

One-hot encoding of the word "am"

0.0	1.0	0.0	0.0	0.0	0.0

Transformer
Training



Transformer
Training

Target Model Outputs

Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.0	0.0	1.0	0.0	0.0	0.0
position #2	0.0	1.0	0.0	0.0	0.0	0.0
position #3	1.0	0.0	0.0	0.0	0.0	0.0
position #4	0.0	0.0	0.0	0.0	1.0	0.0
position #5	0.0	0.0	0.0	0.0	0.0	1.0
	a	am	I	thanks	student	<eos>

Trained Model Outputs

Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.01	0.02	0.93	0.01	0.03	0.01
position #2	0.01	0.8	0.1	0.05	0.01	0.03
position #3	0.99	0.001	0.001	0.001	0.002	0.001
position #4	0.001	0.002	0.001	0.02	0.94	0.01
position #5	0.01	0.01	0.001	0.001	0.001	0.98
	a	am	I	thanks	student	<eos>

Transformer

多头注意力（Multi-headed attention）机制

1、由编码器和解码器组成，在编码器的一个网络块中，由一个多头attention子层和一个前馈神经网络子层组成，整个编码器栈式搭建了N个块。类似于编码器，只是解码器的一个网络块中多了一个多头attention层。为了更好的优化深度网络整个网络使用了残差连接和对层进行了规范化（Add&Norm）。

- Encoder and Decoder Stacks
  - **Attention**
  - Position-wise Feed-Forward Networks
  - **Positional Encoding**
  - Add & Norm

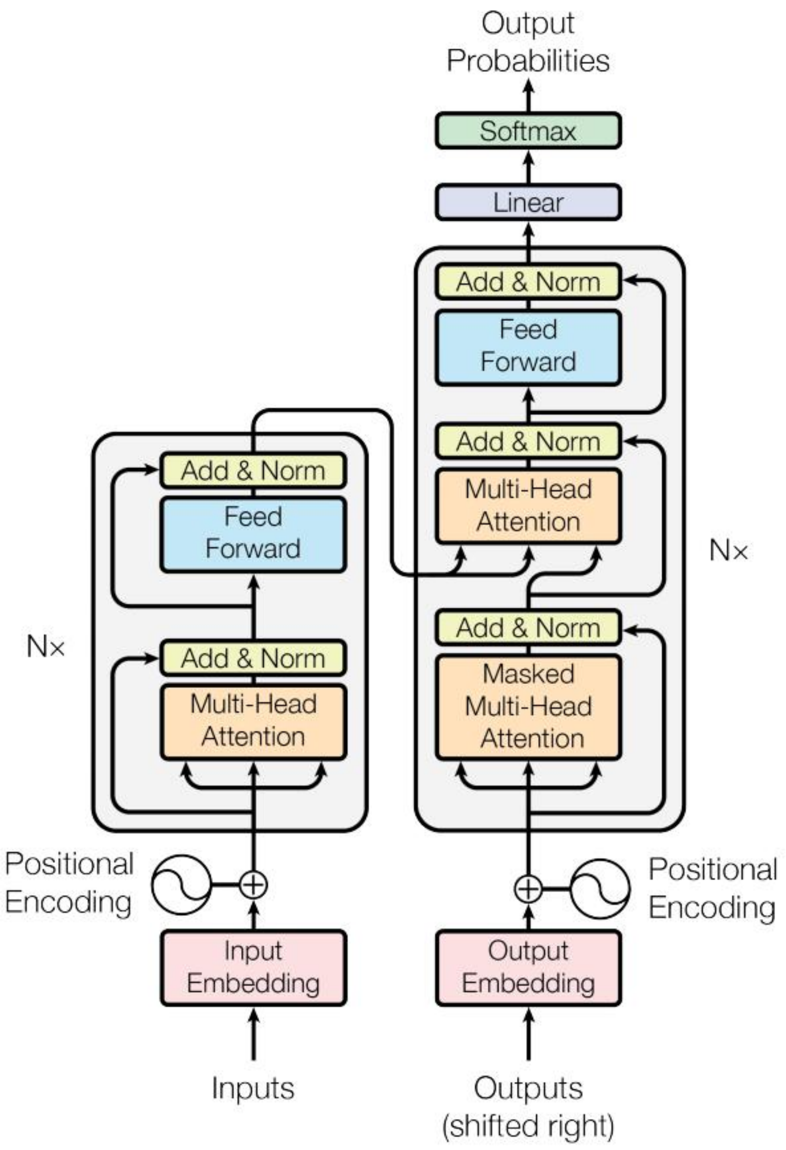


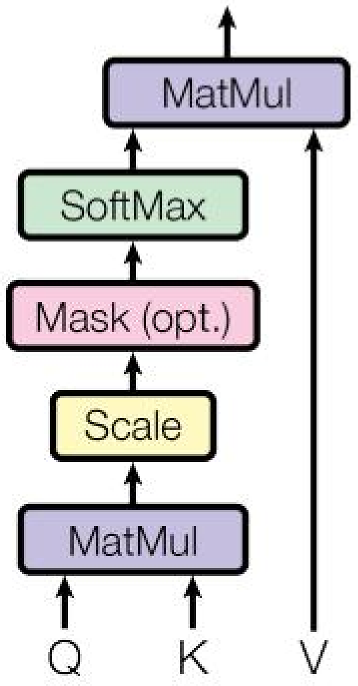
Figure 1: The Transformer - model architecture.

Transformer

2、放缩点积attention（scaled dot-Product attention）。对比我在前面背景知识里提到的attention的一般形式，其实scaled dot-Product attention就是我们常用的使用点积进行相似度计算的attention，只是多除了一个（为*K*的维度）起到调节作用，使得内积不至于太大。

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

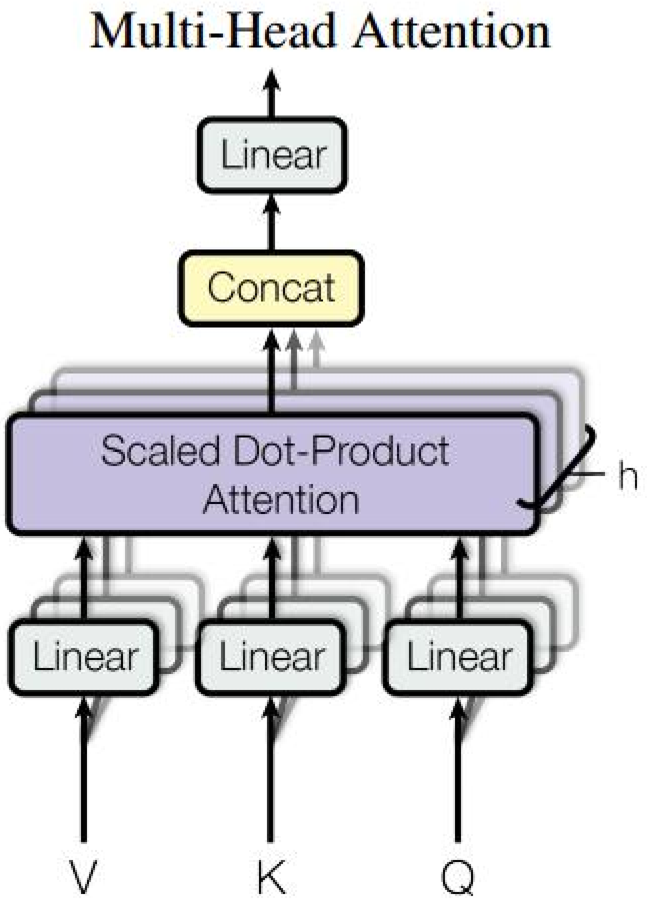
Scaled Dot-Product Attention



Transformer

3、多头attention的Query，Key，Value首先进过一个线性变换，然后输入到放缩点积attention，注意这里要做h次，其实也就是所谓的多头，每一次算一个头。而且每次Q，K，V进行线性变换的参数W是不一样的。然后将h次的放缩点积attention结果进行拼接，再进行一次线性变换得到的值作为多头attention的结果。

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1},...,\mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



其它Transformer结构

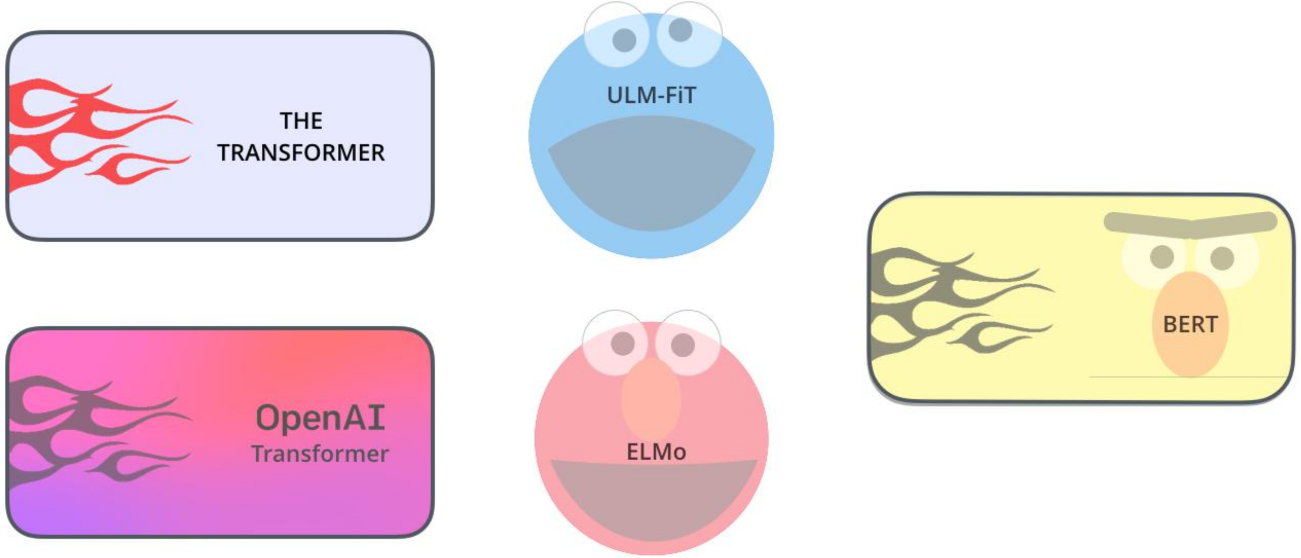
- Weighted Transfomer：
  - https://arxiv.org/pdf/1711.02132.pdf
- Universal Transformer：
  - https://arxiv.org/pdf/1807.03819.pdf
- Gaussian Transformer
- IR Transformer
- NOTE:
  - https://blog.csdn.net/weixin_37947156/article/details/90112176

概念普及

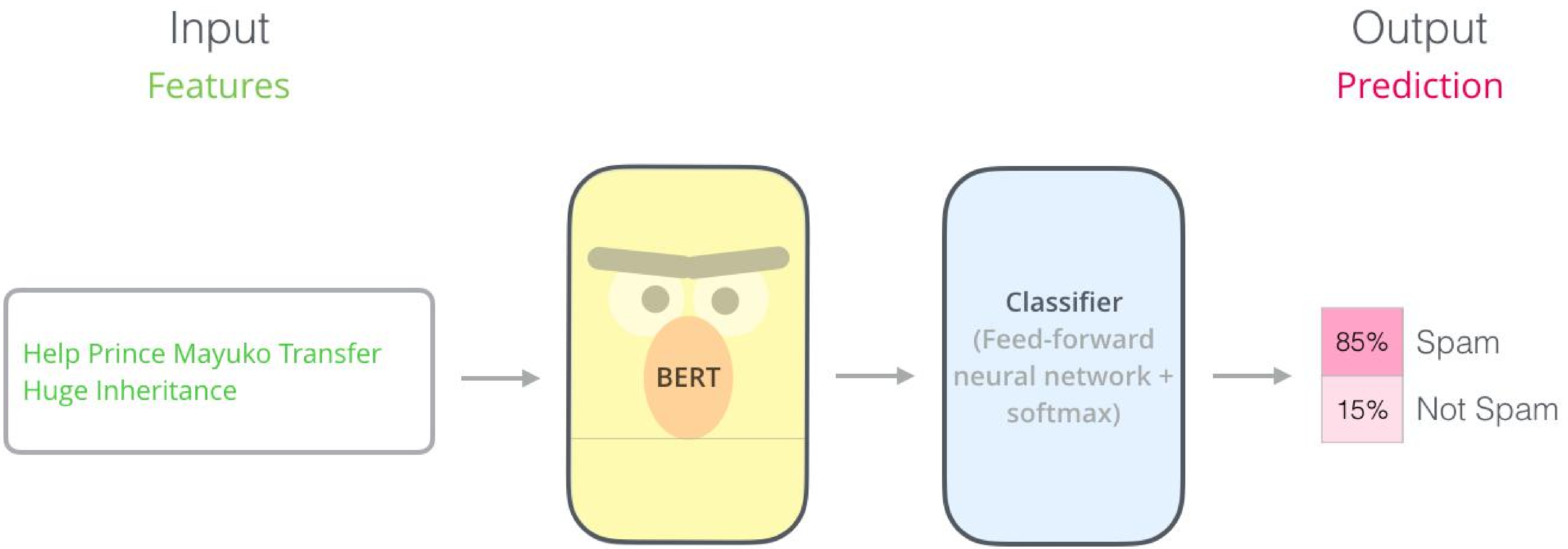
- **语言模型(Language Modeling)**会根据前面单词来预测下一个单词。常用的语言模型有：N-gram、Word2Vec、ELMo、OpenAI GPT、Bert、XLNet；
- Mask：遮挡掩盖的意思，比如：把需要预测的词给挡住。主要出现出现在OpenAI GPT和Bert中。

Bert

- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
  - 2018, Google, https://arxiv.org/pdf/1810.04805.pdf
  - New Features:
    - **Bidirectional Transformers**
    - **Pre-training**
      - **Masked Language Model**
      - **Next Sentence Prediction**

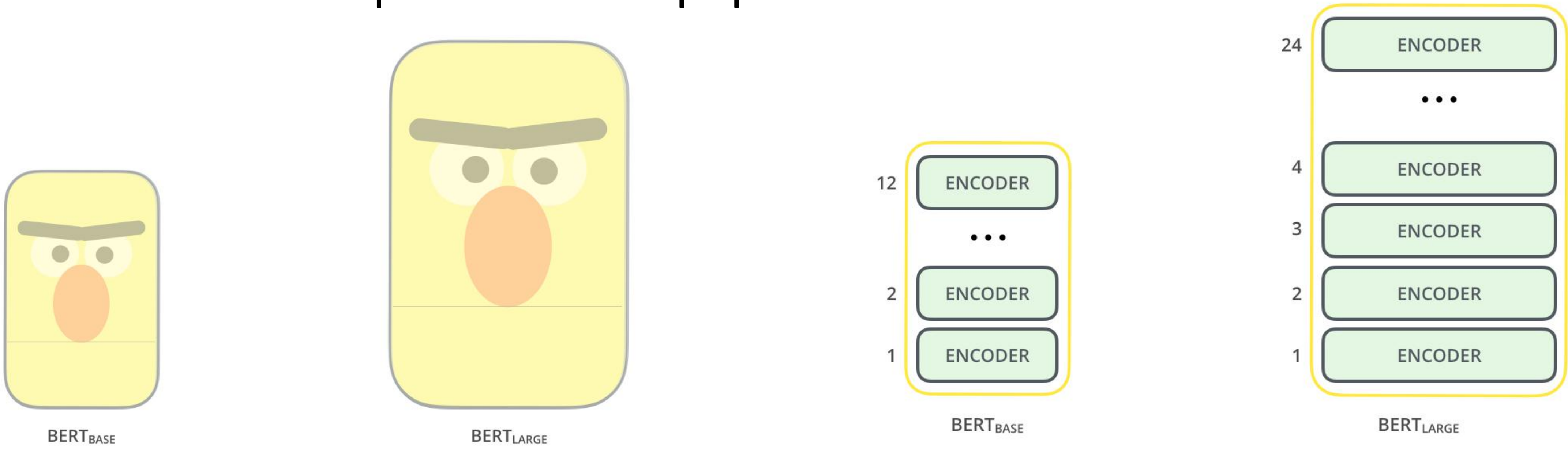


Bert


Input
Features
Output
Prediction
Help Prince Mayuko Transfer Huge Inheritance
BERT
Classifier
(Feed-forward neural network + softmax)
85%
Spam
15%
Not Spam

Bert

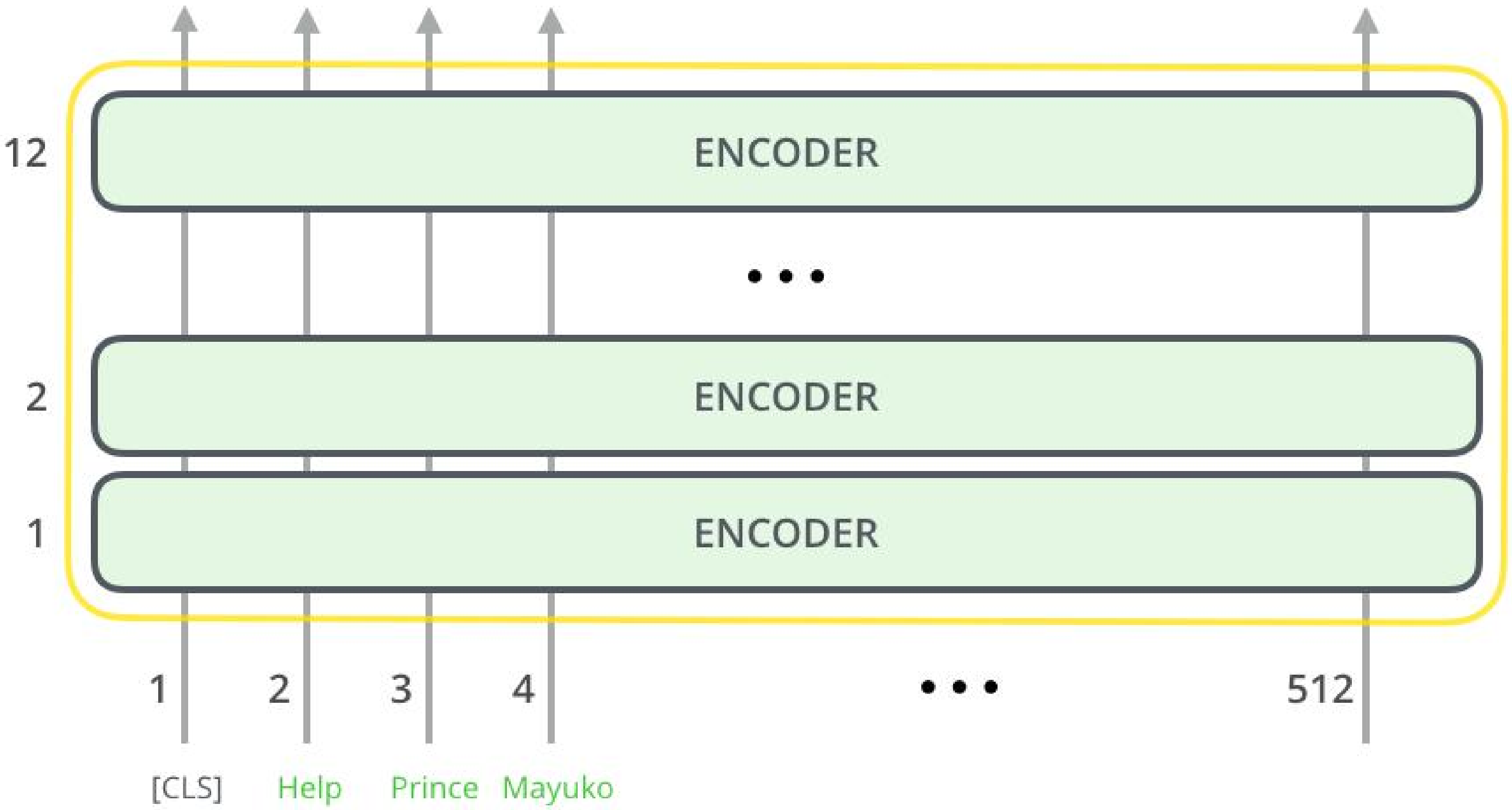
- **BERT BASE**： Comparable in size to the OpenAI Transformer in order to compare performance。
- **BERT LARGE**： A ridiculously huge model which achieved the state of the art results reported in the paper。



Bert

- Compared With Transformer:
  - Encoder Layers: 6 --> 12/24
  - feed-forward NN units: 512 --> 768/1024
  - Multi Headed Attention: 8 --> 12/16
  - Encoder Mask： no --> yes
  - Embedding: word+position --> word+position+segment

Bert
Model Input



BERT

Bert
Model Input

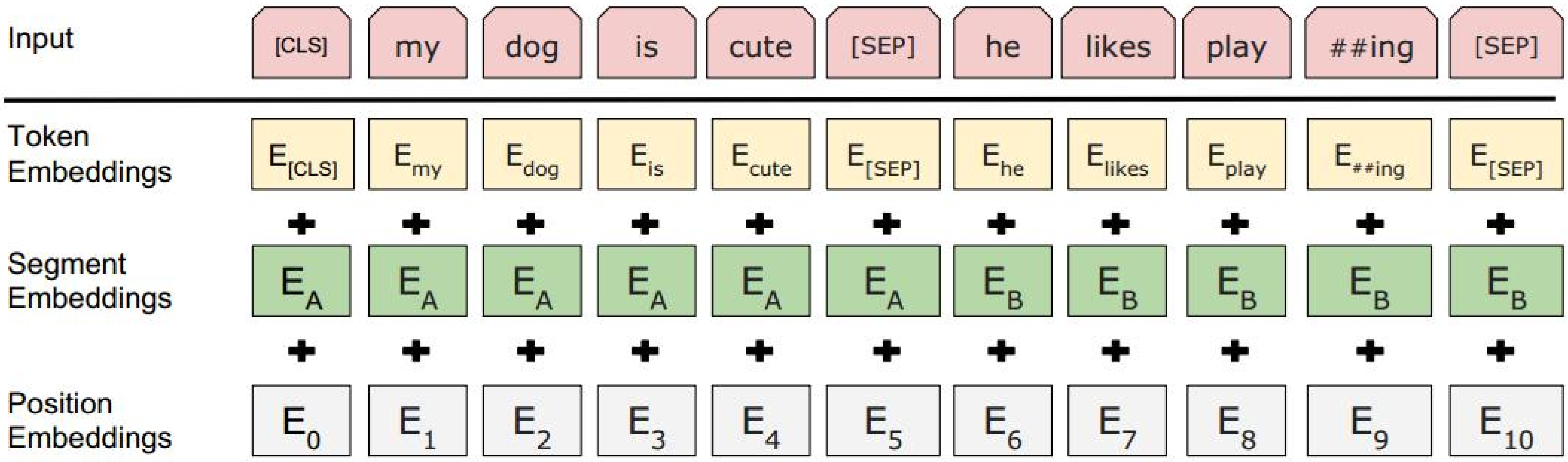
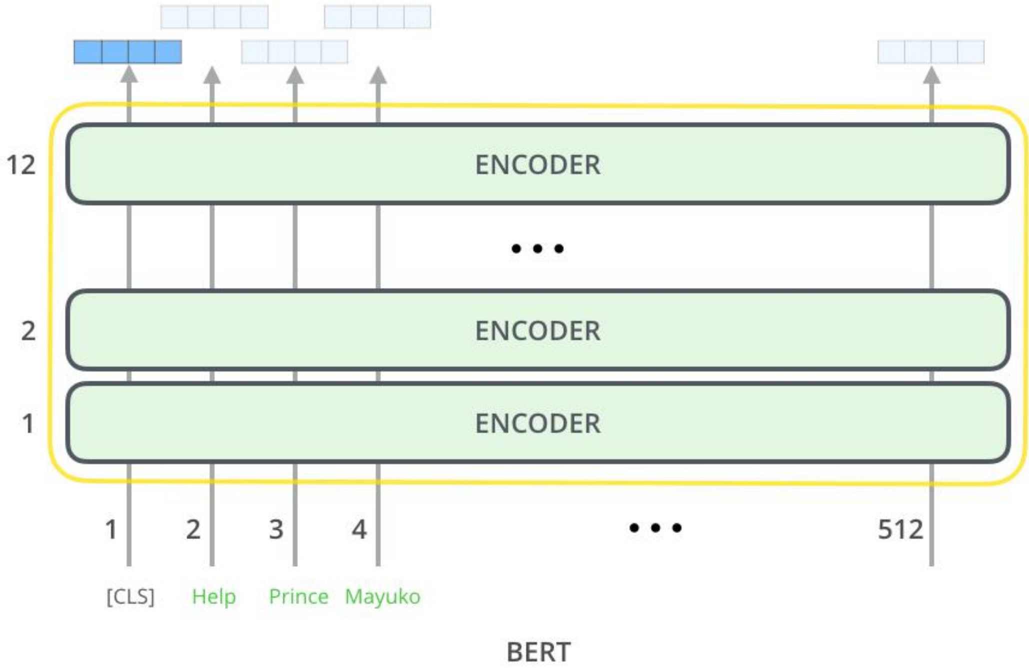
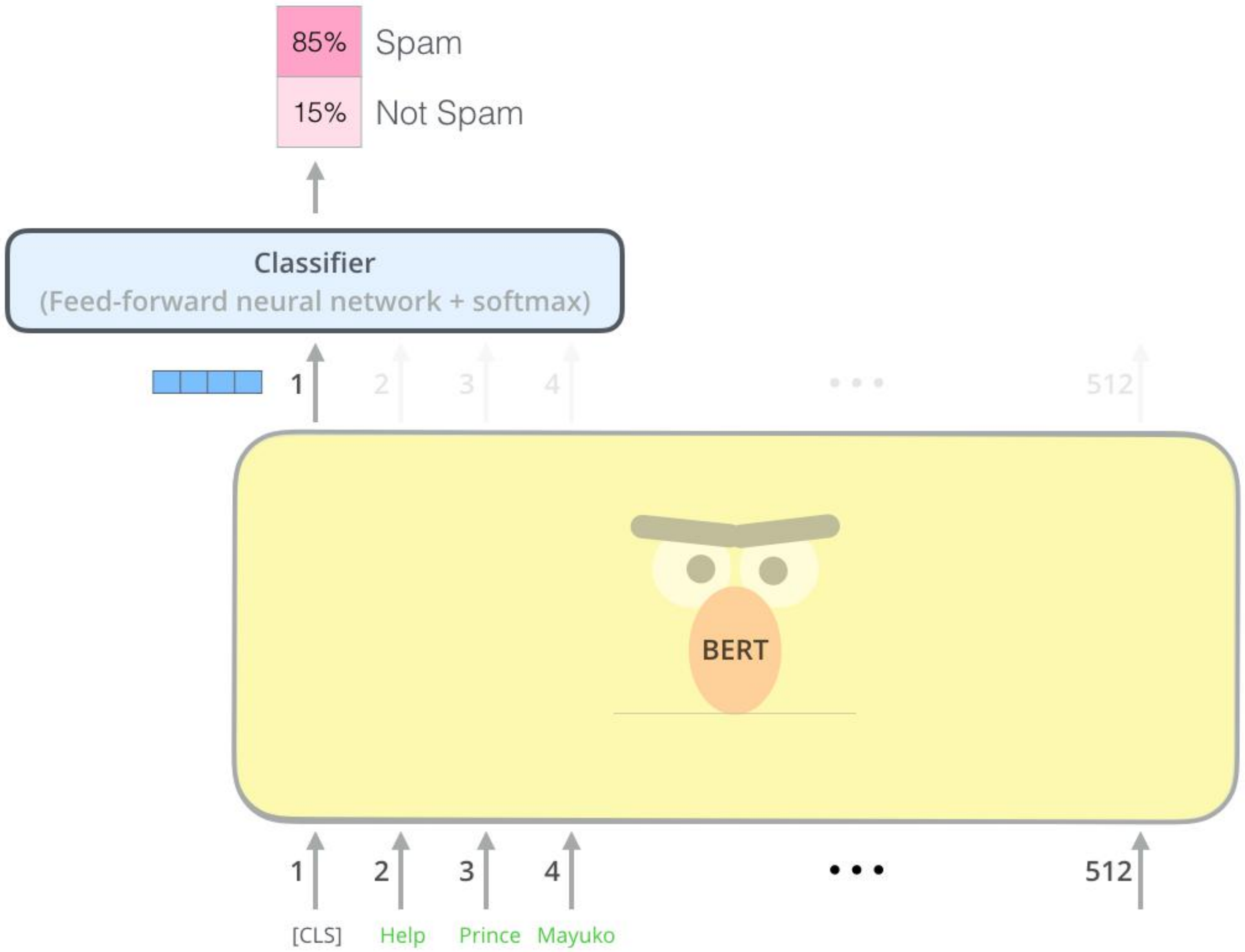


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Not Word Token, Is Word Piece Token(FastText)

Bert
Model Output

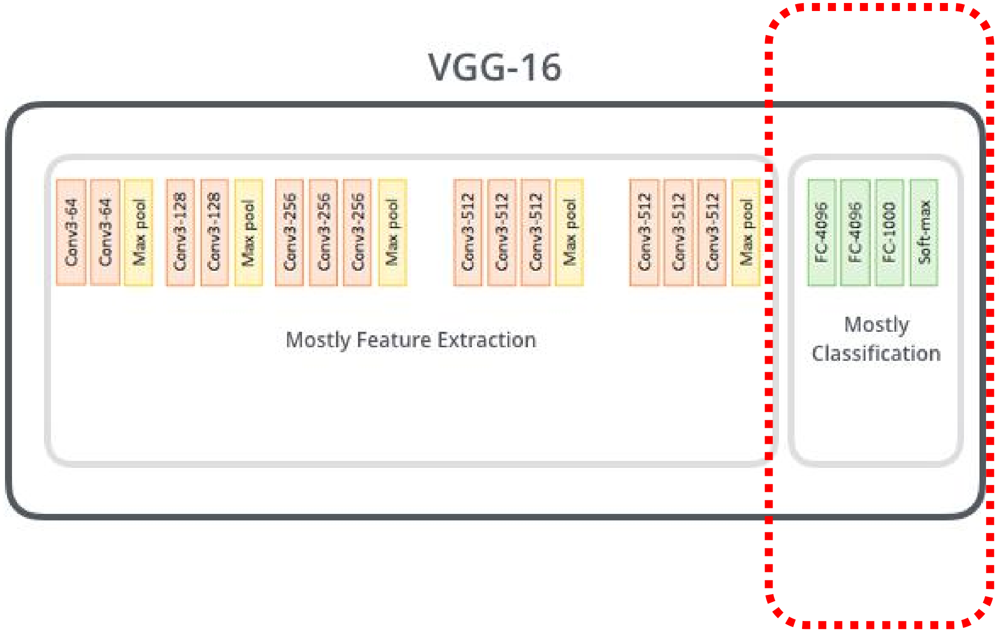




Bert
with VGG

Input

Features



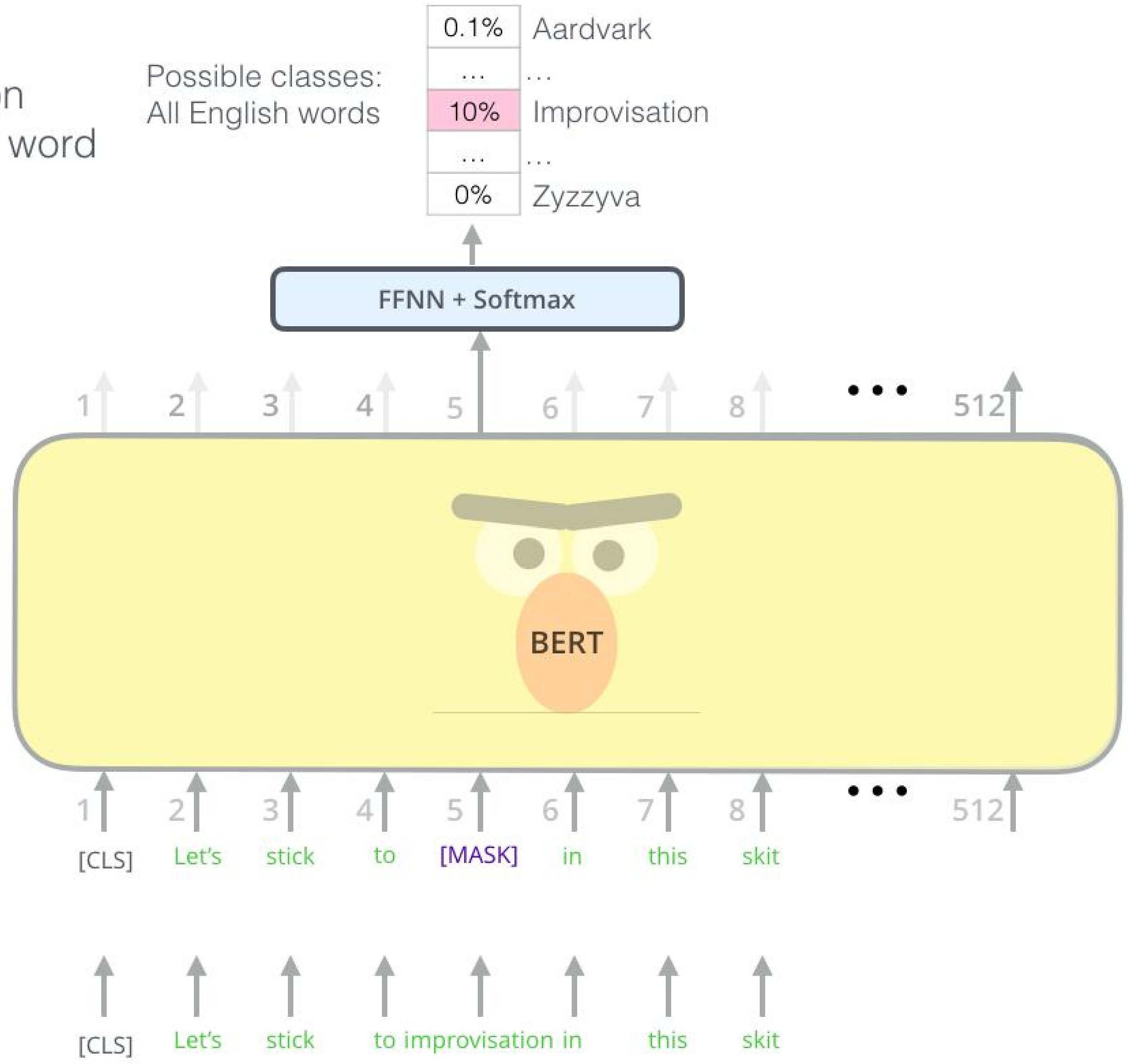
Output

Prediction

0.2%	Kit fox
0.1%	English setter
95%	Egyptian cat
1%	Great Dane
	...
0%	Hotdog

Bert
Masked Language Model

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

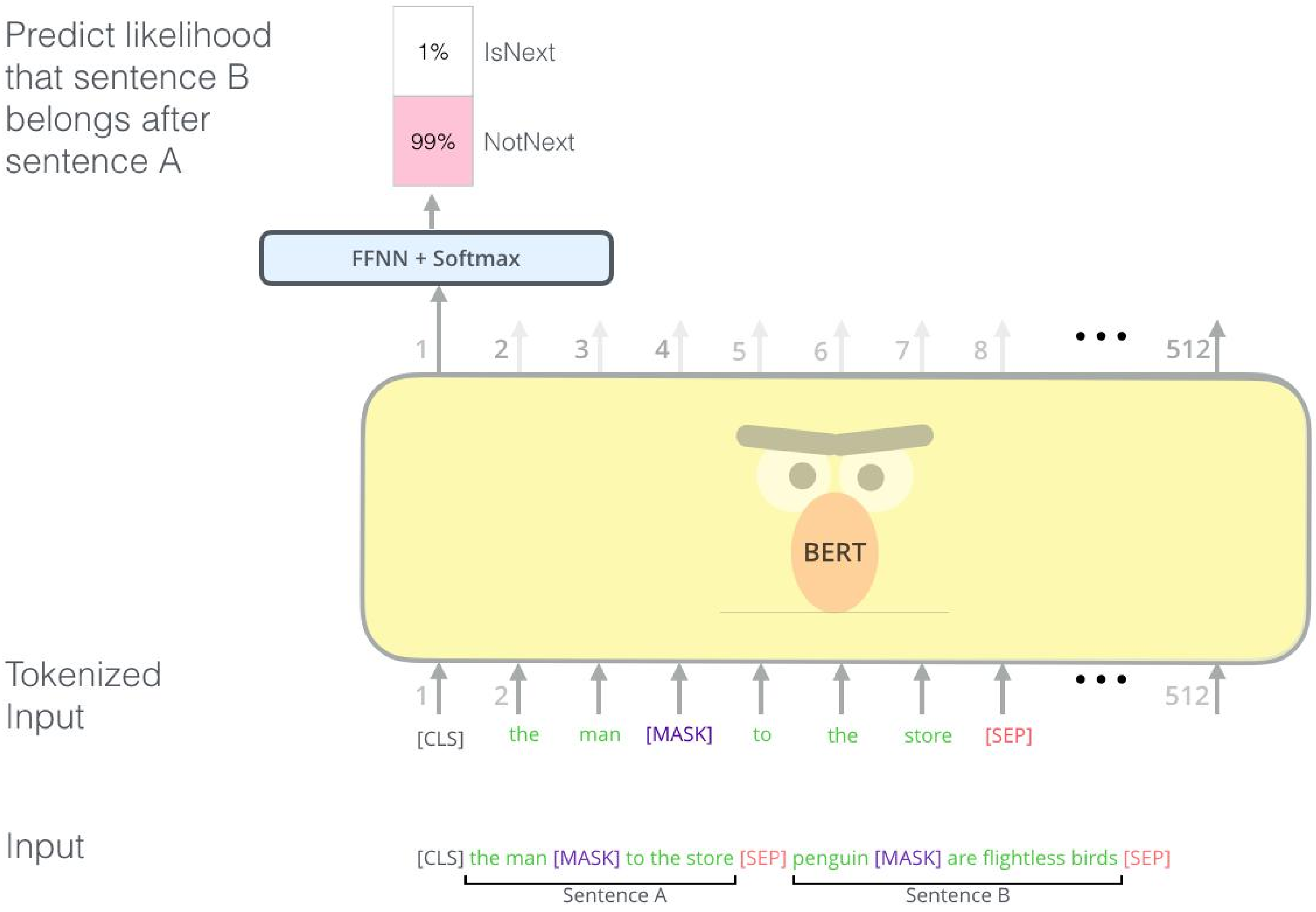


Randomly mask 15% of tokens

Input

Mask + 15%的随机替换

Bert
Next Sentence Prediction



Bert

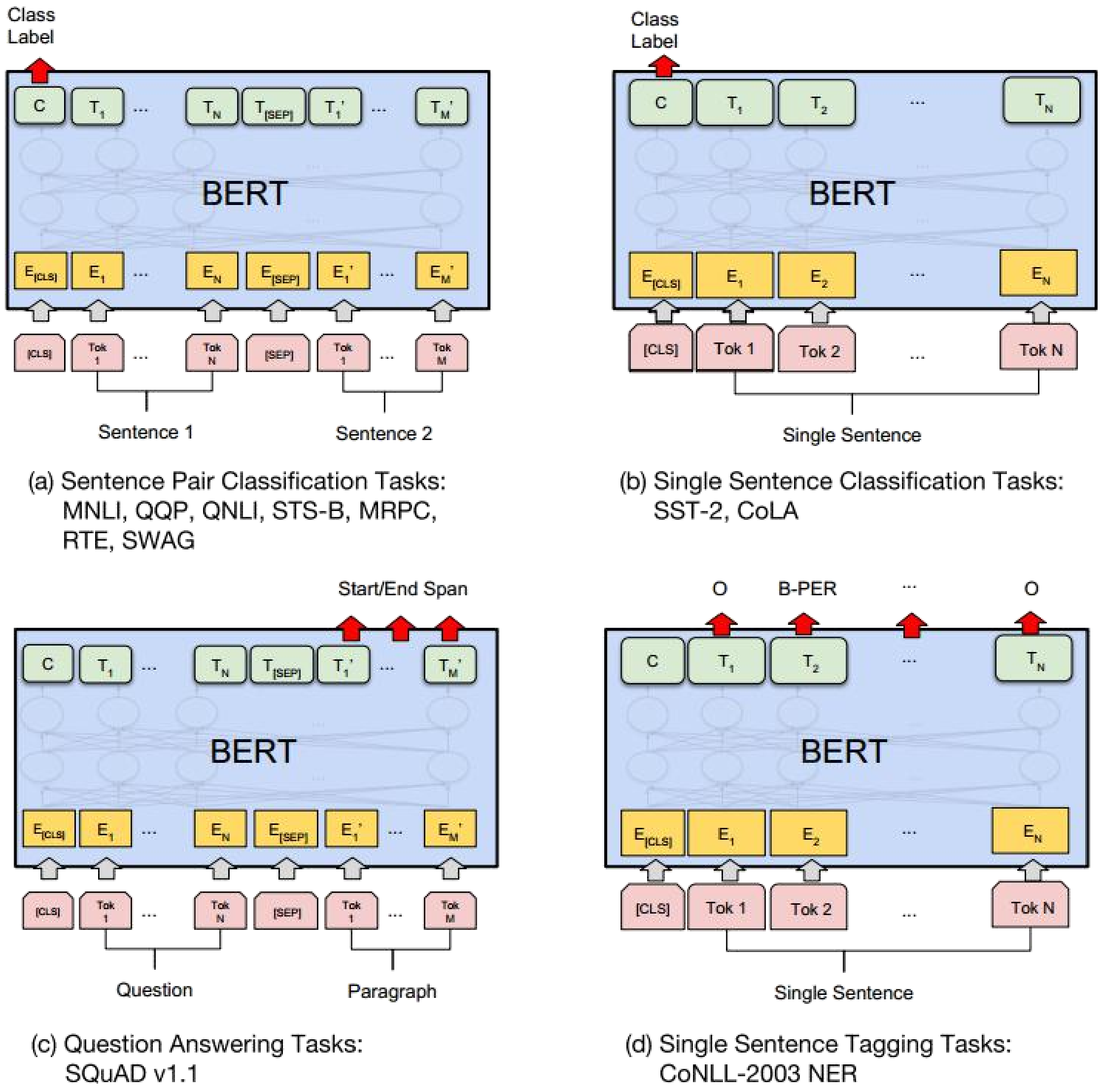
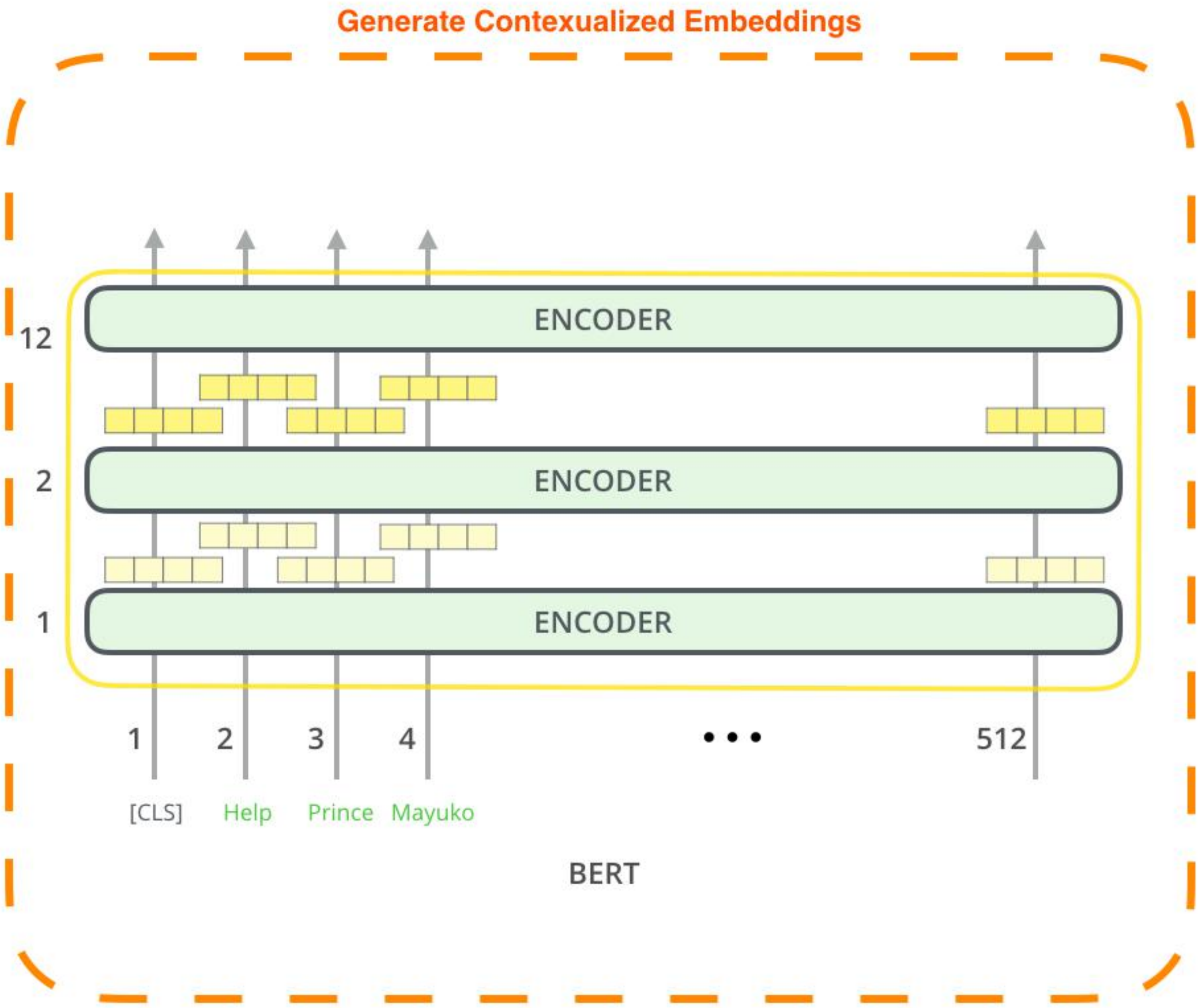
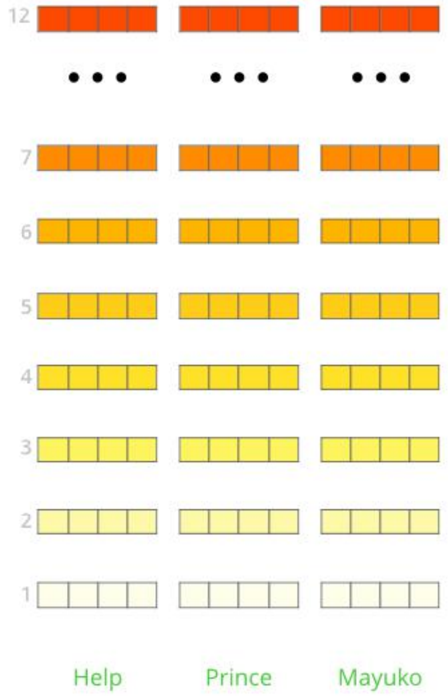


Figure 3: Our task specific models are formed by incorporating BERT with one additional output layer, so a minimal number of parameters need to be learned from scratch. Among the tasks, (a) and (b) are sequence-level tasks while (c) and (d) are token-level tasks. In the figure, E represents the input embedding, T_i represents the contextual representation of token i, [CLS] is the special symbol for classification output, and [SEP] is the special symbol to separate non-consecutive token sequences.

Bert
with Feature Extraction



The output of each encoder layer along each token's path can be used as a feature representing that token.

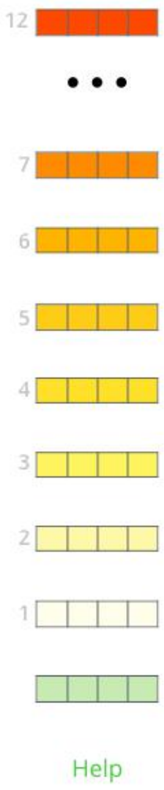


But which one should we use?

Bert
with Feature Extraction

What is the best contextualized embedding for "Help" in that context?

For named-entity recognition task CoNLL-2003 NER



	Dev F1 Score
First Layer	91.0
Last Hidden Layer	94.9
Sum All 12 Layers	95.5
Second-to-Last Hidden Layer	95.6
Sum Last Four Hidden	95.9
Concat Last Four Hidden	96.1

Bert方式

- 参考：

  https://colab.research.google.com/github/tensorflow/tpu/blob/master/tools/colab/bert_finetuning_with_cloud_tpus.ipynb

- 过程：
  - 该模型在modeling.py（BertModel类）中构建。
  - run_classifier.py是微调过程的一个示例。它还构建了监督模型的分类层。如果要构建自己的分类器，请查看该文件中的create_model()方法。
  - 可以下载几种预先训练的模型。涵盖102种语言的多语言模型，这些语言都是在维基百科的数据基础上训练而成的。
  - BERT不会将单词视为tokens。相反，它注重Word Pieces。 tokenization.py是将你的单词转换为适合BERT的wordPieces的tokensizer。
  - 参考：
    - https://github.com/allenai/allennlp
    - https://github.com/huggingface/pytorch-transformers
    - **https://github.com/google-research/bert**

Bert

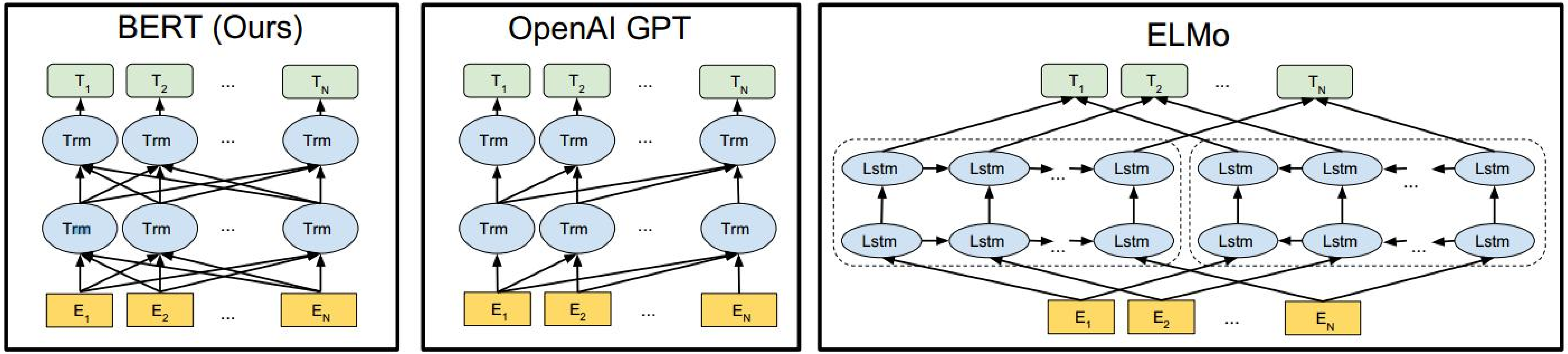
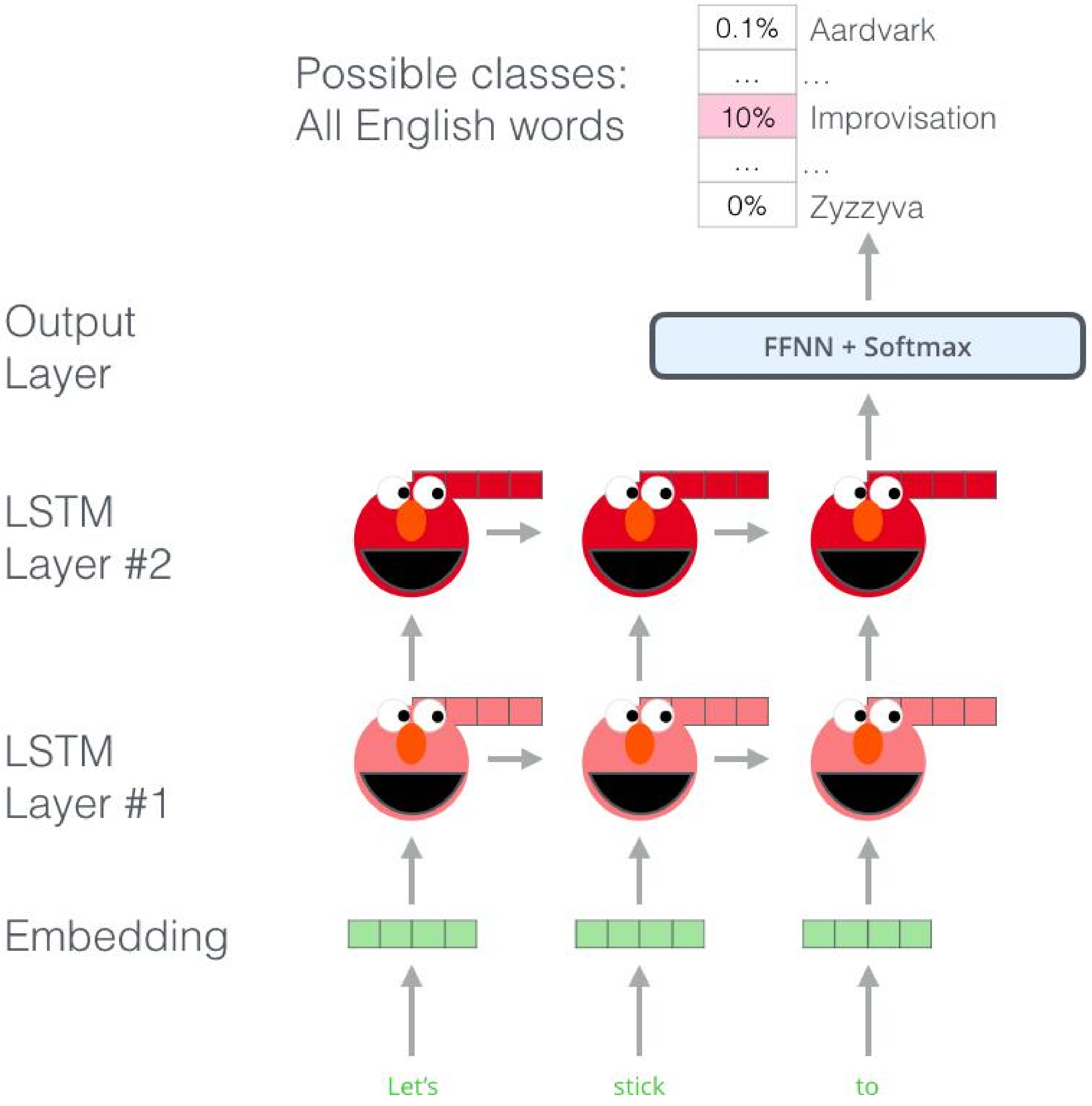


Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

扩展：Bert
with ELMo

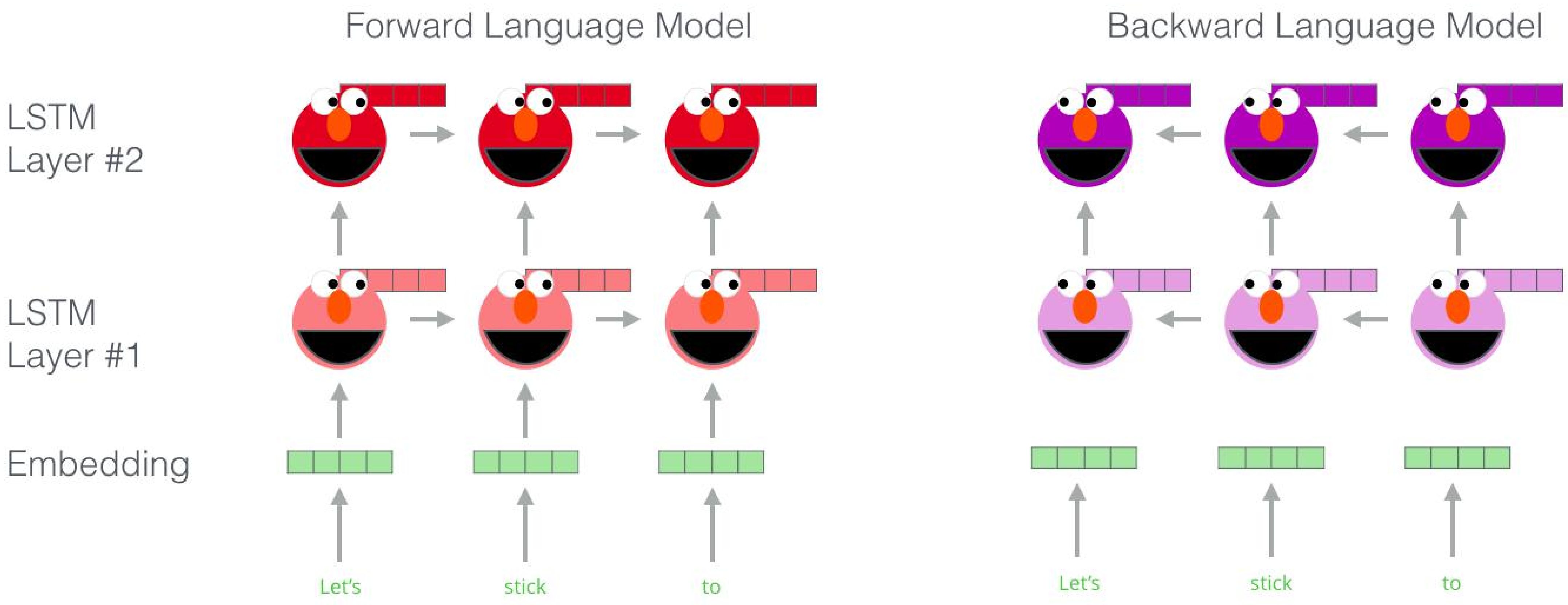
- ELMo： Deep contextualized word representations
  - ELMo gained its language understanding from being trained to predict the next word in a sequence of words - a task called **Language Modeling**.
  - https://arxiv.org/pdf/1802.05365.pdf
  - Features：
    - **Base on Bi-LSTM**

扩展：Bert
with ELMo



扩展：Bert
with ELMo

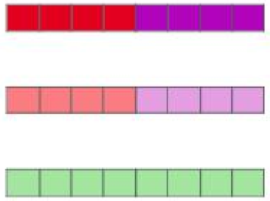
Embedding of “stick” in “Let’s stick to” - Step #1



扩展：Bert
with ELMo

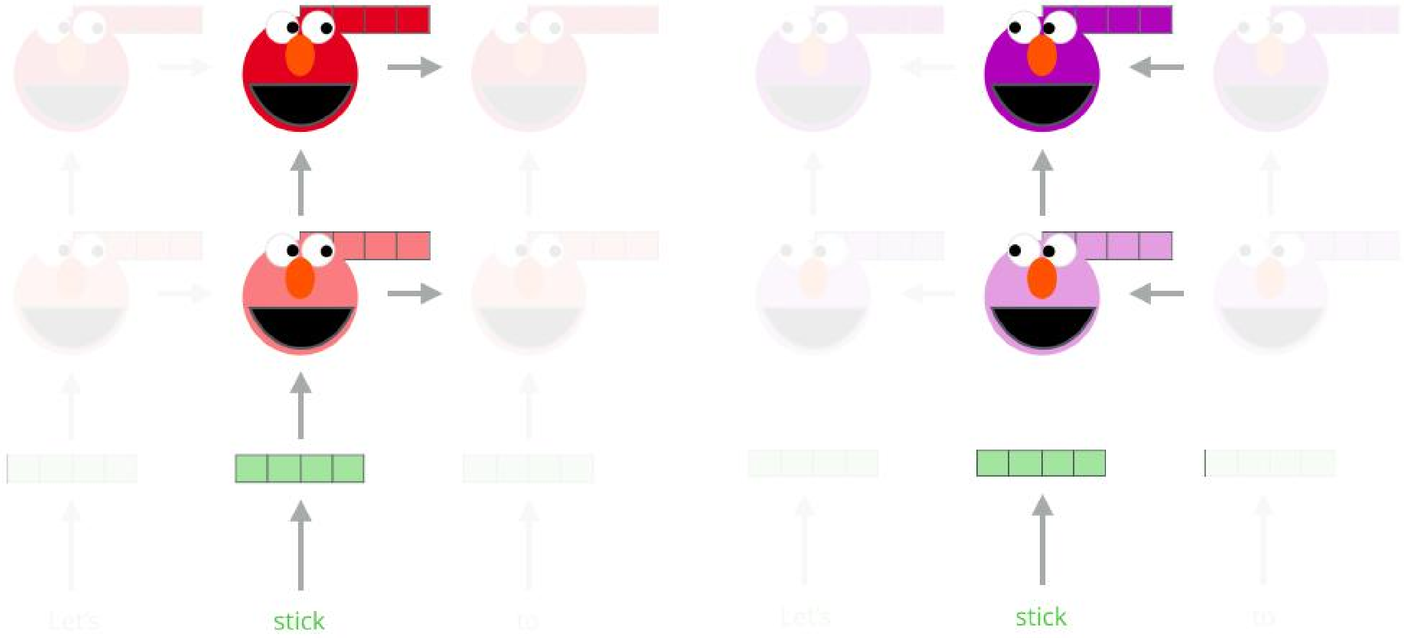
Embedding of “stick” in “Let’s stick to” - Step #2

1- Concatenate hidden layers

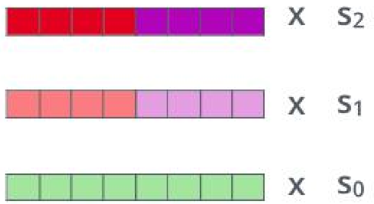


Forward Language Model

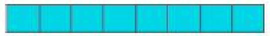
Backward Language Model



2- Multiply each vector by a weight based on the task



3- Sum the (now weighted) vectors



ELMo embedding of “stick” for this task in this context

扩展：Bert

With OpenAI

- OpenAI：
  - 2015，致力于人工智能模型预训练领域的相关技术的开发，主要是对于NLP相关人工智能技术的研发。
  - https://www.openai.com/

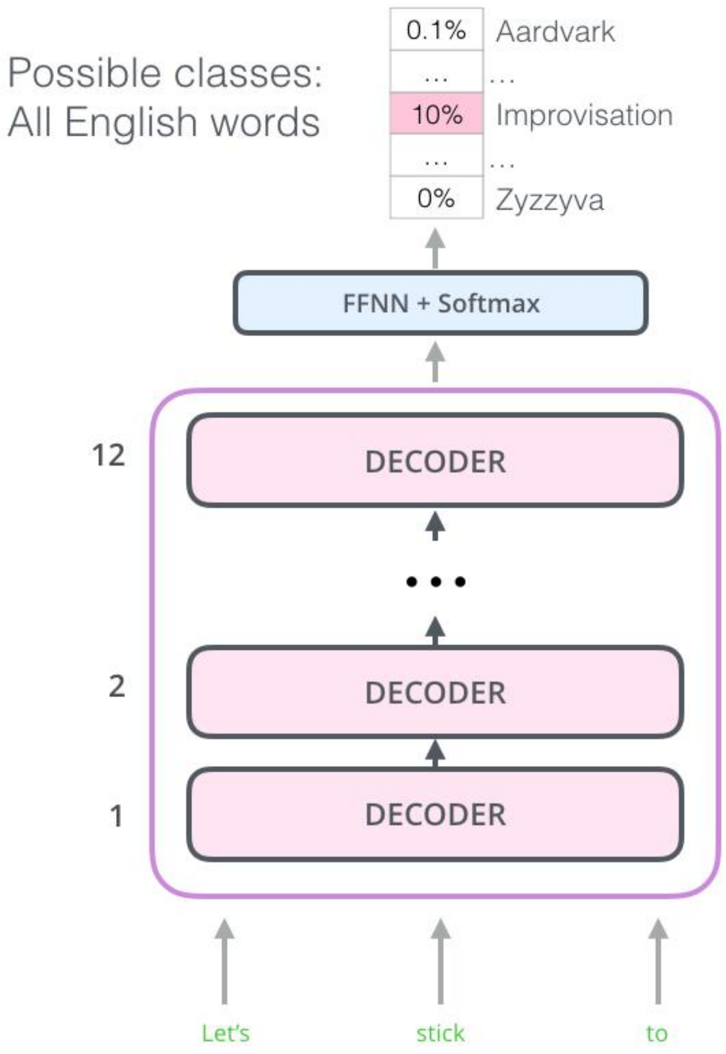
扩展：Bert

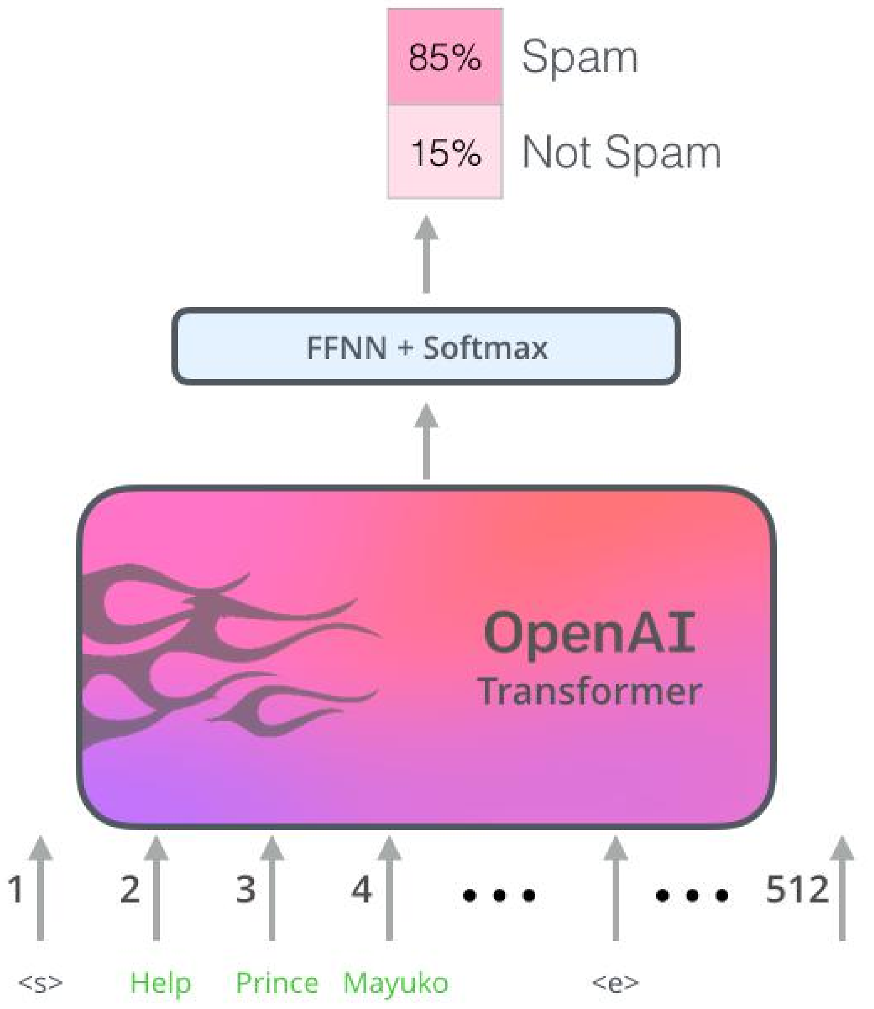
With OpenAI GPT

- GPT在BooksCorpus(800M单词)训练；BERT在BooksCorpus(800M单词)和维基百科(2,500M单词)训练。
- GPT使用一种句子分隔符([SEP])和分类符词块([CLS])，它们仅在微调时引入；BERT在预训练期间学习[SEP]，[CLS]和句子A/B嵌入。
- GPT用一个批量32,000单词训练1M步；BERT用一个批量128,000单词训练1M步。
- GPT对所有微调实验使用的5e-5相同学习率；BERT选择特定于任务的微调学习率，在开发集表现最佳。
- GPT是12层，Bert是24层。
- GTP使用的是Transformer的类似Decoder结构(单向的Transformer，里面没有Encoder-Decoder Attention，只有Mask Self-Attention和FFNN)，Bert使用的是Encoder结构(双向Transformer)

扩展：Bert

With OpenAI GPT





扩展：Bert

With OpenAI