NLP项目

Bert

课程内容

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

Seq2Seq结构回顾

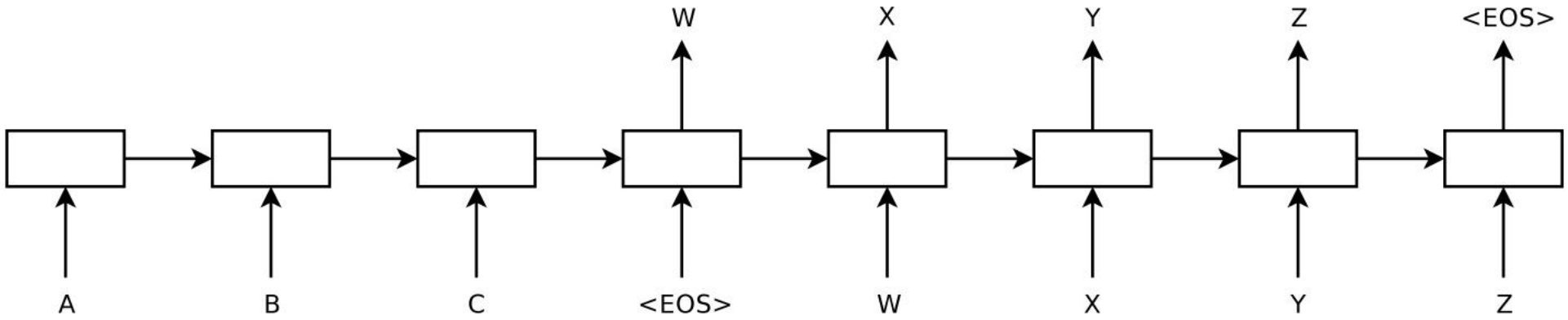
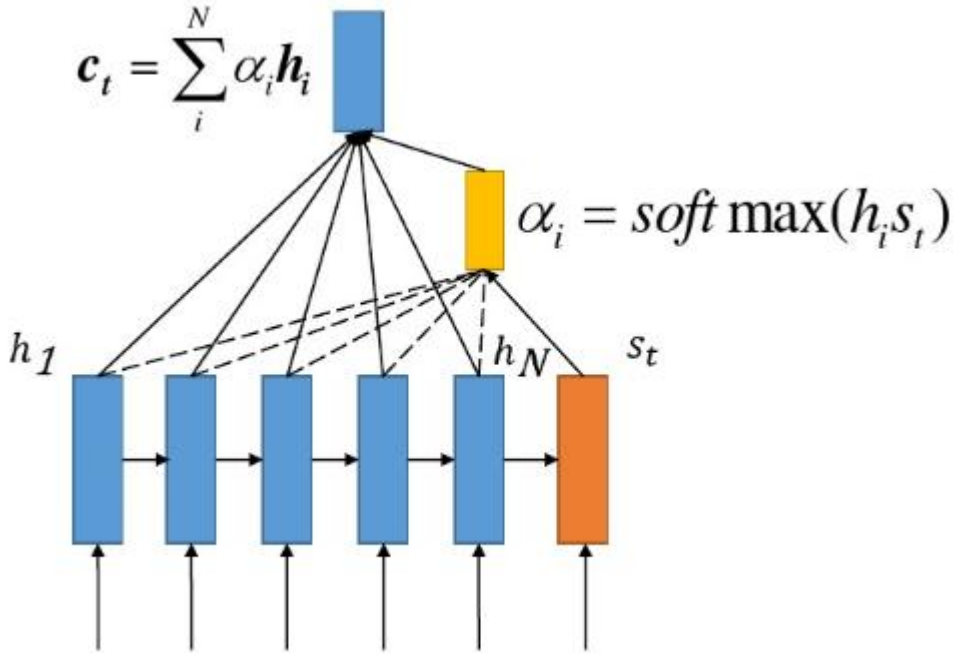
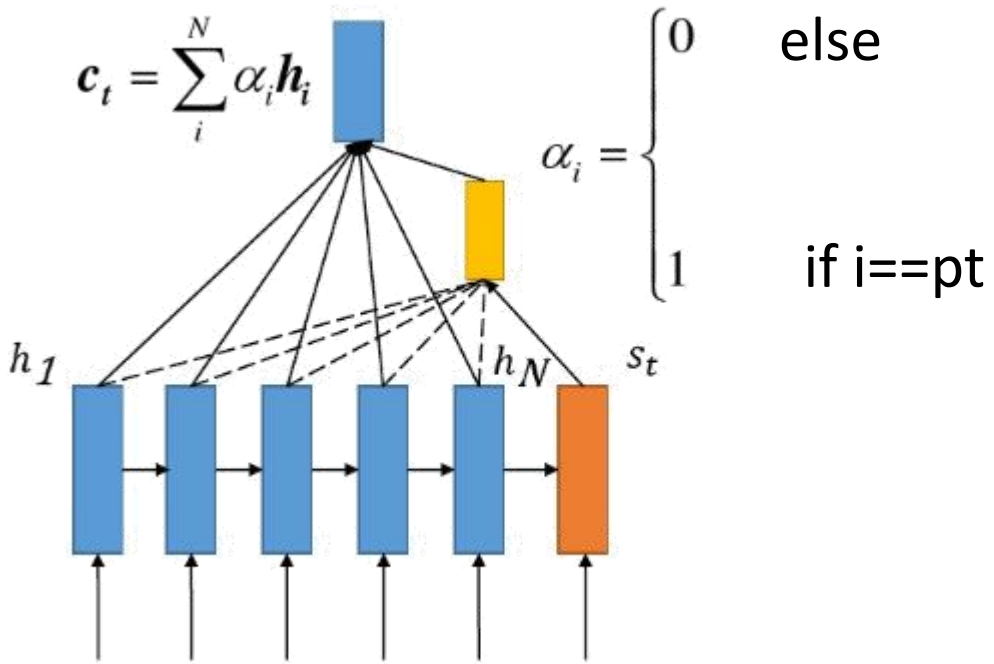


Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that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

Attention结构回顾



Soft Attention



Hard Attention

Attention结构回顾

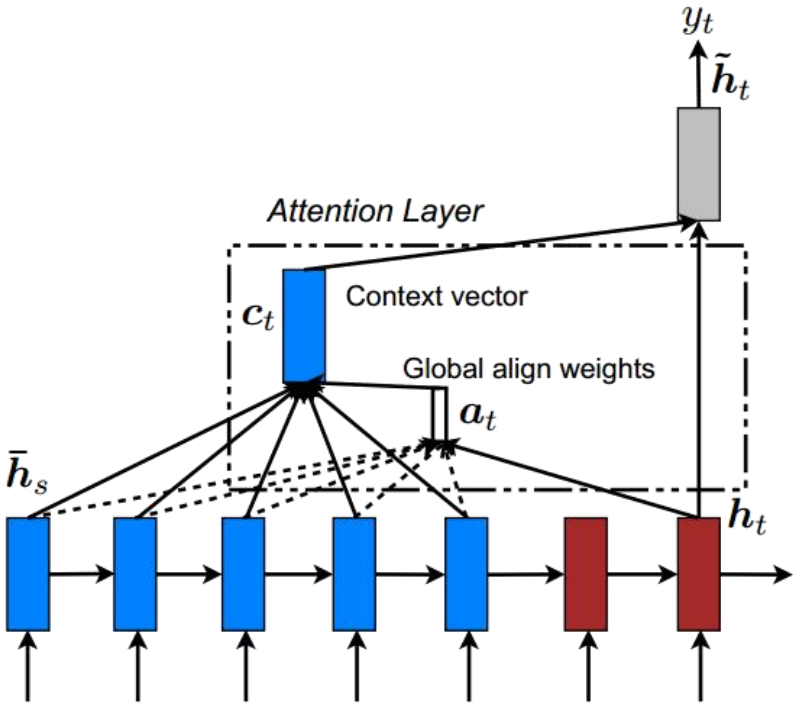


Figure 2: **Global attentional model** – at each time step t, the model infers a *variable-length* alignment weight vector $\boldsymbol{a}_t$ based on the current target state $\boldsymbol{h}_t$ and all source states $\bar{\boldsymbol{h}}_s$. A global context vector $\boldsymbol{c}_t$ is then computed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

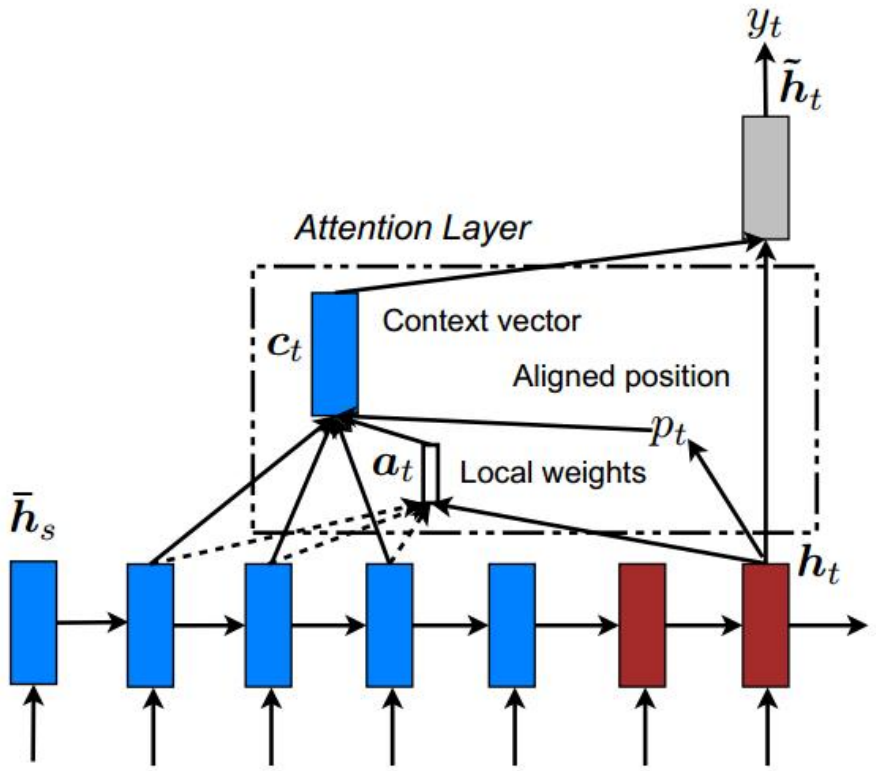
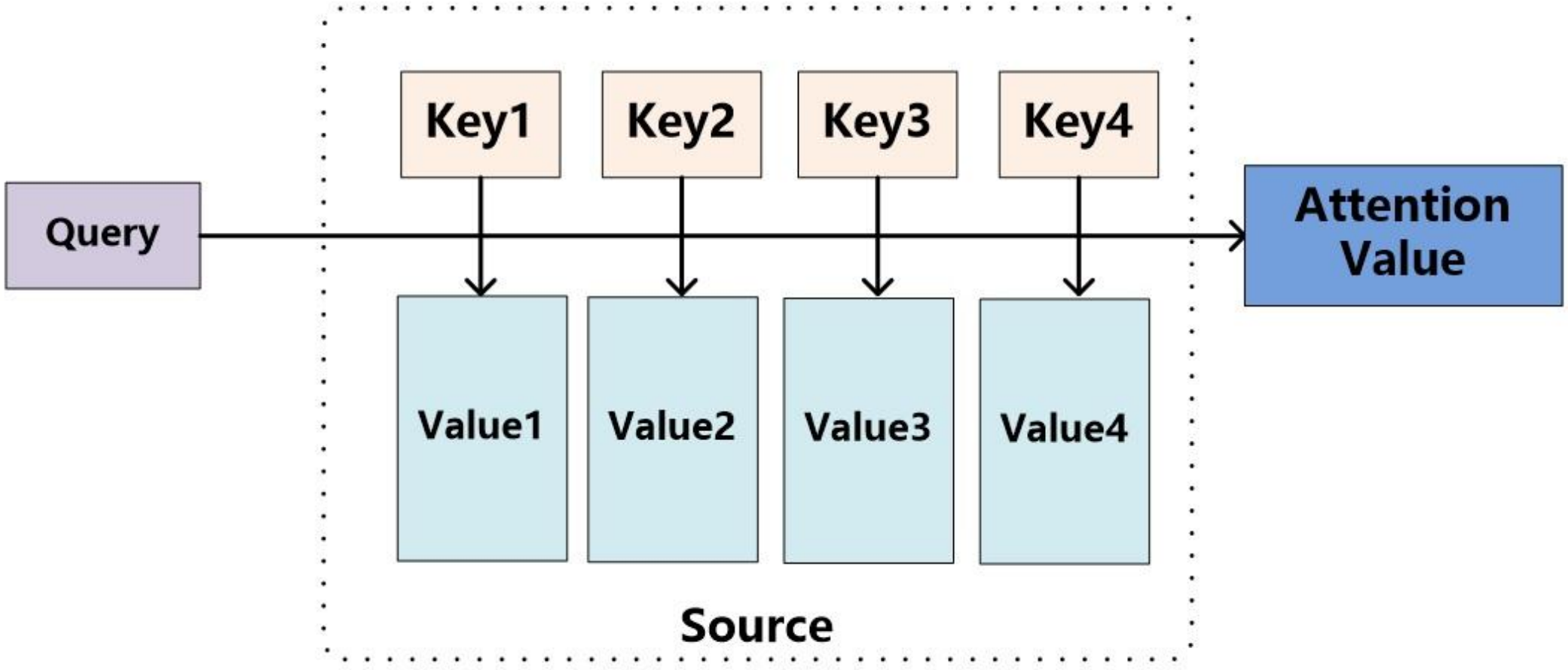


Figure 3: **Local attention model** – the model first predicts a single aligned position p_t for the current target word. A window centered around the source position p_t is then used to compute a context vector $\boldsymbol{c}_t$, a weighted average of the source hidden states in the window. The weights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

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

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

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

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

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

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

Transformer回顾

- 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**

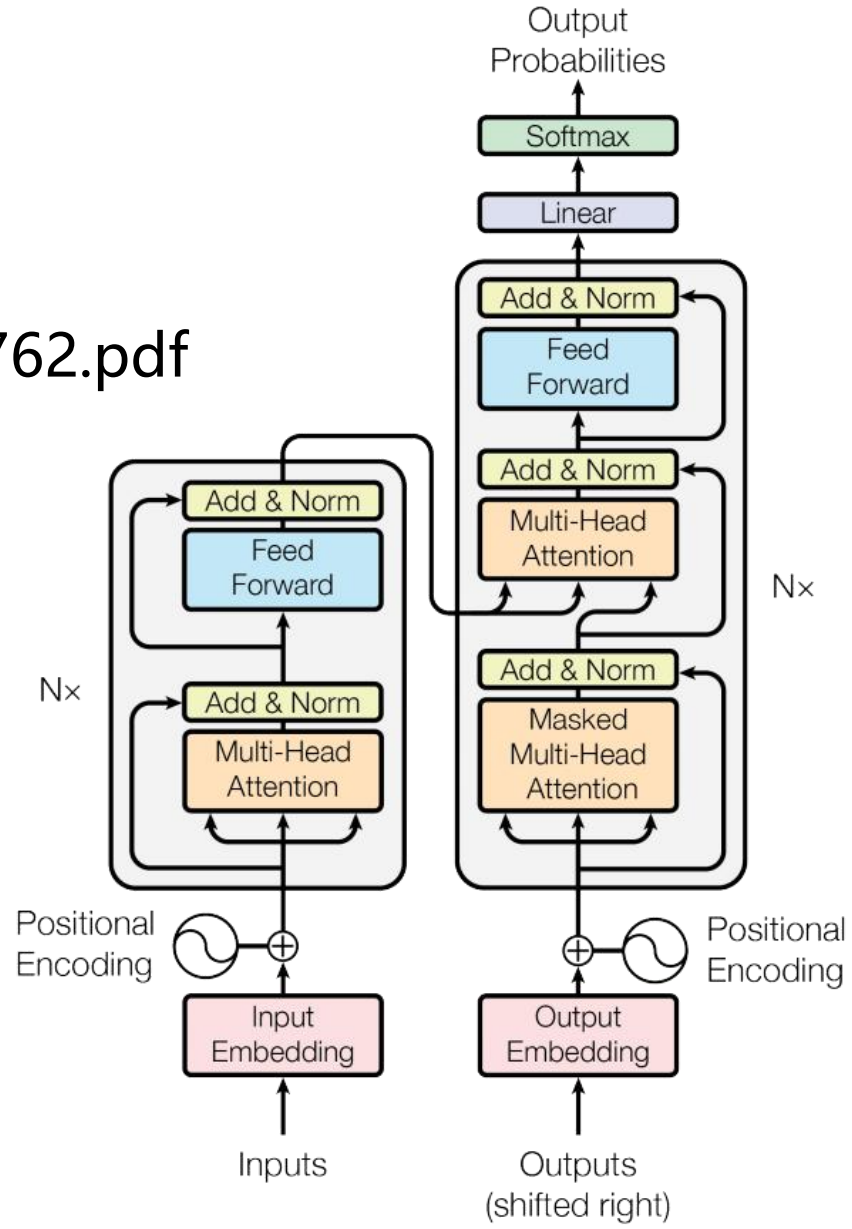
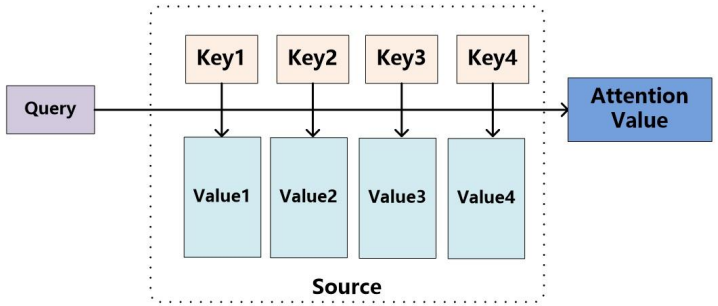
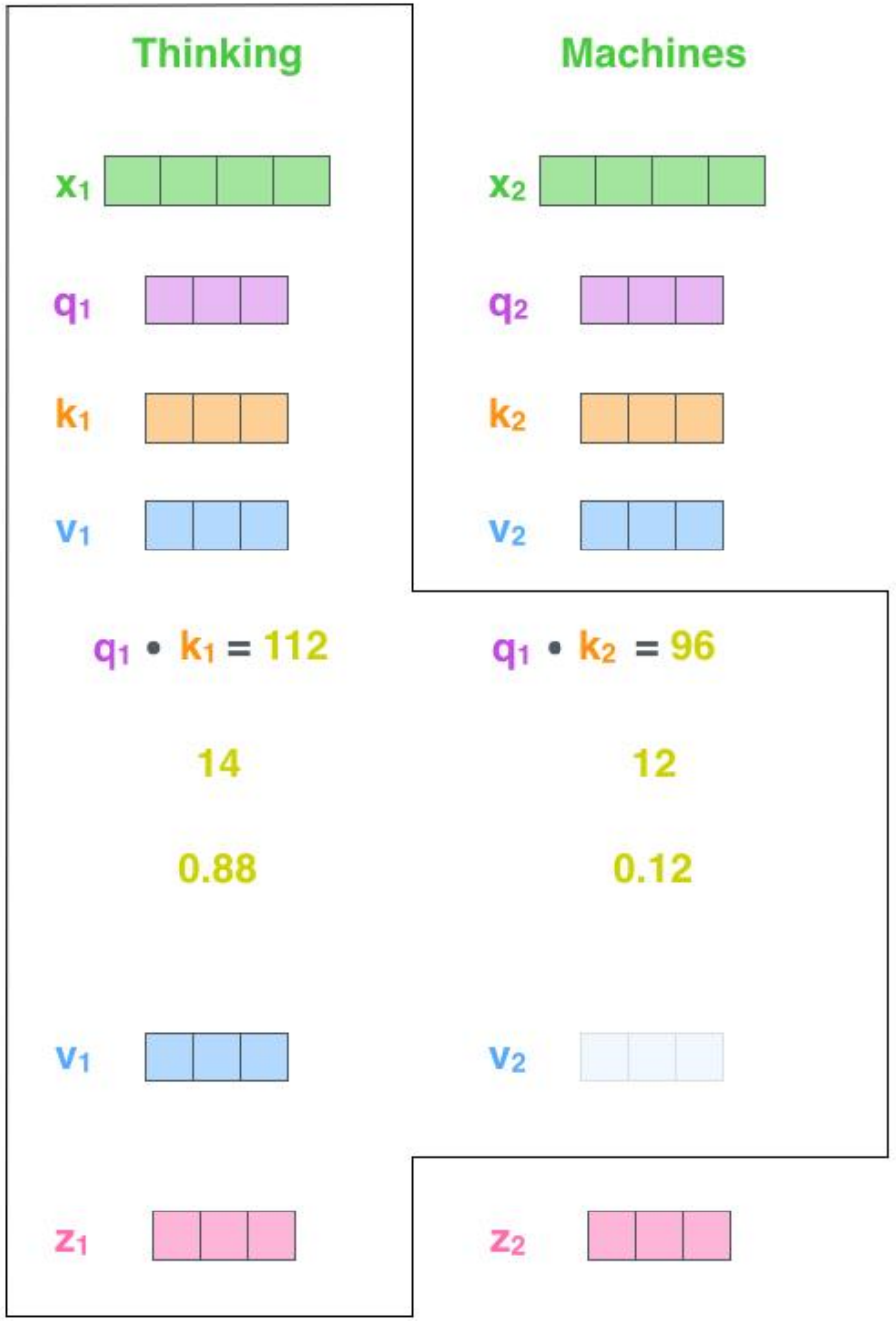


Figure 1: The Transformer - model architecture.

Transformer回顾



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

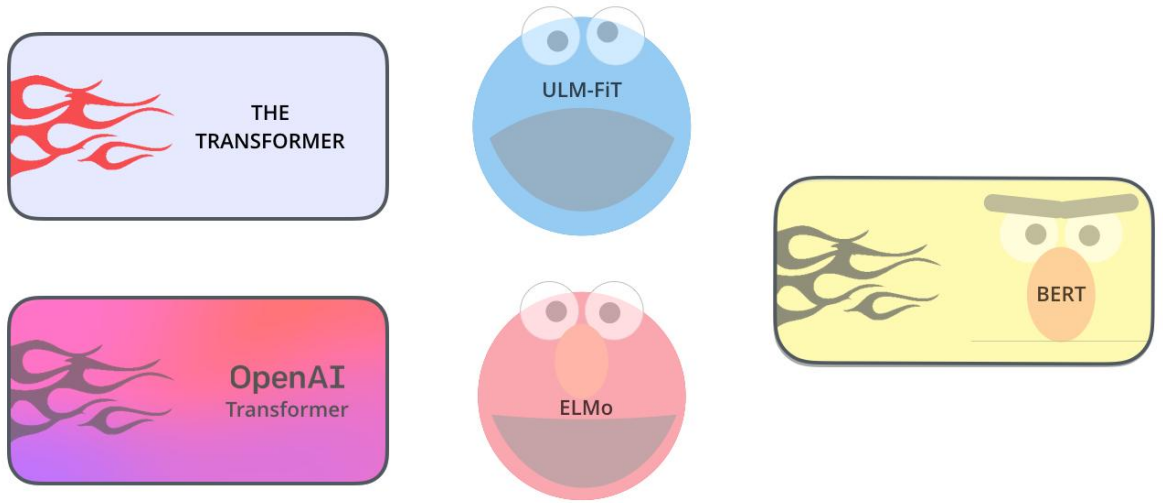


概念普及

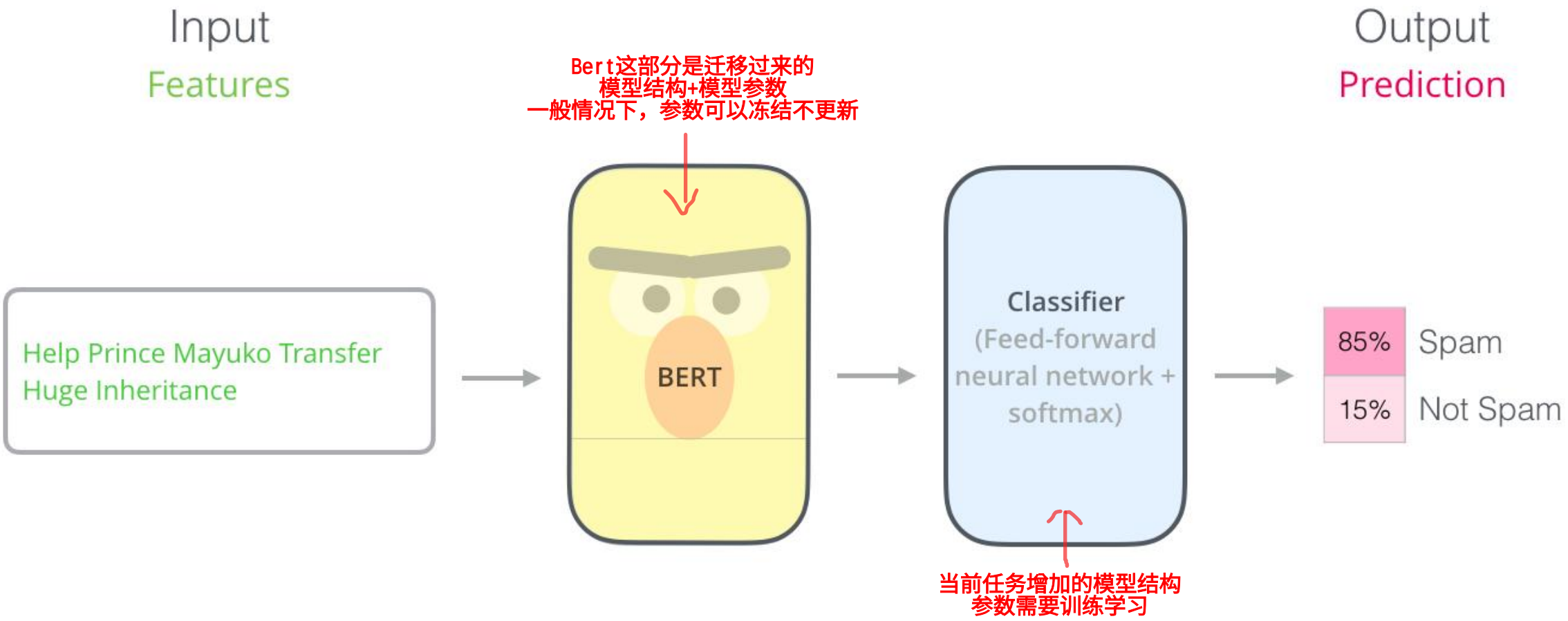
- **语言模型(Language Modeling)**会根据前面单词来预测下一个单词/NLP中最基础的模型结构(可以应用到各个NLP的应用场景中)。常用的语言模型有：N-gram、Word2Vec、ELMo、OpenAI GPT、**Bert**、XLNet、ALBert；
- Mask：遮挡掩盖的意思，比如：把需要预测的词给挡住。主要出现出现在OpenAI GPT<Transformomer>和Bert中。

Bert

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



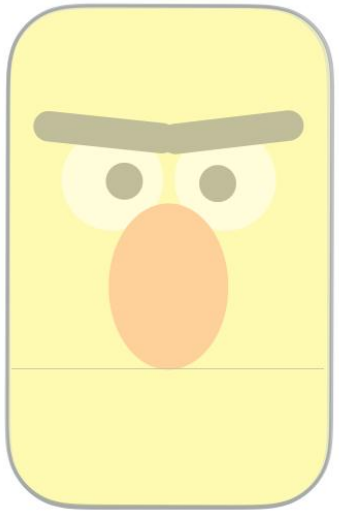
Bert


Input
Features
Bert这部分是迁移过来的
模型结构+模型参数
一般情况下，参数可以冻结不更新
Output
Prediction
Help Prince Mayuko Transfer
Huge Inheritance
BERT
Classifier
(Feed-forward
neural network +
softmax)
85%
Spam
15%
Not Spam
当前任务增加的模型结构
参数需要训练学习

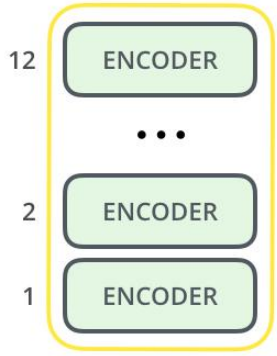
Bert

- **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。

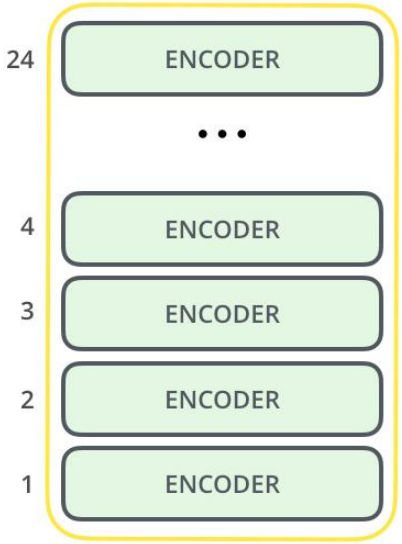
BERT$_{BASE}$



BERT$_{LARGE}$



BERT$_{BASE}$

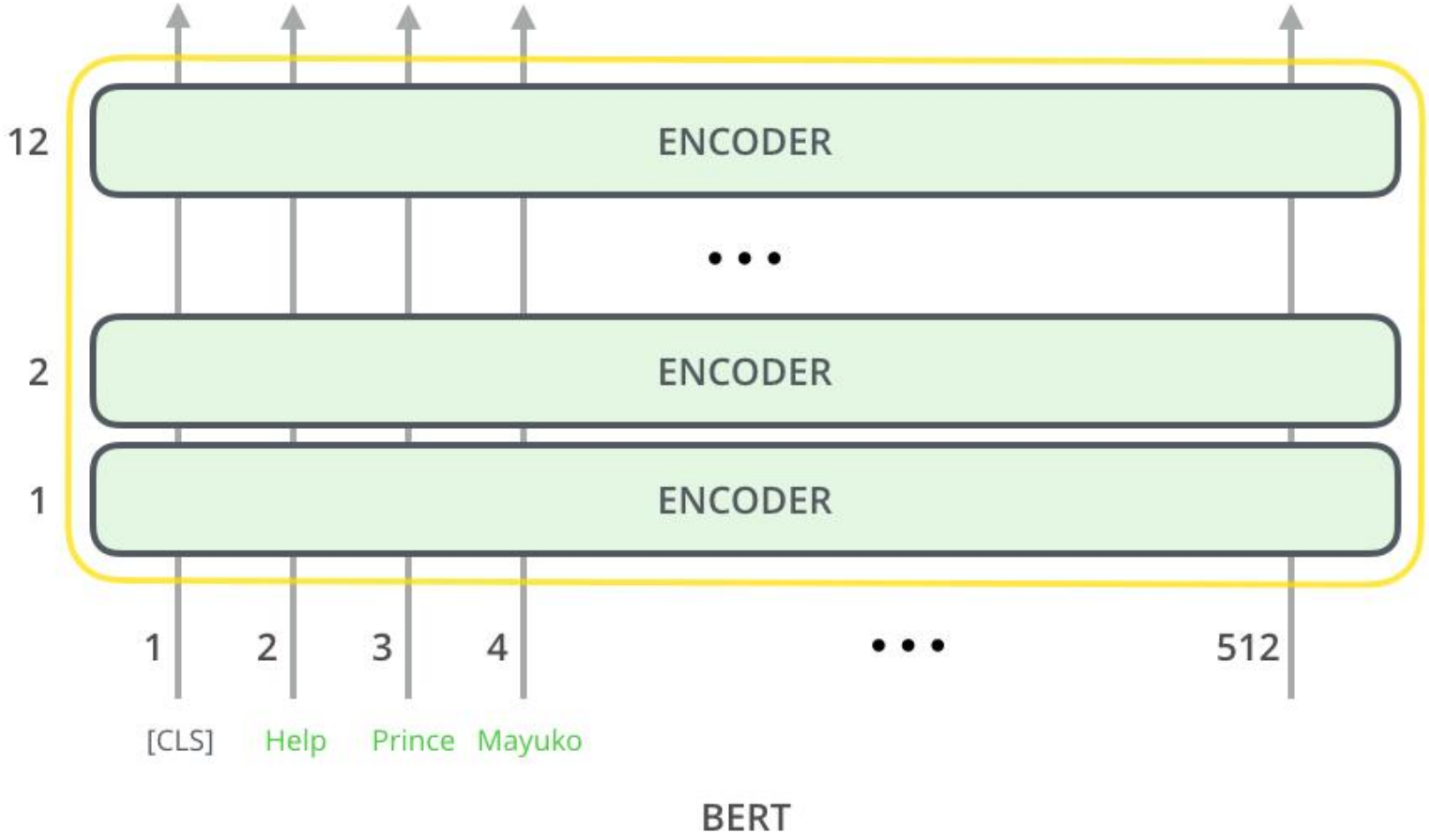


BERT$_{LARGE}$

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增加了几个特殊的token: [CLS]、[SEP]、[MASK]；
  - Bert默认允许输入的token总序列长度最大为512；

Bert
Model Input



Bert
Model Input

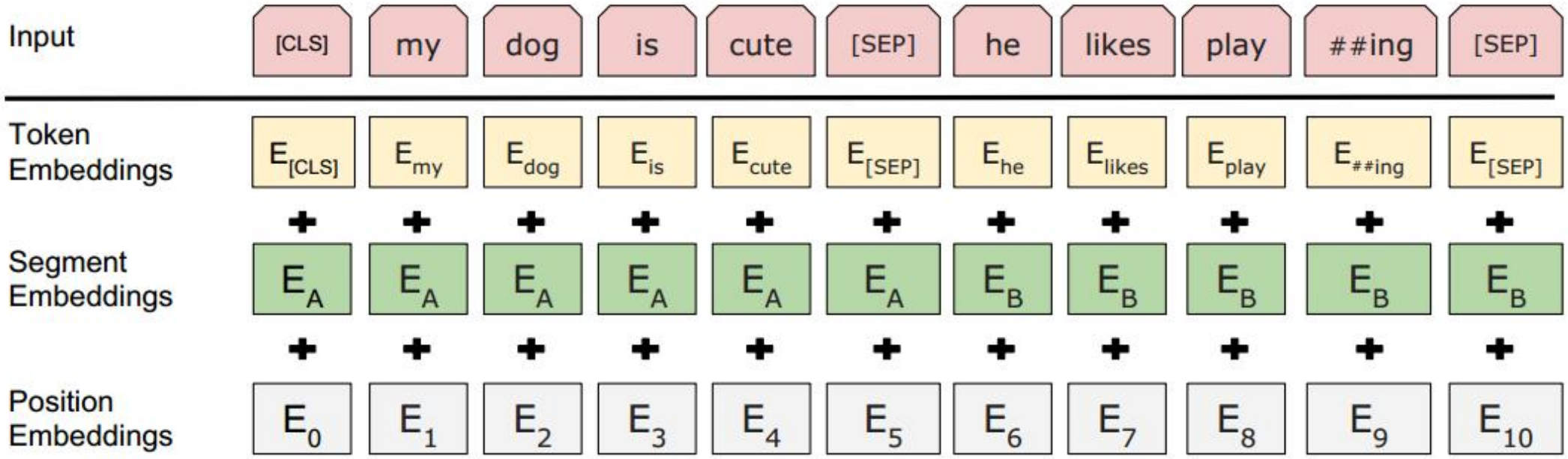
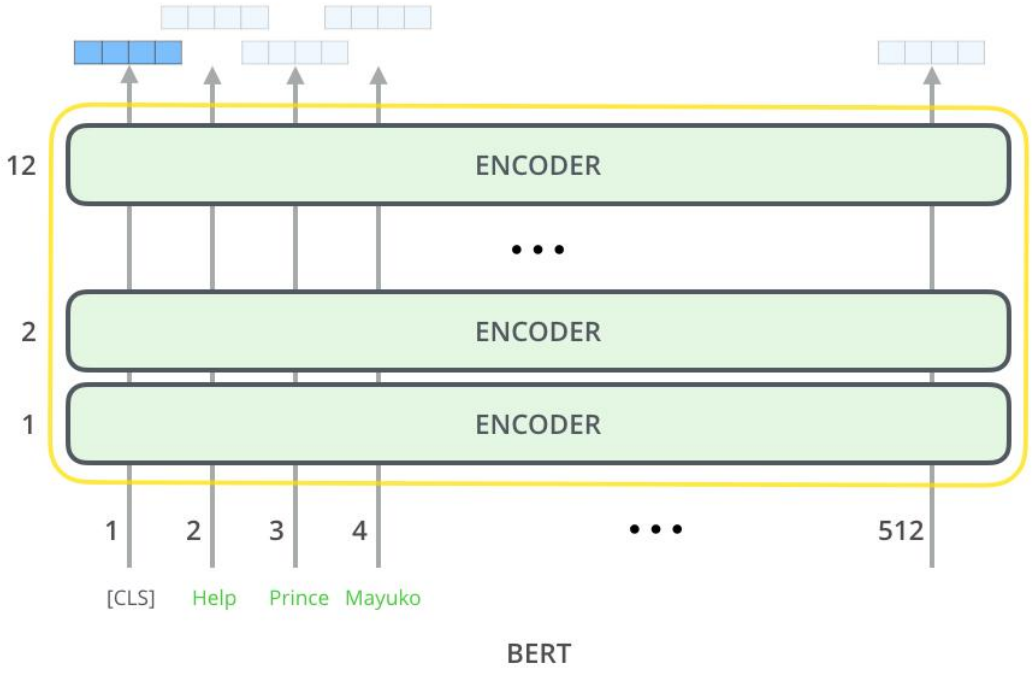
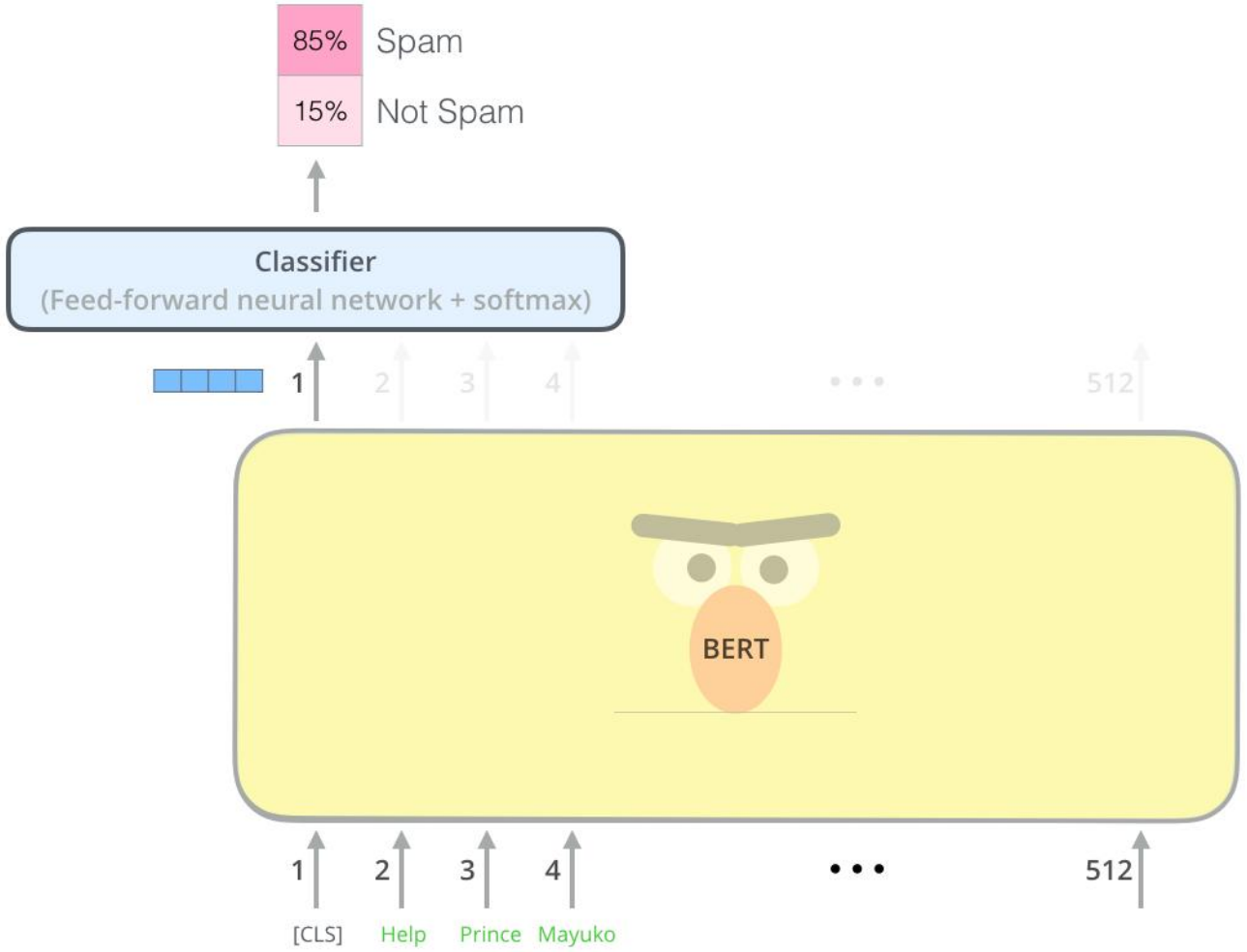


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Not Word Token, Is Word Piece Token(FastText)

Bert
Model Output

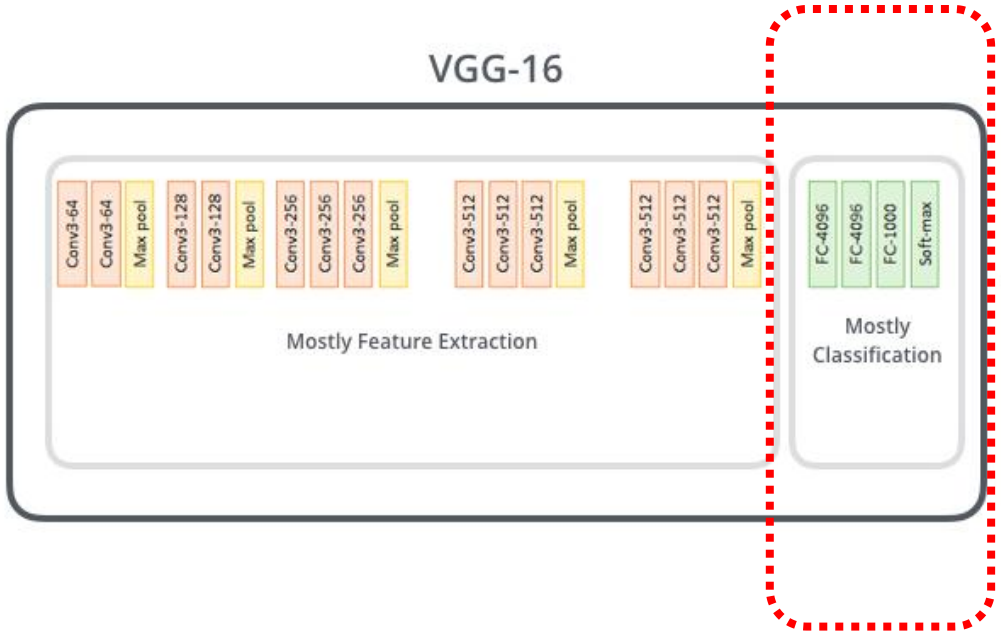




Bert
with VGG

Input

Features



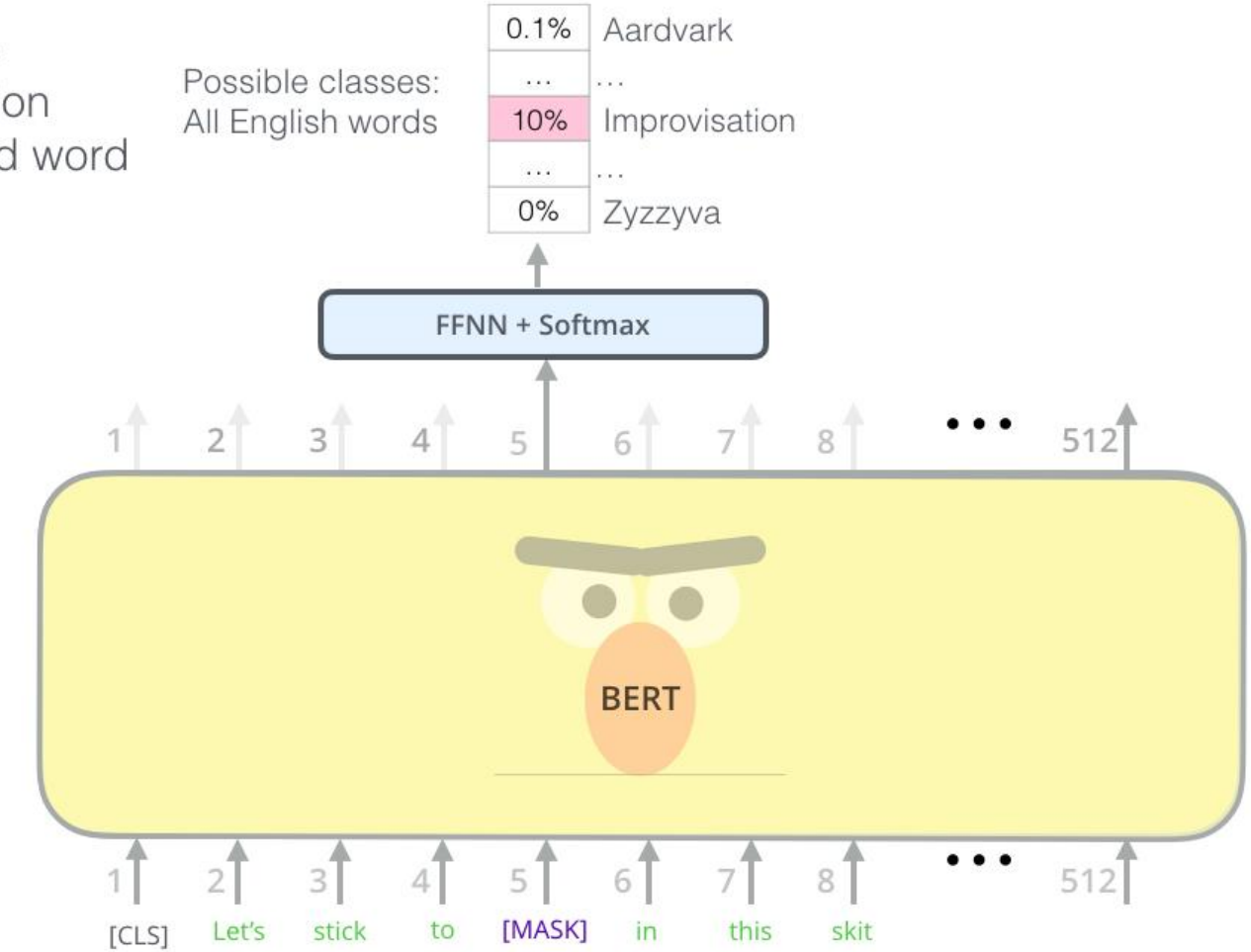
Output

Prediction

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

Bert Pre Train
Masked Language Model

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

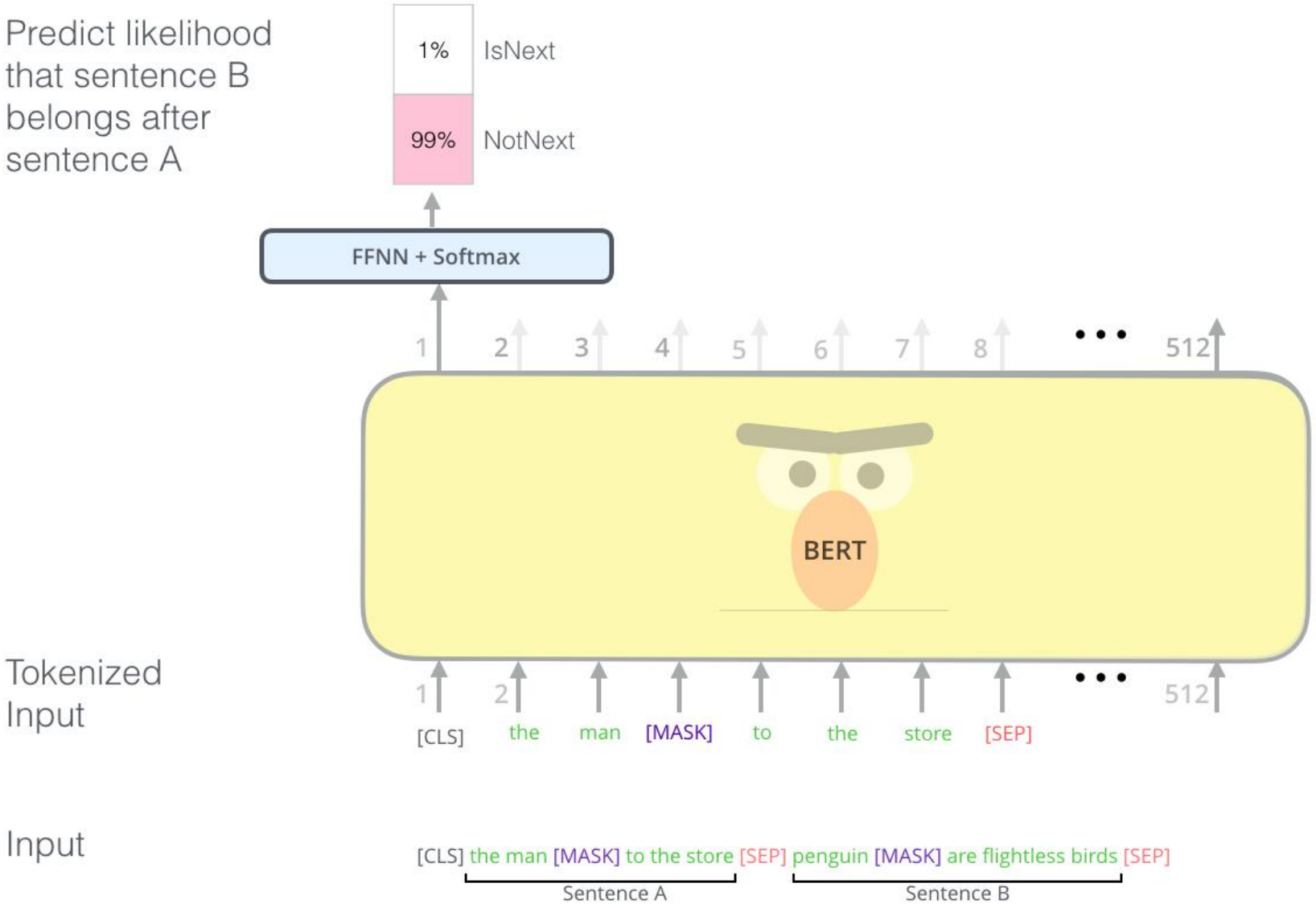


Randomly mask 15% of tokens

Input

Mask + 15%的随机“替换”

Bert Pre Train
Next Sentence Prediction



Bert

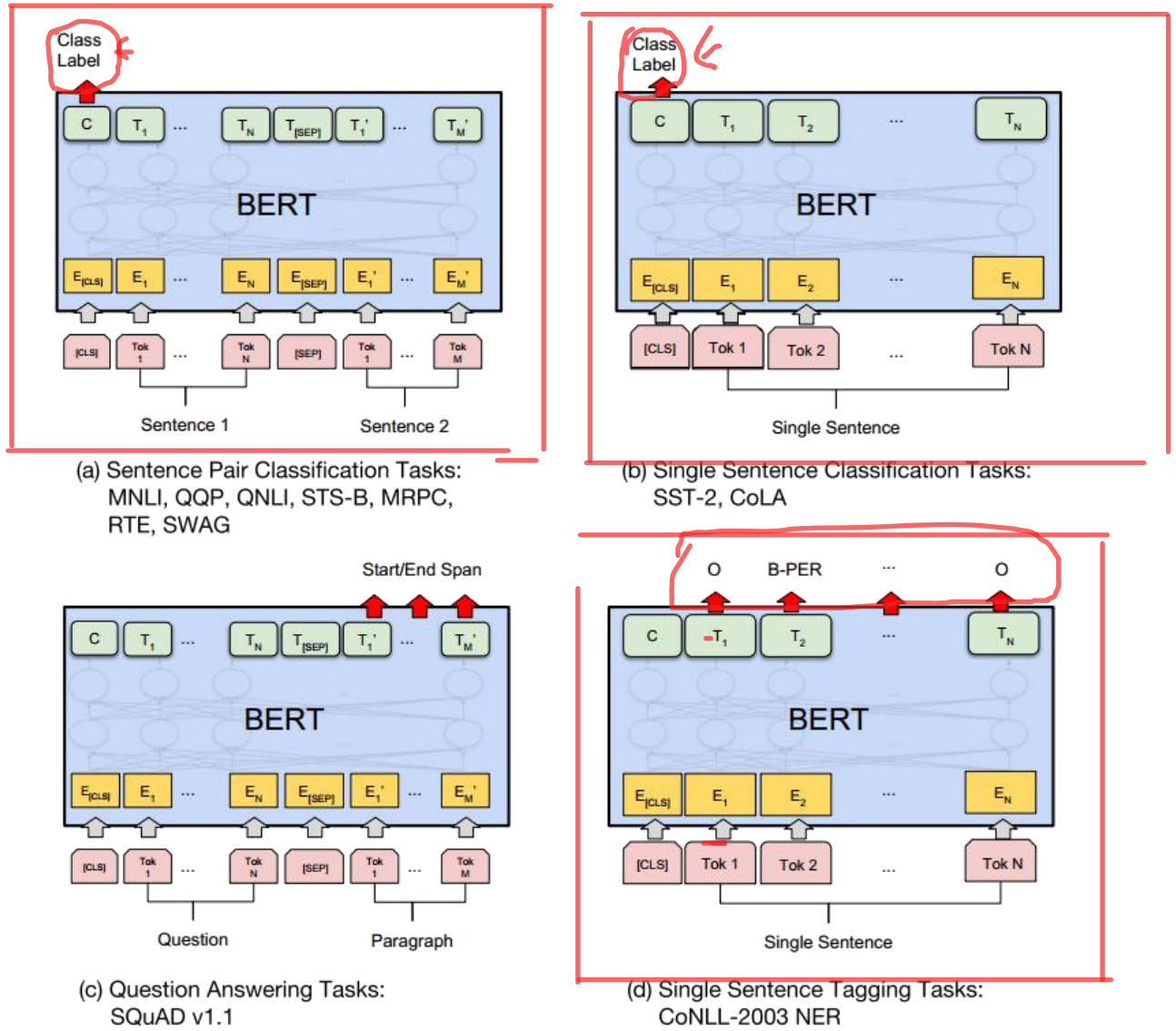
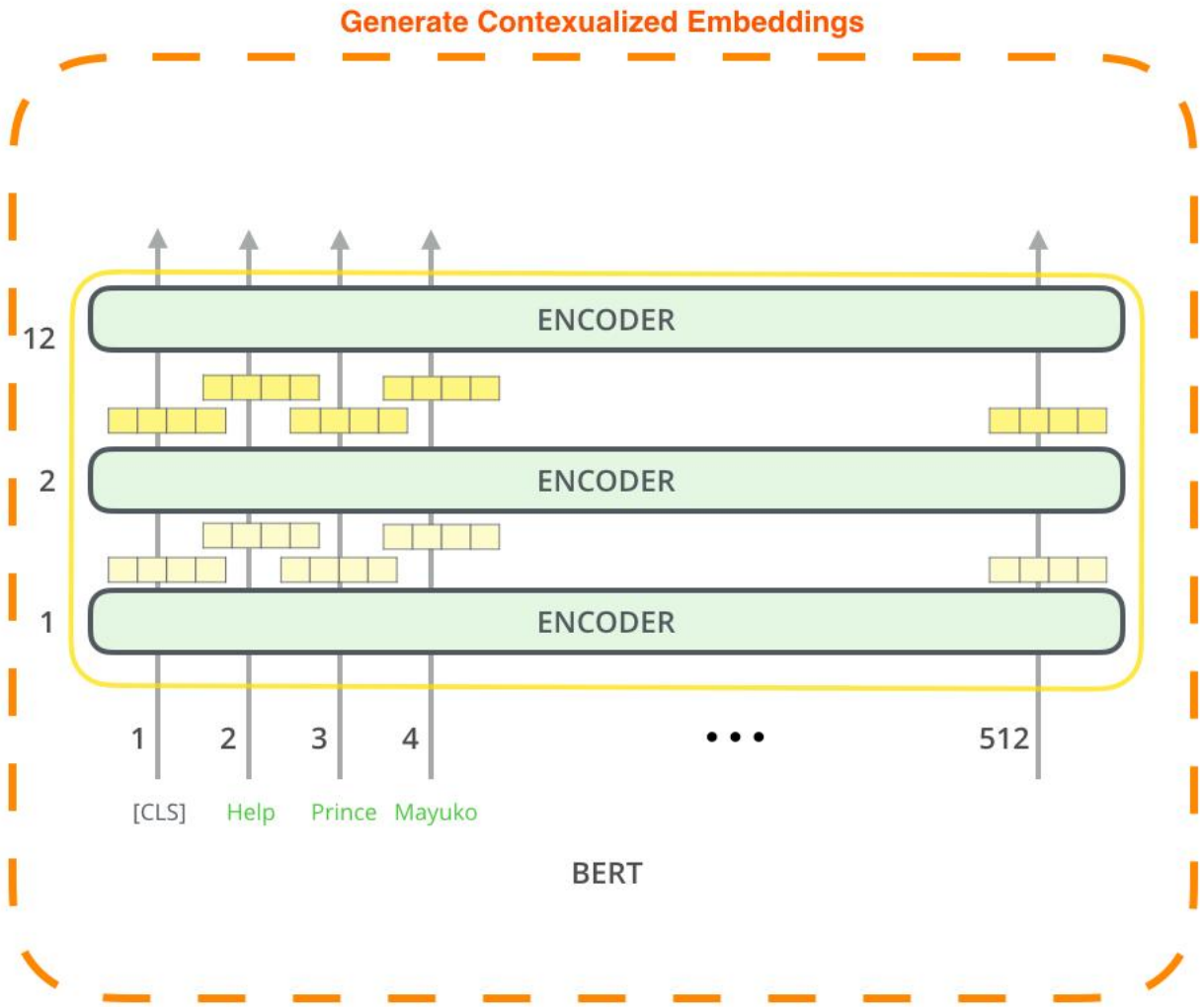
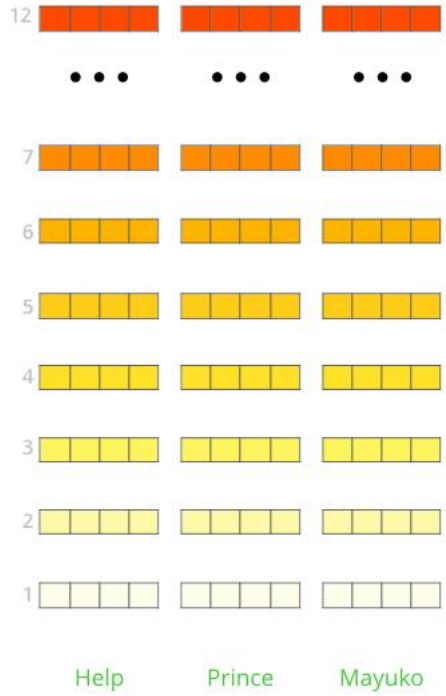


Figure 3: Our task specific models are formed by incorporating BERT with one additional output layer, so a minimal number of parameters need to be learned from scratch. Among the tasks, (a) and (b) are sequence-level tasks while (c) and (d) are token-level tasks. In the figure, E represents the input 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



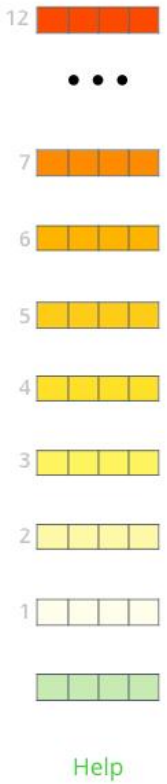
The output of each encoder layer along each token's path can be used as a feature representing that token.

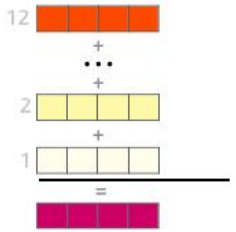
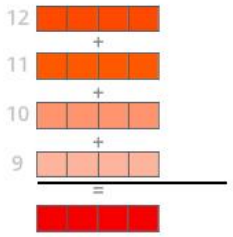


But which one should we use?

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和Transformer(Encoder)区别

- Bert的训练输入有Mask掩码(随机Mask)，Transformer没有；
- Bert的网络结构更加复杂一些(Layer:12/24, Head:12/16)；
- Bert的网络输入存在占位符[CLS], Transformer没有；
- Bert的输入Embedding会比Transformer多一个segment信息；

Bert、GPT、ELMo

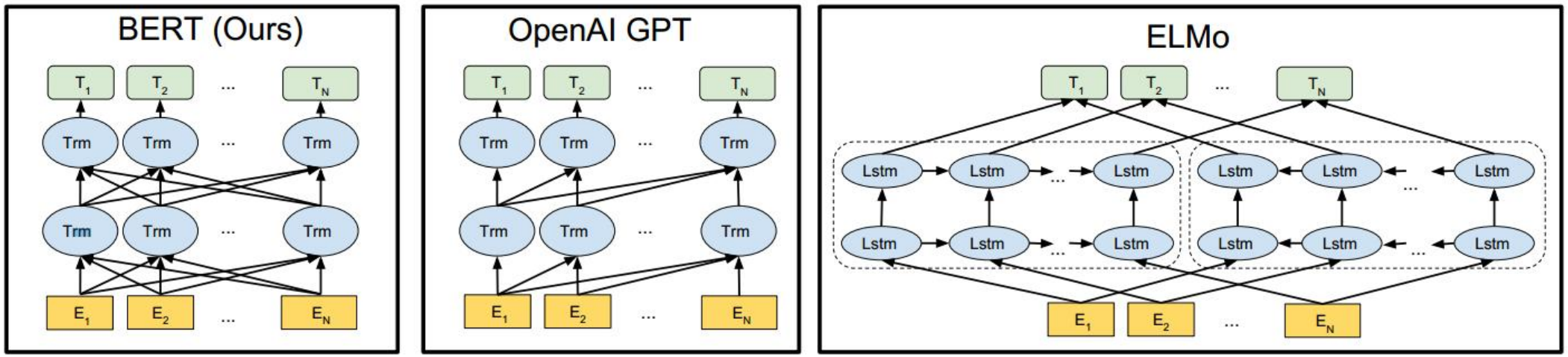
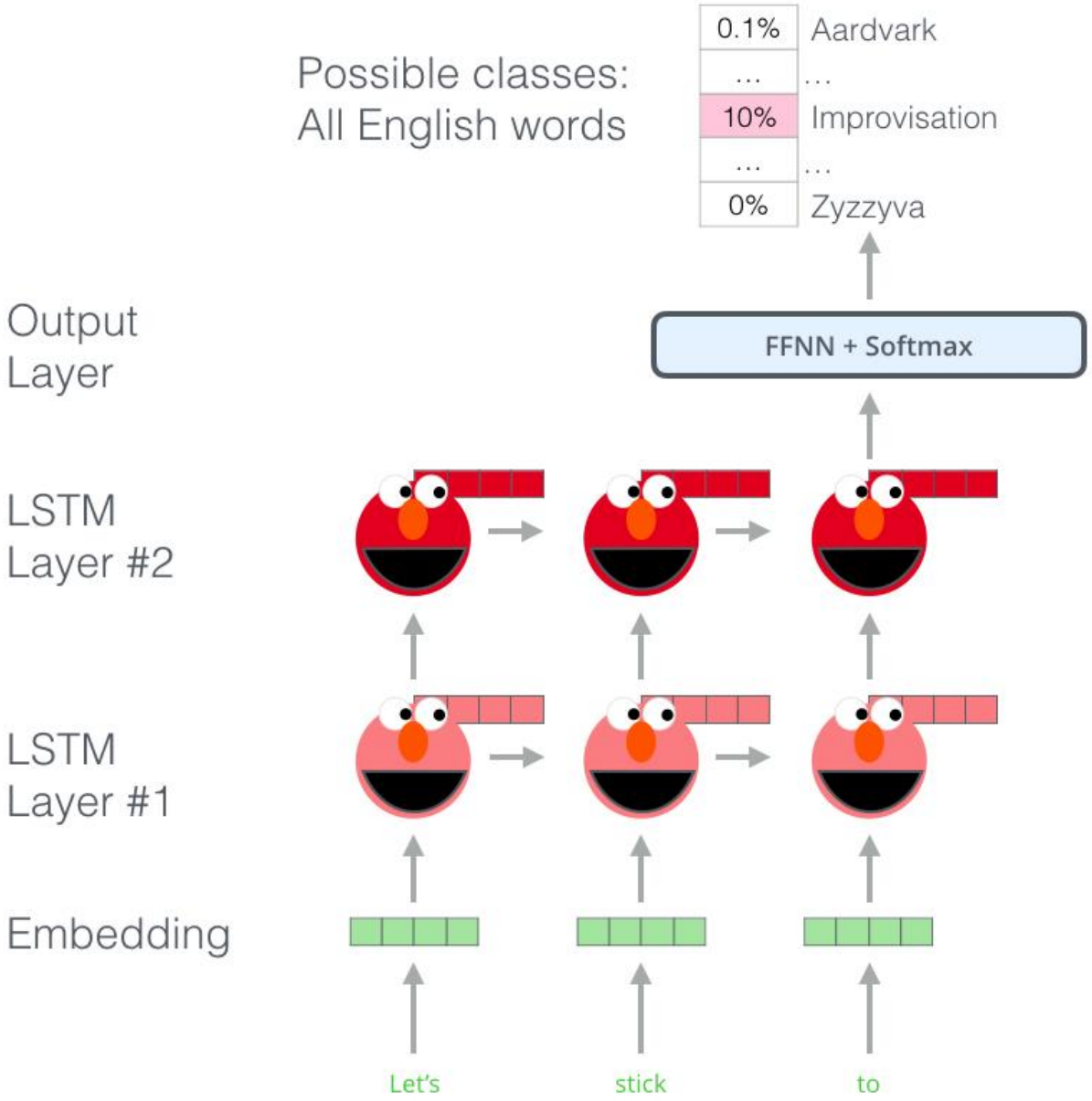


Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT 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.

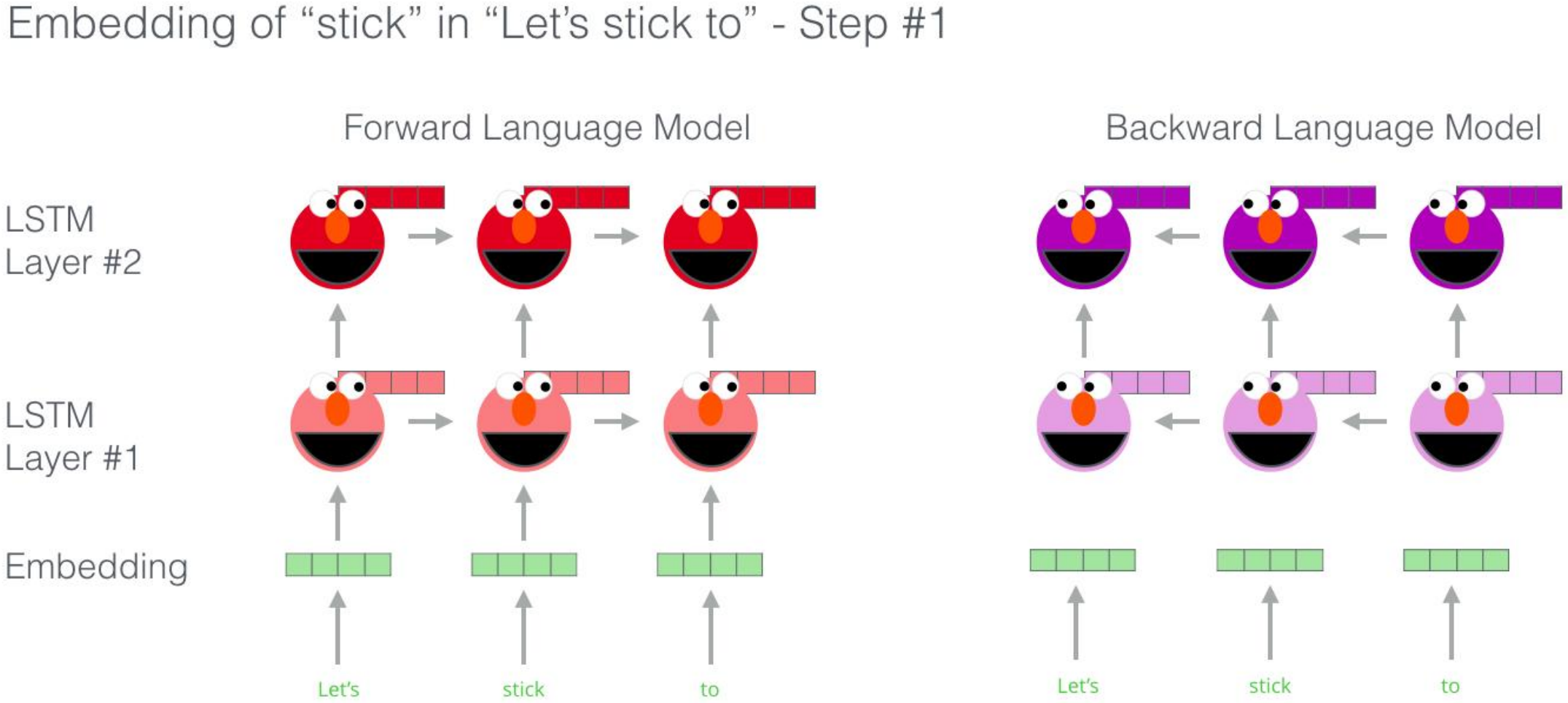
扩展：Bert
with ELMo

- 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**

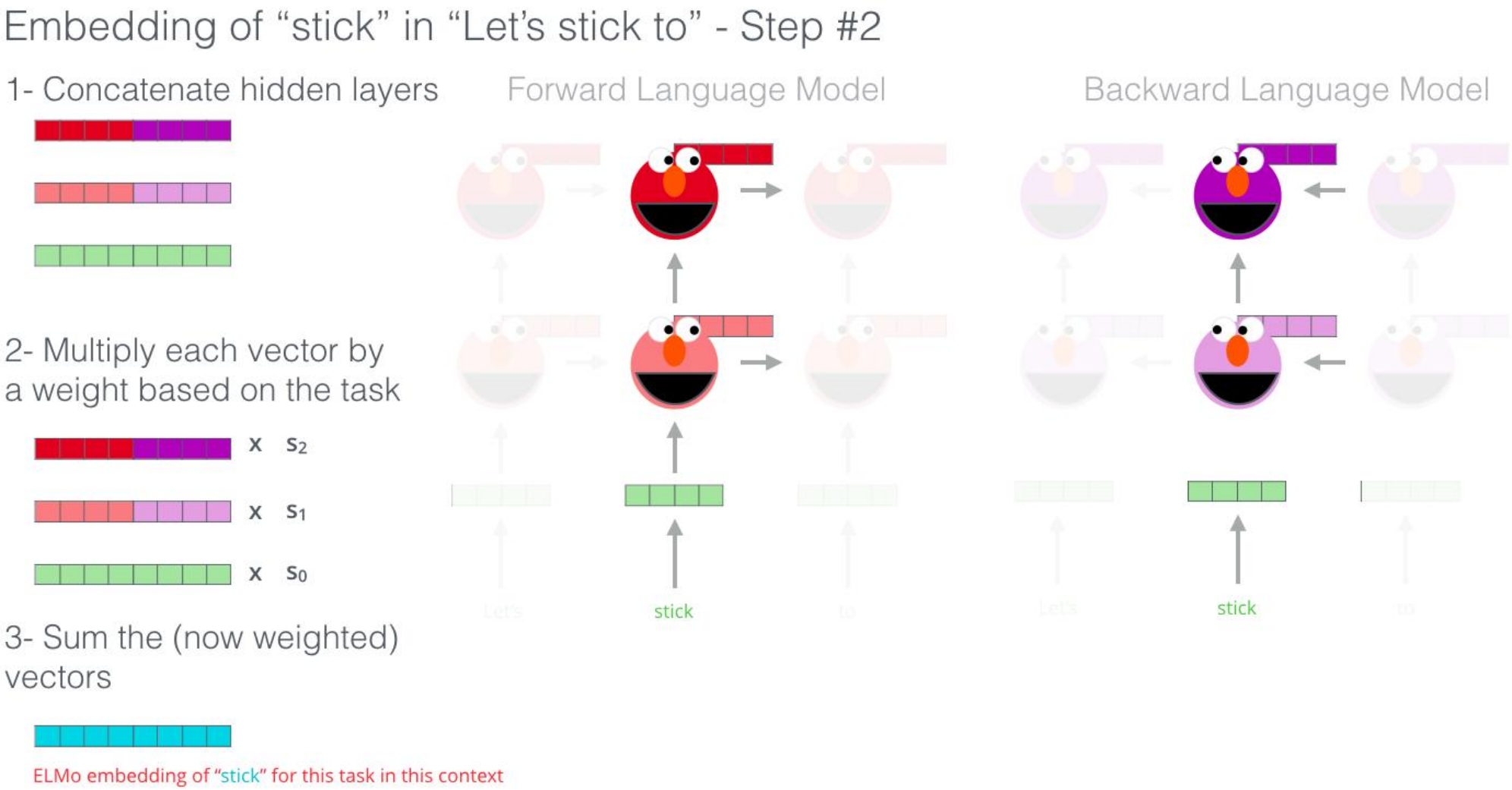
扩展：Bert
with ELMo



扩展：Bert
with ELMo



扩展：Bert
with ELMo



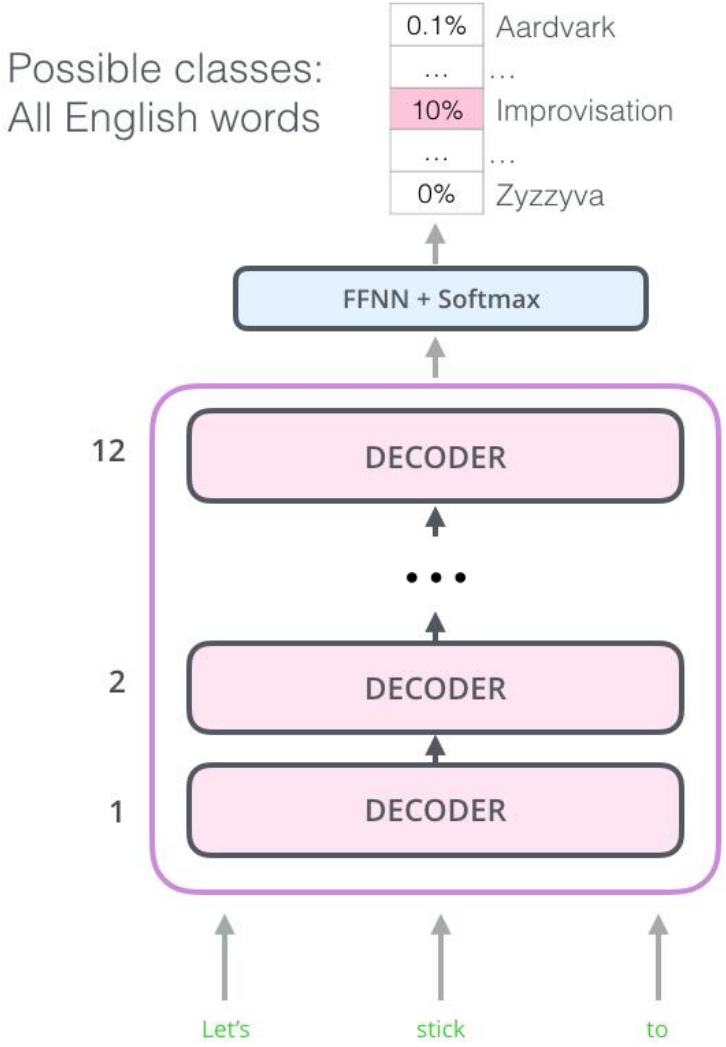
扩展：Bert
With OpenAI

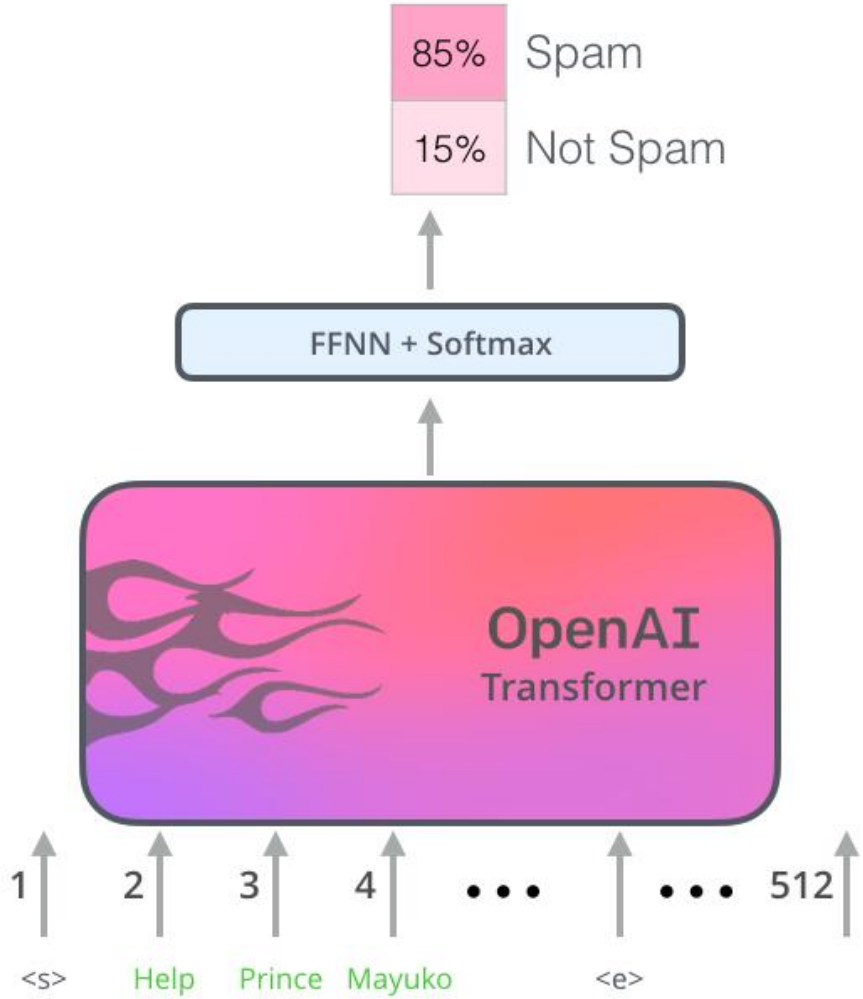
- OpenAI：
  - 2015，致力于人工智能模型预训练领域的相关技术的开发，主要是对于NLP相关人工智能技术的研发。
  - https://www.openai.com/

扩展：Bert
With OpenAI GPT

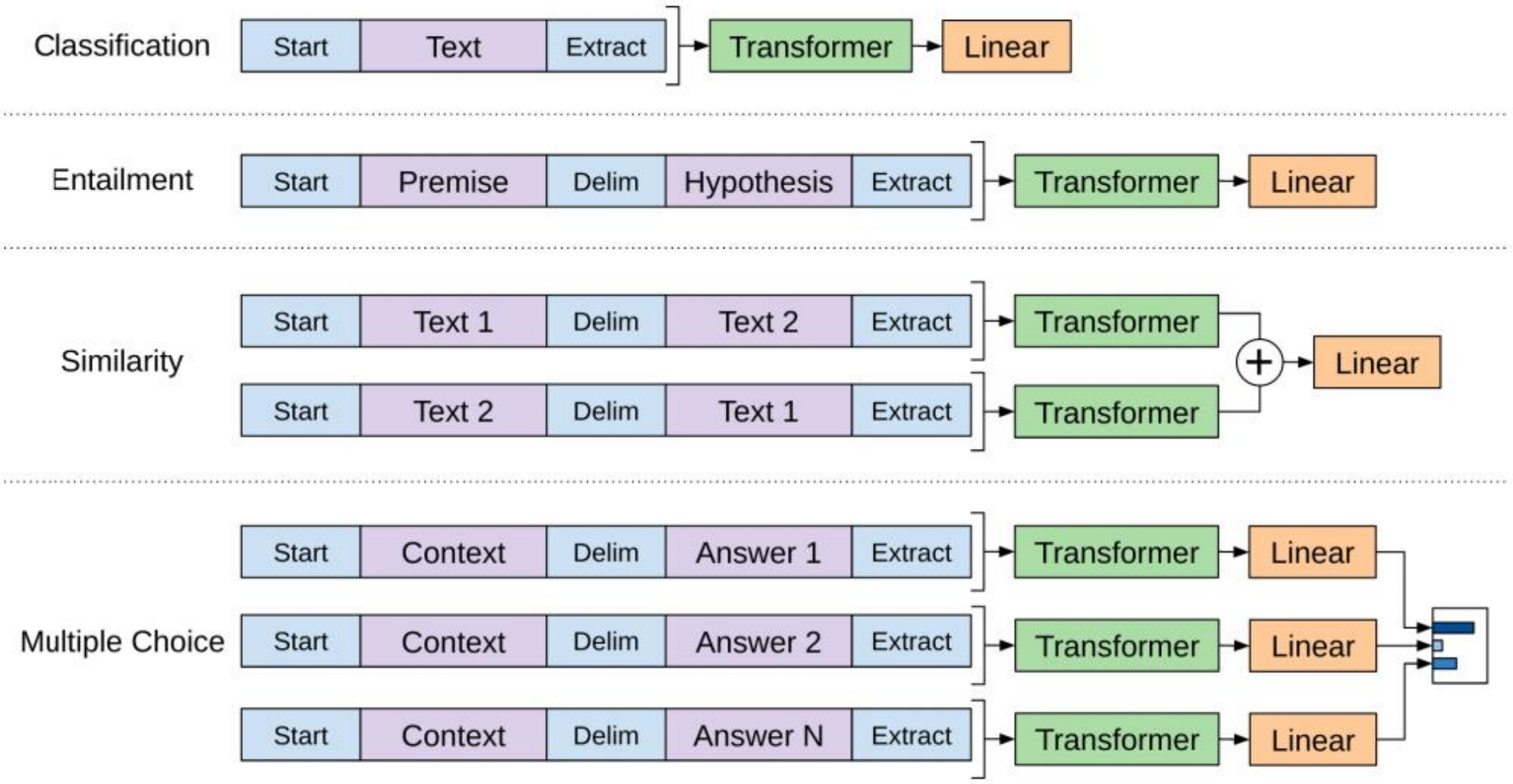
- GPT在BooksCorpus(800M单词)训练；BERT在BooksCorpus(800M单词)和维基百科(2,500M单词)训练。
- GPT使用一种句子分隔符([SEP])和分类符词块([CLS])，它们仅在微调时引入；BERT在预训练期间学习[SEP]，[CLS]和句子A/B嵌入。
- GPT用一个批量32,000单词训练1M步；BERT用一个批量128,000单词训练1M步。
- GPT对所有微调实验使用的5e-5相同学习率；BERT选择特定于任务的微调学习率，在开发集表现最佳。
- GPT是12层，Bert是24层。
- **GTP使用的是Transformer的类似Decoder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





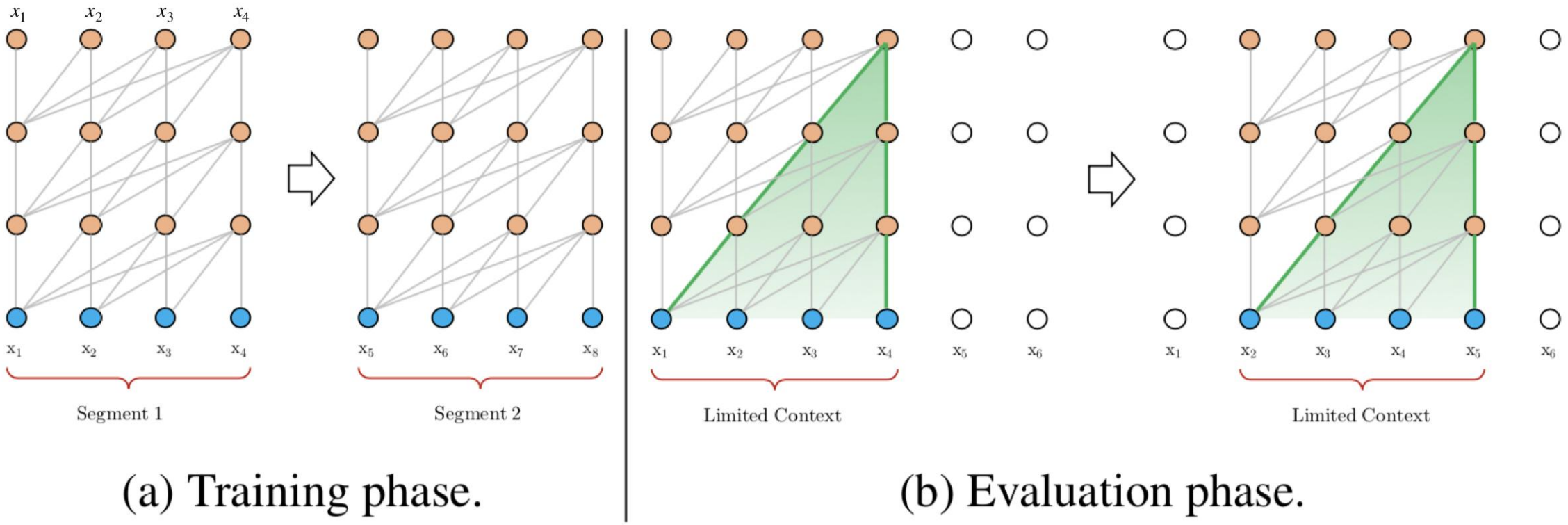
扩展：Bert
With OpenAI



扩展：vanilla transformer

- 属于一种语言模型的训练方式，来根据之前的字符预测片段中的下一个字符；论文中采用64层模型，并仅限于处理512个字符的输入，因此如果序列太长会进行分段，然后每段进行学习训练。
  论文：https://arxiv.org/pdf/1808.04444.pdf
- 缺陷：
  - 上下文长度受限：字符之间的最大依赖距离受输入长度的限制，模型看不到出现在几个句子之前的单词。
  - 上下文碎片：对于长度超过512个字符的文本，都是从头开始单独训练的。段与段之间没有上下文依赖性，会让训练效率低下，也会影响模型的性能。
  - 推理速度慢：在测试阶段，每次预测下一个单词，都需要重新构建一遍上下文，并从头开始计算，这样的计算速度非常慢。

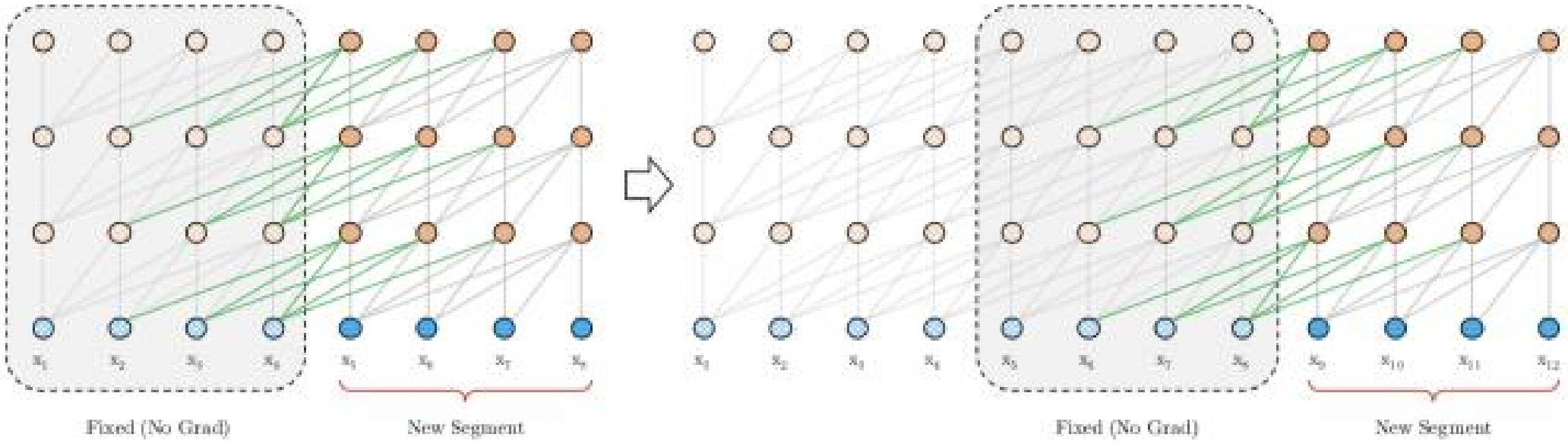
扩展：vanilla transformer



扩展：Transfomer-XL

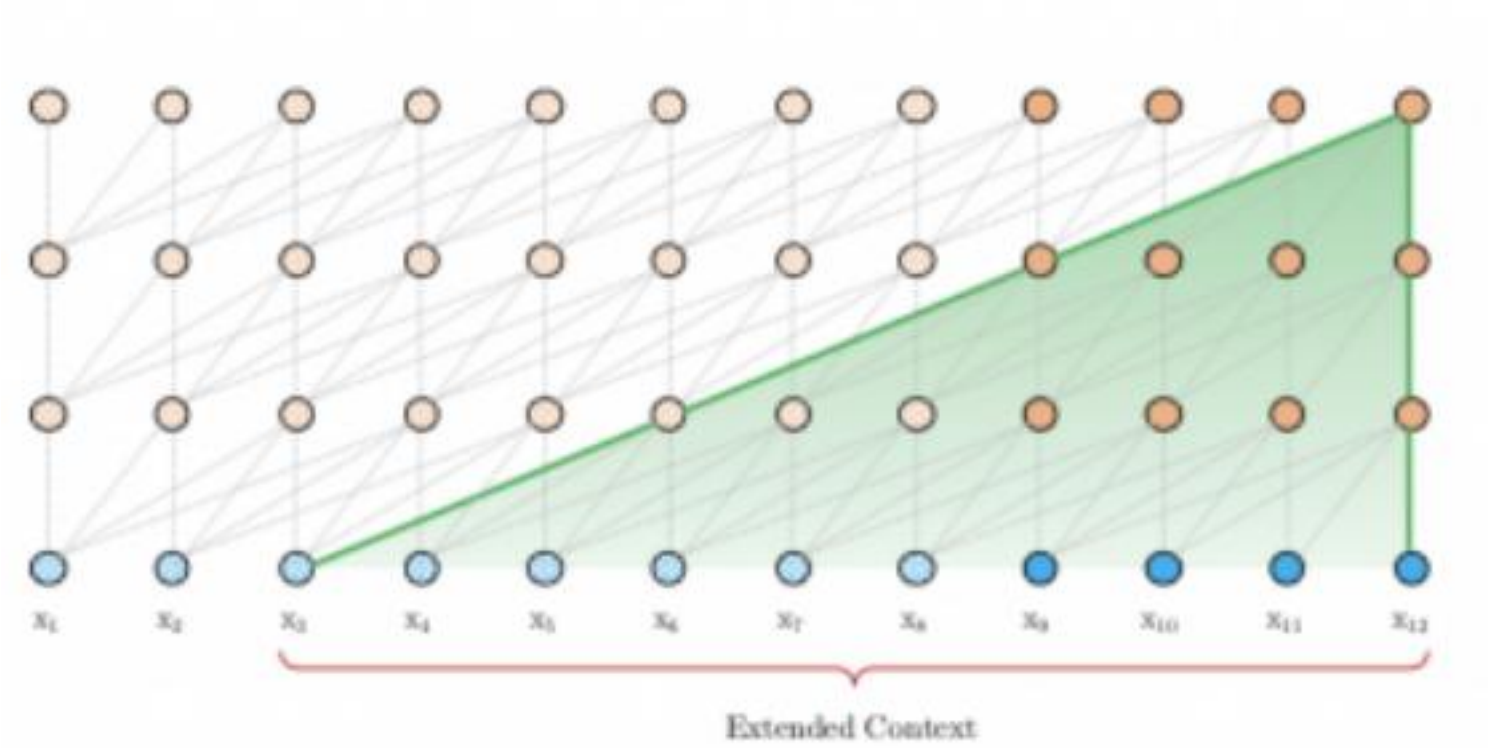
- RNN和Transformer都可以学习序列之间的依赖关系，但是对于长时依赖上都存在一定的局限性，在vanilla Transformer的基础上，Transfomer-XL语言模型引入两点创新：
  - **循环机制(Recurrence Mechanism)**
  - **相对位置编码(Relative Positional Encoding)**
- 论文：https://arxiv.org/pdf/1901.02860.pdf
- 代码：https://arxiv.org/pdf/1901.02860.pdf
- 解析：https://zhuanlan.zhihu.com/p/271984518

扩展：Transfomer-XL



(a) Training phase.

扩展：Transfomer-XL



(b) Evaluation phase.

扩展：XLNet

- XLNet：Generalized Autoregressive Pretraining for Language Understanding
  - https://arxiv.org/pdf/1906.08237.pdf
  - 解决BERT的问题：
    - 训练数据和测试数据的不一致，训练数据中使用了Mask，测试数据/预测数据中没有使用Mask，这个问题叫做：pretrain-finetune discrepancy
    - BERT模型不能用来生成数据。
  - 参考：https://www.zhihu.com/tardis/bd/art/649916898?source_id=1001

扩展：ALBERT

- ALBERT：A Lite BERT for Self-supervised Learning of Language Representations
  - https://arxiv.org/pdf/1909.11942.pdf
  - 解决Bert和XLNet的问题：
    - **模型参数变的更少；**
    - **模型使用更少的内存；**
    - 提升模型效果；
    - 参考: https://github.com/brightmart/albert_zh

THANKS!