

A Brief Intro to BERT

文本表征模型

- Bag of words: one-hot、TF-IDF、Text Rank等;
- 主题模型: LSA (SVD) 、 pLSA、 LDA
- 词嵌入静态表征: word2vec、 GloVe
- 词嵌入动态表征: ELMo、 GPT、 BERT

动态文本表征模型的演变



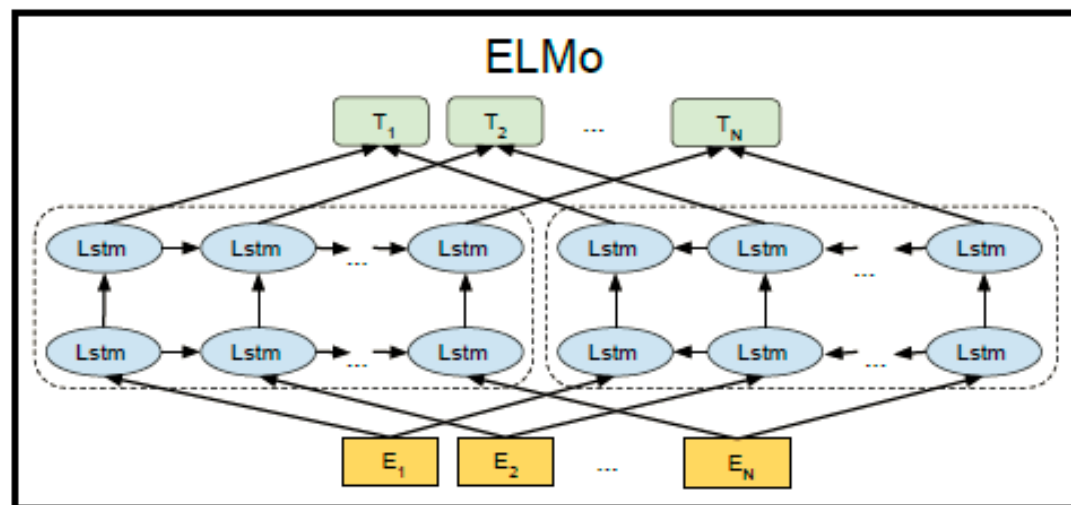
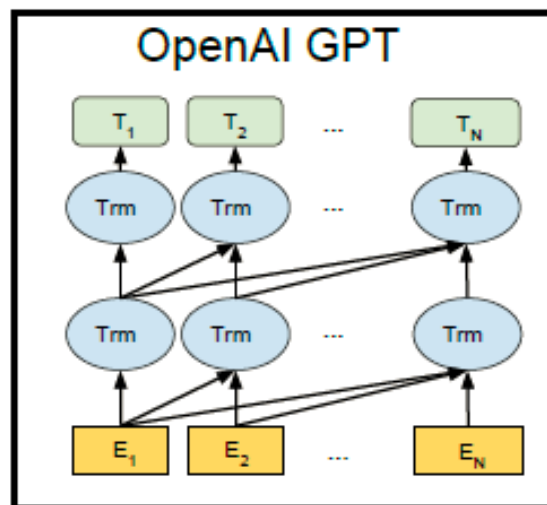
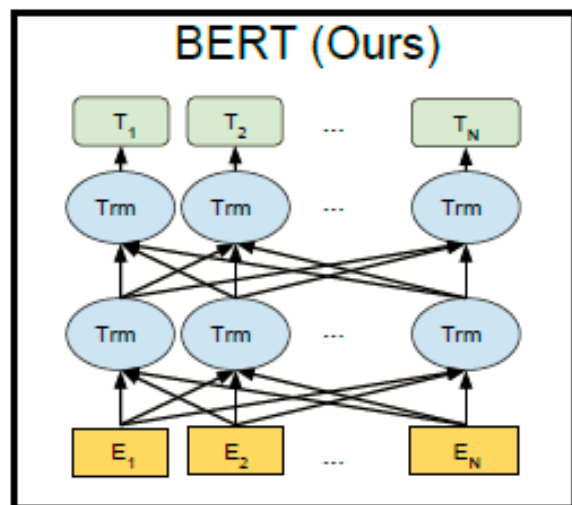
BERT: Bidirectional
Encoder Representations
from Transformers.



截止2018年10月, 刷新11项NLP任务



BERT模型结构



多层双向Transformer , 多层单向Transformer

LM目标函数 :

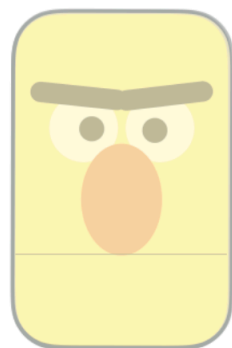
$$P(w_i | w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_n)$$

双层LSTM , 独立双向语义拼接 ,

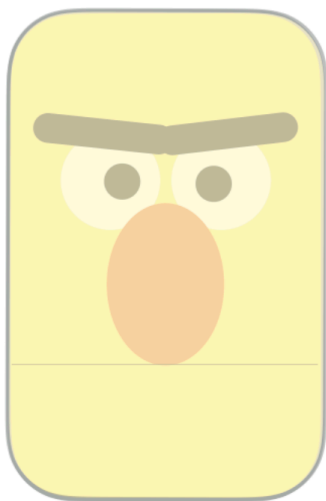
LM目标函数 :

$$P(w_i | w_1, \dots, w_{i-1}) \text{ 和 } P(w_i | w_{i+1}, \dots, w_n)$$

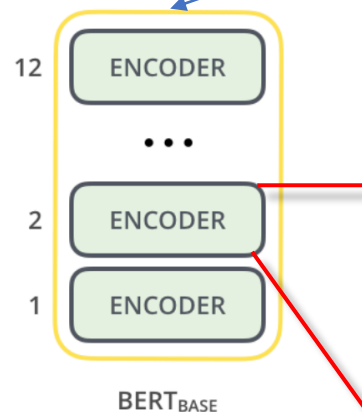
BERT模型结构



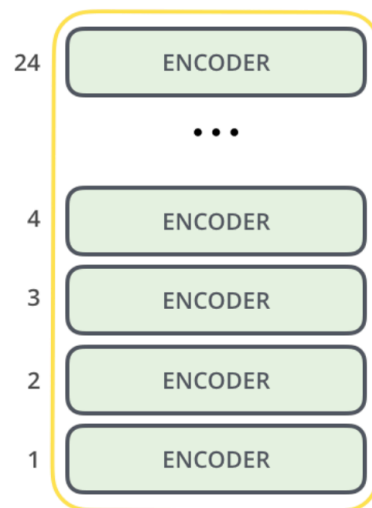
BERT_{BASE}



BERT_{LARGE}



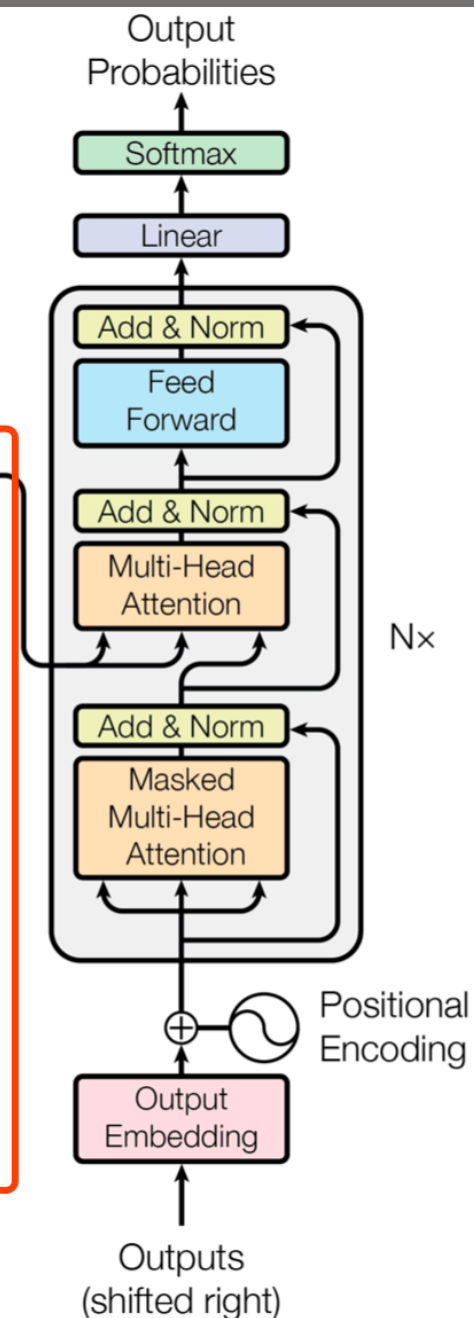
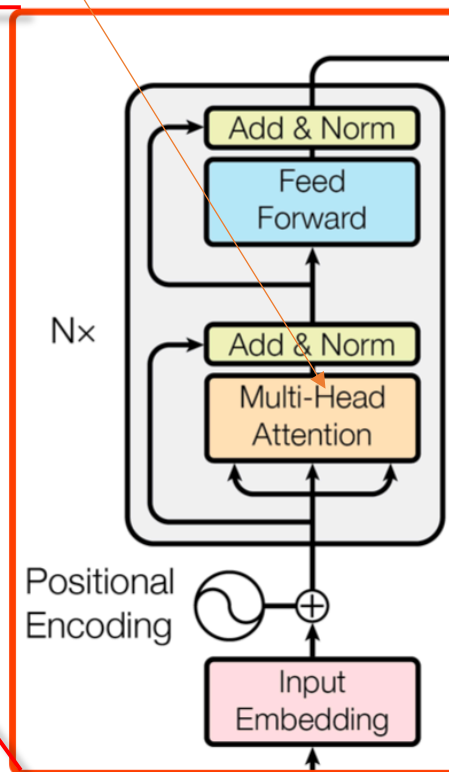
BERT_{BASE}



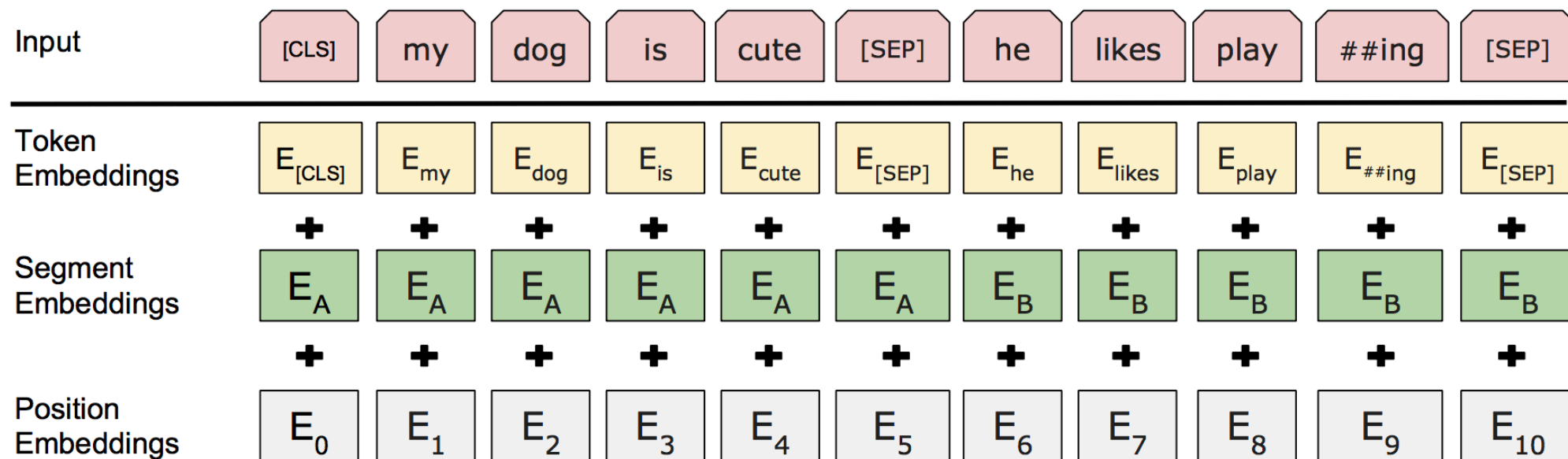
BERT_{LARGE}

$L = 12 / 24, H = 768 / 1024,$

$A = 12 / 16$



BERT的inputs

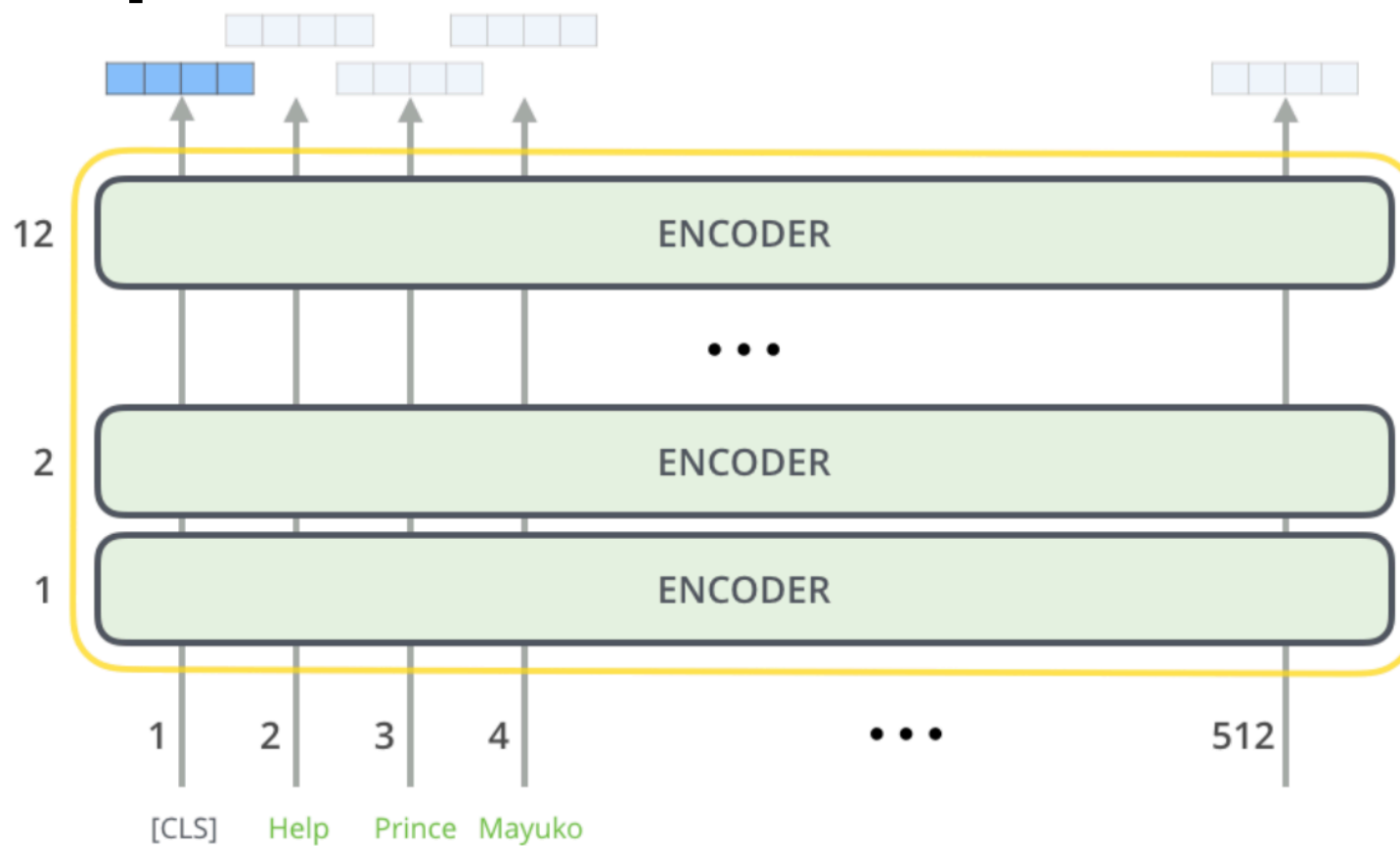


Token：第一个单词用 [CLS] 标识出来，该标识的output可用于分类任务；

Segment：分句编码，用于区分两个句子，预训练的“下一句预测”子任务用到；

Position：位置编码，但不是使用三角函数计算而是在预训练中学习得到的。

BERT的outputs



每个位置输出一个长度为H (768 / 1024) 的向量

BERT的预训练 (Fine-tuning)

Task #1 Masked Language Model

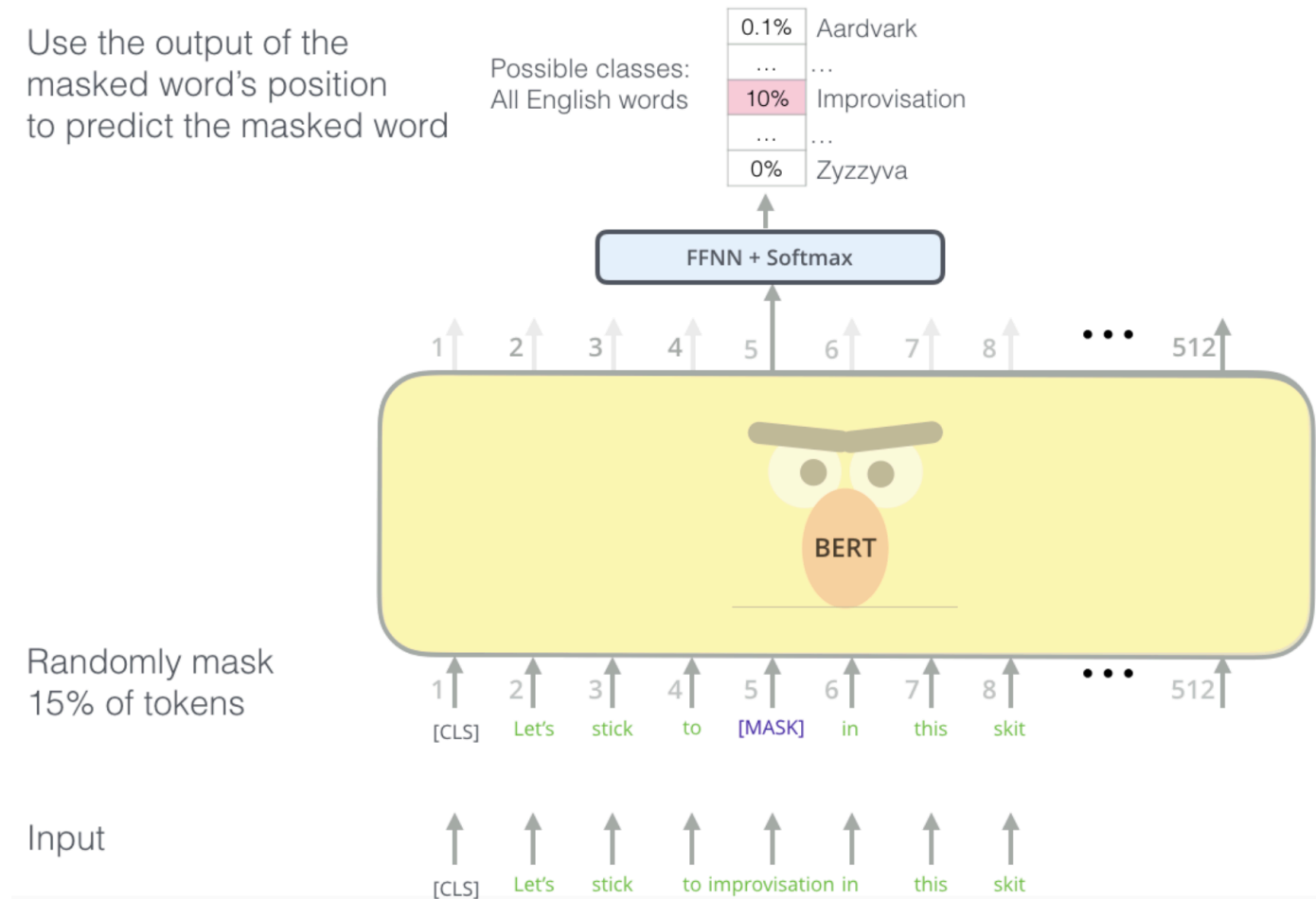
- 采取mask技巧的原因

- (1) 为了捕捉更深层的上下文语义依赖；
- (2) 双向Transformer Encoder网络中，每个词可以**间接**地在多层上下文中**看到自身**，这在实际预测中是不允许的。

- Mask的具体操作

- (1) 随机遮挡15%的词语（80%: “[MASK]”；10%: 保持不变; 10%: 随机替换成另一个词）
- (2) 与CBOW不同，最终损失函数只**计算**被masked掉的token。（但预训练时每个词都要关注，因为不知道哪些词是被masked的）

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



BERT的预训练 (Fine-tuning)

Task #2 Next Sentence Prediction

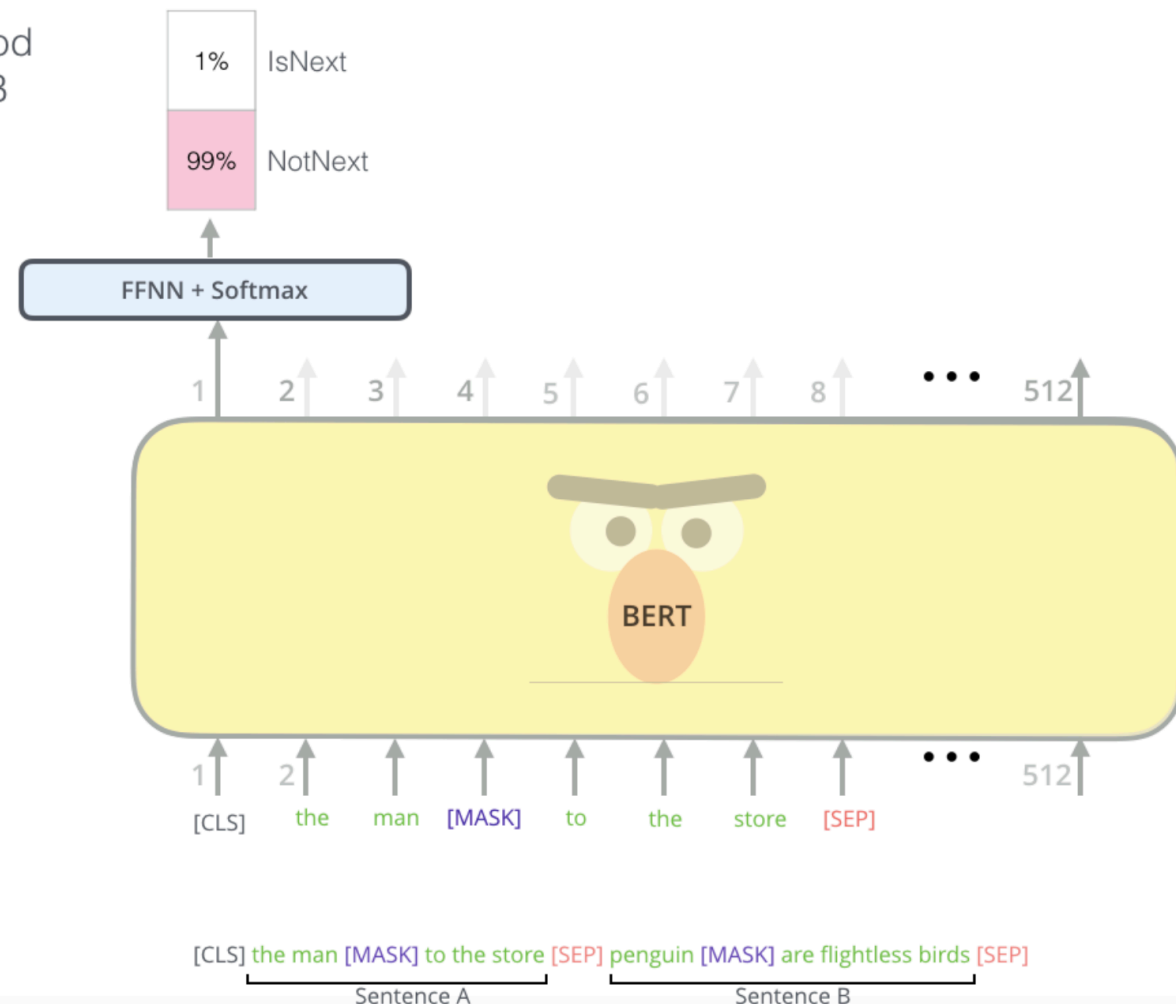
- 目的

- (1) 让模型更好地理解句子之间的联系；
- (2) 使模型更适用于QA和NPI等多句任务。

- 具体操作

- (1) 给定句子A和句子B，B有50%的概率为A的下一句；
- (2) 二分类目标：预测句子B是否为A的下一句话；
- (3) 预训练时准确率达到97%~98%。

Predict likelihood
that sentence B
belongs after
sentence A

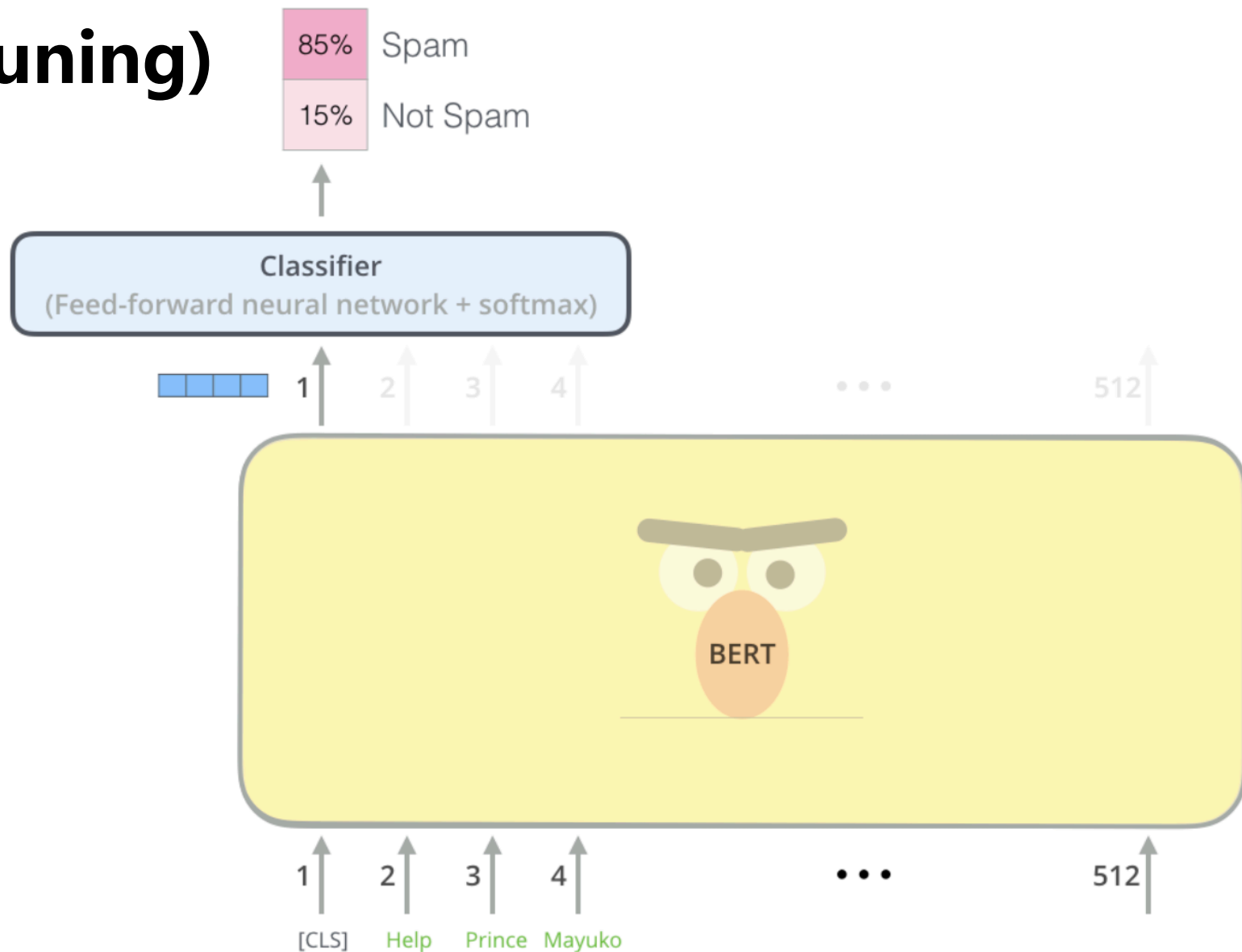


BERT的微调 (Fine-tuning)

1. 分类任务

sequence-level类的文本分类，
直接取第一个token 即 [CLS]
的最后一层encoder隐层输出
 $C \in \mathbb{R}^H$ ，加上一层权重 $W \in \mathbb{R}^{K \times H}$ ，然后softmax预测label proba，如下：

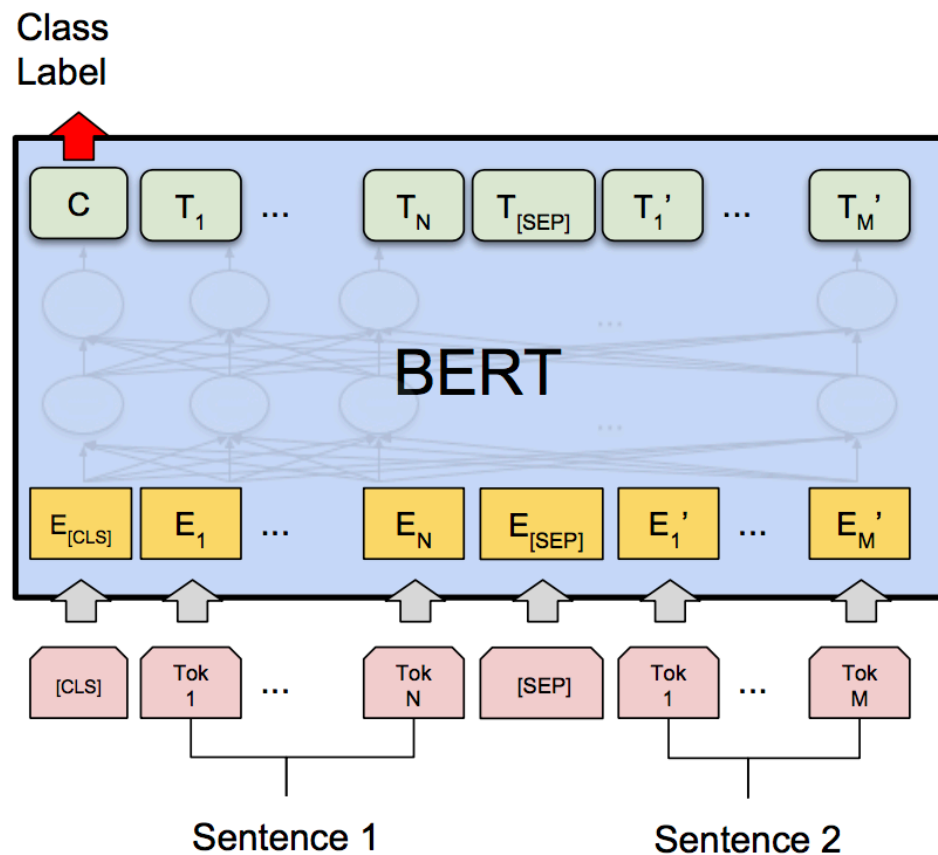
$$P = \text{softmax}(CW^T)$$



BERT的微调 (Fine-tuning)

2. 双句分类任务

与单句分类任务相似，依然是取第一个token 即 [CLS] 的最后一层 encoder 隐层输出 C ，加上一层权重 W ，然后softmax预测label proba，注意在输入的时候，对两个句子进行分句标识[SEP]。



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

BERT的微调 (Fine-tuning)

3. 问答任务

Input : 一个问题和一篇包含问题答案的文章

Return : 问题答案在文章中的起止位置

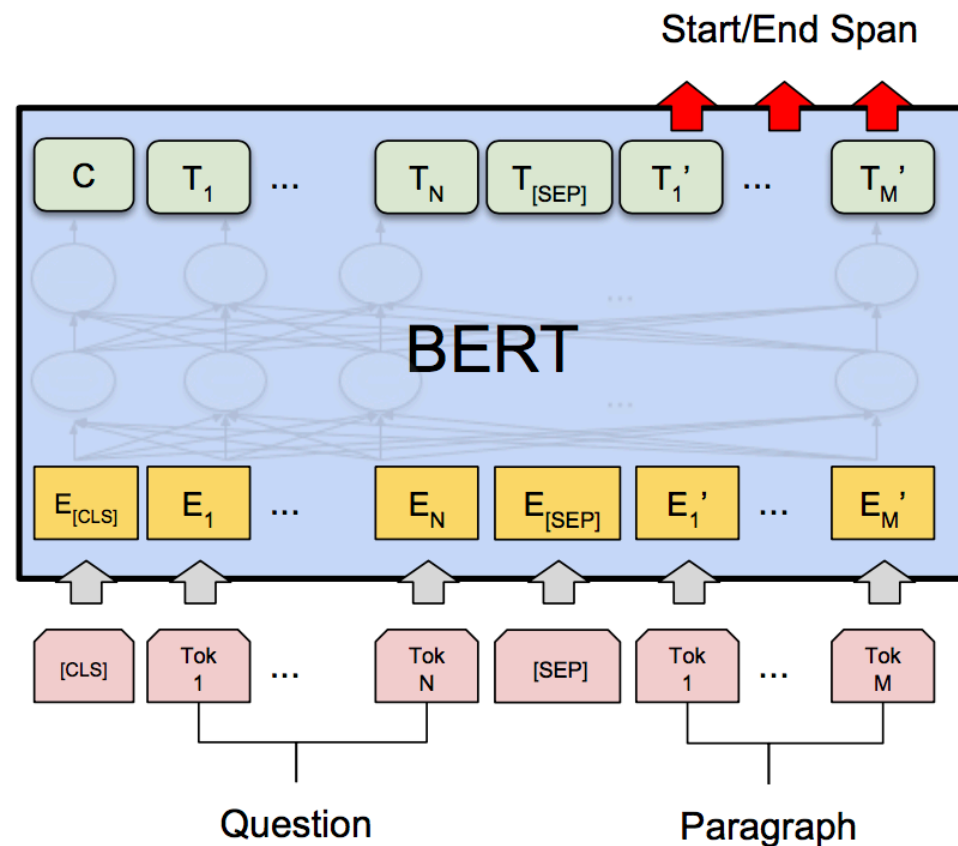
(相当于一个答案抽取的过程)

具体过程 :

(1) 引入两个向量 , $S \in R^H$, $E \in R^H$, 分别代表答案起始和结束位置的判断向量 ;

(2) 用词 T_i 和 S 的内积表示起始概率 , 词 T_i 和 E 的内积表示结束概率 ;

(3) 一个答案片段得分为 $S \times T_i + E \times T_j$, 得分最大的片段即为答案。



(c) Question Answering Tasks:
SQuAD v1.1

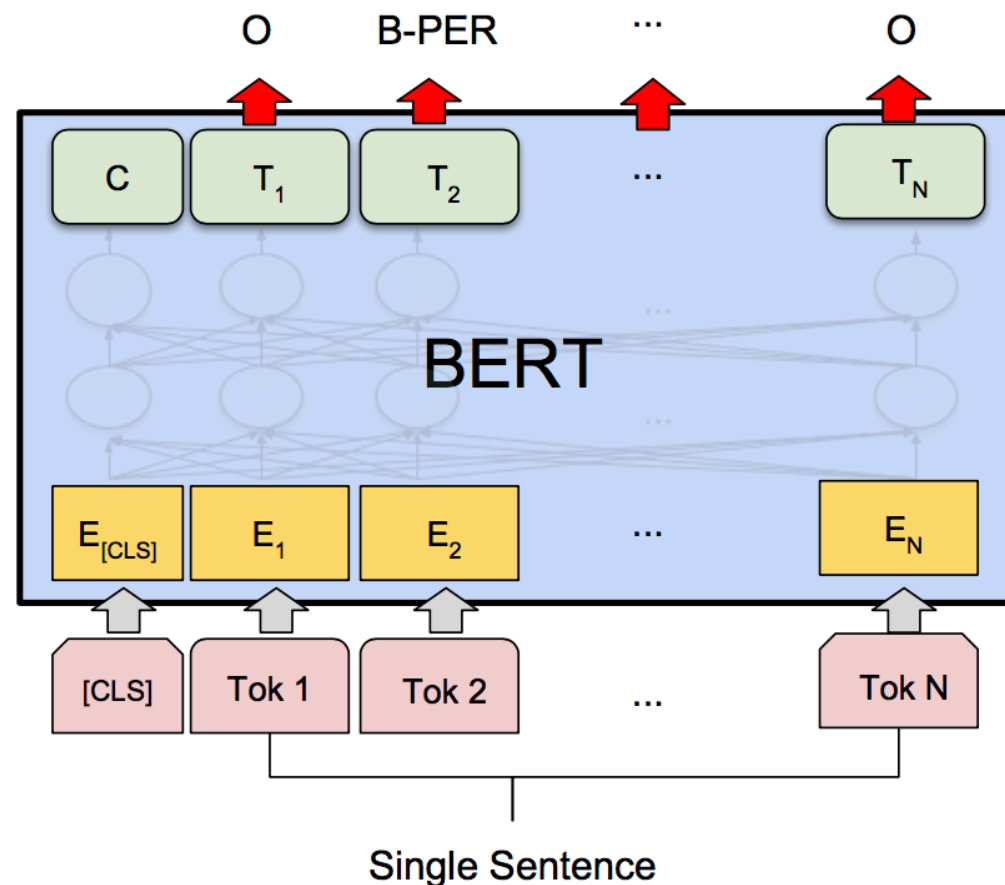
BERT的微调 (Fine-tuning)

4. NER标记任务

对于中文NER任务，使用字向量bert embeddings；

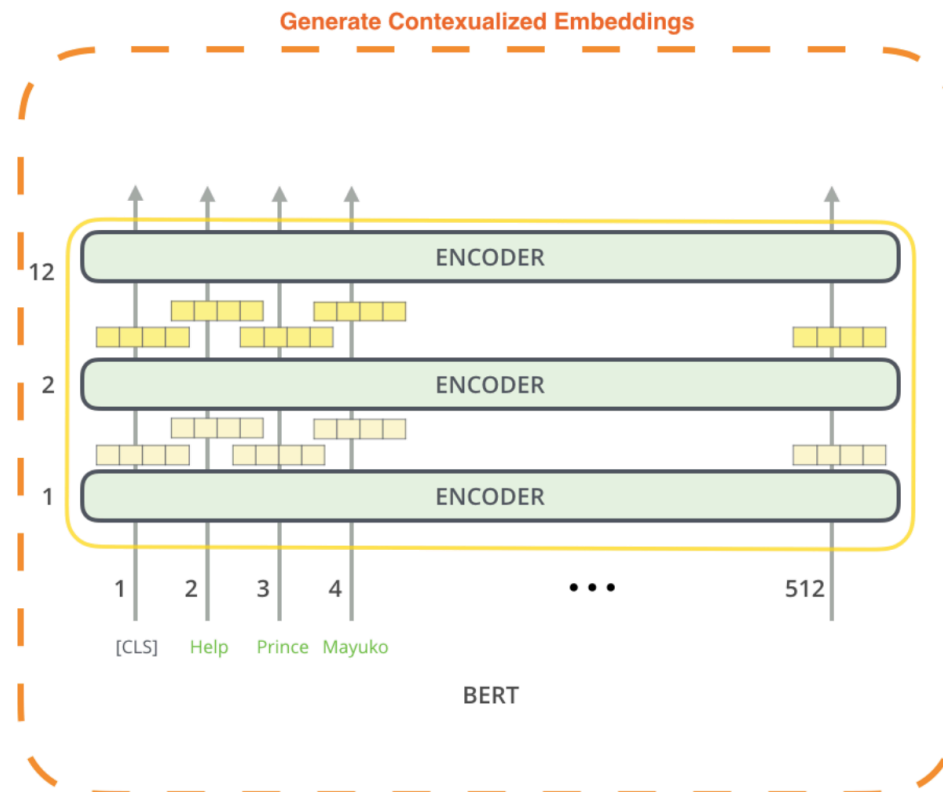
字向量的获取：kashgari package

```
(`from kashgari.embeddings import BERTEmbedding;  
embedding = BERTEmbedding( "bert-base-chinese" , 200);  
`)
```

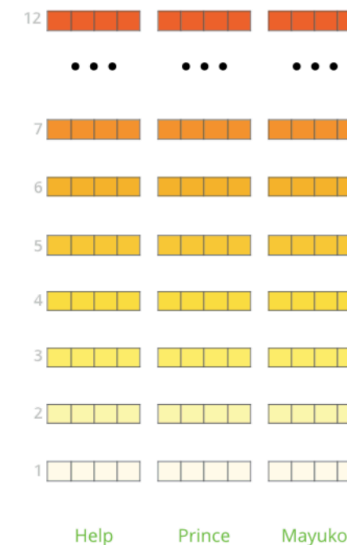


BERT用于特征抽取

- Fine-tuning不是使用BERT的唯一方式，可以使用BERT预训练模型获取字 / 词 / 句embeddings，用于其他下游任务；
- 关键在于：应该使用哪一层encoder的hidden states



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

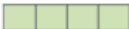

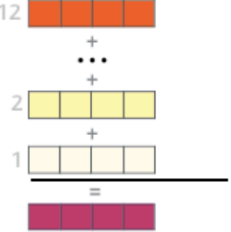


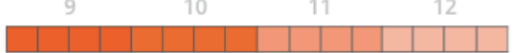


But which one should we use?

BERT用于特征抽取

- Fine-tuning不是使用BERT的唯一方式，可以使用BERT预训练模型获取字 / 词 / 句embeddings，用于其他下游任务；
- 关键在于：应该使用哪一层encoder的hidden states；
- 在CoNLL-2003 NER任务中，使用最后四层的拼接作为embedding，F1得分最高。

What is the best contextualized embedding for “**Help**” in that context?
For named-entity recognition task CoNLL-2003 NER

		Dev F1 Score
First Layer	Embedding 	91.0
Last Hidden Layer	12 	94.9
Sum All 12 Layers		95.5
Second-to-Last Hidden Layer	11 	95.6
Sum Last Four Hidden		95.9
Concat Last Four Hidden		96.1

Thank you !