自 然 语 言 处 理

Natural language Processing

第八章、预训练语言表示模型

第8章

CONTENTS

01 双向Transformer编码器

02 预训练模型的预训练

03 预训练模型的微调

04 提示学习（Prompting)

语言丰富性

一个表达流利且丰富的人应该具备哪些特点？

一个美国成年人的词汇量大概在3万-10万的范围内

日常使用的单词量非常有限，绝大部分单词通过阅读来认识

基于分布假设的单词学习

单词与单词的共现或之间的关联性可以表示知识

回顾第六章中的向量语义，
在训练过程中生成的静态语义向量(训练过程)，可以用于新环境，例如：

- 单词间的相似性度量
- 随时间增长词义变化相关的研究等

动态上下文向量

Word2vec/GloVe 中，

- 一个单词对应的一个单独的静态词向量

本章讲述的BERT/GPT，

- 一个单词在不同的上下文中词向量表达也不同。

预训练语言模型

两种范式：

1. **自回归的语言模型**：从左到右循序的训练模型
   - Transformer (第九章)
     - 可用于需要自回归生成的任务(上下文生成，机器翻译等)

2.**非自回归的语言模型**：双向Transformer编码器+掩码语言建模
   - Bert(本章内容)
     - 允许模型同时看到文本双向的全部内容(包括右到左)

Part.01

双向Transformer编码器

双向Transformer编码器

BERT- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers

RoBERTa - A **R**obustly **O**ptimized **BERT** Pretraining **A**pproach

SpanBERT - Improving Pre-training by Representing and Predicting **Span**s

单向Transformer的缺点

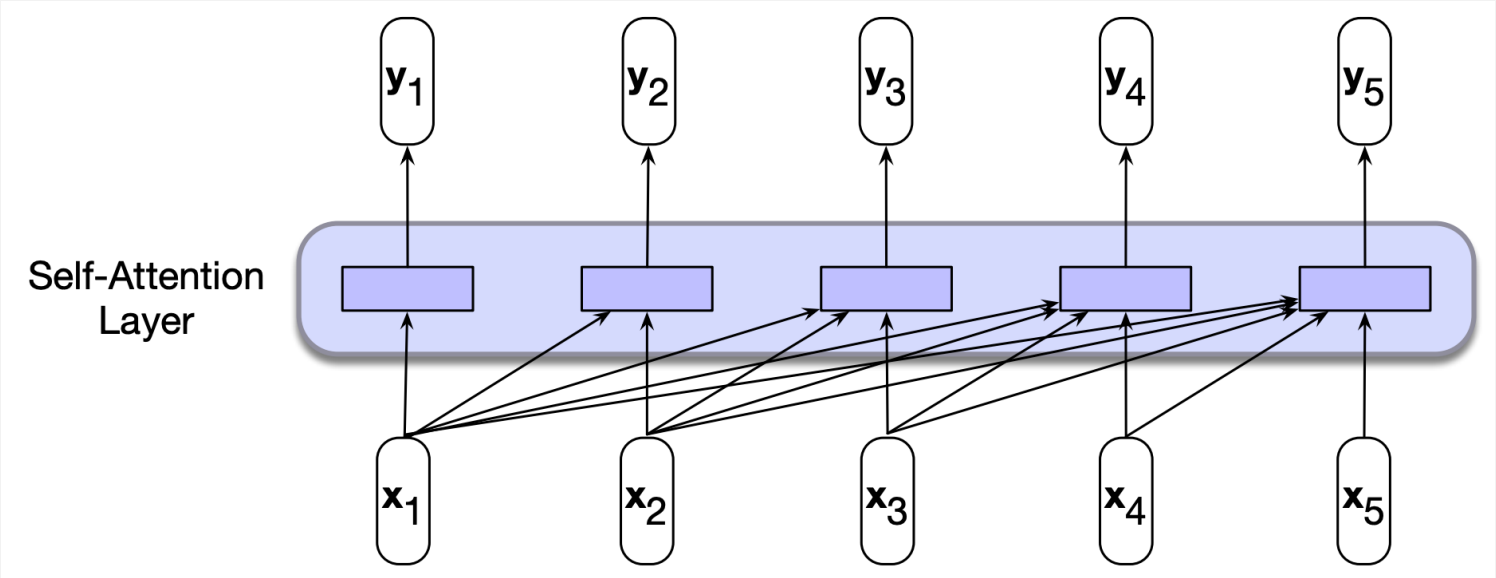
单向Transformer是增量式的，从左到右的序列处理方式。

当处理序列分类、序列标记或者其他复杂语言标记任务时(解析标签)

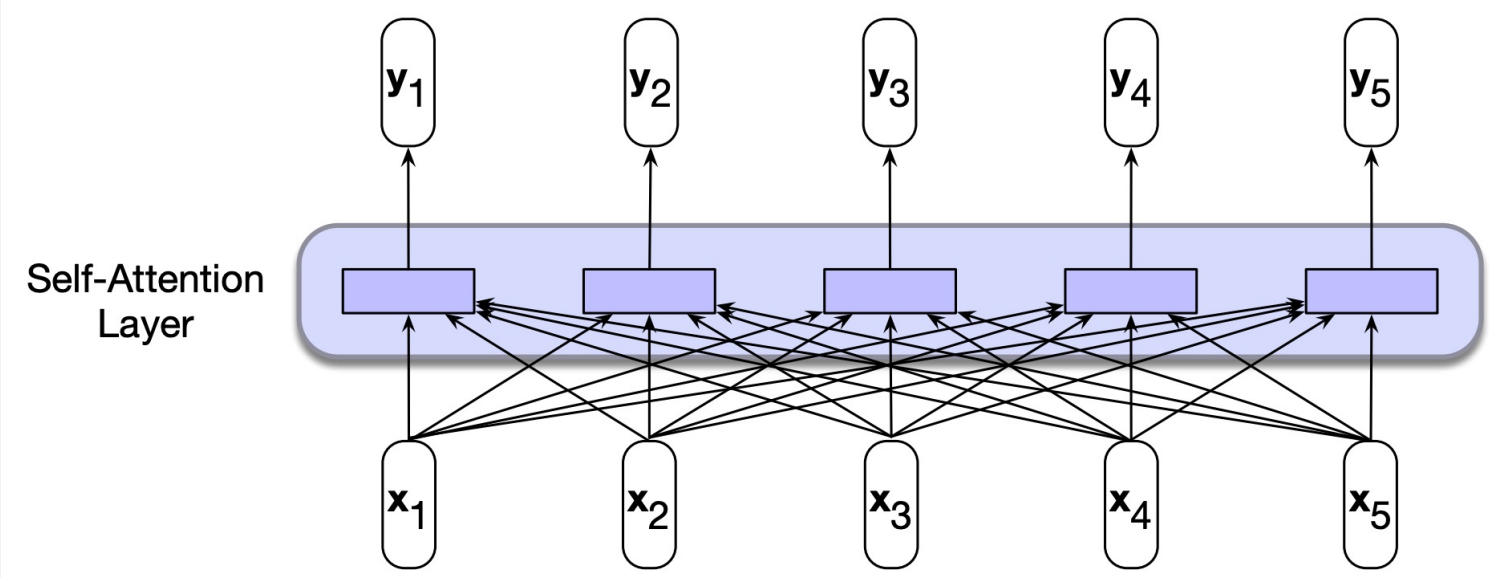
- 需同时考虑双向的信息才能更好完成这类任务

单向与双向Transformer对比

单向



双向



双向Transformer编码器

使用与单向Transformer一样的自注意力机制

使用可以训练学习到的查询，键，值三个矩阵，为序列中的每一个输入

- 生成对应的查询，键，值嵌入

$$\mathbf{q}_i = \mathbf{W}^{\mathbf{Q}}\mathbf{x}_i;\ \ \mathbf{k}_i = \mathbf{W}^{\mathbf{K}}\mathbf{x}_i;\ \ \mathbf{v}_i = \mathbf{W}^{\mathbf{V}}\mathbf{x}_i$$

每个输入所对应的输出yi是由序列中所有输入的加权和求解得出的

$$\mathbf{y}_i = \sum_{j=i}^{n} \alpha_{ij}\mathbf{v}_j$$

双向Transformer编码器

权重 α：基于当前元素查询向量和其他元素键向量对比分数的，softmax分布得出的。

$$\alpha_{ij} = \frac{\exp(\text{score}_{ij})}{\sum_{k=1}^{n} \exp(\text{score}_{ik})}$$

$$\text{score}_{ij} = \mathbf{q}_i \cdot \mathbf{k}_j$$

对比分数即两两向量间的点积

双向Transformer编码器

每个y的计算都是独立的，因此该过程是可以通过矩阵操作来进行并行化处理的。

1. 将所有输入向量x整合为一个矩阵X $\mathbf{X} \in \mathbb{R}^{N \times d_h}$ (N是序列长度, d_h是词向量维度)
   - 每一行是序列中一个元素的词向量(一共N行)

2. 将 X乘以W^Q，W^K，W^V (三个矩阵的维度为dxd，这里的d等于d_h)

   - 生成矩阵 $\mathbf{Q} = \mathbf{X}\mathbf{W}^{\mathbf{Q}};\ \mathbf{K} = \mathbf{X}\mathbf{W}^{\mathbf{K}};\ \mathbf{V} = \mathbf{X}\mathbf{W}^{\mathbf{V}}$ $\mathbf{Q} \in \mathbb{R}^{N \times d}$ $\mathbf{K} \in \mathbb{R}^{N \times d}$ $\mathbf{V} \in \mathbb{R}^{N \times d}$
   - 这三个矩阵里包含了序列里每个t元素所对应的查询、键、值向量。
   - 那么所有需要的对比分数也可以通过乘以Q和K转置来计算得到。

3. 对所有对比分数进行归一化，再进行softmax计算，将结果乘以值矩阵V
   - 计算得出自注意力分数的矩阵

$$SelfAttention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\intercal}}{\sqrt{d_k}}\right)\mathbf{V}$$

$N \times N$ $\mathbf{QK}^{\mathsf{T}}$ 矩阵

双向Transformer中，当前输入会与序列中的每一项元素作比较

N

q1•k1	q1•k2	q1•k3	q1•k4	q1•k5
q2•k1	q2•k2	q2•k3	q2•k4	q2•k5
q3•k1	q3•k2	q3•k3	q3•k4	q3•k5
q4•k1	q4•k2	q4•k3	q4•k4	q4•k5
q5•k1	q5•k2	q5•k3	q5•k4	q5•k5

N

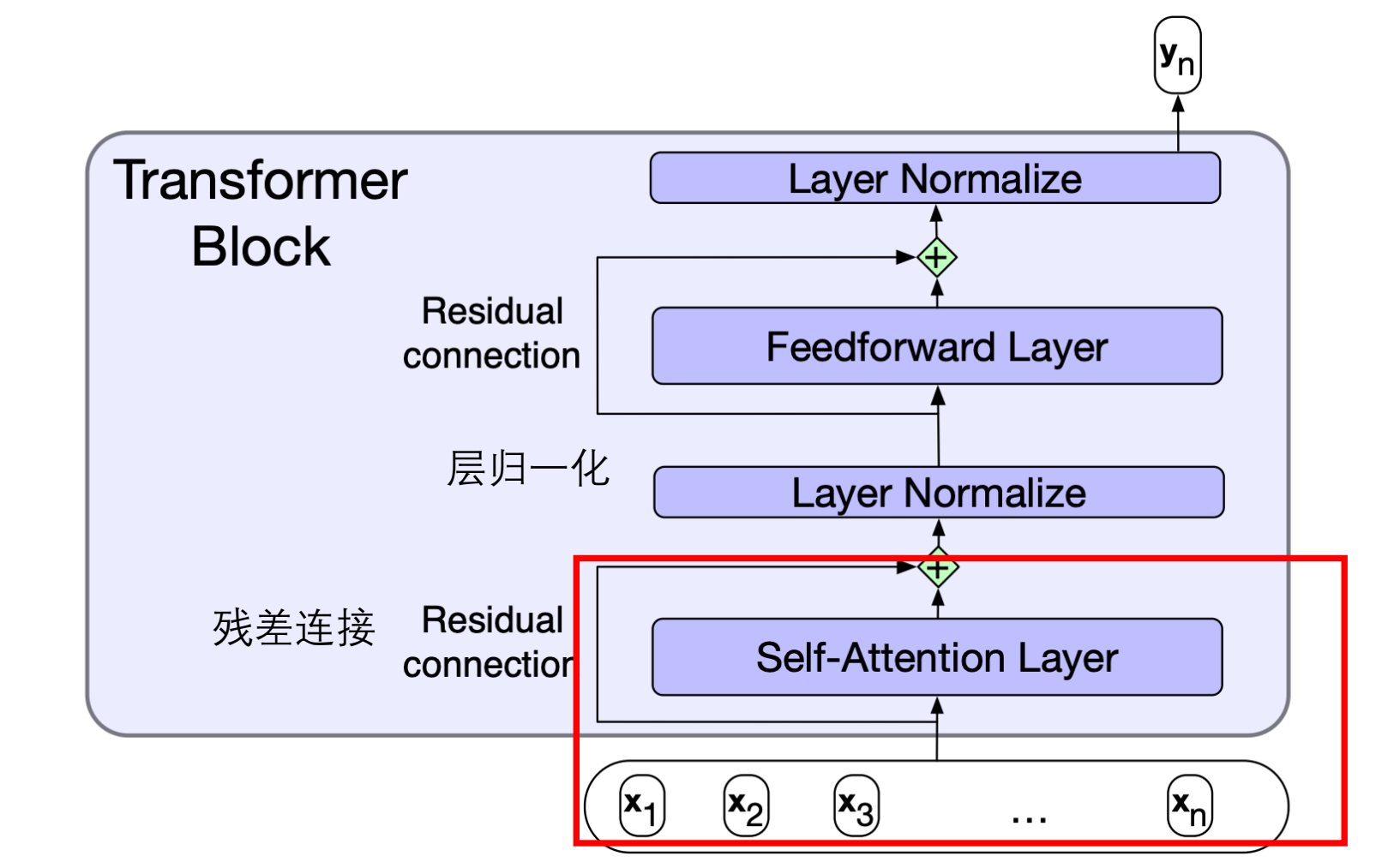
单向Transformer

N

q1•k1	$-\infty$	$-\infty$	$-\infty$	$-\infty$
q2•k1	q2•k2	$-\infty$	$-\infty$	$-\infty$
q3•k1	q3•k2	q3•k3	$-\infty$	$-\infty$
q4•k1	q4•k2	q4•k3	q4•k4	$-\infty$
q5•k1	q5•k2	q5•k3	q5•k4	q5•k5

N

BERT中的Transformer块



注意力层的计算使用的是上下文信息，而非上文信息

Part.02

预训练语言模型的训练

预训练语言模型的训练过程

1. **预训练（Pre-training）：** 构建基于transformer的预训练语言模型后，通过海量无标注的文本数据进行学习，获取词义或句意表达的学习过程。

如何进行预训练，是预训练语言模型关注的重点？

2. **微调(Fine-tuning)：** 针对不同的下游任务，利用下游任务的标注语料对上预训练模型的参数进行调整。

迁移学习

预训练+微调

迁移学习：从一项任务或者领域里获取知识，再将知识转移并用于解决新的任务的方法

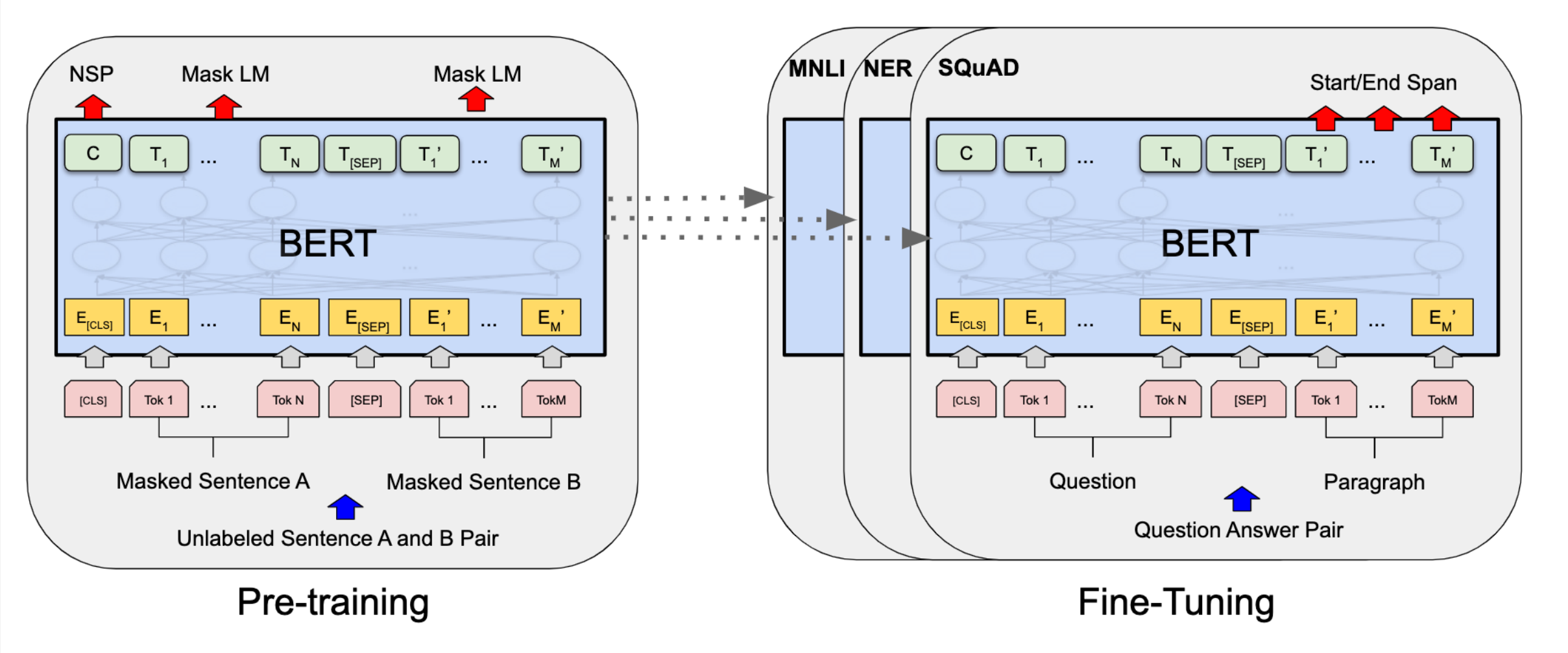
BERT中的完形填空任务

RNN、LSTM、Transformer等序列模型可以用于下一单词预测

Please turn your homework _____ .

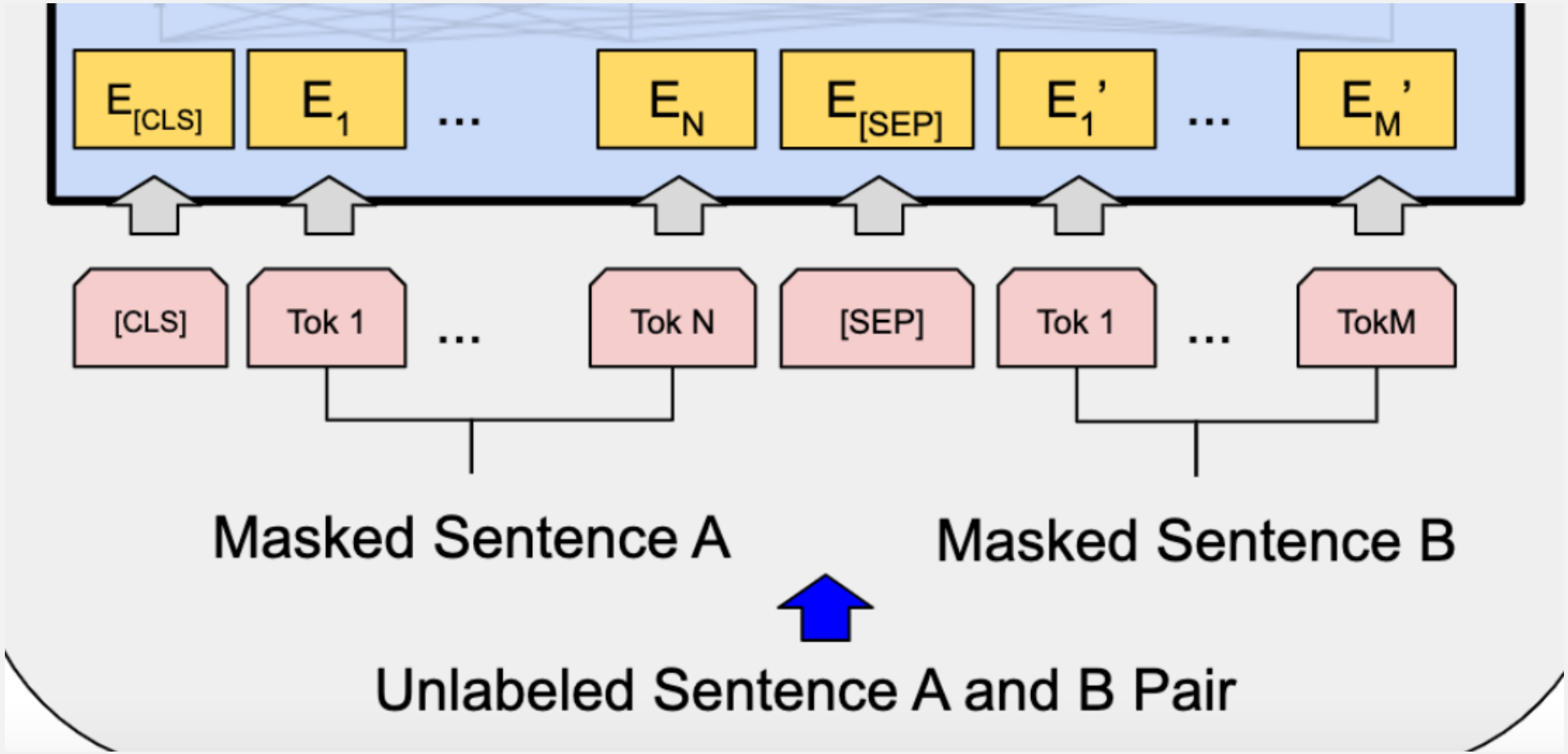
不仅有考虑上文信息，BERT模型还考虑当前输入的下文信息，因此可以用于类似完形填空的预测任务

Please turn _____ homework in.



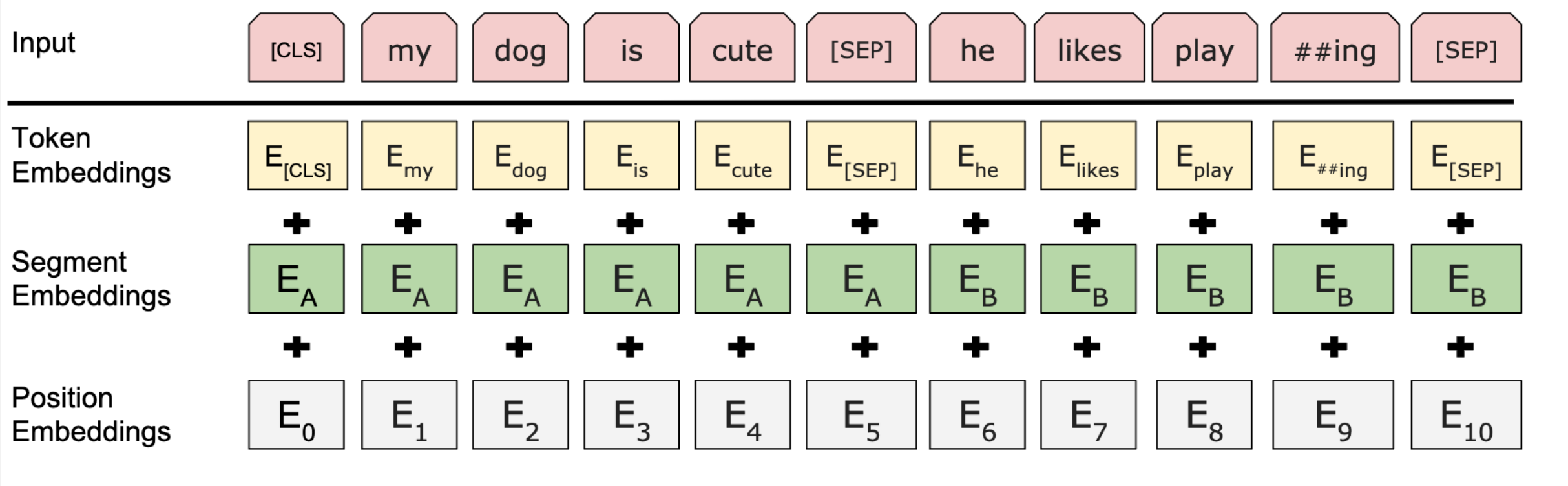
预训练语言模型BERT的“预训练”和“微调”

BERT的输入



特殊符号：
分类符号[CLS]
句子分割[SEP]

BERT的输入表示



BERT的输入表示由标记嵌入向量、段嵌入向量和位置嵌入向量求和得到

预训练---掩码语言建模(MLM)

为了训练一个深层的双向表示，BERT只需**随机屏蔽一定百分比的输入标记**(token)，然后预测那些被屏蔽的标记。BERT把这个过程称为掩码语言建模，尽管它在文献中经常被称为完形填空。

与标准的语言模型一样，与掩码标记相对应的最终隐藏向量通过输入到softmax进行预测。 然后使用交叉熵对预测值和真实值构造损失函数，对网络进行优化。

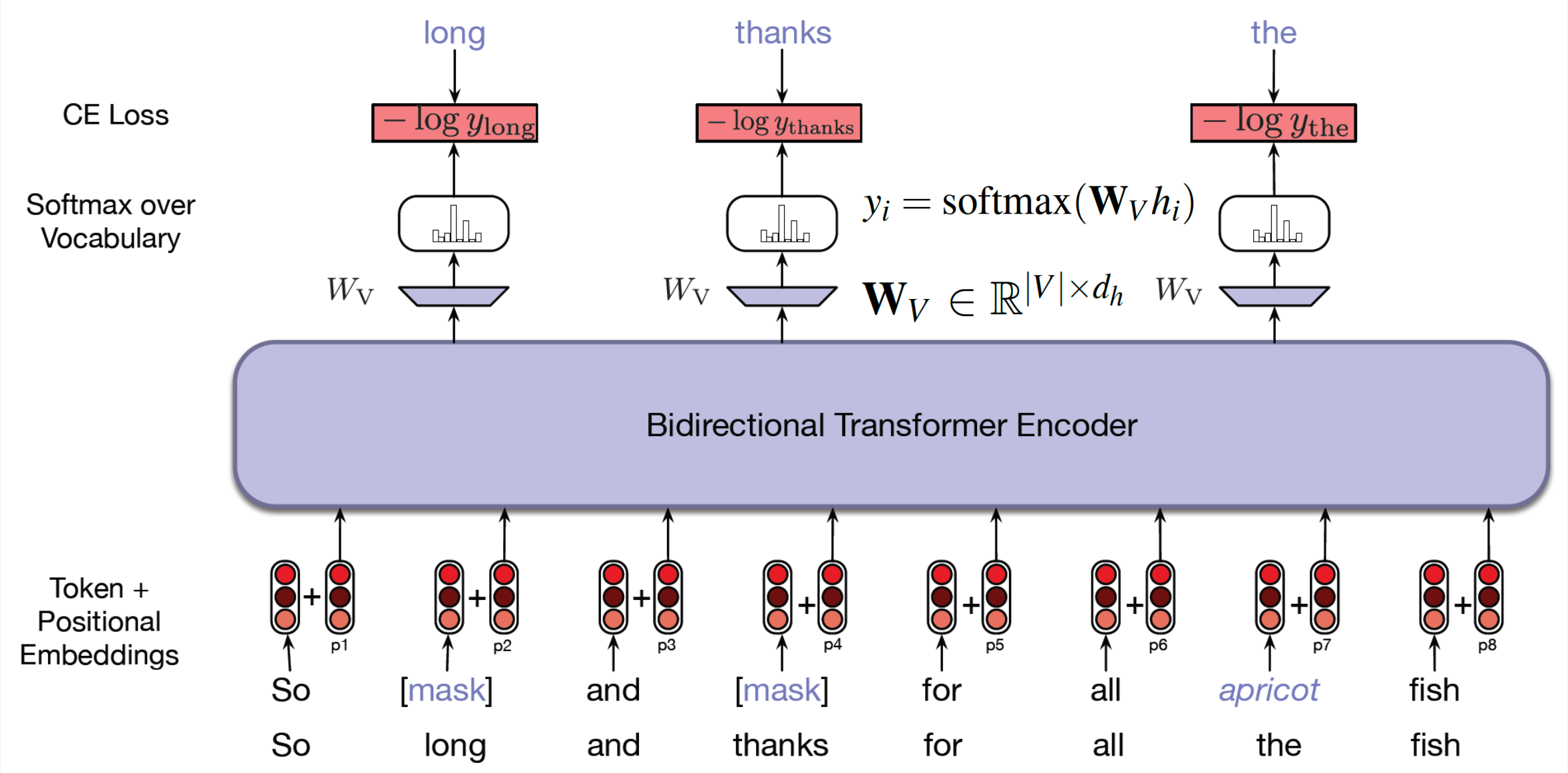
掩码语言建模(MLM)

尽管这允许获得一个双向的预训练模型，但缺点是在预训练和微调之间造成了不匹配，因为[MASK]标记在微调过程中不会出现。

为了缓解这种情况，BERT并不总是用实际的[MASK]标记替换“MASK”单词。训练数据生成器随机选择15%的标记位置进行预测。
如果选择了第i个标记，BERT将第i个标记替换为：

- 80%的概率为[MASK]标记；
- 10%的概率为随机标记；
- 10%的概率不更改的第i标记。

单词掩码


long
thanks
the
CE Loss
$-\log y_{\text{long}}$
$-\log y_{\text{thanks}}$
$-\log y_{\text{the}}$
Softmax over Vocabulary
$y_i = \text{softmax}(\mathbf{W}_V h_i)$
W_V
$\mathbf{W}_V \in \mathbb{R}^{|V| \times d_h}$
Bidirectional Transformer Encoder
Token + Positional Embeddings
p1
p2
p3
p4
p5
p6
p7
p8
So
[mask]
and
[mask]
for
all
apricot
fish
So
long
and
thanks
for
all
the
fish

预训练-下句预测任务 (NSP)

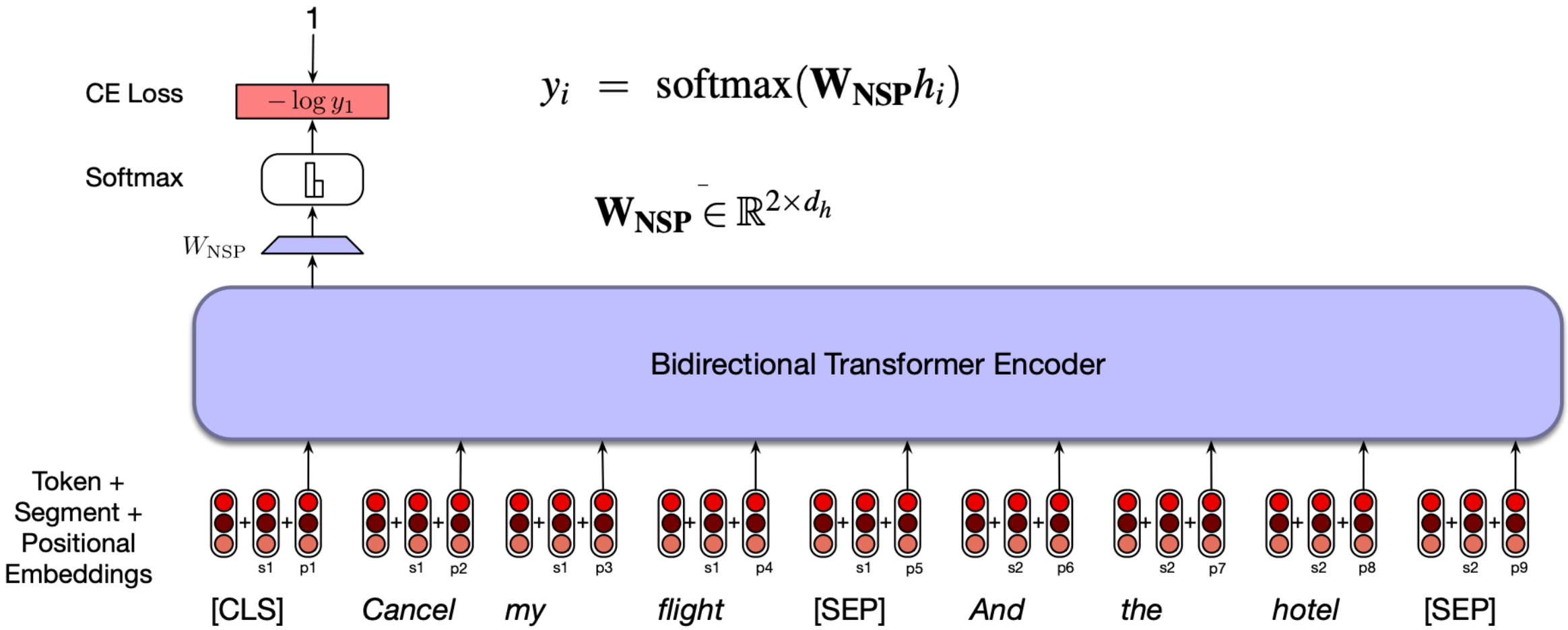
检测两个句子是否：

- 有相似释义
- 有蕴含关系
- 内容冲突
- 是连贯的

BERT训练数据：50%原始正向句对、50%句对的后一句从数据集中随机选择。

即BERT尝试将两种句对进行区分来训练模型。

BERT中的下句预测任务



$$y_i = \text{softmax}(\mathbf{W_{NSP}} h_i)$$

$$\mathbf{W_{NSP}} \in \mathbb{R}^{2 \times d_h}$$

训练好的BERT的两个版本

- BERT使用WordPiece算法生成token(第二章内容)
  - 大约一共有30,000tokens

In this work, we denote the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A.[3] We primarily report results on two model sizes: $\mathbf{BERT_{BASE}}$ (L=12, H=768, A=12, Total Parameters=110M) and $\mathbf{BERT_{LARGE}}$ (L=24, H=1024, A=16, Total Parameters=340M).

BERT中的训练细节

训练语料库：BookCorpus （**800M words**）& Wikipedia (**2500M words**)

训练数据为句子对，句对的最大长度为512个tokens

使用MLM对句对进行掩盖后，作为输入

损失函数：由MLM的损失结合NSP的损失构成

使用训练数据集对模型训练将近40次以达到收敛(适应绝大部分训练数据)

训练结束后将获得，所有参与训练的单词词向量和训练好的所有权重(用于预测)

其他的预训练语言模型

从普通的文本中，BERT利用了两种自监督信号训练Transformer双向编码器：掩码语言模型和下句预测

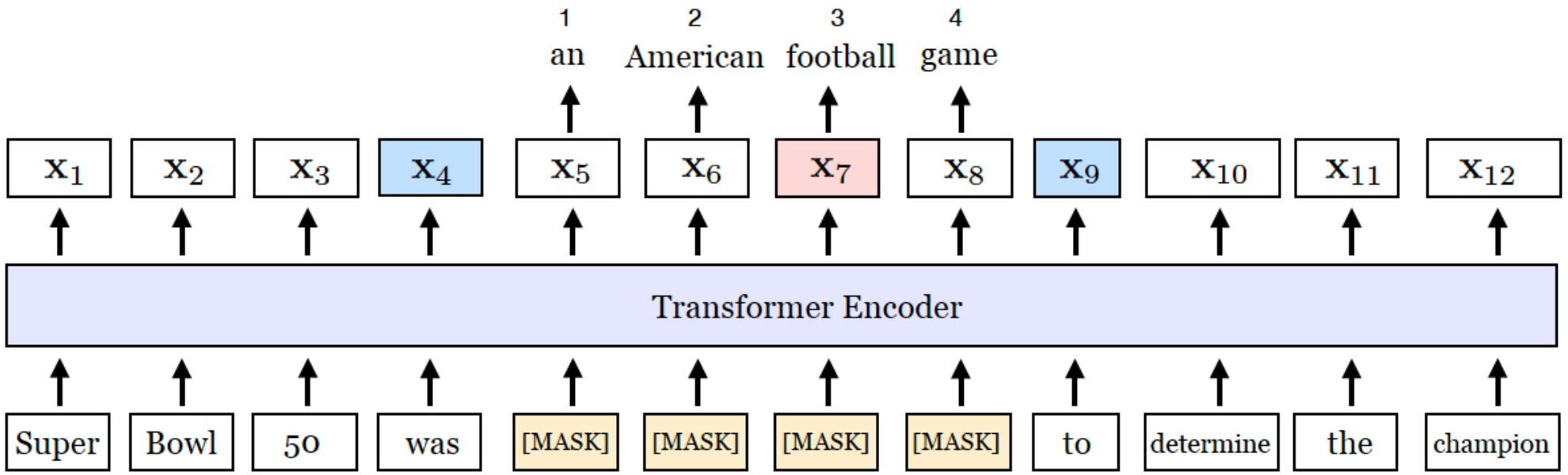
是不是更多的自监督信号，也可以拿来训练？

SpanBERT

- 提出了**更好的 Span Mask 方案**，也再次展示了随机遮盖连续一段字要比随机遮盖掉分散字好；
- 通过**加入 Span Boundary Objective (SBO) 训练目标**，增强了 BERT 的性能，特别在一些与 Span 相关的任务，如抽取式问答；
- 用实验获得了和 XLNet 类似的结果，发现**不加入 Next Sentence Prediction (NSP) 任务，直接用连续一长句训练效果更好**。

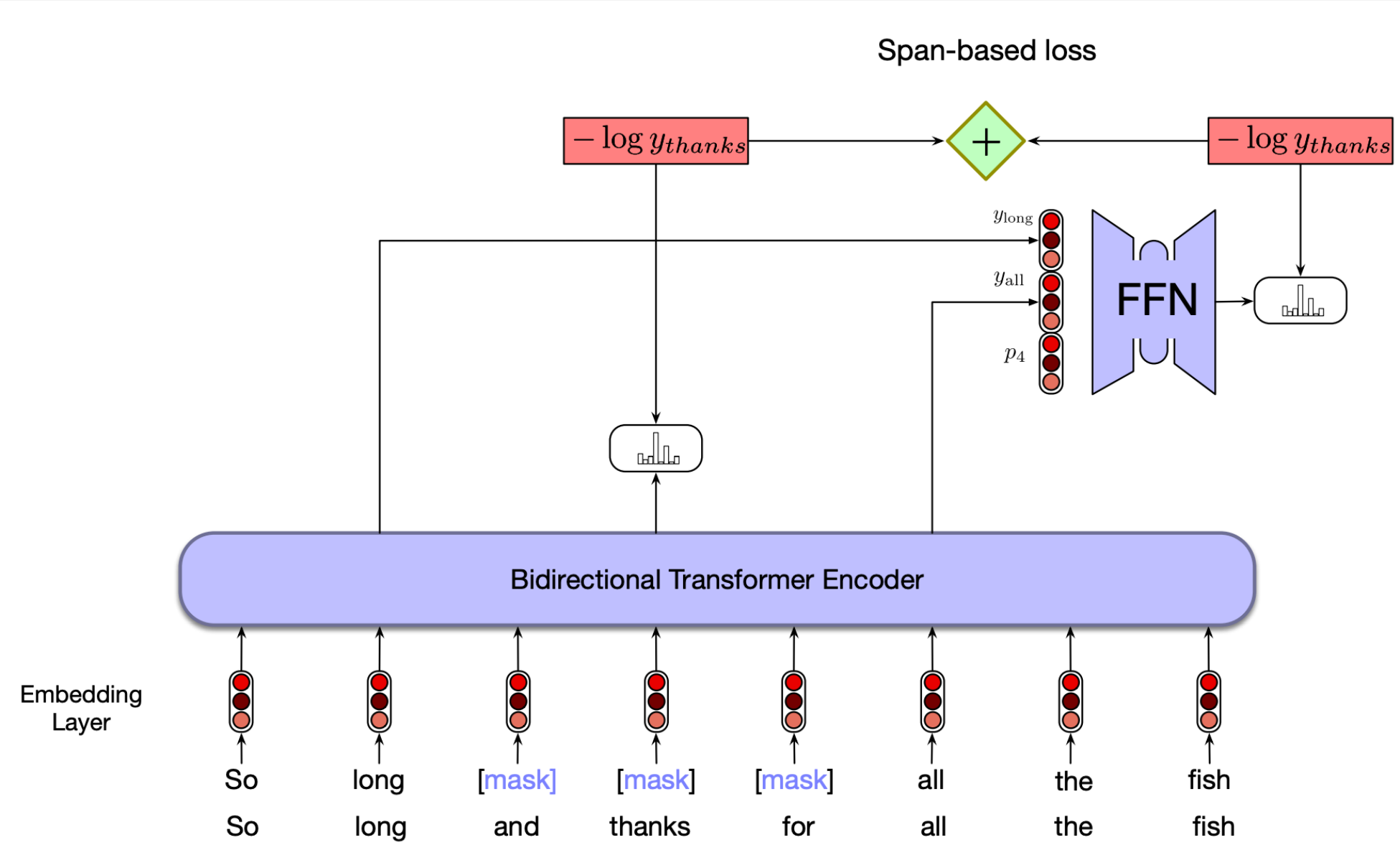
SpanBERT

$$\mathcal{L}(\text{football}) = \mathcal{L}_{\text{MLM}}(\text{football}) + \mathcal{L}_{\text{SBO}}(\text{football})$$
$$= -\log P(\text{football} \mid \mathbf{x}_7) - \log P(\text{football} \mid \mathbf{x}_4, \mathbf{x}_9, \mathbf{p}_3)$$



Span “An American football game” 被掩盖。 Span Boundary Objective (SBO) 用x和x的表示预测Span中的每个词语

SpanBERT模型训练



词段掩码 - SpanBERT

基于Span的下游任务依赖于集合Span中的所有单词token的表达，

- 包括Span的起点终点或边界

SpanBERT通过Span边界目标(SPO)增加了MLM的学习目标。

- SPO依赖于一个模型在预测基于掩码词段上下文的Span中所有单词的能力。
- 该预测通过使用和掩码词段上下文相关的输出向量来完成。
  - Span中所有单词相对应的位置编码也被预测了。

词段掩码 - SpanBERT

s表示Span之前一个单词的位置，e表示Span之后一个单词的位置

Span中位置i的单词生成是通过

1. 拼接s和e位置的输出向量和i的位置编码
   - (该输出向量即s/e两个位置的边界向量)
2. 将结果输送入一个双层的前馈网络

$$\mathbf{y}_i = f(\mathbf{x}_{s-1}, \mathbf{x}_{e+1}, \mathbf{p}_{i-s+1})$$

$$\begin{aligned}
\mathbf{h}_0 &= [\mathbf{x}_{s-1}; \mathbf{x}_{e+1}; \mathbf{p}_{i-s+1}] \\
\mathbf{h}_1 &= \mathrm{LayerNorm}(\mathrm{GeLU}(\mathbf{W}_1\mathbf{h}_0)) \\
\mathbf{y}_i &= \mathrm{LayerNorm}(\mathrm{GeLU}(\mathbf{W}_2\mathbf{h}_1))
\end{aligned}$$

SpanBERT中的损失函数

$$
\begin{aligned}
\mathcal{L}(x_i) &= \mathcal{L}_{\mathrm{MLM}}(x_i) + \mathcal{L}_{\mathrm{SBO}}(x_i) \\
&= -\log P\left(x_i \mid \mathbf{x}_i\right) - \log P\left(x_i \mid \mathbf{y}_i\right)
\end{aligned}
$$

SpanBERT中的性能

	SQuAD 1.1		SQuAD 2.0	
	EM	F1	EM	F1
Human Perf.	82.3	91.2	86.8	89.4
Google BERT	84.3	91.3	80.0	83.3
Our BERT	86.5	92.6	82.8	85.9
Our BERT-1seq	87.5	93.3	83.8	86.6
SpanBERT	**88.8**	**94.6**	**85.7**	**88.7**

Table 1: Test results on SQuAD 1.1 and SQuAD 2.0.

Part.03

预训练模型的微调

微调

预训练语言模型可以从大量文本信息中提取知识，如：语义、词性等

- 有益于不同的NLP下游任务

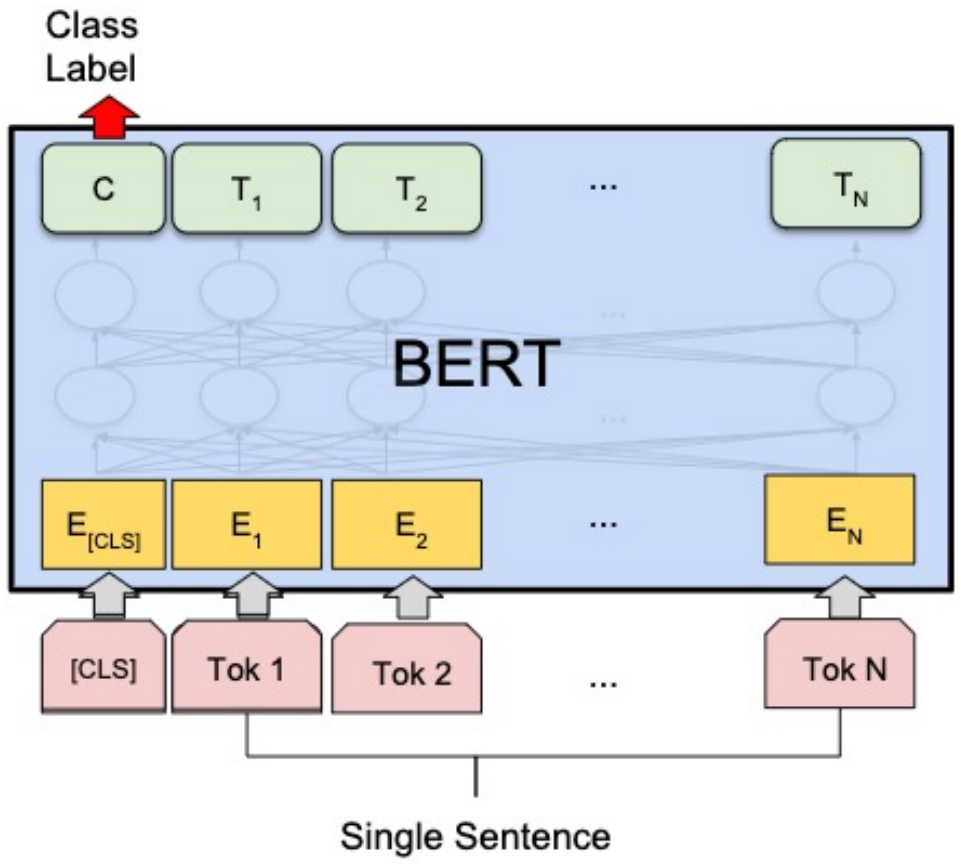
这些知识的使用需要通过使用微调创建一个接口适应到不同任务

微调通过添加少数任务特定的参数，推进基于预训练模型应用的实现

这些参数的训练是通过学习具体任务中的带标签数据，来完成的

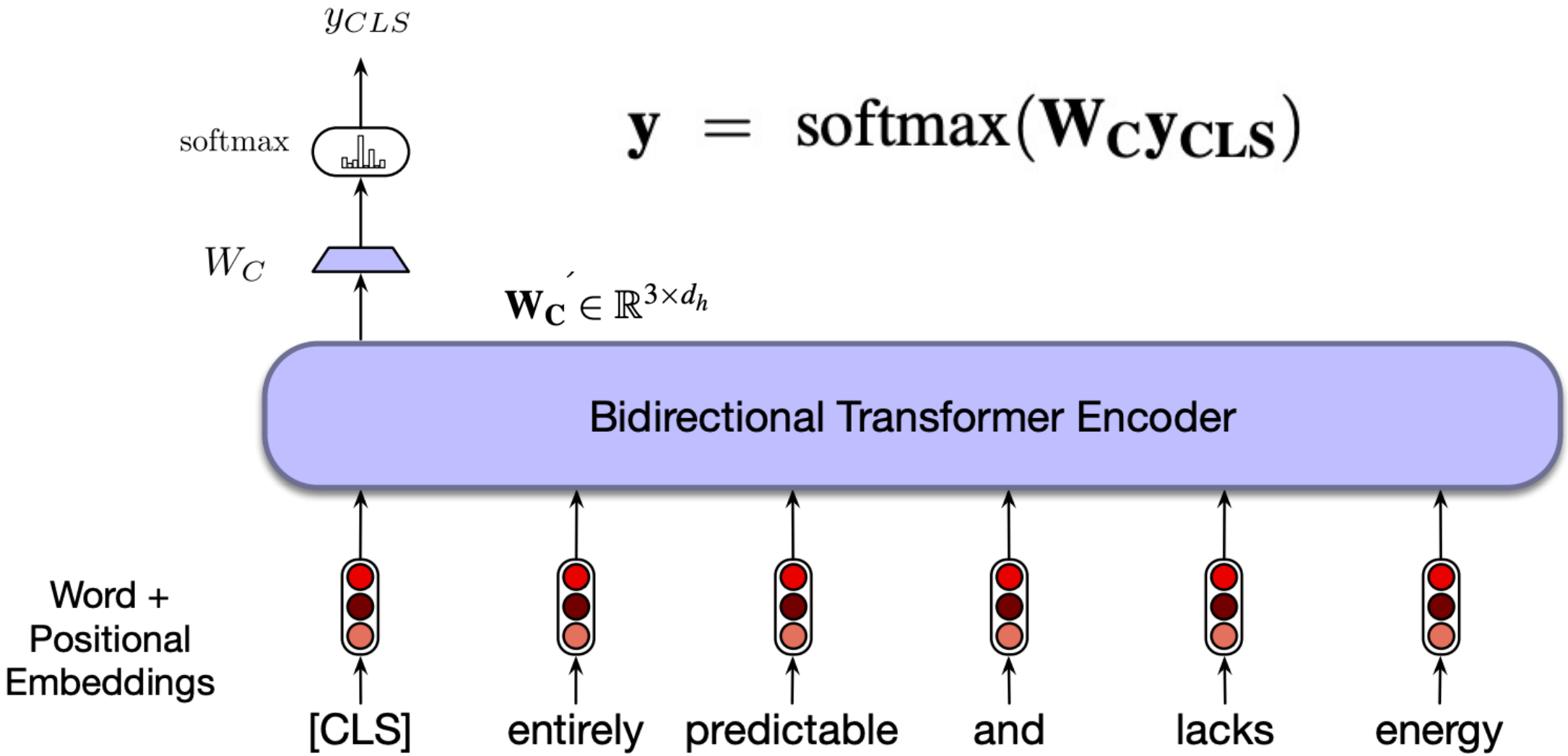
- 通常这个训练过程为保留原有预训练语言模型参数
- 或者对原有预训练语言模型参数进行微小调整

单句分类任务



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

BERT微调 – 情感分类



$$\mathbf{y} = \text{softmax}(\mathbf{W_C y_{CLS}})$$

BERT微调 – 情感分类

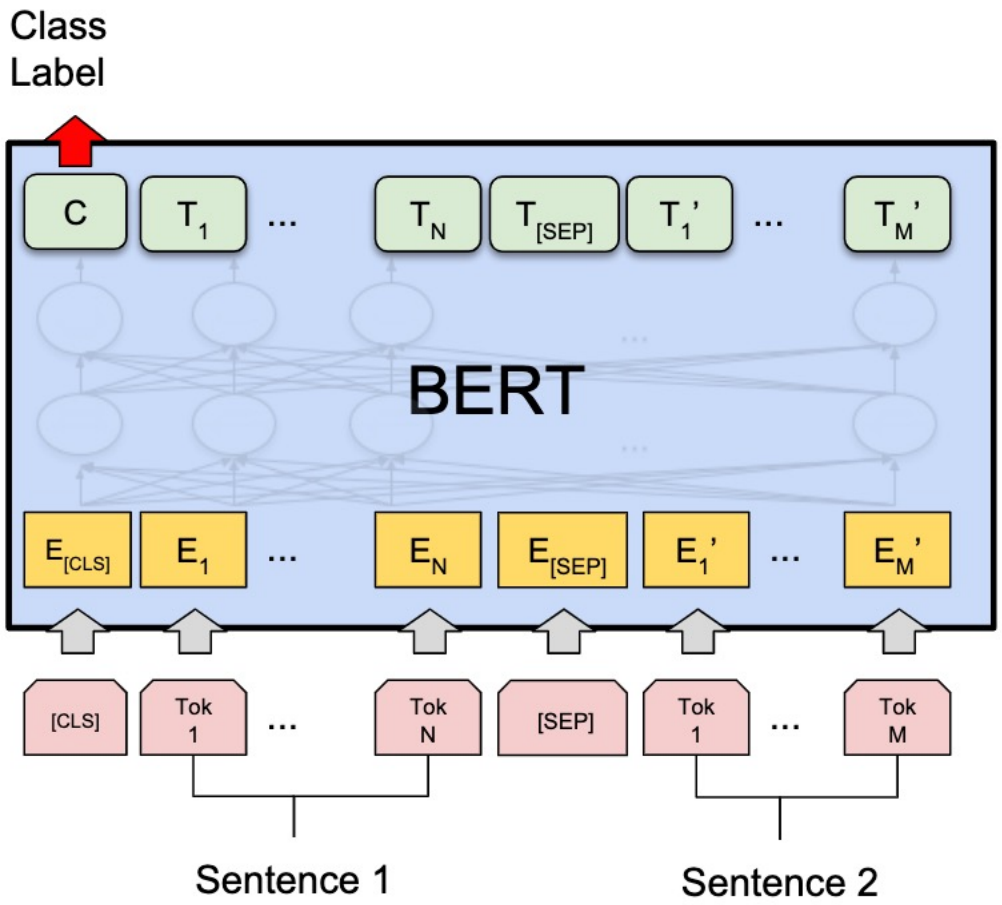
需要有带标签的训练数据来进行权重训练

- 使用softmax输出的分类和正确答案之间的交叉熵损失
- 推动参数矩阵 $\mathbf{W_C}$ 的学习和训练

微调中的损失函数可以用于：

- 训练分类器的参数
- 更新预训练语言模型的参数

序列(句)对：微调分类



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

序列(句)对：微调分类

句对相关的任务：

- 逻辑蕴涵、释义检测、话语分析等

训练数据为带标签的句对集。

- 句对的两个句子之间，使用[SEP]进行分隔

与序列分类一样，句对中[CLS]对应的输出向量代表整个句对序列。

训练过程：

1. [CLS]的对应向量乘以一个可学习的参数矩阵，
2. softmax，
3. 损失函数，梯度下降
4. 驱动权重更新

举例：逻辑蕴涵

- `Neutral`

  a: Jon walked back to the town to the smithy.

  b: Jon traveled back to his hometown.

- `Contradicts`

  a: Tourist Information offices can be very helpful.

  b: Tourist Information offices are never of any help.

- `Entails`

  a: I'm confused.

  b: Not all of it is very clear to me.

举例：逻辑蕴涵

三种标签：entails(蕴涵)、contradicts(矛盾)、 和 neutral(中性).
真实标签：[1,0,0]　　[0,1,0]　　[0,0,1]

数据准备：
[CLS] I’ m confused. [SEP] Not all of it is very clear to me.

$1 \times d_h$

其输出向量代表整个序列，乘以$3 \times d_h$的权重矩阵Wc，与真实标签进行损失计算

序列标记

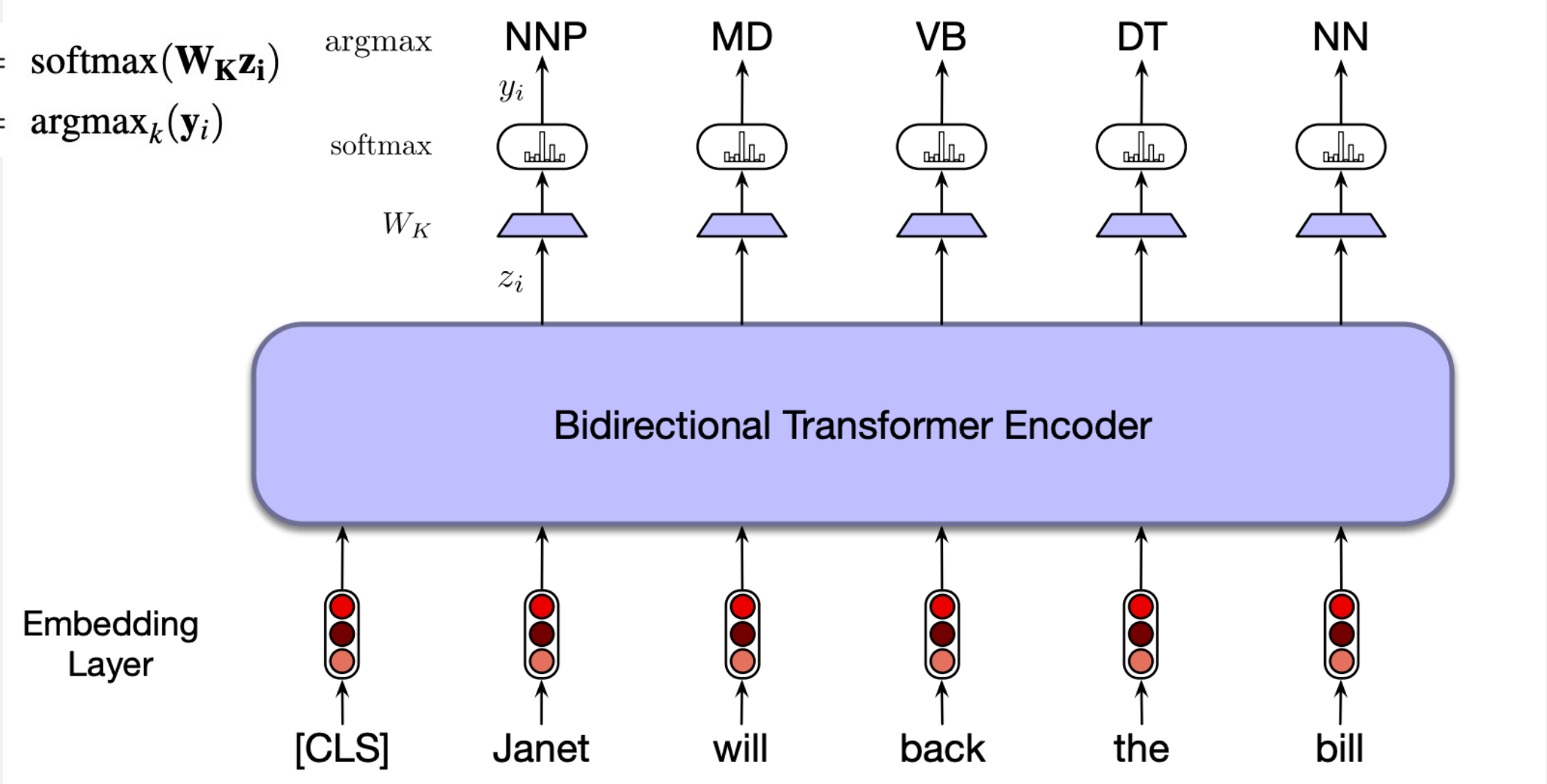
POS标记、基于BIO的命名实体识别

对每一个token进行词性标记，因此：

- 对每个token对应的最终层输出向量，
- 经过一个可学习的权重矩阵 $\mathbf{W_K} \in \mathbb{R}^{k \times d_h}$
- 进行softmax (标签个数为k)
- 经过argmax预测当前标签
- 最后根据真实标签构造损失函数，以驱动权重更新

序列标记(POS标记)

$$\mathbf{y_i} = \text{softmax}(\mathbf{W_K z_i})$$
$$\mathbf{t_i} = \text{argmax}_k(\mathbf{y}_i)$$



序列标记(命名实体识别)

[$_{LOC}$ **Mt. Sanitas**] is in [$_{LOC}$ **Sunshine Canyon**]

Mt.	*Sanitas*	*is*	*in*	*Sunshine*	*Canyon*	*.*
B-LOC	I-LOC	O	O	B-LOC	I-LOC	O

WordPiece生成的tokens

```
'Mt', '.', 'San', '##itas', 'is', 'in', 'Sunshine', 'Canyon' '.'
```

BERT在不同任务上的实验结果

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

微调：基于Span的应用

面向Span的应用介于token级和序列级应用的中间。

- 即生成或操作，序列中连续的tokens

常见的操作包括：

- 包括识别感兴趣的Span，
- 根据标记方案对Span进行分类
- 确定Span间的关系等

相关的应用有：

- 命名实体识别
- 问答
- 句法解析
- 语义角色标记
- 共指消解

微调：基于Span的应用

输入序列x ： $(x_1, x_2, \ldots, x_T)$

Span 起点为i，终点为j : 1≤i≤j ≤T

所有可能的Span个数为 $\frac{T(T-1)}{2}$

实际应用中，Span的长度不超过L ： $j - i < L$

所有有效的Span集合 S(x)

微调：基于Span的应用

第一步：使用上下文输入向量为所有的Span生成表达

该表达包含两部分：

1. Span的边界表达
2. 内容的总结表达

将两个表达进行拼接(concatenation)，生成统一的Span表达

$$g_{ij} = \frac{1}{(j-i)+1} \sum_{k=i}^{j} h_k$$

$$\text{spanRep}_{ij} = [h_i; h_j; g_{i,j}]$$

Span的表达生成

上述方法的缺点：
无法区分词嵌入做为Span起点和Span终点的使用

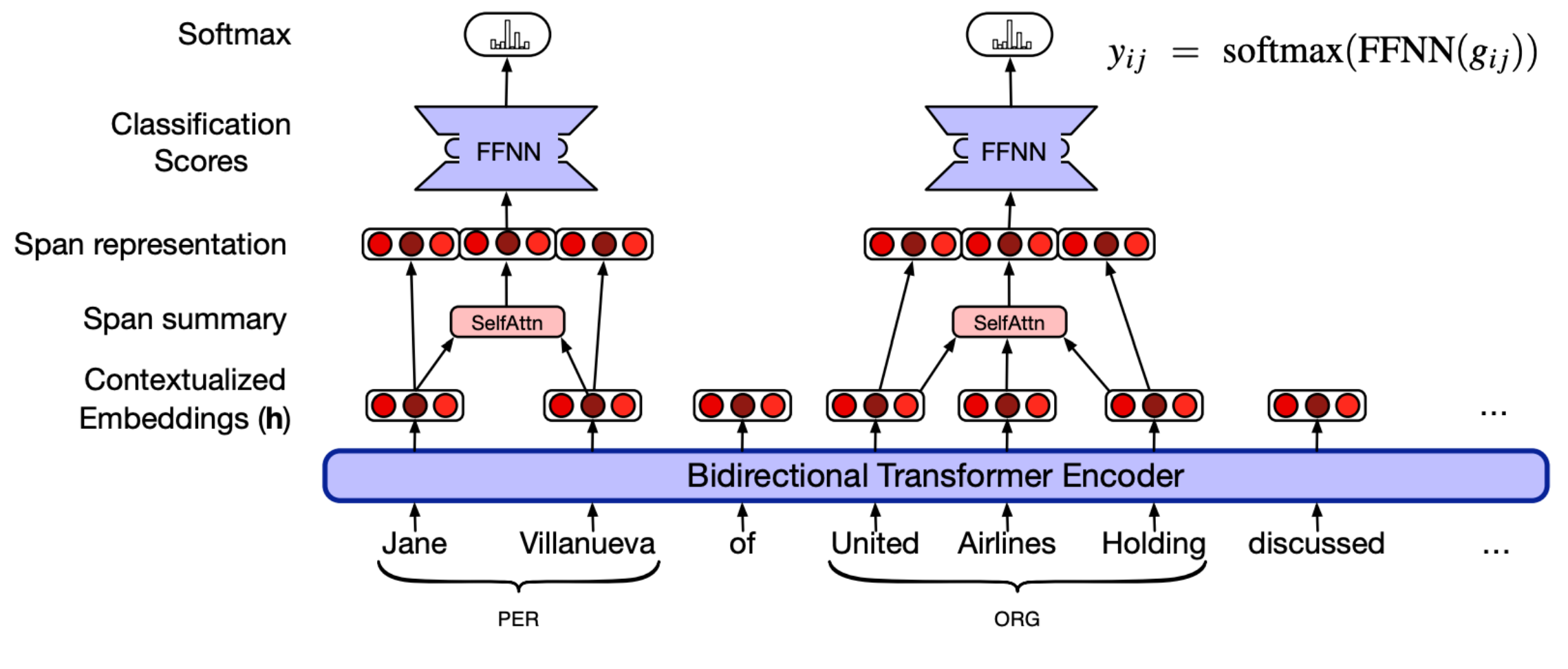
一种更优的方法：

$$
\begin{aligned}
s_i &= \mathrm{FFNN}_{start}(h_i) \\
e_j &= \mathrm{FFNN}_{end}(h_j) \\
\mathrm{spanRep}_{ij} &= [s_i; e_j; g_{i,j}]
\end{aligned}
$$

另外、Span的向量平均无法表示词顺序，因此：

$$\mathbf{g}_{ij} = \mathrm{SelfATTN}(\mathbf{h}_{i:j})$$

微调：基于Span的应用



命名实体分类

微调：基于Span的应用

优点：

1. 避免BIO方法中的标签不匹配问题

2. 可以自适应嵌入命名实体

   - 例如United Airlines 和 United Airlines Holding

Part.04

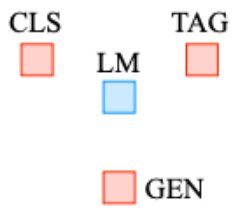
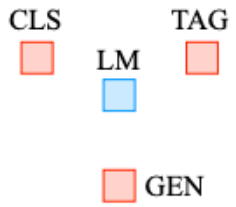
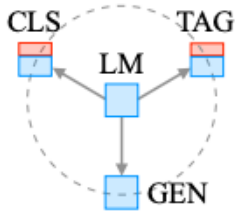
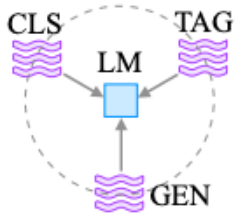
Prompting learning
（提示学习）

提示学习

前面说到NLP领域的一个范式“预训练-微调”，这个范式利用那些 已经在**大规模 未标记 数据**上通过 **自监督学习完成预训练** 的模型，让它们在下游任务上使用 **少量人工标记数据 进行微调**。这个模型其实挺不错，效率高，并且在小样本学习当中十分有效。

在原先“预训练-微调”的方式下，也许是时机不够成熟或是预训练模型本身不够强大，预训练模型不得不“献出自己的一部分”（改动预训练模型自身的参数），去迁就下游任务，以此来成全这门“喜事”。

提示学习

Paradigm	Engineering	Task Relation
a. Fully Supervised Learning (Non-Neural Network)	Features (e.g. word identity, part-of-speech, sentence length)	
b. Fully Supervised Learning (Neural Network)	Architecture (e.g. convolutional, recurrent, self-attentional)	
c. Pre-train, Fine-tune	Objective (e.g. masked language modeling, next sentence prediction)	
d. Pre-train, Prompt, Predict	Prompt (e.g. cloze, prefix)	

提示学习

举一个例子：

> 我们的 预训练模型 是GPT，bert这样的模型，
> 我们的 下游任务 是 句子的情感分类问题。

现在我拿到一句话"我喜欢这本电影"

我要判断"我喜欢这个电影" 这句话的情感（"正面" 或者 "负面"）

提示学习

我们使用prompt的方法去改造下游任务，让我们的预训练模型可以做这个任务，那我们可以将这句话变成一个完型填空。

> 在"我喜欢这本电影"这个输入后面，
> 加上一个模板"**[X]整体上来看，这是一个 [Z] 的电影**"
> 即"我喜欢这个电影，整体上来看，这是一个 __ 的电影"

在这里面，[Z]是我们预训练模型要预测的内容，"我喜欢这个电影，整体上来看，这是一个[Z]的电影"这样完形填空式的输入是预训练模型喜欢且擅长做的输入形式。

提示学习

之后，我们给出两个选项，让模型预测。

A：无聊的 B：有趣的

其中选项A就是无聊的对应的是负面情感，B有趣的对应的就是正面情感。

Z=A -> X=y+ Z=B -> X=y-

就这样，一个下游情感分类的任务改造成了预训练模型可以处理的形态。

提示学习

但是在这里，一个合适的prompt（提示）对于模型的效果至关重要，像是提示的长度提示词汇的位置等等，一点prompt的微小差别，可能会造成效果的巨大差异。

比如，我们离谱一点，你在上面的话中加一个但字，"我喜欢这本电影，但这本电影是___"，很明显预测出来的东西就完全相反了。

Prompt	P@1
[X] is located in [Y]. *(original)*	31.29
[X] is located in which country or state? [Y].	19.78
[X] is located in which country? [Y].	31.40
[X] is located in which country? In [Y].	51.08

完形填空型-情感分类

input： " I love this movie "
output： " positive/negative "

模板 ： " [X] Overall, it was a [Z] movie "
修改后的输入：" I love this movie. Overall it was a [Mask] movie. "

输入文本

预测词(可映射到分类标签pos/neg)

提示 – 前缀提示型

前缀提示：整个输入都在需预测的[z]之前。

同一个输入，可以用于不同的任务，通过模板对任务进行提示，输出结果

input：“How to understand Changing lake movie.”

翻译任务模板：“English: How to understand Changing lake movie. French:[Z]”
问答任务模板：“Question: How to understand Changing lake movie. Answer:[Z]”

提示 – 前缀提示型

Type	Task	Input ([X])	Template	Answer ([Z])
Text CLS	Sentiment	I love this movie.	[X] The movie is [Z].	great fantastic ...
	Topics	He prompted the LM.	[X] The text is about [Z].	sports science ...
	Intention	What is taxi fare to Denver?	[X] The question is about [Z].	quantity city ...
Text-span CLS	Aspect Sentiment	Poor service but good food.	[X] What about service? [Z].	Bad Terrible ...
Text-pair CLS	NLI	[X1]: An old man with ... [X2]: A man walks ...	[X1]? [Z], [X2]	Yes No ...
Tagging	NER	[X1]: Mike went to Paris. [X2]: Paris	[X1][X2] is a [Z] entity.	organization location ...
Text Generation	Summarization	Las Vegas police ...	[X] TL;DR: [Z]	The victim ... A woman ... ...
	Translation	Je vous aime.	French: [X] English: [Z]	I love you. I fancy you. ...

连续提示 (软提示)

可以是真实token的序列，也可以是虚拟token(例如使用数字ID进行表示)

- 这些虚拟token会被嵌入到一个连续空间中
- 或直接生成连续向量

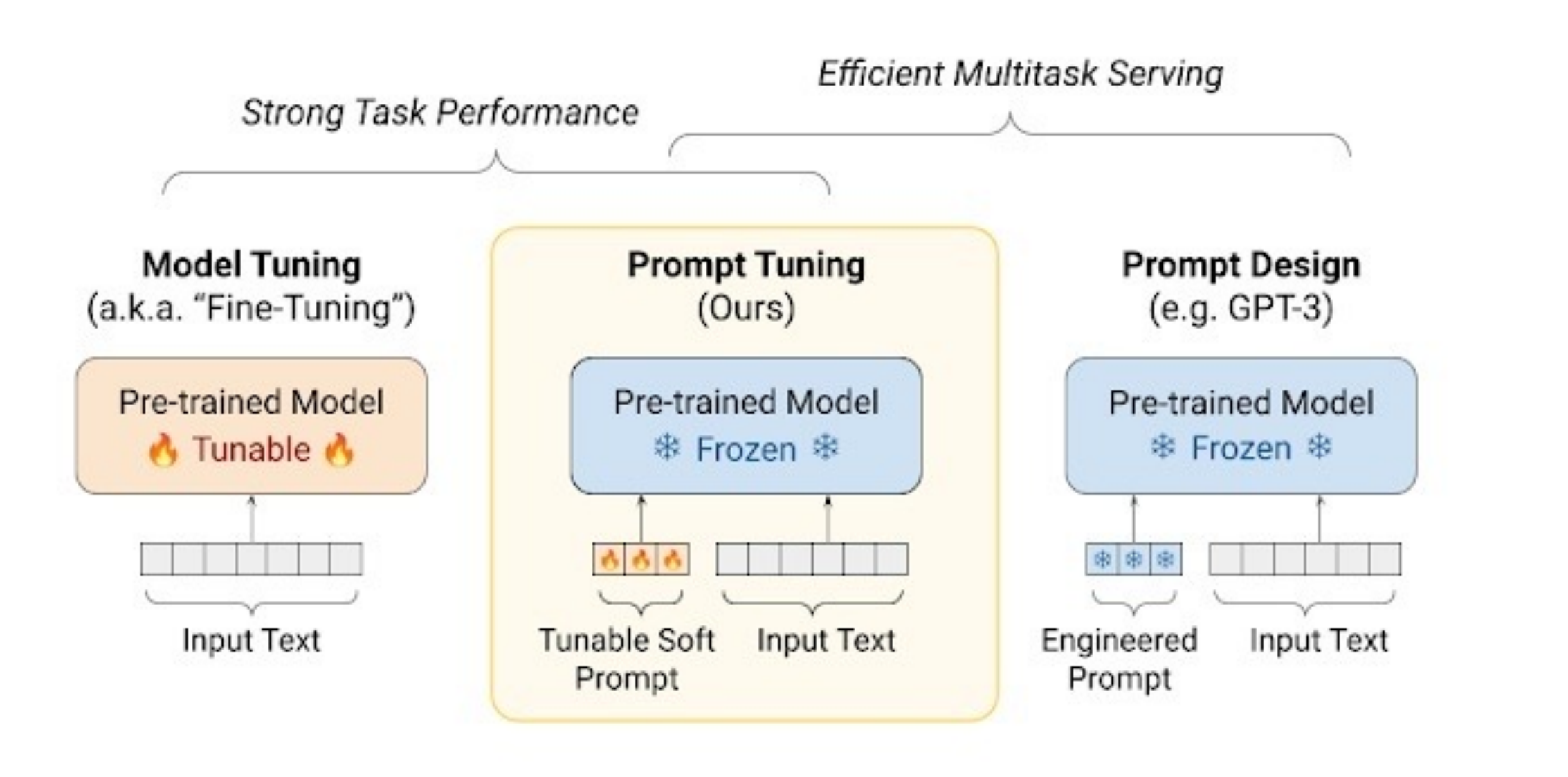
即不受限于使用人工干预(单词级别)的提示模板，
而直接在向量空间中(词向量级别)进行提示。

优势：

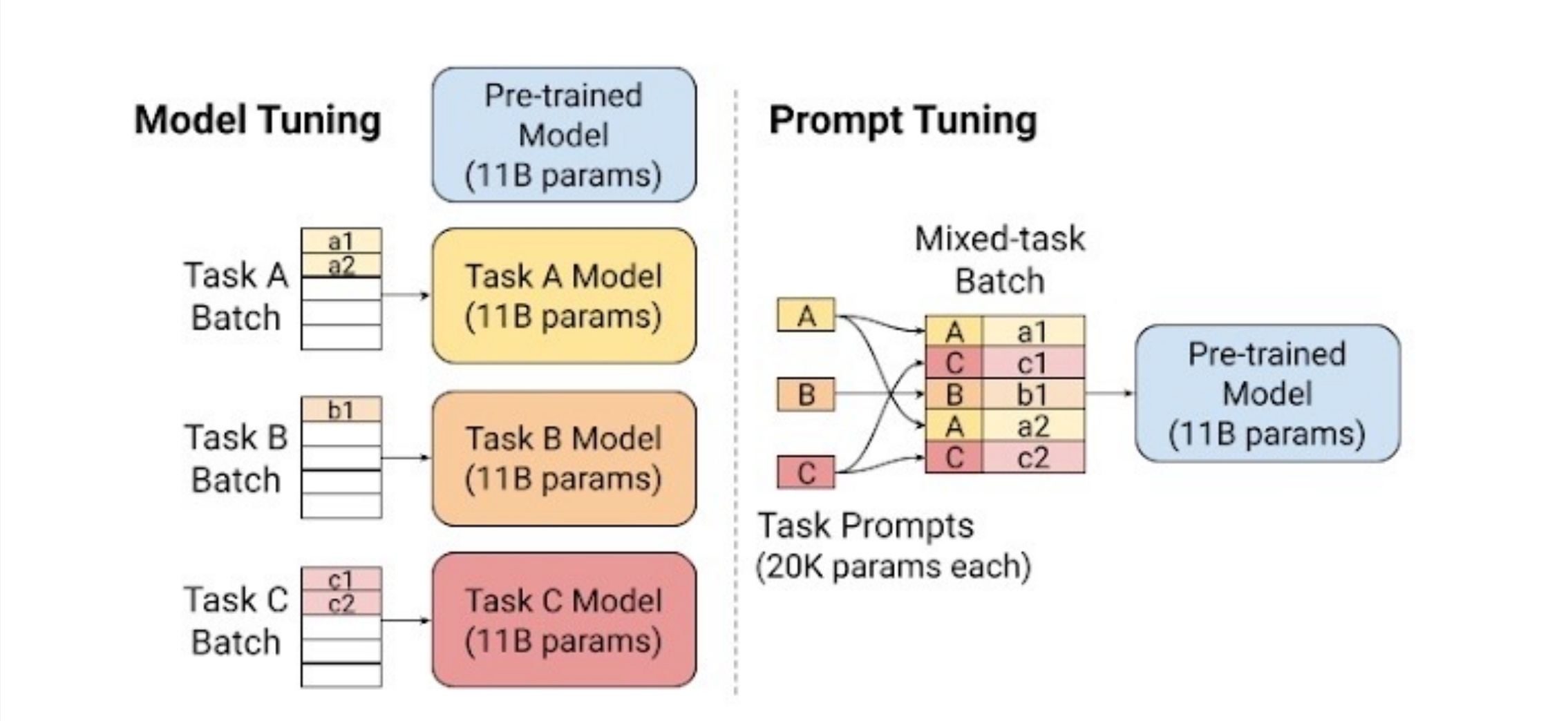
不受限于单词或者单词向量，可以生成任意连续向量的模板

不受限于预训练语言模型的参数，有自身可学习的参数

连续提示 (软提示)


Strong Task Performance
Efficient Multitask Serving
Model Tuning
(a.k.a. "Fine-Tuning")
Pre-trained Model
🔥 Tunable 🔥
Input Text
Prompt Tuning
(Ours)
Pre-trained Model
❄ Frozen ❄
Tunable Soft Prompt
Input Text
Prompt Design
(e.g. GPT-3)
Pre-trained Model
❄ Frozen ❄
Engineered Prompt
Input Text

连续提示 (软提示)


Model Tuning
Pre-trained Model (11B params)
Task A Batch
a1
a2
Task A Model (11B params)
Task B Batch
b1
Task B Model (11B params)
Task C Batch
c1
c2
Task C Model (11B params)
Prompt Tuning
Mixed-task Batch
A
B
C
A a1
C c1
B b1
A a2
C c2
Pre-trained Model (11B params)
Task Prompts (20K params each)

思考

微调和prompt在语言任务中的区别有哪些？

prompt有什么优势？

相较于硬提示模板，连续提示模板有哪些好处？

总结

使用双向编码器(BERT)，为输入生成上下文表达

掩码 - BERT预训练语言模型 (token掩码和Span掩码)

微调 – 对预训练语言模型进行微调，适应不同的下游任务

提示 – 对下游任务进行调整，以适应预训练语言模型

谢 谢