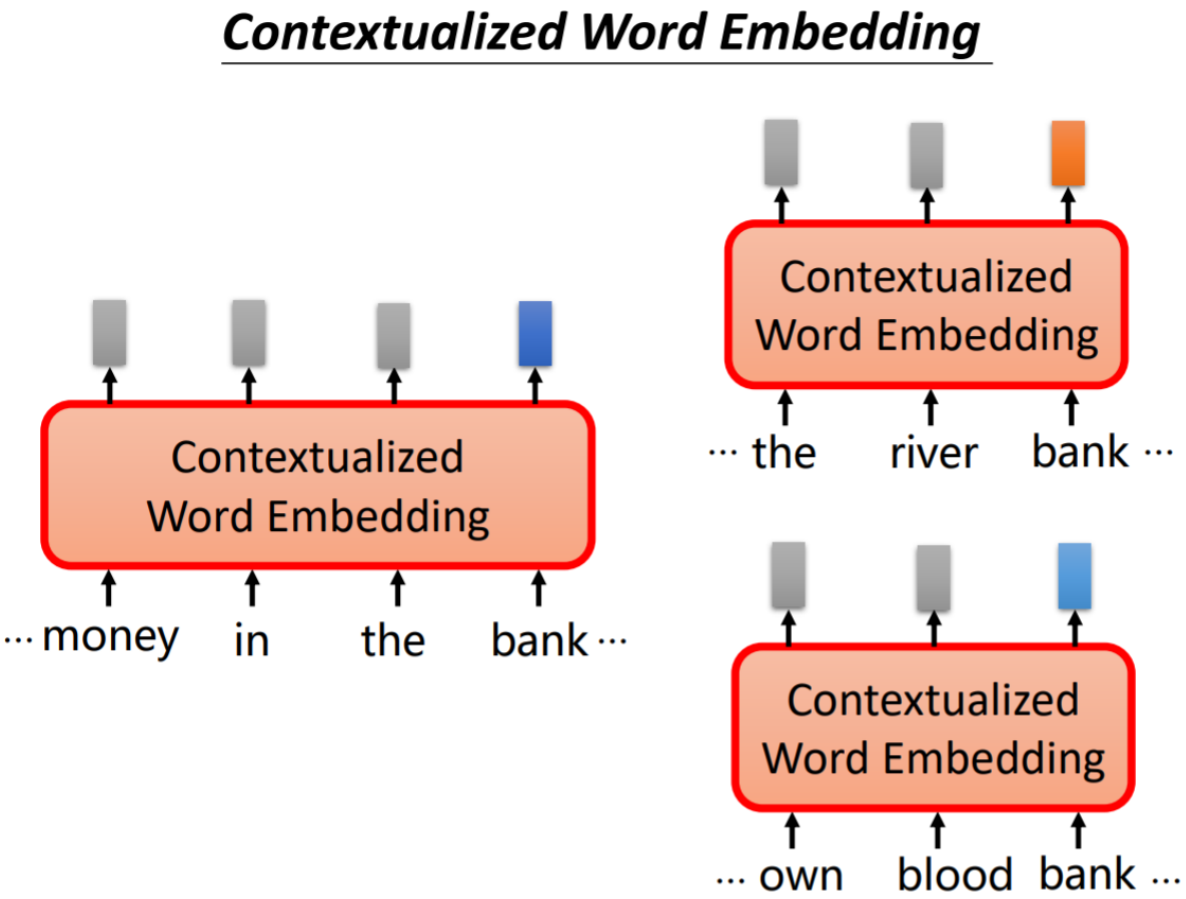
Bert

0、前言

同一个单词可能有不同的语义。



每一个单词token都有自己的embedding。如果两个两个token的意思接近，那么他们的embedding向量会距离比较近。

1、BERT模型概述

BERT，全称是 Bidirectional Encoder Representation from Transformers， Google 于 2018 年发表，它在 11 项自然语言处理任务中均表现出惊人的成绩。BERT即双向Transformer的Encoder，因为decoder是不能获要预测的信息的。模型的主要创新点都在pre-train方法上，即用了Masked LM和Next Sentence Prediction两种方法分别捕捉词语和句子级别的representation。

要理解 Bert，5 个关键词帮你理解其思想，分别是 **Pre-training、Deep、Bidirectional、Transformer、Language Understanding。**

Deep：Bert 与 Transformer 不同，Bert 的神经网络层更深，意味着它能更准确表达语义的理解，提取更多特征。

Bidirectional：双向的，BERT 被设计成一个深度双向模型，使得神经网络更有效地从第一层本身一直到最后一层捕获来自目标词的左右上下文的信息。意思是表示它在处理一个词的时候，能考虑到该词前面和后面单词的信息。

传统上，我们要么训练语言模型预测句子中的下一个单词(GPT 中使用的从右到左的上下文)，要么训练语言模型预测从左到右的上下文。这使得我们的模型容易由于信息丢失而产生错误。

Transformer：Bert 是基于 Tranformer 的深度双向语言表征模型，也就是利用 Transformer 结构构造了一个多层双向的Encoder 网络。它的特点之一就是所有层都联合上下文语境进行预训练。

Bert 的目标是生成预训练语言模型，所以只需要 Encoder 机制。Transformer 的 Encoder 是一次性读取整个文本序列，而不是从左到右或者从右到左按顺序读取。

Pre-training：pre-training 的意思是，作者认为，确实存在通用的语言模型，先用大量数据预训练一个通用模型，然后再微调模型，使其适用于下游任务。为了区别于针对语言生成的 Language Model，作者给通用的语言模型，取了一个名字，叫语言表征模型 Language Representation Model。

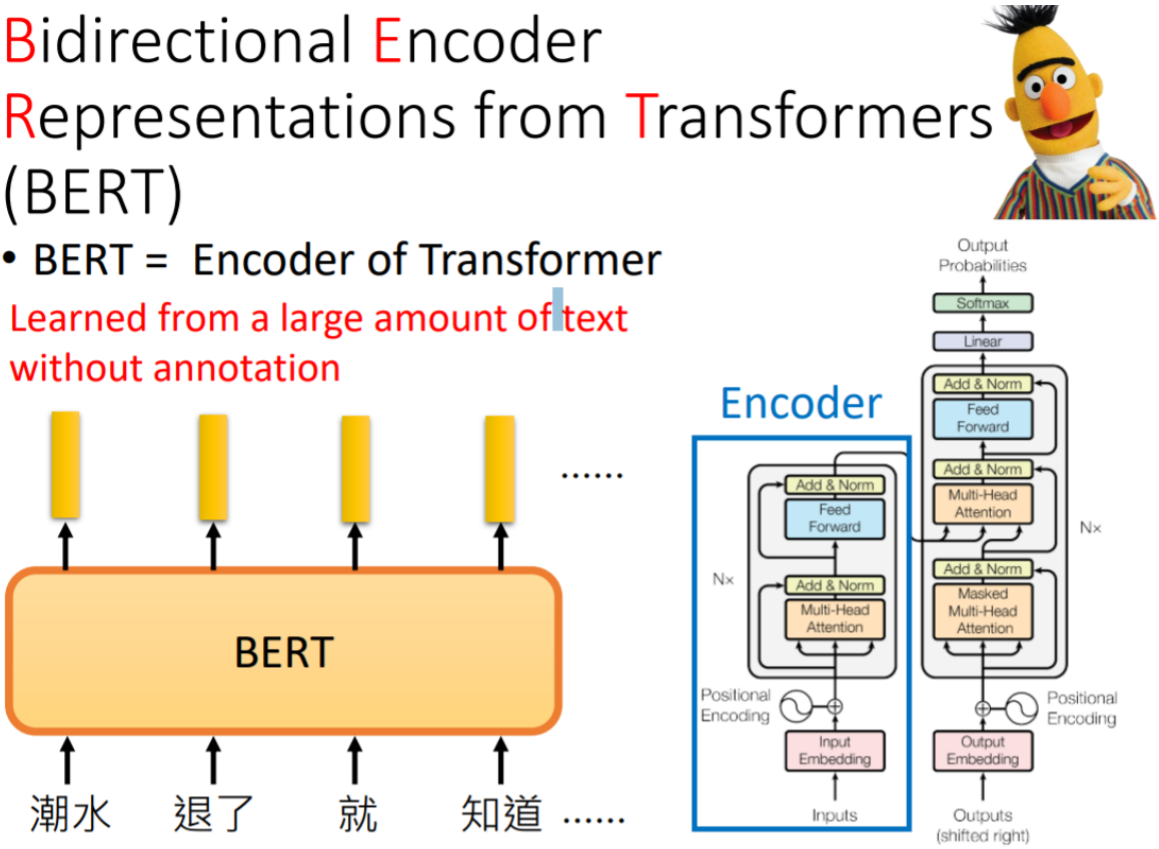
深度学习就是表征学习，大多只在预训练表征微调的基础上加一个线性分类器作为输出就可以完成下游任务。

Language Understanding：Bert 是一个语言表征模型，能实现语言表征目标训练，通过深度双向 Transformer 模型达到语义理解的目的。

整合以上特点，我们就可以很直观的理解 Bert，Bert 是一个用 Transformer 作为特征抽取器的深度双向预训练语言理解模型。Bert 就是一个预训练模型，利用双向 Transformer，通过大量数据训练一个语言表征模型，这是一个通用模型，通过对其微调来适用下游任务，包括分类，回归，机器翻译，问答系统等等任务。

2、BERT模型核心点

1、BERT的架构

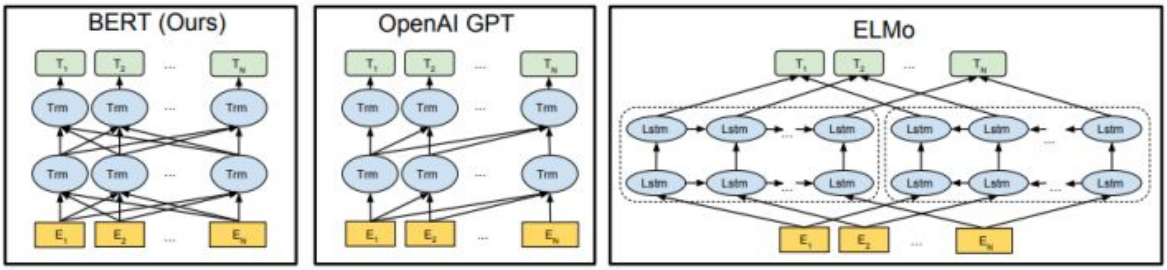


（注意如果要做中文的embedding的话，最好用单字，而不是词，词的组合太多，维度会非常大，而字的数量是有限的）

BERT的模型架构基于了Transformer，实现了多层双向的Transformer编码器，也就是蓝色框出来的部分。文中有两个模型，一个是1.1亿参数的base模型，一个是3.4亿参数的large模型。里面所设置的参数如下：

Model	Transformer层数(L)	Hidden units(H)	self-attention heads(A)	总参数
BERT(base)	12	768	12	1.1亿
BERT(large)	24	1024	16	3.4亿

BERT的模型结构如下最左图：



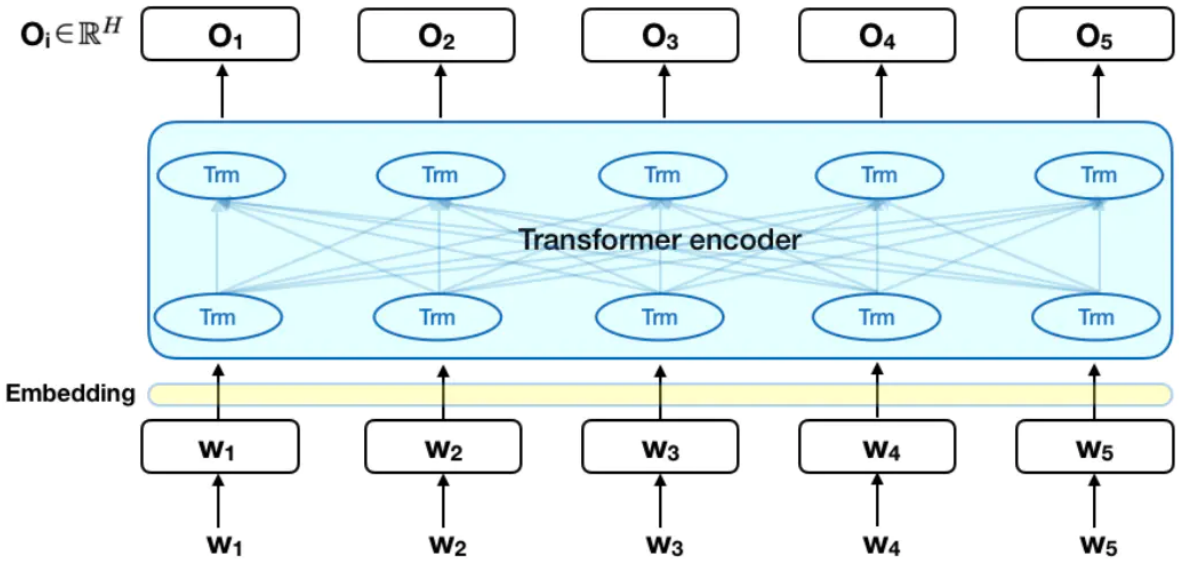
Bert 采用了 Transformer Encoder，也就是每时每刻的 Attention 计算都能够得到全部时刻的输入。BERT是双向的Transformer block连接

OpenAI GPT 采用 Transformer 的 Decoder，每个时刻的 Attention 计算只能依赖于该时刻前的所有时刻的输入，因为 OpenAI GPT 是单向语言模型。

ELMO 则采用的是 LSTM，这个模型虽然可以学习到前后语义，但是输出依赖于前面的输入，决定了 EMLO 的网络层数不会太多，会造成大量耗时，这就决定了 ELMO 提取的特征有限。

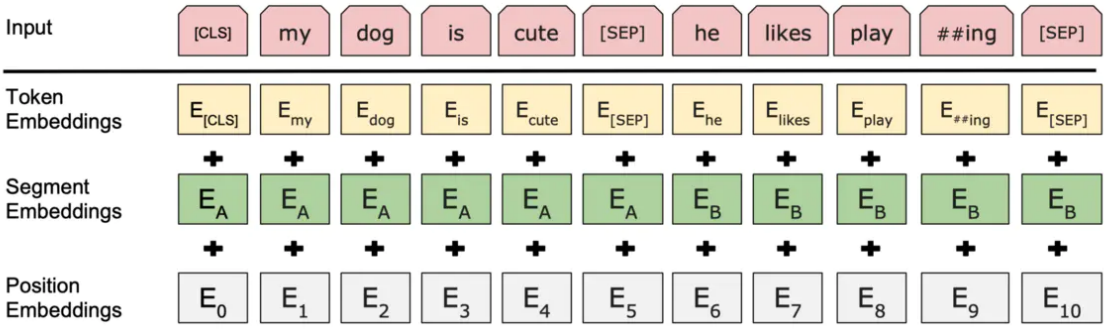
（不明白）对比ELMo，虽然都是"双向"，但目标函数其实是不同的。ELMo是分别以 $P(w_i|w_1,...w_{i-1})$ 和 $P(w_i|w_{i+1},...w_n)$ 作为目标函数，独立训练处两个representation然后拼接，而BERT则是以 $P(w_i|w_1,...,w_{i-1},w_{i+1},...,w_n)$ 作为目标函数训练LM。

其中base模型的参数和OpenAI的GPT的参数一致。目的就是为了同GPT的效果进行一个比较。下图是详细模型：



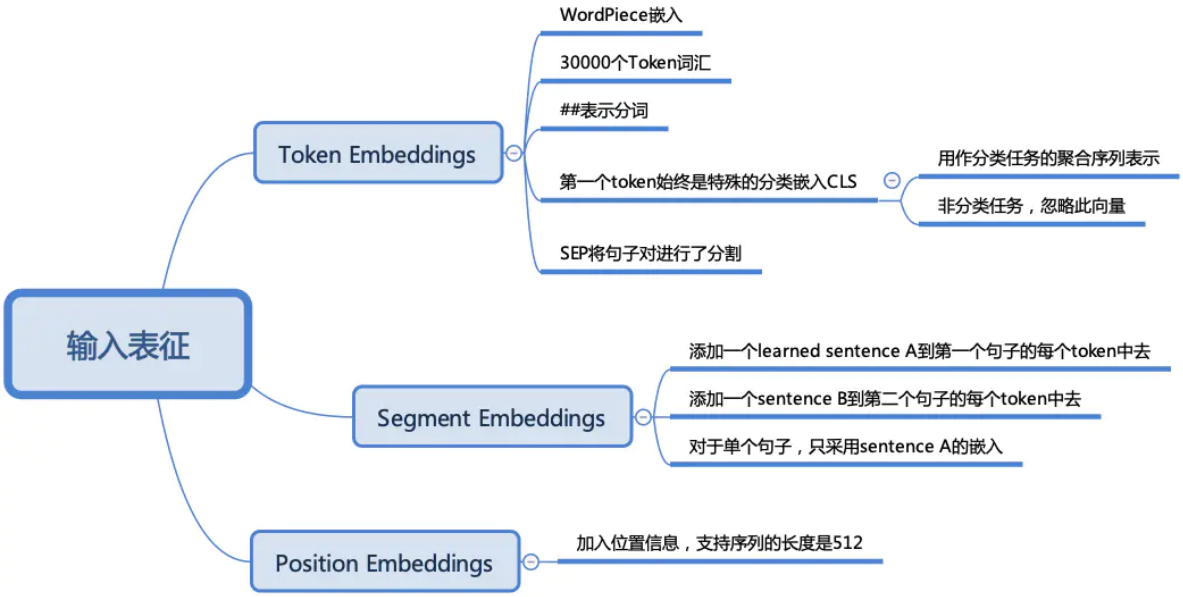
2、BERT的输入表征

下图表示了BERT的输入表征



Bert 的输入 Input 是两个句子："my dog is cute"，"he likes playing"。首先会在第一句开头加上特殊 Token [CLS] 用于标记句子开始，用 [SEP] 标记句子结束。

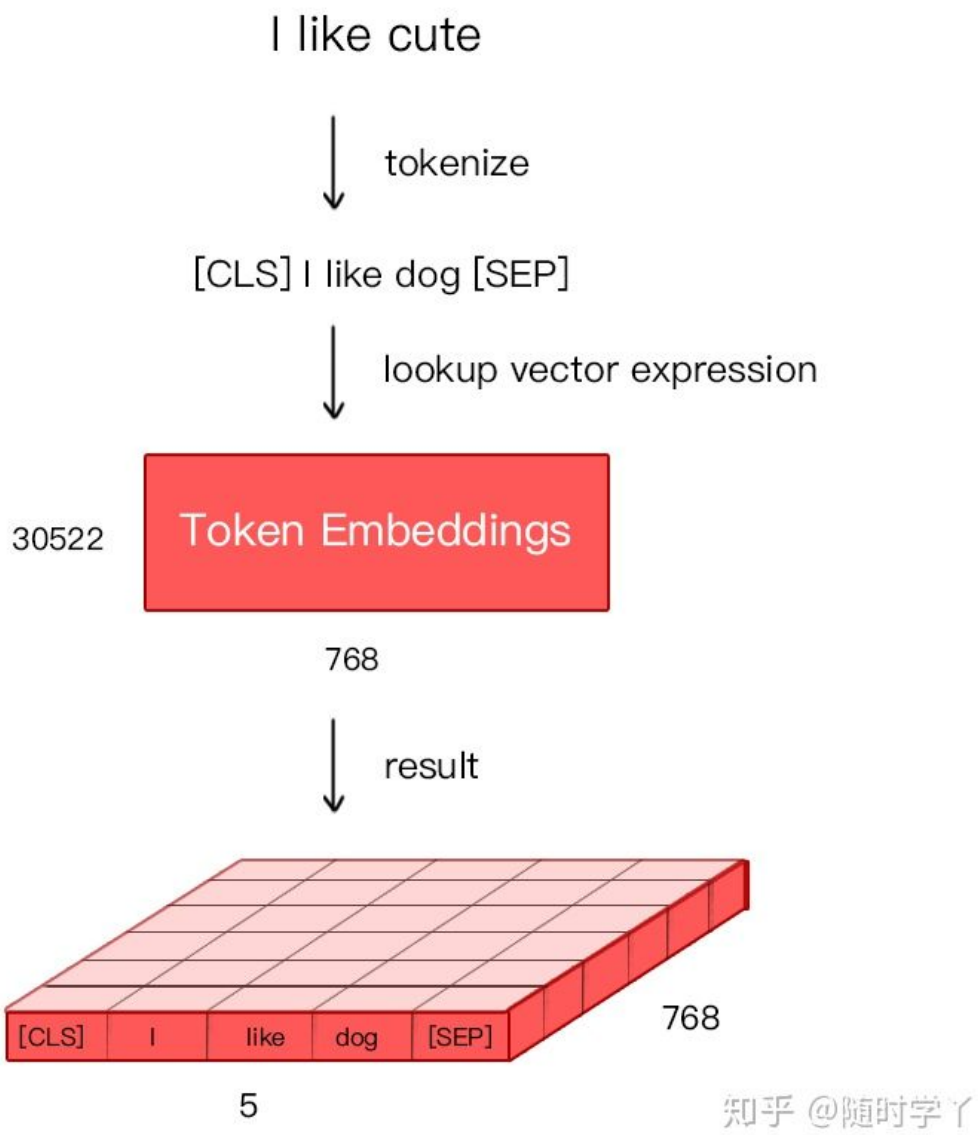
下面的思维导图说明了各个部分的作用是什么：



词的Token Embedding:

作用：BERT将输入文本中的每一个词（token)送入token embedding层，通过建立字向量表将每个字转换成一个一维向量。英文词汇会做更细粒度的切分，比如playing 或切割成 play 和 ##ing，切割成更细粒度的word piece是为了解决未登录词的常见方法。在BERT中，每个词会被转换成768维的向量表示。

假设输入文本是 “I like cute”。下面这个图展示了 Token Embeddings 层的实现过程:



输入文本在送入token embeddings 层之前要先进行tokenization处理。此外，两个特殊的token会被插入到tokenization的结果的开头 ([CLS])和结尾 ([SEP]) 。它们视为后面的分类任务和划分句子对服务的。

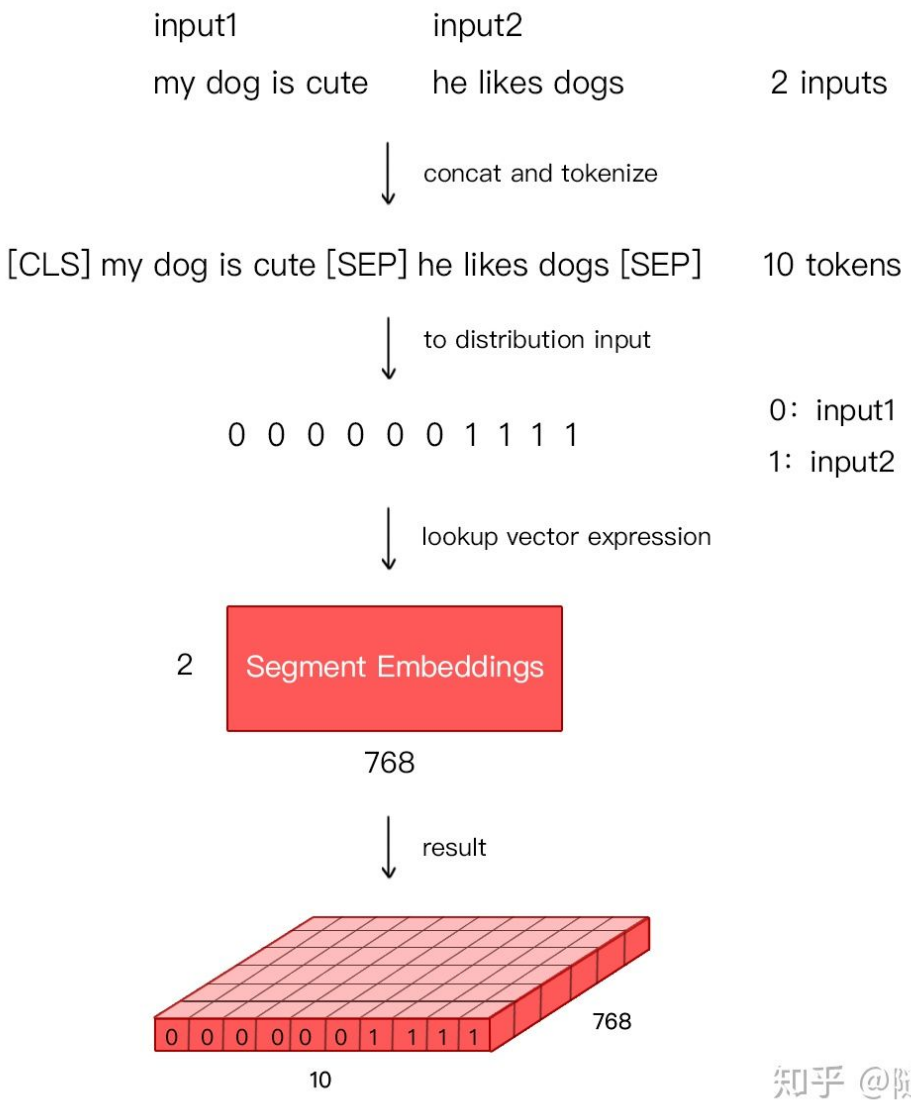
tokenization使用的方法是WordPiece tokenization. 这是一个数据驱动式的tokenization方法，旨在权衡词典大小和oov词的个数。这种方法把例子中的“strawberries”切分成了“straw” 和“berries”。这种方法的详细内容不在本文的范围内。有兴趣的读者可以参阅 Wu et al. (2016) 和 Schuster & Nakajima (2012)。使用WordPiece tokenization让BERT在处理英文文本的时候仅需要存储30,522 个词，而且很少遇到oov的词。

Token Embeddings 层会将每一个wordpiece token转换成768维的向量。这样，例子中的5个token就被转换成了一个(5, 768) 的矩阵或者是(1, 5, 768)的张量（如果考虑batch_size的话）。

句子的Segment Embeddings:

作用：当BERT处理句子分类问题时，需要判断两个文本是否语义相似，将两个句子拼接在一起送入模型中。其中，Segment Embeddings可以区分一个句子中的两个句子。

如："my dog is cute,he likes dogs"，下面说明了segment embeddings如何帮助BERT区分两个句子:



Segment Embeddings 层只有两种向量表示。前一个向量是把0赋给第一个句子中的各个token, 后一个向量是把1赋给第二个句子中的各个token。如果输入仅仅只有一个句子，那么它的segment embedding就是全0。

位置的Position Embeddings

RNN 能够让模型隐式的编码序列的顺序信息，而Transformers无法编码输入的序列的顺序性。如"I think, therefore I am"，第一个I 和 第二个I 应该有不同的向量表示。加入Position Embeddings可以让BERT理解。

Transformer 中通过植入关于 Token 的相对位置或者绝对位置信息来表示序列的顺序信息。作者测试用学习的方法来得到 Position Embeddings，最终发现固定位置和相对位置效果差不多，所以最后用的是固定位置的，而正弦可以处理更长的 Sequence，且可以用前面位置的值线性表示后面的位置。

偶数位置，使用正弦编码，奇数位置，使用余弦编码。

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

BERT能够处理最长512个token的输入序列，长度超过 512 会被截取，Bert 在各个位置上学习一个向量来表示序列顺序的信息编码进来。这意味着Position Embeddings layer 实际上就是一个大小为 (512, 768) 的lookup表，表的第一行是代表第一个序列的第一个位置，第二行代表序列的第二个位置，以此类推。因此，如果有这样两个句子“Hello world” 和“Hi there”, “Hello” 和“Hi”会由完全相同的position embeddings，因为他们都是句子的第一个词。同理，“world” 和“there”也会有相同的position embedding。

合成

我们已经介绍了长度为n的输入序列将获得的三种不同的向量表示，分别是：

- Token Embeddings， (1, n, 768)，词的向量表示
- Segment Embeddings， (1, n, 768)，辅助BERT区别句子对中的两个句子的向量表示
- Position Embeddings，(1, n, 768)，让BERT学习到输入的顺序属性

这些表示会被按元素相加，用求和的方式得到一个大小为(1, n, 768)的合成表示。这一表示就是BERT编码层的输入了。

3、BERT中最核心的部分

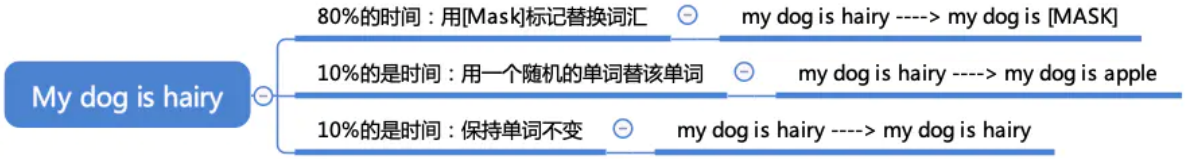
（1）Masked Language Model(MLM)

Maked LM 是为了解决单向信息问题，现有的语言模型的问题在于，没有同时利用双向信息，如 ELMO 号称是双向LM，但实际上是两个单向 RNN 构成的语言模型的拼接，由于时间序列的关系，RNN模型预测当前词只依赖前面出现过的词，对于后面的信息无从得知。

MLM：随机屏蔽掉部分输入token，然后再去预测这些被屏蔽掉的token。

Bert 预训练过程就是模仿我们学习语言的过程，要准确的理解一个句子或一段文本的语义，就要学习上下文关系，从上下文语义来推测空缺单词的含义。而 Bert 的做法模拟了英语中的**完形填空**，随机将一些单词遮住，让 Bert 模型去预测这个单词，以此达到学习整个文本语义的目的。

例子：句子= my dog is hairy， 选择的token是hairy。执行的流程为：



随机 mask 预料中 15% 的 Token，然后预测 [MASK] Token，与 masked token 对应的最终隐藏向量被输入到词汇表上的 softmax 层中。这虽然确实能训练一个双向预训练模型，但这种方法有个缺点，因为在预训练过程中随机 [MASK] Token 由于每次都是全部 mask，预训练期间会记住这些 MASK 信息，但是在fine-tune期间从未看到过 [MASK] Token，导致预训练和 fine-tune 信息不匹配。

而为了解决预训练和 fine-tune 信息不匹配，Bert 并不总是用实际的 [MASK] Token 替换 masked 词汇。操作方法如上图。

这里实现的时候有两个缺点
缺点1：预训练与微调之间的不匹配，因为微调期间是没有看到[MASK] Token。

Solution：不是总用实际的[Mask]token替换被“masked”的词汇，而是采用训练数据生成器随机去选择15%的token。

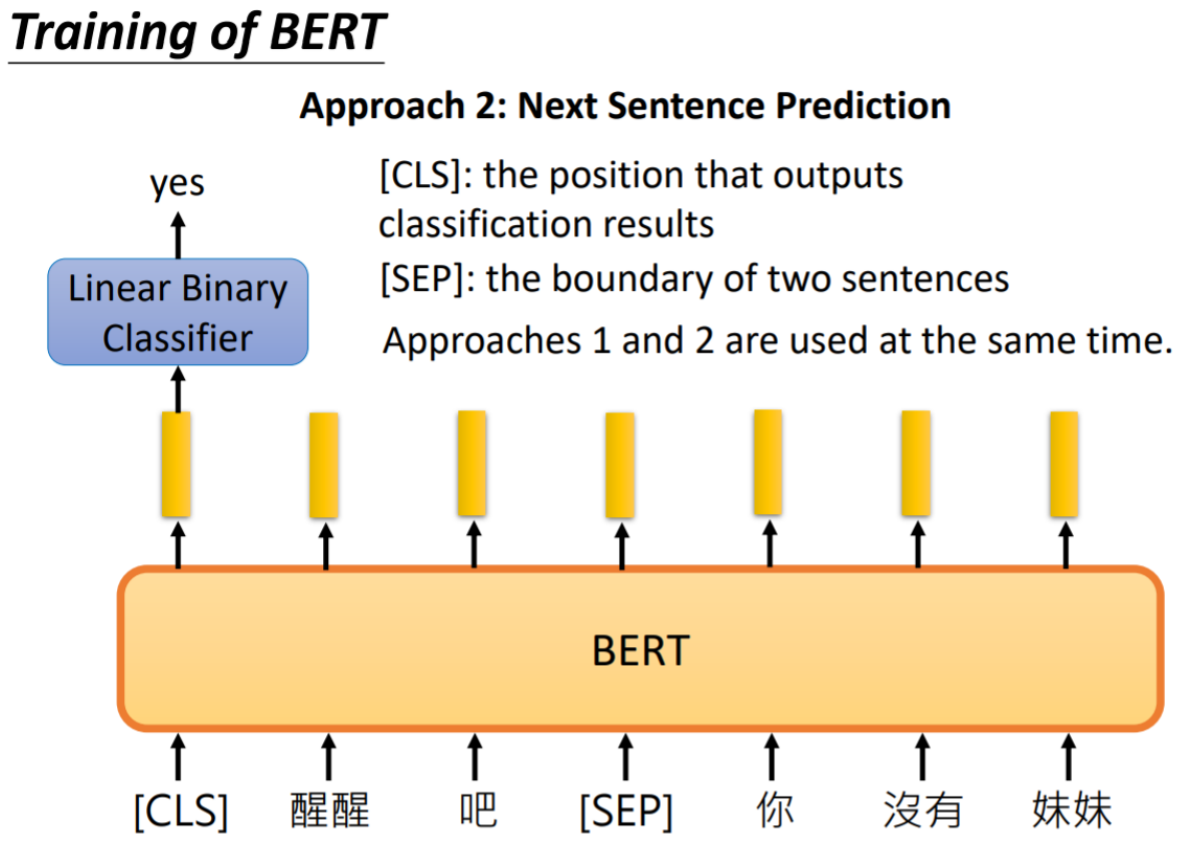
Transformer不知道它将被要求预测哪些单词或哪些单词已被随机单词替换，因此它被迫保持每个输入词块的分布式语境表征。此外，因为随机替换只发生在所有词块的1.5%(即15%的10%)，这似乎不会损害模型的语言理解能力。
缺点2：每个batch只预测了15%的token，这说明了模型可能需要更多的预训练步骤才能收敛。

(2) Next Sentence Prediction

在许多下游任务中，如问答系统 QA 和自然语言推理 NLI，都是建立在理解两个文本句子之间的关系基础上，这不是语言模型能直接捕捉到的。

为了训练一个理解句子关系的模型，作者提出 Next Sentence Prediction，也即是预训练一个下一句预测的二分类任务，这个任务就是每次训练前都会从语料库中随机选择句子 A 和句子 B，50% 是正确的相邻的句子，50% 是随机选取的一个句子，这个任务在预训练中能达到 97%-98% 的准确率，并且能很显著的提高 QA 和 NLI 的效果。

```
Input = [CLS] the man went to [MASK] store [SEP]
        he bought a gallon [MASK] milk [SEP]
Label = IsNext
•
Input = [CLS] the man [MASK] to the store [SEP]
        penguin [MASK] are flight ##less birds [SEP]
Label = NotNext
```

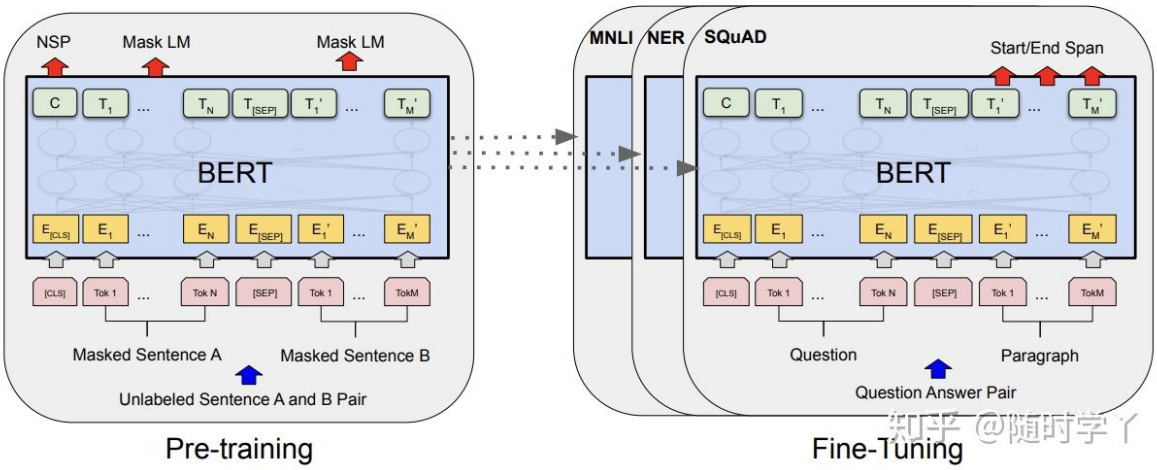


在句子开头插入一个特殊的token：cls 。告诉bert需要做分类模型，判断两个句子是不是用在用一时间的。

关于为什么把cls放在开头，因为bert里边不是RNN，它里边是一个Transformer的Encoder，不管cls放在什么位置，输出都是并行一起输出的。

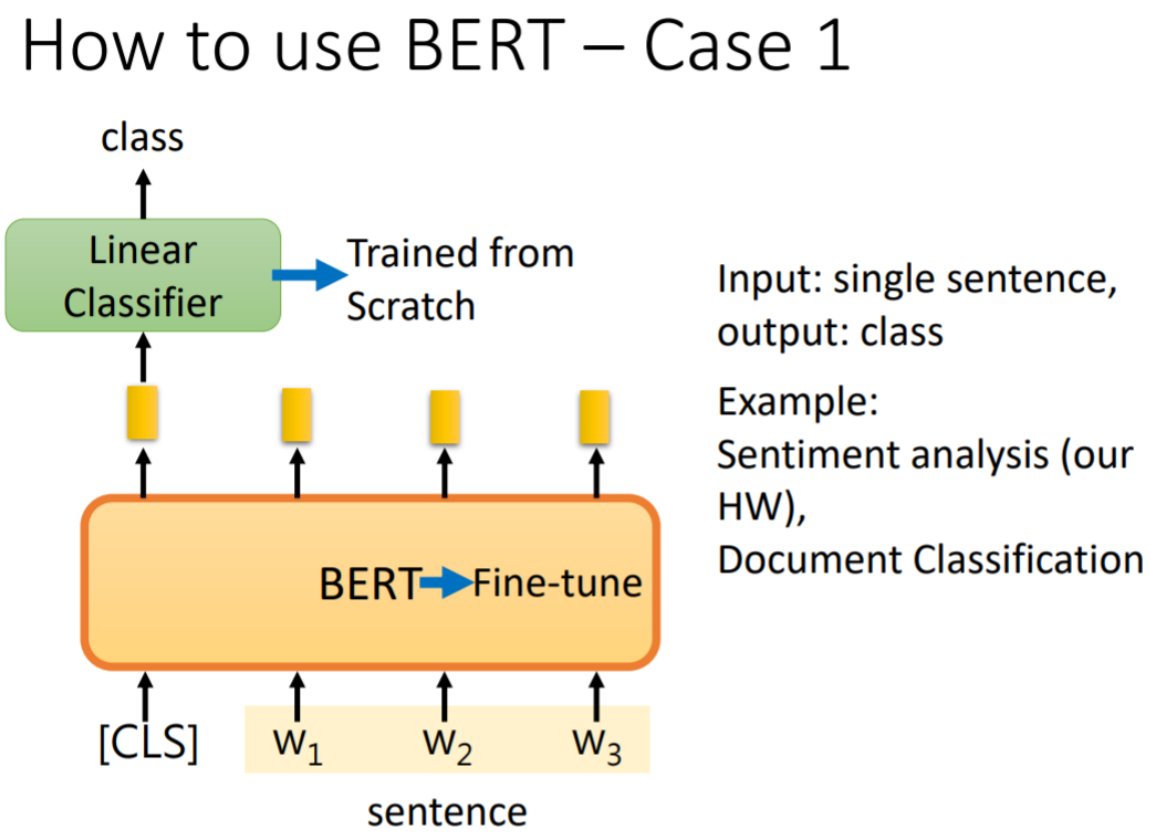
模型通过对 Masked LM 任务和 Next Sentence Prediction 任务进行联合训练，使模型输出的每个字 / 词的向量表示都能尽可能全面、准确地刻画输入文本（单句或语句对）的整体信息，为后续的微调任务提供更好的模型参数初始值。

Bert如何实现fine-tune



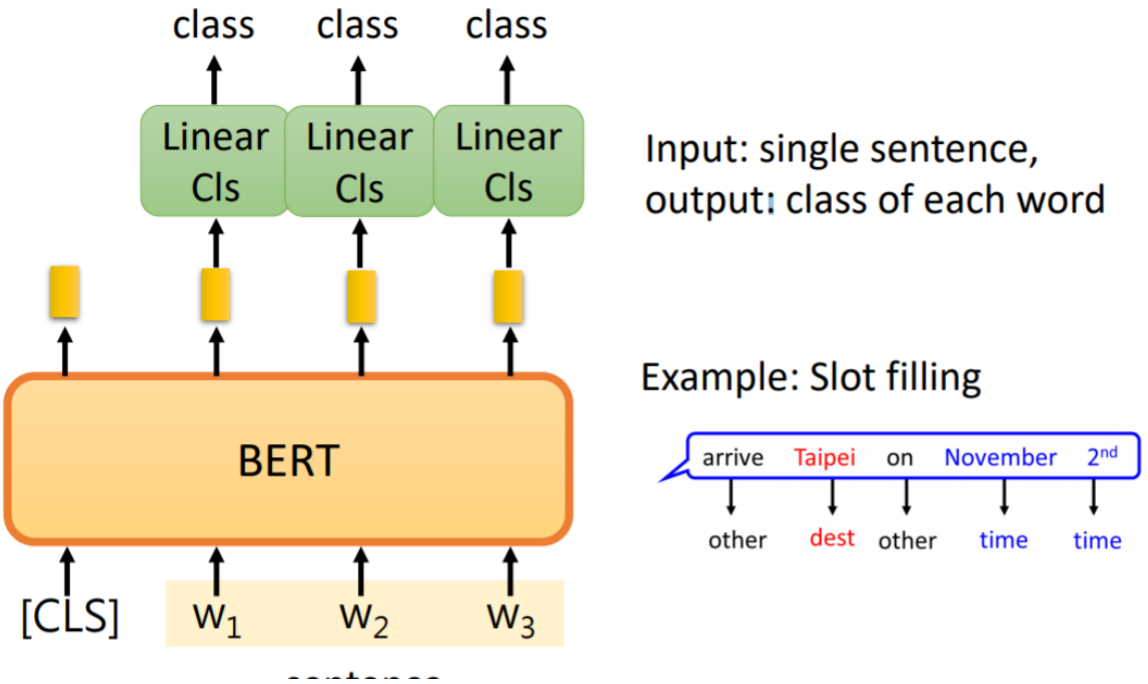
fine-tune 就是指在已经训练好的语言模型基础上，使用有标签的数据对参数进行调整，使其更好的适用于下游任务。如对于分类问题在语言模型基础上加一层 softmax 网络，然后再新的预料上重新训练进行 fine-tune。

4、如何使用BERT



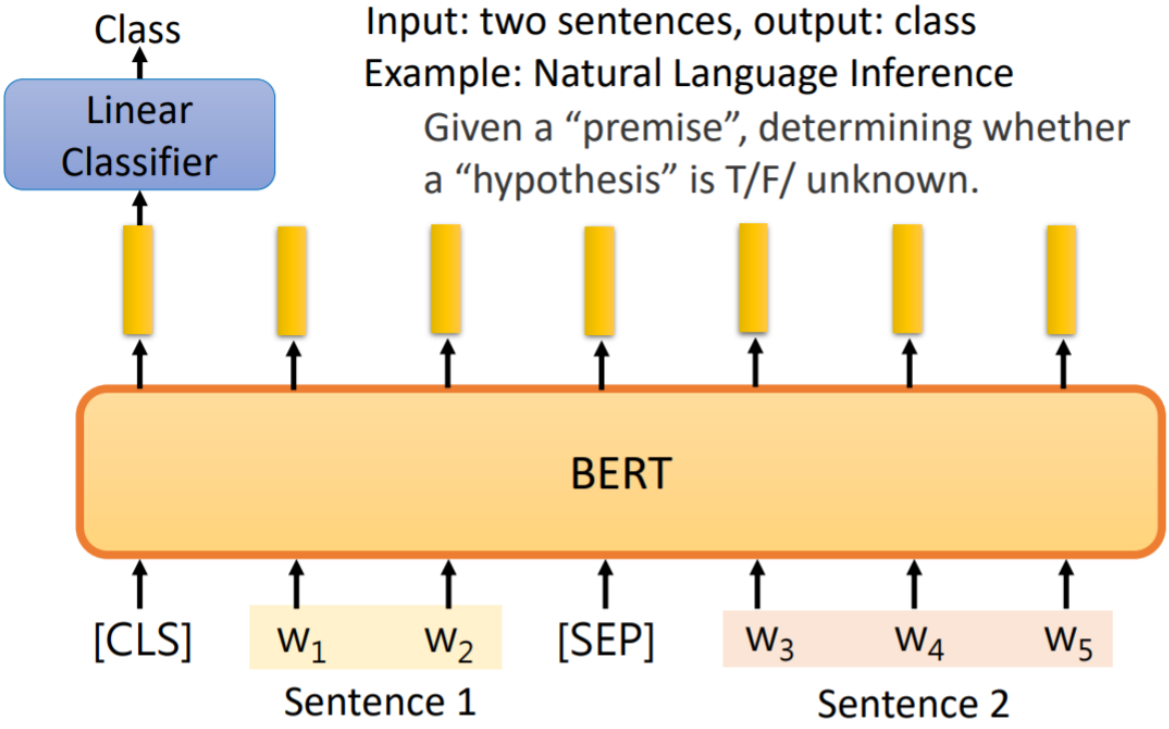
比如做情感分类，输入一个句子，在句子的开头加入CLS，在CLS的输出接入一个Linear Classifier, 这个分类器是从头开始学，而BERT是只需要微调参数就好，BERT多数参数已经学的很好了。

How to use BERT – Case 2



输入一个句子，把每个单词对应的输出都接一个分类器，然后从头开始训练。

How to use BERT – Case 3



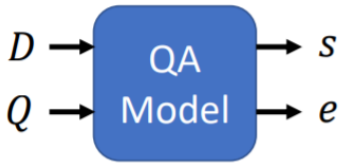
给两个句子，在第一个句子条件下，第二个句子是对的，错的，不知道

How to use BERT – Case 4

- Extraction-based Question Answering (QA) (E.g. SQuAD)

Document: $D = \{d_1, d_2, \cdots, d_N\}$

Query: $Q = \{q_1, q_2, \cdots, q_N\}$



output: two integers (s, e)

Answer: $A = \{q_s, \cdots, q_e\}$

In meteorology, precipitation is any product of the condensation of 17 spheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain 77 atte 79 cations are called "showers".

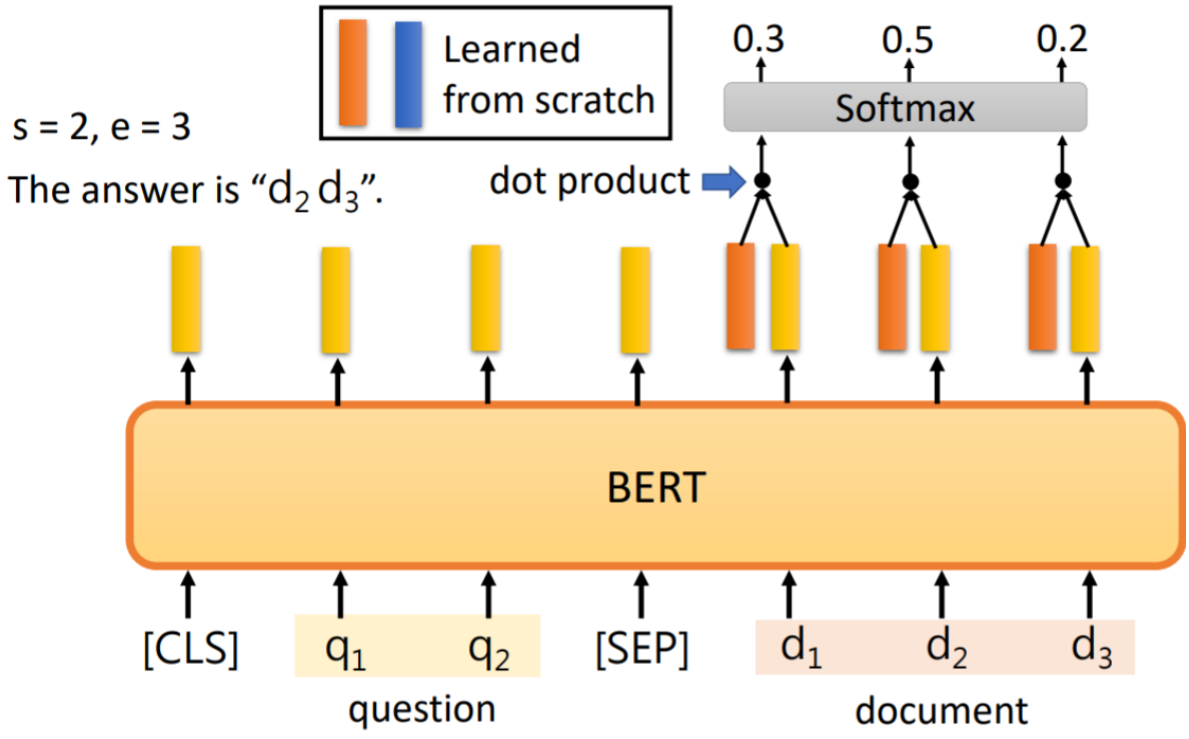
What causes precipitation to fall?
gravity $s = 17, e = 17$

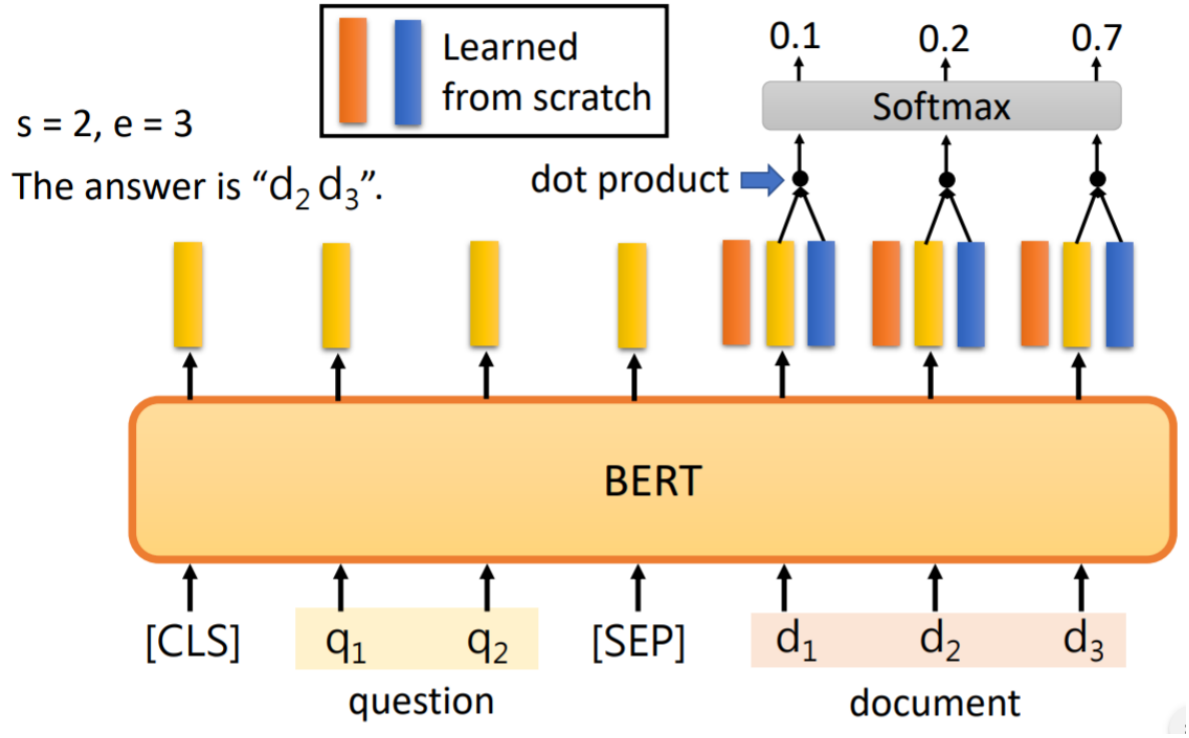
What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud $s = 77, e = 79$

给定一个文章，回答系统。

输入D是文章，Q是问题。输出是s，e分别是文章的下标是答案。



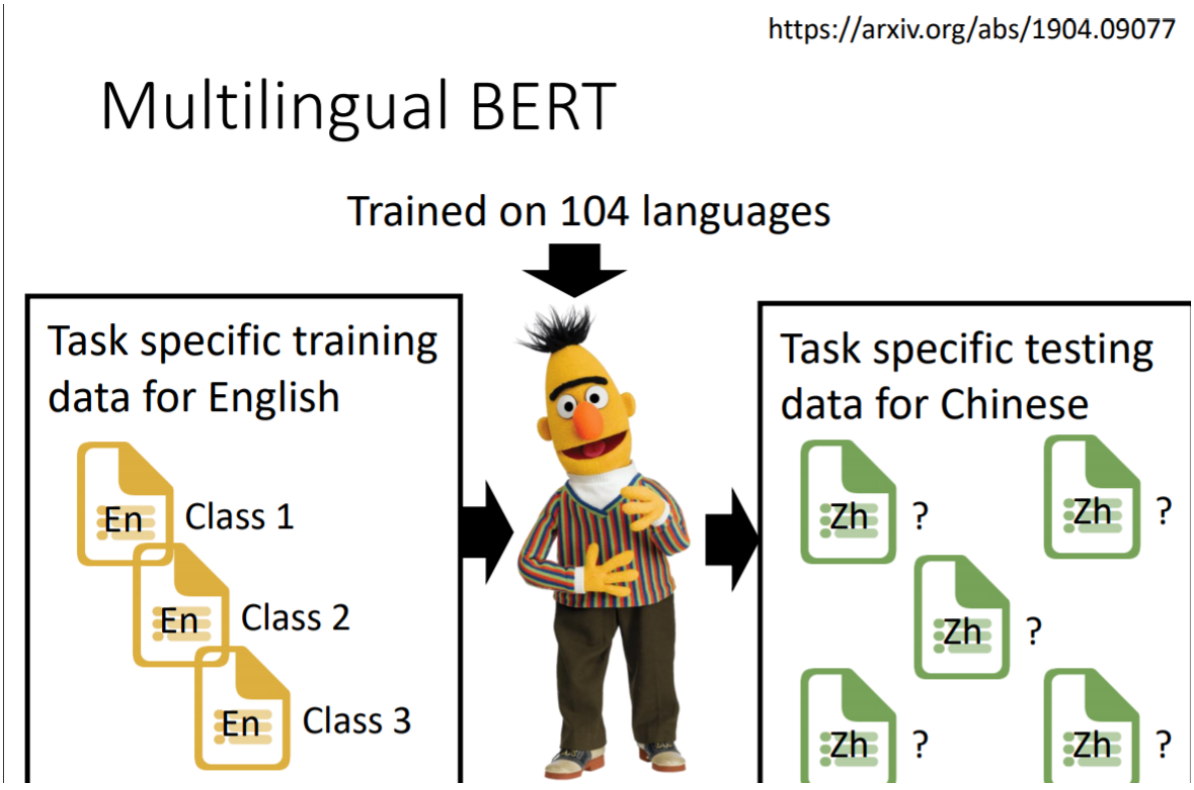


训练一个红色和蓝色的向量，分别和文章的词汇的向量做dot product，分别求出分数，取最高的分别是s和e。

ERNIE

Enhanced Representation through Knowledge Integration (ERNIE)

原理与BERT相似，但是专门为了中文设计的。在BERT中，经常mask掉一个值，但是对于中文是非常容易猜出来的。所以ERNIE是每次mask掉一个词。



多语言BERT，通过爬英语的维基百科，教他学会分类。然后就自己学会了中文的文章的分类。

5、Bert的主要贡献

Bert 采用深度双向 Transformer 语言模型，通过 Mask LM 来达到训练深度双向预训练模型，较之前使用单向语言模型训练更准确，信息量更大，且语义理解更准确。

论文表明，预训练模型能省去特定工程需要修改体系架构的麻烦，Bert 是第一个基于 fine-tune 的语言模型，它在大量句子级和 Token 级任务上展现了很好的性能。

Bert 的成功，一个重要原因就是数据量大，计算资源丰富。BERT 训练数据采用了英文的开源语料 BooksCropus以及英文维基百科数据，一共有 33 亿个词。同时 BERT 模型的标准版本有 1 亿的参数量，与 GPT 持平，而 BERT的大号版本有 3 亿多参数量，这应该是目前自然语言处理中最大的预训练模型了。

当然，这么大的模型和这么多的数据，训练的代价也是不菲的。谷歌用了 16 个自己的 TPU 集群（一共 64 块 TPU）来训练大号版本的 BERT，一共花了 4 天的时间。

对于是否可以复现预训练，作者在 Reddit 上有一个大致的回复，指出 OpenAI 当时训练GPT用了将近 1 个月的时间，而如果用同等的硬件条件来训练 BERT 估计需要 1 年的时间。不过他们会将已经训练好的模型和代码开源，方便大家训练好的模型上进行后续任务。

参考：

1. 奔向算法的喵
2. 为什么BERT有3个嵌入层，它们都是如何实现的
3. 论文解读：Bert原理深入浅出
4. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
5. 李宏毅BERT

其他推荐阅读：

1. 超细节的BERT/Transformer知识点