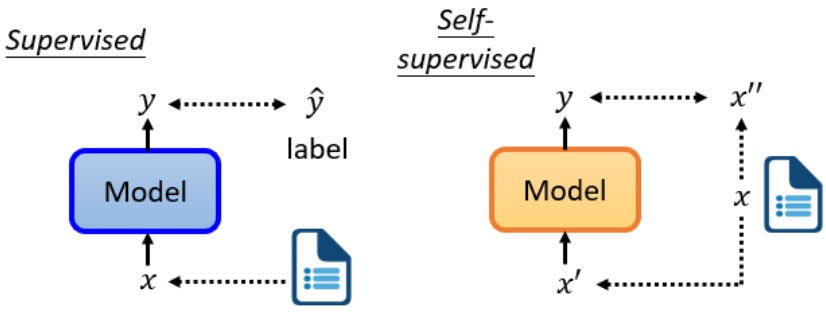
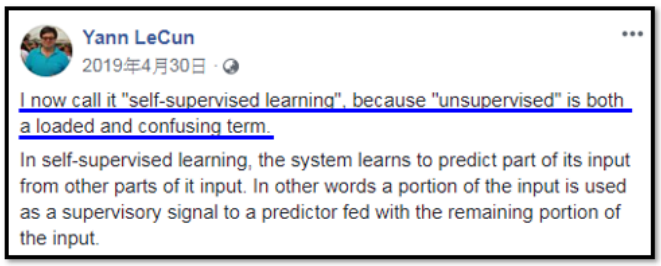
8月1日

一、李宏毅2021春机器学习课程第7.1节：自监督学习（一）

1 什么是自监督学习





- 首先我们来回忆一下监督学习，当我们做监督学习时，我们只有一个模型，这个模型的输入是x，输出是y。假设我们做情感分析，那就是让机器阅读一篇文章，而机器需要对这篇文章进行分类，是正面的还是负面的，为了对机器进行训练，我们必须先找到大量的文章，需要对所有的文章进行label。我们需要**有标签和文章数据来训练监督模型。**

  > 监督学习利用大量的标注数据来训练模型，模型的预测和数据的真实标签产生损失后进行反向传播（计算梯度、更新参数），通过不断的学习，最终可以获得识别新样本的能力。

- 有监督学习和无监督学习最主要的区别在于模型在训练时是否需要**人工标注**的**标签信息。**

  > 无监督学习不依赖任何标签值，通过对数据内在特征的挖掘，找到样本间的关系，比如聚类相关的任务。

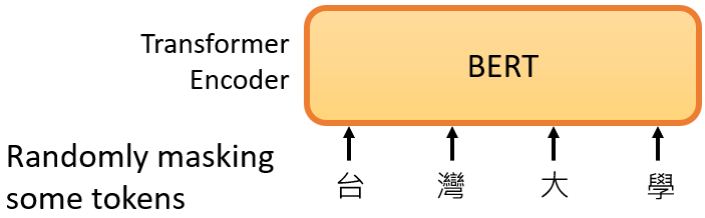
- 自监督学习 是用另一种方式来监督，**没有标签。**假设我们只有一堆没有label的文章，例如，一篇文章叫x，我们把x分成两部分，**一部分叫x'，另一部分叫x''**，然后把**x'输入模型，让它输出y。**这里的x''充当了label的作用，但是是从原本无标签的训练数据中划分出来的，所以叫做自监督学习。由于在Self-supervised学习中不使用标签，我们可以说，**Self-supervised学习也是一种无监督的学习方法。**但之所以叫Self-supervised Learning，是为了让定义更清晰。

> 自监督学习主要是利用辅助任务（pretext）从大规模的无监督数据中挖掘自身的监督信息，通过这种构造的监督信息对网络进行训练，从而可以学习到对下游任务有价值的表征。

---

2 训练 BERT 的方法一：Masking Input

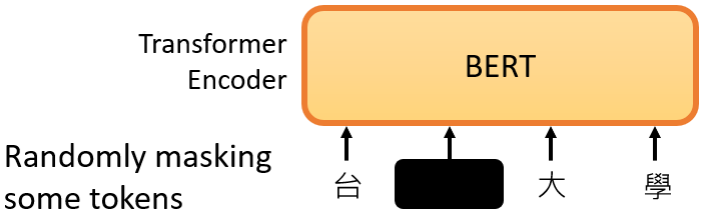
下面我们直接拿BERT模型来说明自监督学习的典型训练方法。首先，**BERT是一个transformer的Encoder**，里面有很多**Self-Attention**和**Residual connection**，还有**Normalization**等等，之前已经提到过了。



如果你已经忘记了Encoder里有哪些部件，那这里只需要记住的关键点是：**BERT可以输入一行向量，然后输出另一行向量，输出的长度与输入的长度相同。**

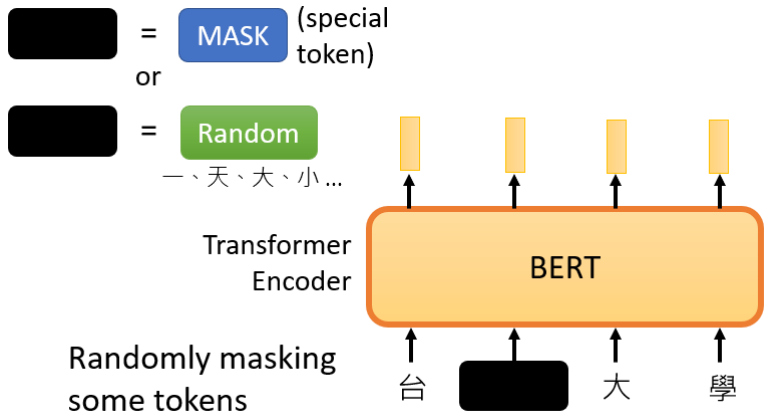
BERT一般用于自然语言处理，用于文本场景，所以一般来说，它的输入是一串文本，也是一串数据。当我们真正谈论Self-Attention的时候，我们也说**不仅文本是一种序列，而且语音也可以看作是一种序列，甚至图像也可以看作是一堆向量。**所以BERT不仅用于NLP，或者用于文本，它也可以用于语音和视频。

接下来我们需要做的是，随机**盖住**一些输入的文字，**被mask的部分是随机决定的。**例如，我们输入100个token，在中文文本中，我们通常把一个汉字看作是一个token，当我们输入一个句子时，其中的一些词会被随机mask。

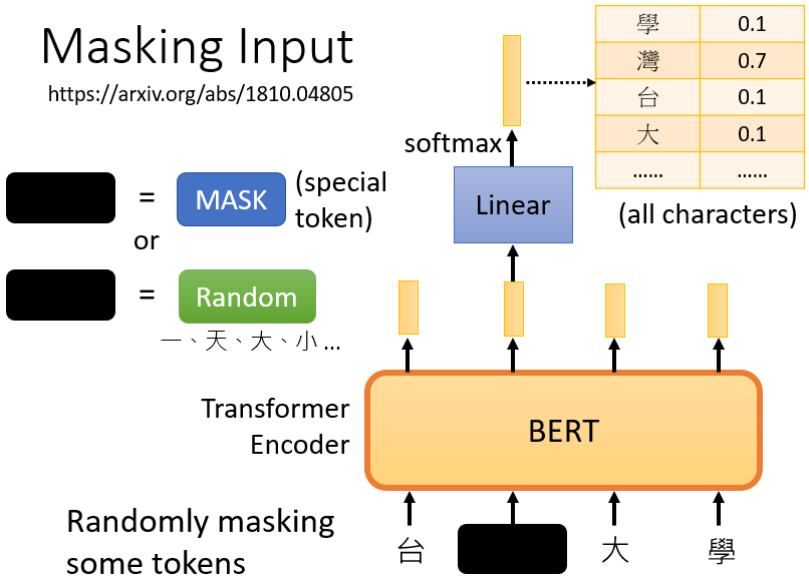


mask的具体实现有**两种方法**：

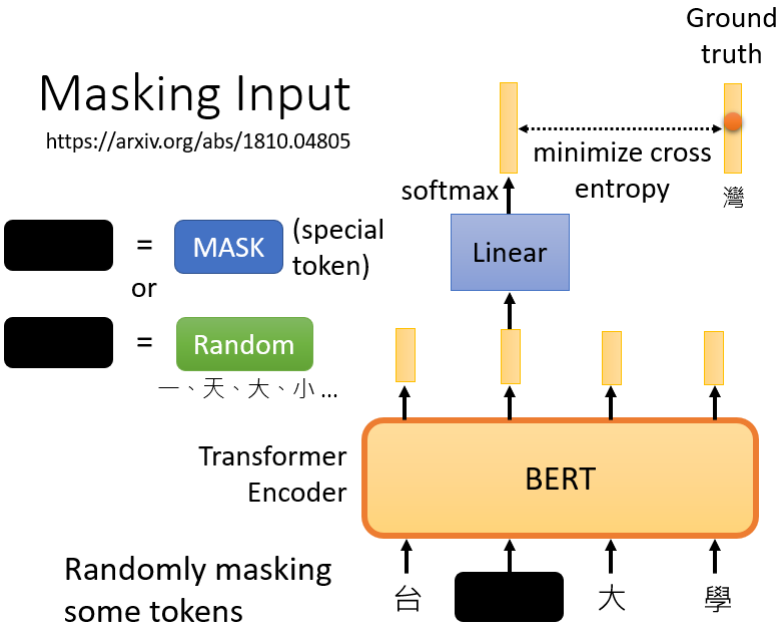
- 第一种方法是，用一个**特殊的符号替换句子中的一个词**，我们用 "MASK "标记来表示这个特殊符号，你可以把它看作一个新字，这个字完全是一个新词，它不在你的字典里，这意味着mask了原文。
- 另外一种方法，**随机**把某一个字**换成另一个字。**中文的 "湾"字被放在这里，然后你可以选择另一个中文字来替换它，它可以变成 "一 "字，变成 "天 "字，变成 "大 "字，或者变成 "小 "字，我们只是用随机选择的某个字来替换它。



两种方法都可以使用。**使用哪种方法也是随机决定的。**因此，当BERT进行训练时，向BERT输入一个句子，**先随机决定哪一部分的汉字将被mask。**mask后，一样是输入一个序列，我们把BERT的相应输出看作是另一个序列，接下来，我们在输入序列中寻找mask部分的相应输出，然后，这个向量将通过一个**Linear transform。**所谓的Linear transform是指，输入向量将与一个**矩阵相乘**，然后做**softmax**，输出一个分布。



在训练过程中，我们知道被mask的字符是什么，而BERT不知道，我们可以用一个one-hot vector来表示这个字符，我们训练的目标就是使输出vector和真实字符的vector之间的交叉熵损失最小。



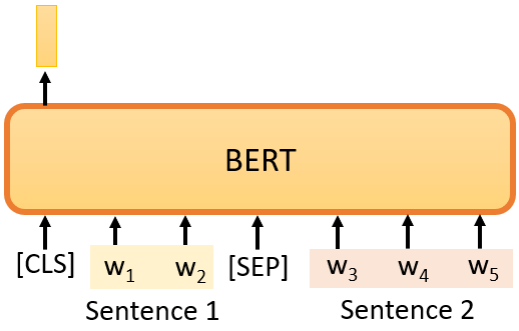
或者说得简单一点，我们实际上是在解决一个**分类问题**。现在，BERT要做的是，**预测什么被盖住**。在上面的例子中被掩盖的字符，属于 "湾"类，这就是训练BERT的方法之一。

---

3 训练 BERT 的方法二：Next Sentence Prediction

事实上，当我们训练BERT时，除了mask之外，我们还会使用另一种方法，这种额外的方法叫做Next Sentence Prediction 。

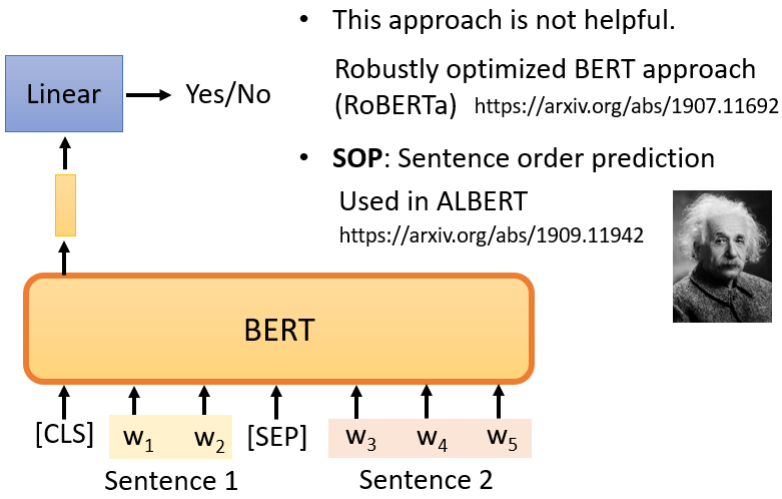
它的意思是，我们从数据库中拿出两个句子，这是我们通过在互联网上抓取和搜索文件得到的大量句子集合，我们在这**两个句子之间**添加一个**特殊标记**。这样，BERT就可以知道，这两个句子是不同的句子，因为这两个句子之间有一个分隔符。



我们还将在句子的**开头**添加一个**特殊标记**，这里我们用CLS来表示这个特殊标记。

现在，我们有一个很长的序列，包括**两个句子**，由SEP标记和前面的CLS标记分开。如果我们把它传给BERT，它应该输出一个序列，因为输入也是一个序列，这毕竟是Encoder的目的。

我们将**只看CLS的输出**，我们将把它通过一个Linear transform。



现在它必须做一个**二分类问题**，有两个可能的输出：是或不是。这个方法被称为Next Sentence Prediction ，所以我们需要预测，第二句是否是第一句的后续句。

然而，后来的研究发现，对于BERT要做的任务来说，**Next Sentence Prediction 并没有真正的帮助。**它帮助不大可能的原因之一是，**Next Sentence Prediction 太简单了**，是一项容易的任务。

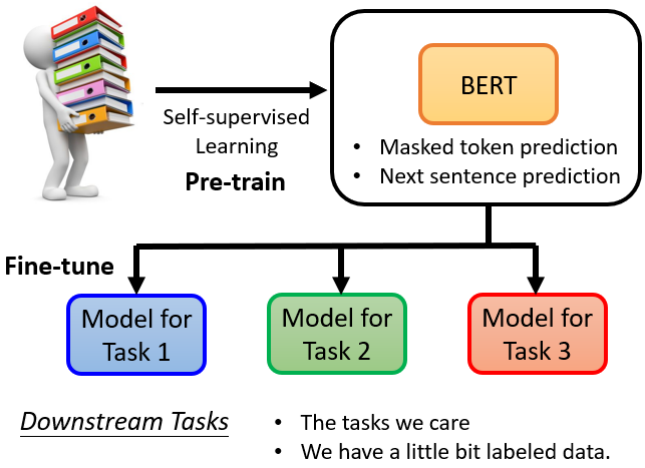
这个任务的典型方法是，首先随机选择一个句子，然后从数据库中或随机选择要与前一个句子相连的句子。通常，当我们随机选择一个句子时，它看起来与前一个句子有很大不同。对于BERT来说，预测两个句子是否相连并不是太难。因此，在训练BERT完成Next Sentence Prediction 的任务时，**没有学到什么太有用的东西。**

还有一种类似于Next Sentence Prediction 的方法，它看起来更有用，它被称为Sentence order prediction，简称SOP。

这个方法的主要思想是，我们最初挑选的两个句子可能是相连的。可能有两种可能性：要么句子1在句子2后面相连，要么句子2在句子1后面相连。有两种可能性，我们问BERT是哪一种。

也许因为这个任务更难，它似乎更有效。它被用在一个叫ALBERT的模型中，这是BERT的高级版本。

所以结合BERT训练的两个任务来说，BERT它学会了如何填空。BERT的神奇之处在于，在你训练了一个填空的模型之后，它还可以**用于其他任务。**这些任务**不一定与填空有关**，也可能是完全不同的任务，但BERT仍然可以用于这些任务，这些任务是BERT实际使用的任务，它们被称为Downstream Tasks(下游任务)。



所谓的 "Downstream Tasks "是指你真正关心的任务。但是，当我们想让BERT学习做这些任务时，我们仍然**需要一些标记的信息**。

总之，BERT只是学习填空，但是，以后可以用来做各种你感兴趣的Downstream Tasks 。它就像**胚胎中的干细胞**,它有各种无限的潜力，我们只需要给它一点数据来激发它，它就能做到很多比填空看起来更加复杂的问题。

BERT分化成各种任务的功能细胞，被称为Fine-tune(微调)。所以，我们经常听到有人说，他对BERT进行了微调，也就是说他手上有一个BERT，他对这个BERT进行了微调，使它能够完成某种任务，与微调相反，在微调之前产生这个BERT的过程称为预训练。所以，生成BERT的过程就是Self-supervised学习。但是，你也可以称之为预训练。

今天，为了测试Self-supervised学习的能力，通常你会在**多个任务上测试**它。因为我们刚才说，BERT就像一个胚胎干细胞，它要分化成各种任务的功能细胞，我们通常不会只在一个任务上测试它的能力，你会让这个BERT分化成各种任务的功能细胞，看看它在每个任务上的准确性，然后我们取其平均值，得到一个总分。这种不同任务的集合，我们可以称之为任务集。任务集中最著名的基准被称为GLUE，它是**General Language Understanding Evaluation**的缩写。

GLUE

General Language Understanding Evaluation (GLUE)

https://gluebenchmark.com/

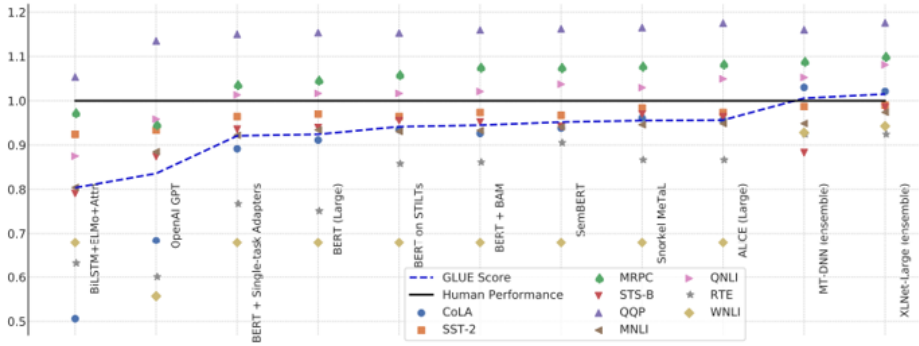
- Corpus of Linguistic Acceptability (CoLA)
- Stanford Sentiment Treebank (SST-2)
- Microsoft Research Paraphrase Corpus (MRPC)
- Quora Question Pairs (QQP)
- Semantic Textual Similarity Benchmark (STS-B)
- Multi-Genre Natural Language Inference (MNLI)
- Question-answering NLI (QNLI)
- Recognizing Textual Entailment (RTE)
- Winograd NLI (WNLI)

GLUE also has Chinese version (https://www.cluebenchmarks.com/)

在GLUE中，总共有9个任务，所以你实际上会得到9个模型，用于9个单独的任务。你看看这**9个任务的平均准确率**，然后得到一个值。这个值代表这个Self-supervised模型的性能。

下图是BERT在GLUE上的性能测试结果：

- GLUE scores



Source of image: https://arxiv.org/abs/1905.00537

在这张图中，**横轴表示不同的模型**，这里你可以发现除了ELMO和GPT，还有很多其他的BERT，从左到右出现的模型也是按时间顺序来到，可以看到**BERT的GLUE得分，也就是9个任务的平均得分，确实逐年增加。**

黑色的线表示人类在这个任务上的准确度，我们把这个当作1，这里每一个点代表一个任务，如果这些模型的表现比人类好，这些点的值就会大于1，如果比人类差，这些点的值就会小于1。

所以你会发现，在原来的9个任务中，只有1个任务，机器可以比人类做得更好。随着越来越多的技术被提出，越来越多的任务可以比人类做得更好。对于那些之前看来远不如人类的任务，它们也在逐渐追赶。

蓝色曲线表示机器**GLUE得分的平均值。**还发现最近的一些强势模型，例如XLNET，甚至超过了人类。当然，这只是这些数据集的结果，并不意味着机器真的在总体上超过了人类。它**在这些数据集上超过了人类。**这意味着这些数据集并不能代表实际的表现，而且难度也不够大。

所以，在GLUE之后，有人做了Super GLUE。他们找到了更难的自然语言处理任务，让机器来解决。好了！展示这幅图的意义主要是告诉大家，有了BERT这样的技术，机器在自然语言处理方面的能力确实又向前迈进了一步。

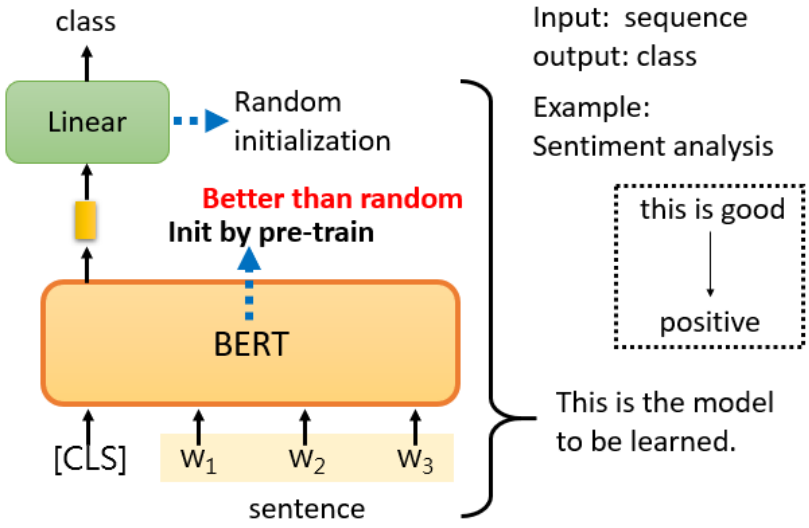
BERT到底是怎么用的呢？我们将给出4个关于BERT的应用案例。

---

4 BERT 的应用案例解析

4.1 情感分析

第一个案例是这样的，我们假设我们的下游任务是输入一个序列，然后输出一个class，也就是说是一个**分类问题。**比如说Sentiment analysis情感分析，就是给机器一个句子，让它判断这个句子附带的情感是正面的还是负面的。



对于BERT来说，它是如何解决情感分析的问题的？

你只要给它一个句子，也就是你想用它来判断情绪的句子，然后把**CLS标记放在这个句子的前面**，在上图的例子中，这4个输入实际上对应着4个输出。然后，我们**只看CLS的部分的输出。**CLS在这里输出一个向量，我们对它进行Linear transform，也就是将它乘以一个Linear transform的矩阵，这里省略了Softmax。

然而，在实践中，你必须为你的下游任务提供**标记数据**，换句话说，BERT没有办法从头开始解决情感分析问题，你仍然需要向BERT提供一些标记数据，你需要向它提供大量的句子，以及它们的正负标签，来训练这个BERT模型。

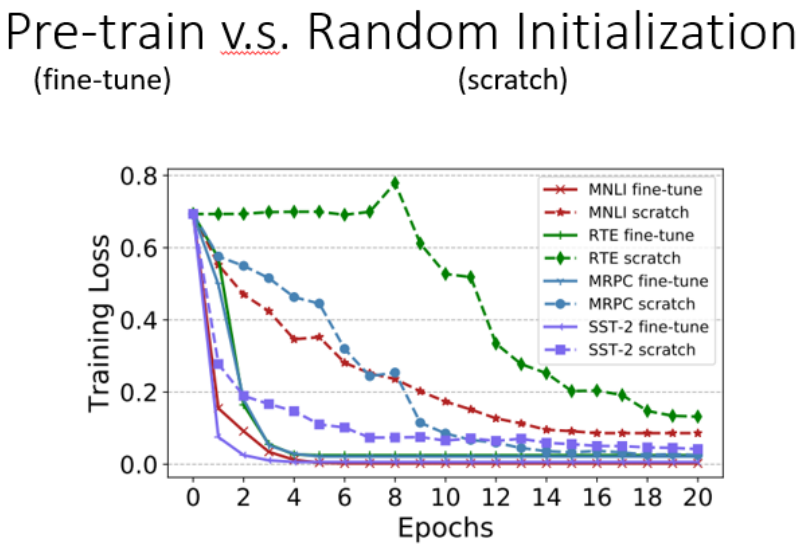
在训练的时候，Linear transform和BERT模型都是利用Gradient descent来更新参数的。

- Linear transform的参数是**随机初始化**的
- 而BERT的参数是由**学会填空的BERT初始化**的。

每次我们训练模型的时候，我们都要初始化参数，我们利用梯度下降来更新这些参数，然后尝试minimize loss，

例如，我们正在做情感分类，但是，我们现在有BERT。我们不必随机初始化所有的参数。,我们唯一随机初始化的部分是Linear这里。BERT的骨干是一个巨大的transformer的Encoder。这个网络的参数不是随机初始化的。把学过填空的BERT参数，放到这个地方的BERT中作为参数初始化。

我们为什么要这样做呢？为什么要用学过填空的BERT，再放到这里呢？**最直观和最简单的原因是，它比随机初始化新参数的网络表现更好。**当你把学会填空的BERT放在这里时，它将获得比随机初始化BERT更好的性能。



Source of image: https://arxiv.org/abs/1908.05620

在这里有篇文章中有一个例子。横轴是训练周期，纵轴是训练损失，到目前为止，大家对这种图一定很熟悉，随着训练的进行，损失当然会越来越低，在这个图中：

- fine-tune是指模型的BERT部分使用了预训练的参数来初始化的。
- scratch表示整个模型都是随机初始化的。

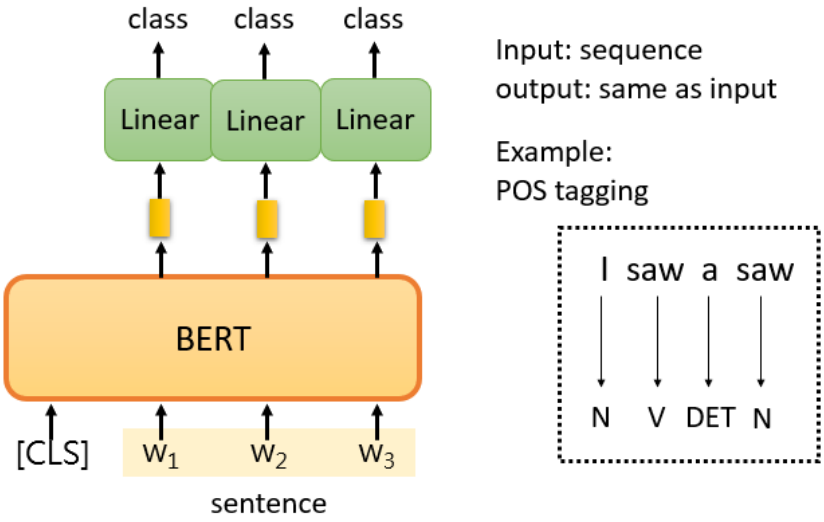
首先，在训练网络时，scratch与用学习填空的BERT初始化的网络相比，**损失下降得比较慢**，最后，用随机初始化参数的网络的损失仍然高于用学习填空的BERT初始化的参数。

- 当你进行Self-supervised学习时，你使用了大量的**无标记数据**。
- 另外，Downstream Tasks 需要少量的**标记数据**。

所谓的 "半监督 "是指，你有大量的无标签数据和少量的有标签数据，这种情况被称为 "半监督"，**所以使用BERT的整个过程是连续应用Pre-Train和Fine-Tune，它可以被视为一种半监督方法。**

4.2 词性标记

第二个案例是，输入一个序列，然后输出另一个序列，而输入和输出的长度是一样的。我们在讲Self-Attention的时候也举了类似的例子， 例如POS tagging，也就是**词性标记**的问题。



所谓的词性标记，就是你给机器一个句子，它必须告诉你这个句子中每个词的词性，即使这个词是相同的，也可能有不同的词性。

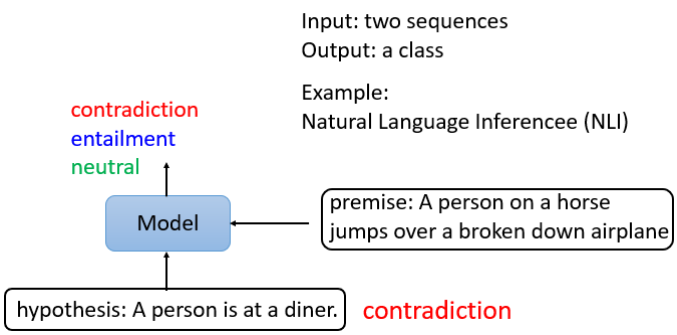
你只需向BERT输入一个句子。之后，对于这个句子中的每一个标记，它是一个中文单词，有一个代表这个单词的相应向量。然后，这些向量会依次通过Linear transform和Softmax层。最后，网络会预测给定单词所属的类别，例如，它的词性。

当然，类别取决于你的任务，如果你的任务不同，相应的类别也会不同。接下来你要做的事情和案例1完全一样。换句话说，你需要有一些标记的数据。这仍然是一个典型的分类问题。唯一不同的是，BERT部分，即网络的Encoder部分，其参数不是随机初始化的。在**预训练过程中，它已经找到了不错的参数。**

当然，我们在这里展示的例子属于自然语言处理。但是你可以把这些例子改成**其他任务**，例如，你可以把它们改成语音任务，或者改成计算机视觉任务。因为之前就提到了语音、文本和图像都可以表示为一排向量。

4.3 自然语言推断

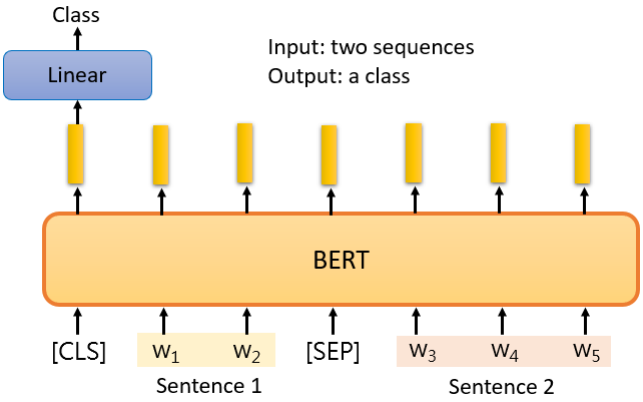
第三个案例是，模型输入两个句子，输出一个类别。什么样的任务采取这样的输入和输出？ 最常见的是Natural Language Inference ，它的缩写是NLI。



所谓NLI，机器要做的就是判断是否有可能**从前提中推断出假设**。这个前提与这个假设相矛盾吗？或者说它们不是相矛盾的句子？在这个例子中，我们的前提是，一个人骑着马，然后他跳过一架破飞机，这听起来很奇怪。但这个句子实际上就是这样的，这是一个基准语料库中的一个例子。这里的**假设**是，这个人在一个餐馆。所以**推论**说这是一个**矛盾**。

所以机器要做的是，把两个句子作为输入，并输出这两个句子之间的关系。这种任务很常见。它可以用在哪里呢？例如，舆情分析。给定一篇文章，下面有一个评论，这个消息是同意这篇文章，还是反对这篇文章？该模型想要预测的是每条评论的位置。事实上，有很多应用程序接收两个句子，并输出一个类别。

BERT是如何解决这个问题的？你只要给它两个句子，我们在这两个句子之间放一个**特殊的标记SEP**，并在最开始放CLS标记。



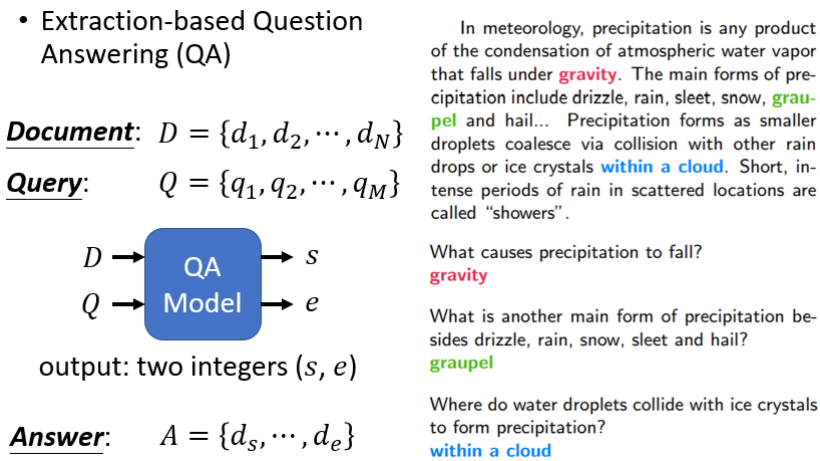
这个序列是BERT的输入。但我们只把CLS标记作为Linear transform的输入。它决定这两个输入句子的类别。对于NLI，你必须问，这两个句子是否是矛盾的。它是用一些**预先训练好的权重来初始化的**。

4.4 基于信息抽取的问答

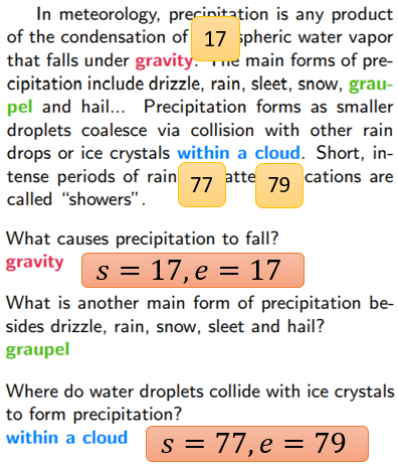
第四个案例是用BERT实现一个问题回答系统。也就是说，在机器读完一篇文章后，你问它一个问题，它将给你一个答案。

但是，这里的问题和答案稍有限制。这是Extraction-based的QA。也就是说，我们假设**答案必须出现在文章**中。答案必须是文章中的一个片段。

在这个任务中，一个输入序列包含**一篇文章**和**一个问题**，文章和问题都是一个**序列**。对于中文来说，每个d代表一个汉字，每个q代表一个汉字。你把d和q放入QA模型中，我们希望它输出**两个正整数s和e**。根据这两个正整数，我们可以直接从文章中**截取一段**，它就是这个问题的答案。



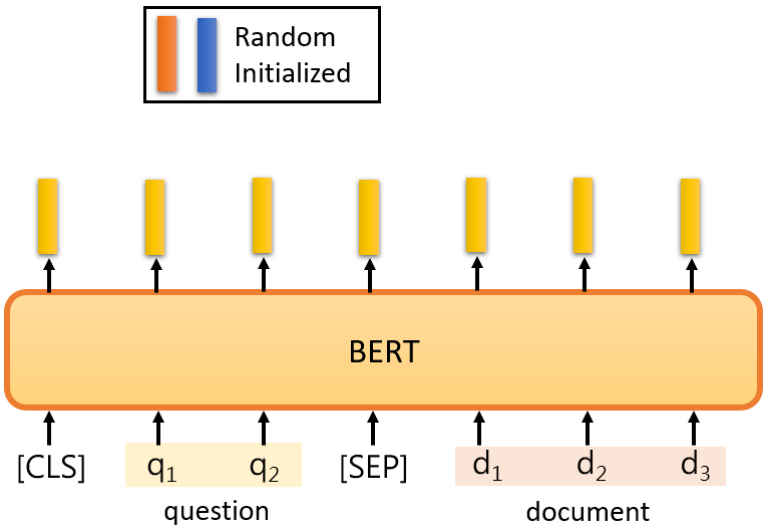
这听起来很疯狂，但是这是现在使用的一个相当标准的方法。举一个具体的例子来说，这里有一个问题和一篇文章，正确答案是 "gravity"。机器如何输出正确答案?



你的模型应该输出，s等于17，e等于17，来表示gravity。因为它是整篇文章中的第17个词，所以s等于17，e等于17，意味着输出第17个词作为答案。

再举另一个例子，答案是，"within a cloud"，这是文章中的第77至79个词。你的模型要做的是，输出77和79这两个正整数，那么文章中从第77个词到第79个词的分割应该是最终的答案。

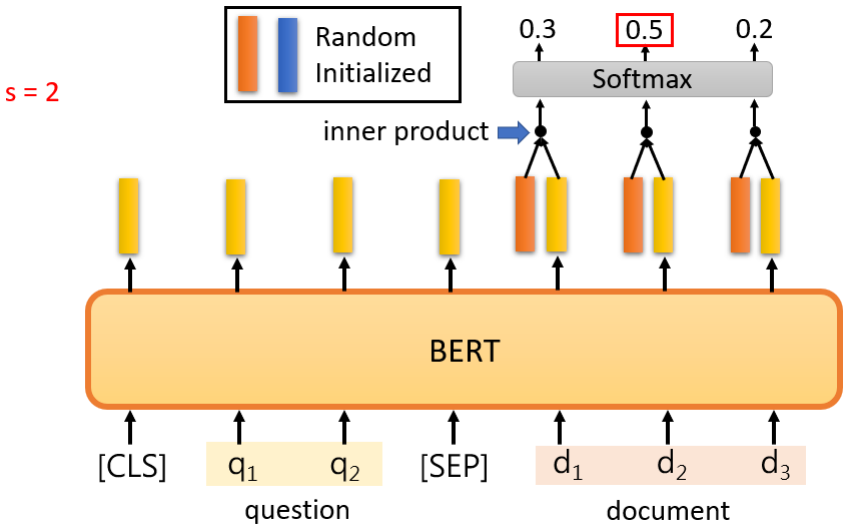
当然，我们不是从头开始训练QA模型，为了训练这个QA模型，我们**使用BERT预训练的模型。**



这个解决方案是这样的。对于BERT来说，你必须向它展示一个问题，一篇文章，以及在问题和文章之间的一个特殊标记，然后我们在开头放一个CLS标记。

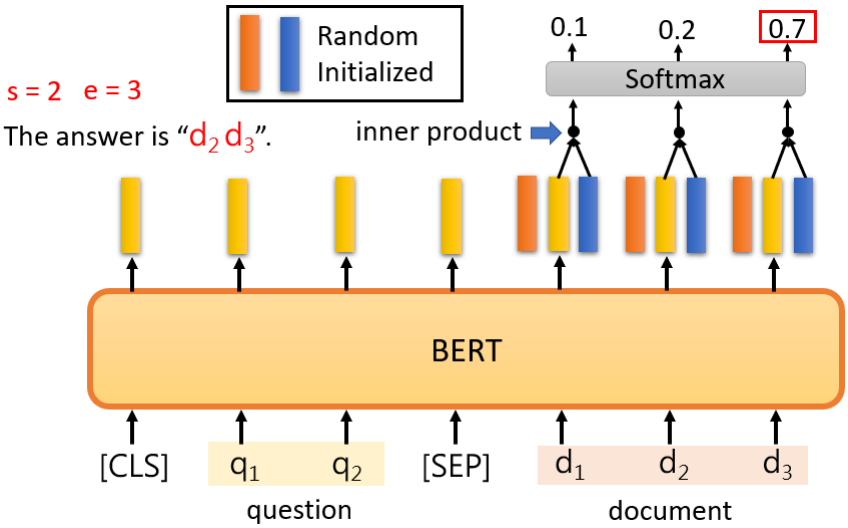
在这个任务中，你唯一需要**从头训练**的只有**两个向量**。"从头训练 "是指**随机初始化**。这里我们用橙色向量和蓝色向量来表示，这两个向量的长度与BERT的输出相同。

假设BERT的输出是768维的向量，这两个向量也是768维的向量。那么，如何使用这两个向量?



- 首先，计算这个**橙色向量**和那些与文件相对应的**输出向量的内积**，由于有3个代表文章的标记，它将输出三个向量，计算这三个向量与橙色向量的内积，你将得到三个值，然后将它们通过**softmax**函数，你将得到另外三个值。

  这个内积**和attention很相似**，你可以把橙色部分看成是query，黄色部分看成是key，这是一个attention，那么我们应该尝试找到分数最大的位置，就是这里，橙色向量和d2的内积，如果这是最大值，s应该等于2，你输出的起始位置应该是2



- 蓝色部分做的是完全一样的事情。蓝色部分代表答案的终点，我们计算这个蓝色向量与文章对应的黄色向量的内积，然后，我们在这里也使用softmax，最后，找到最大值，如果第三个值是最大的，e应该是3，正确答案是d2和d3。

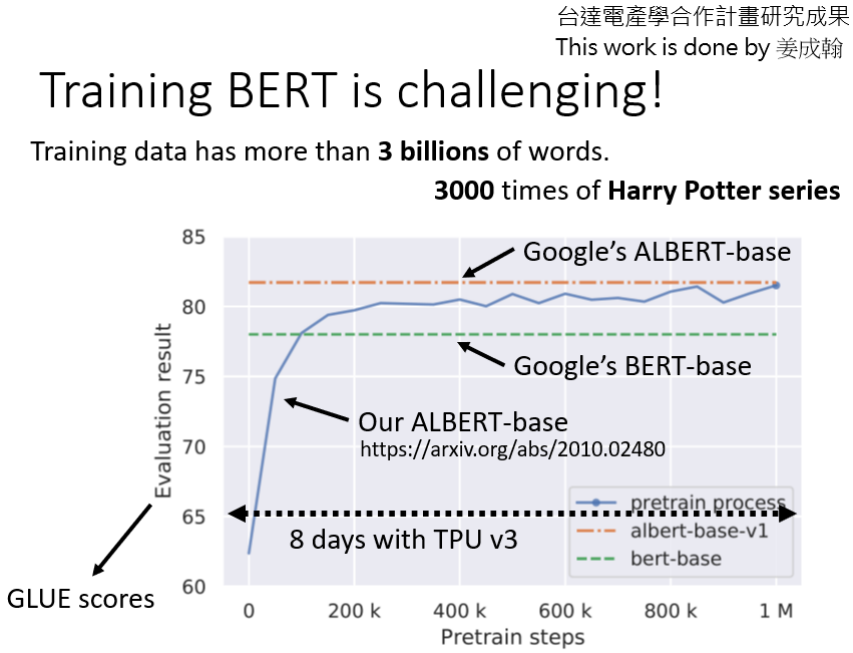
因为答案必须在文章中，如果**答案不在文章中，你就不能使用这个技巧**。这就是一个QA模型需要做的。注意，**这两个向量是随机初始化的，而BERT是通过它预先训练的权重初始化的。**

---

5 BERT 的训练并不容易

BERT真的是一个非常出名的模型，它可以做任何事情，那么你可能会认为BERT在预训练中只是做填空题，应该比较容易训练，但是**BERT的训练并没有想象中的那么容易。**

首先，谷歌最早的BERT，它使用的**数据规模已经很大**了，它的数据中包含了**30亿个词汇**，30亿个词汇有多少？是《哈利波特全集》的3000倍。所以训练数据处理起来会比较痛苦，**更痛苦的是训练过程**，李老师实验室有一个学生，也是现在的助教之一，他自己试着训练一个ALBERT，看究竟能不能重现谷歌的结果。



ALBERT是BERT的一个高级版本，**谷歌的结果是橙色的线，蓝线是我们自己训练的ALBERT**，但是我们实际训练的不是最大版本，BERT有一个base版本和一个large版本。对于大版本，我们很难自己训练它，所以我们尝试用最小的版本来训练，看它是否与谷歌的结果相同。

横轴是训练过程，要达到谷歌base相同的效果，**参数大约有一百万次的更新**，用TPU运行了8天，如果你在Colab上做，这个至少要运行200天，你甚至可能到明年才能得到结果。

所以，你真的很难自己训练这种BERT模型。幸运的是，当前我们的使用场景大都是在预训练好了的模型上进行微调，来服务于我们想要完成的下游任务。但是，如果你想从头开始训练它，这将需要大量的时间，是十分困难的。

---

6 BERT 胚胎学

谷歌已经训练了BERT，而且这些Pre-Train模型是公开的，我们自己训练一个，结果和谷歌的BERT差不多，这有什么意义呢?

其实是想建立BERT胚胎学。"BERT胚胎学是什么意思？"

我们知道在BERT的训练过程中需要非常大的计算资源，所以我们想知道有没有可能，**节省这些计算资源**？有没有可能让它训练得更快？,要知道如何让它训练得更快，也许我们可以从观察它的训练过程开始。

过去没有人观察过BERT的训练过程。因为在谷歌的论文中，他们只是告诉你，我有这个BERT。然后它在各种任务中做得很好。

BERT在学习填空的过程中，学到了什么？"它在这个过程中何时学会填动词？什么时候学会填名词？ 什么时候学会填代词？ 没有人研究过这个问题。

所以我们自己训练BERT后，可以观察到BERT什么时候学会填什么词汇，它是如何提高填空能力的？ 这里放一篇论文的链接供大家参考。
[Pretrained Language Model Embryology: The Birth of ALBERT (arxiv.org)]
