



**[课设]基于Bert论文构建**

**Question-Answering模型**

**课程名称 文本信息挖掘概论**

**学生学院 计算机学院**

**专业班 18级软件工程2班**

**学 号 3118005002**

**学生姓名 胡梓泽**

**指导老师 杨易杨**

**日期：2021年6月26日**

目录

[***Paper from [BERT: Pre-training of Deep Bidirectional*** 4](#_Toc75860145)

[***Transformers for Language Understanding*]** 4](#_Toc75860146)

[0. 前言 4](#_Toc75860147)

[1. 论文的简短介绍 5](#_Toc75860148)

[**1.1 涉及领域，前人工作等** 5](#_Toc75860149)

[1.2 中心思想 5](#_Toc75860150)

[1.3 论文思路及创新点 5](#_Toc75860151)

[1.4 BERT四大下游任务之问答任务 6](#_Toc75860152)

[2. 基于Bert的QA模型 7](#_Toc75860153)

[2.1 程序简要说明 7](#_Toc75860154)

[2.2 BERT的输入格式 8](#_Toc75860155)

[2.3 简单的一个QA 9](#_Toc75860156)

[2.4 可视化单词分数 11](#_Toc75860157)

[2.4.1 显示每个输入单词作为“start”单词的得分 12](#_Toc75860158)

[2.4.2 显示作为“end”单词的每个输入单词的得分。 12](#_Toc75860159)

[2.4.3 合并start和end 13](#_Toc75860160)

[2.5 汇总画图和QA模型 13](#_Toc75860161)

[2.6 长文本的QA例子 13](#_Toc75860162)

[3. 大规模数据集的尝试 15](#_Toc75860163)

[3.1 数据来源及展示 15](#_Toc75860164)

[3.2 程序结果及展示 17](#_Toc75860165)

[3.2.1 训练模型并保存 17](#_Toc75860166)

[3.2.2 测试集结果展示 17](#_Toc75860167)

[附录 19](#_Toc75860168)

***Paper from [BERT: Pre-training of Deep Bidirectional***

***Transformers for Language Understanding***]

|  |
| --- |
| (link):<https://arxiv.org/abs/1810.04805> |

**基于Bert论文构建Question-Answering模型**

胡梓泽a

a广东工业大学计算机学院

摘要 本文拜读了提出**Bert**模型的论文，考虑了在Bert中算法模型的实现.比较了Bert与其他如**Transformer、GPT**等热门**NLP**模型.BERT在概念上很简单，在经验上也很强大。它推动了11项自然语言处理任务的最新技术成果，而这11项NLP任务可分类为四大自然语言处理下游任务，本文聚焦分析实现一个简单的问答任务，使用了来自BERT-large的训练集，测试数据集(作业四爬取的wiki数据集),测试结果令人满意。

关键词 **Bert； Transformer； GPT; NLP; 问答任务**

# 0. 前言

* 由于本课设用的BERT-large真的巨巨巨巨大。有24层，1024大小的embedding，总共340M个参数！总的来说，它是1.34GB。然而国内下载实在太慢，故使用Google Colab提供的jupyter来运行相关代码。
* 想做但没有做出的：利用bertviz(<https://github.com/jessevig/bertviz>),一个Transformer模型中可视化注意力的工具，时间有些仓促，原本觉得能更好地展示Bert的运行过程，种种原因没有细看。
* 可视化的替代方案：将文本语段中的开始词和结束词分别评分，然后将其可视化。对于简短的文本有较好的效果。
* 数据集根据课设，一个简单联动作业四的wiki数据集中的文本内容(Demo)，由于爬取的数据没有答案标签，因此只简单判断答案。
* 大规模数据集以及部分参考代码来自[HUNG-YI LEE (李宏毅)](https://speech.ee.ntu.edu.tw/~hylee/index.html)的HW7问答，数据集(包括训练集、验证集和测试集[文章+问题+答案])。

# 1. 论文的简短介绍

**1.1 涉及领域，前人工作等**

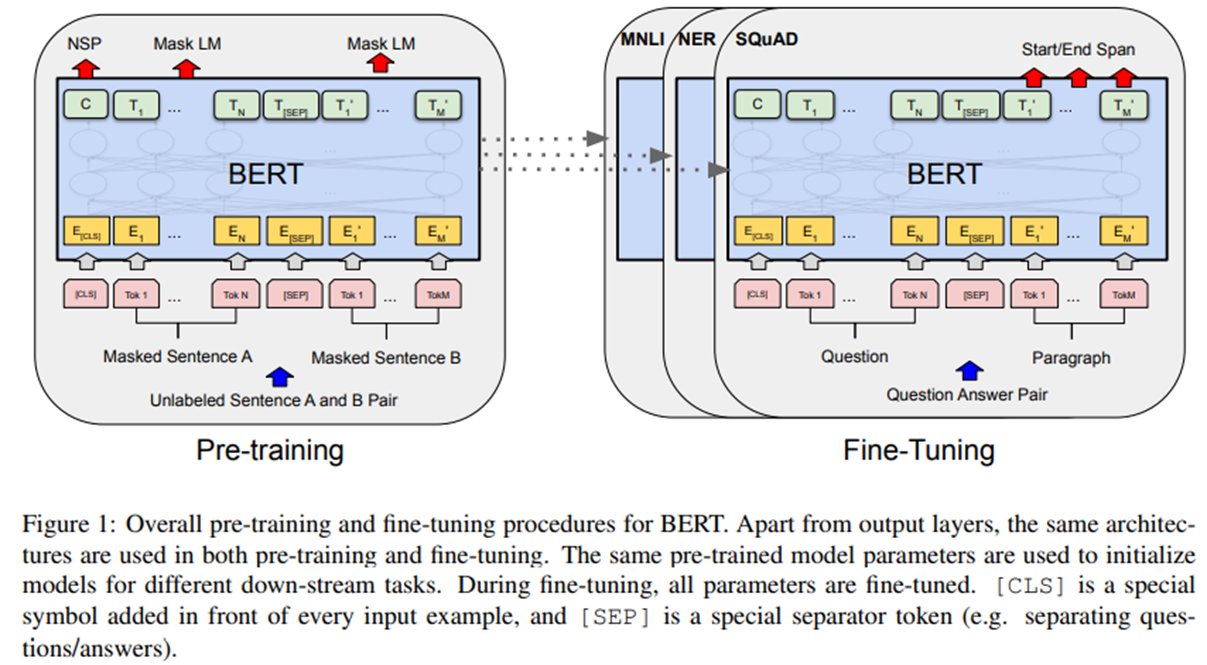
本文主要涉及NLP的一种语言模型，之前已经出现了【1】ELMo和【2】GPT这些较为强大的模型，ELMo的特征提取器不是很先进，GPT没有使用双向，这篇论文结合两者的思想或做法，大大提升了最终效果。

## 1.2 中心思想

本论文文在前人研究基础上，沿用了pre-train和fine-tuning结构。使用双向transformer结构（不同于ELMO的双向，而是"掩蔽语言模型"(MLM)），并加入Next Sentence Prediction（NSP），在11个自然语言处理任务上获得了新的最先进的结果。

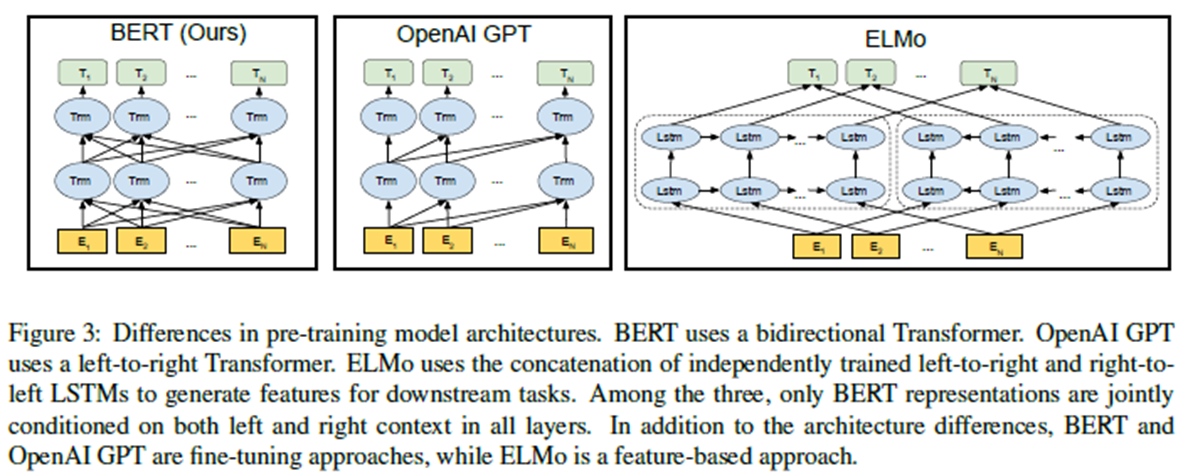
## 1.3 论文思路及创新点

框架图和重要部分如下图：



主要组成构建有：pre-training：**Embedding、Masked LM、Next Sentence Prediction 和**fine-tuning

创新点有：



1. 与GPT比较，使用了双向（GPT只用了单向结构）
2. 与ELMo比较，使用了更强的提取器：transformer，而且由于它的双向——使用了独立训练的从左到右和从右到左的LMs的浅连接。

论文中做种与GPT进行了对比，BERT和GPT的培训方式还有其他一些不同之处:

1. GPT在BooksCorpus上训练(800M单词);BERT被训练成书虫维基百科(2500万字)。
2. GPT使用一个句子分隔符([SEP])和分类器标记([CLS])，它们只在微调时引入;BERT在在预处理时，学习[SEP]， [CLS]和A/B嵌入。
3. GPT训练1M步，批处理32000字;BERT被训练1M步，批处理大小为12.8万字。
4. GPT对所有微调实验使用5e-5相同的学习率;BERT选择特定于任务的微调学习率，该学习率在开发集上执行得最好。

## 1.4 BERT四大下游任务之问答任务

BERT等预训练模型的提出，简化了我们对NLP任务精心设计特定体系结构的需求，我们只需在BERT等预训练模型之后下接一些网络结构，即可出色地完成特定任务。BERT在概念上很简单，在经验上也很强大。它推动了11项自然语言处理任务的最新技术成果，而这11项NLP任务可分类为四大自然语言处理下游任务。下面讲简单分析其中的问答任务。

SQuAD v1.1：Rajpurkar等人[25]提出的斯坦福问答数据（Stanford Question Answering Dataset）是10万个问题/答案对的集合。给定一个问题以及Wikipedia中包含答案的段落，任务是预测段落中的答案文本范围（start，end）。

到目前为止，所有提出的BERT微调方法都是在预训练模型中添加了一个简单的分类层，并且在下游任务上共同对所有参数进行了微调。然而，并非所有任务都可以轻松地由BERT体系结构表示，因此需要添加特定于任务的模型体系结构。如图 8.11（c）所示，阅读理解任务首先需要将问题和文本用“[SEP]”连接起来，并输入模型。然后，我们再将BERT最后一层向量输入到输出层。具体运算逻辑是初始化输出层的权重矩阵，并通过如下公式计算答案指针概率向量logit：

|  |  |  |
| --- | --- | --- |
|  |  | (1.1) |

其中L为序列的长度，K为2，表示logit是个L行2列的矩阵，第1列为答案开头start的指针概率向量，第2列为答案结尾end的指针概率向量。

因为K为2，所以我们能分别抽出答案的开头start\_logit和答案end\_logit的结尾。并根据两者与真实答案对(start, end)之间的差值计算start\_logit和end\_logit,最后求出总的loss，如下公式所示，我们便可以进行梯度下降训练了。

|  |  |  |
| --- | --- | --- |
|  |  | (1.2) |

# 2. 基于Bert的QA模型

# 2.1 程序简要说明

模型：

预训练模型bert-large-uncased-whole-word-masking-finetuned-squad

数据来源：自拟测试语句+科比wiki英文网介绍+自拟问题语句

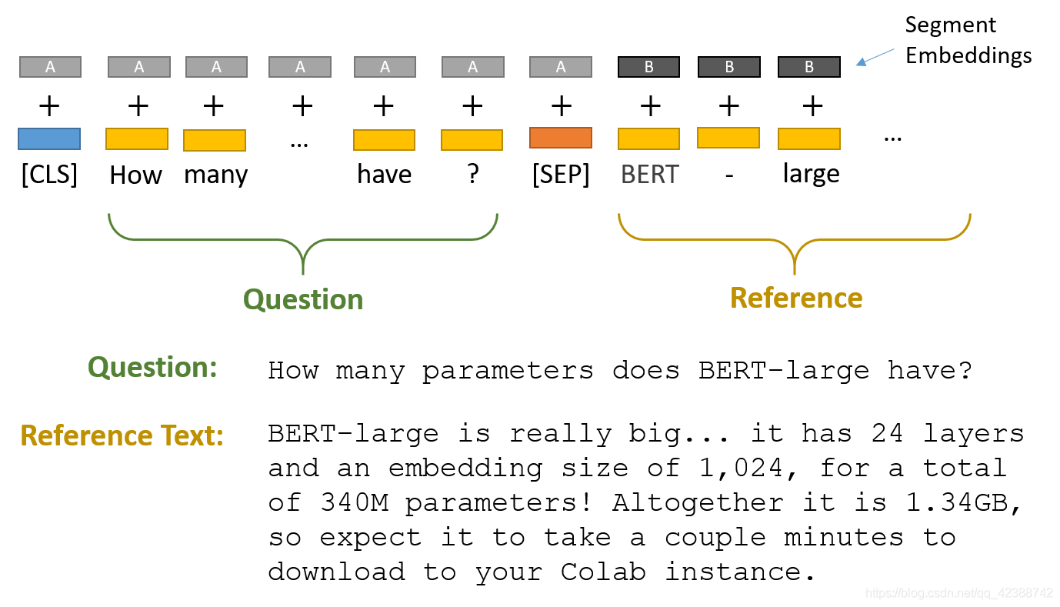
代码来源：个人编写,由于预训练模型较大和本地环境问题，采用Colab线上编程

(<https://colab.research.google.com/drive/1CyWQWwSamJuKAHw7T2U0uEvQ3YaWIzIz?usp=sharing>)

## 2.2 BERT的输入格式

**2.2中的图片来源：https://blog.csdn.net/qq\_42388742/article/details/113575739**

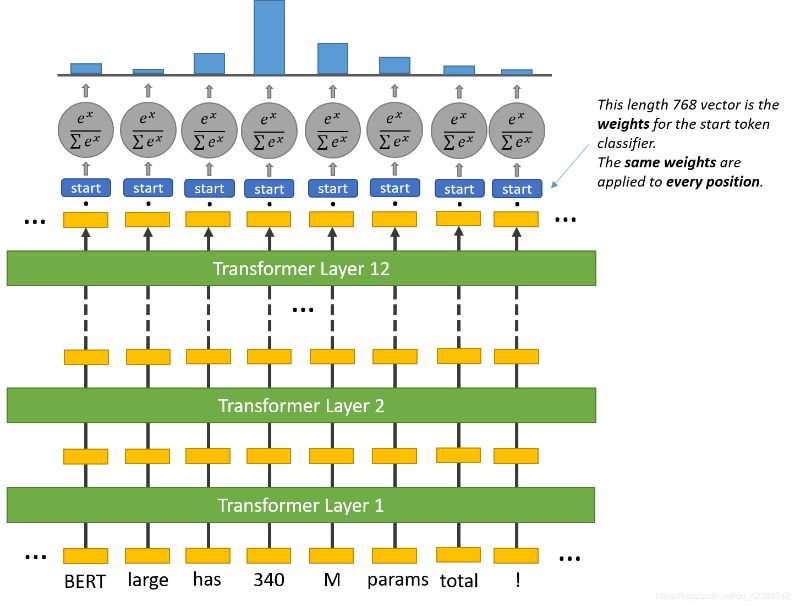
为了向BERT提供QA任务，我们将问题和文本打包到输入中。



这两段文本由特殊的标记[SEP]分隔。

BERT还使用了"Segment Embeddings"来区分问题和文本。这只是BERT学习到的两个嵌入(对于片段“A”和“B”)，在将它们输入到输入层之前，它将它们添加到token embeddings中。

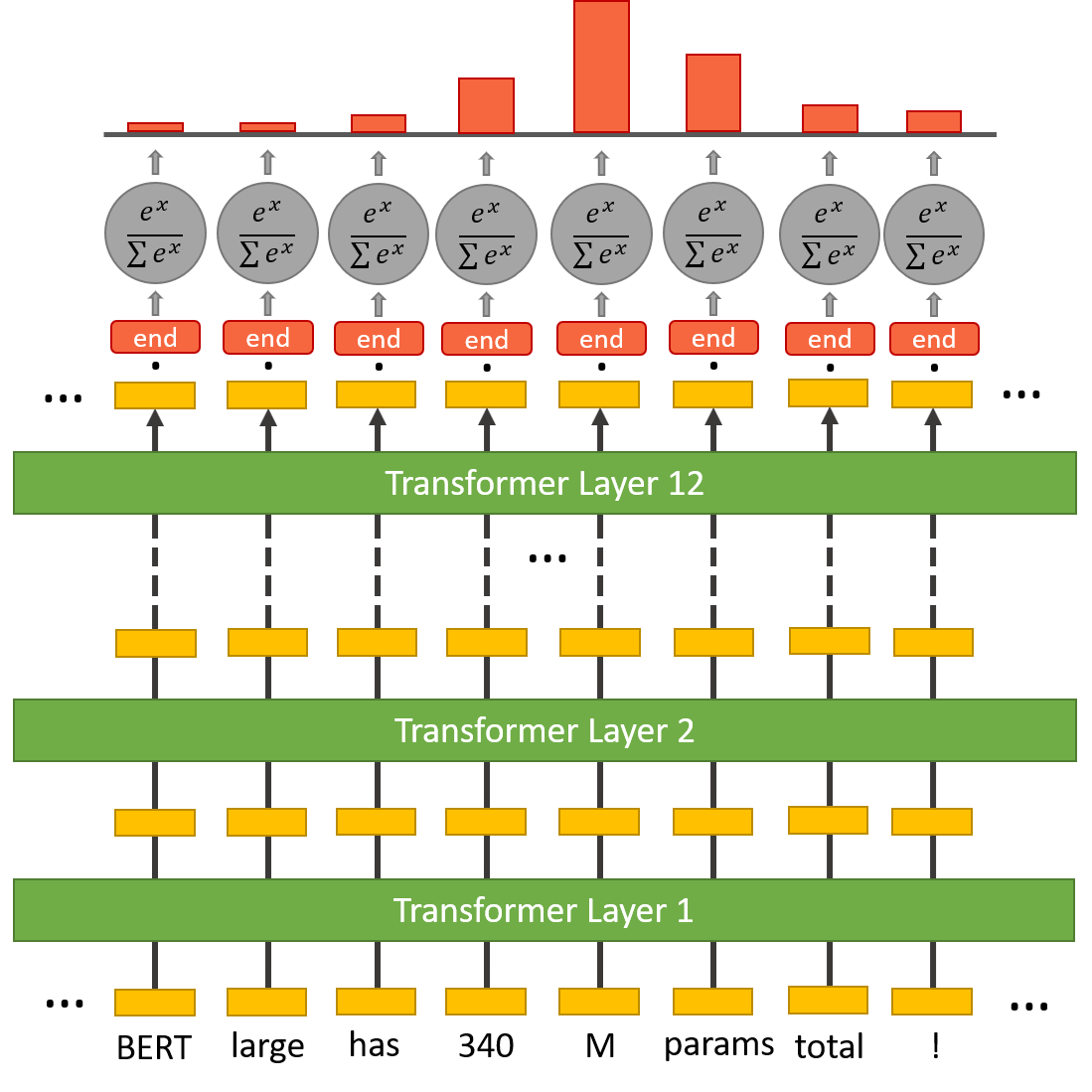
BERT需要高亮显示包含答案的文本“span”(答案)——这表示为简单地预测哪个token标志答案的开始，哪个token标志答案的结束。



对于在文档中的每个单词，返回一个最终的嵌入到分类器中。开始tokens 的分类器只有一组权值(由上图中的蓝色“strat”矩形表示)它应用于每个单词。

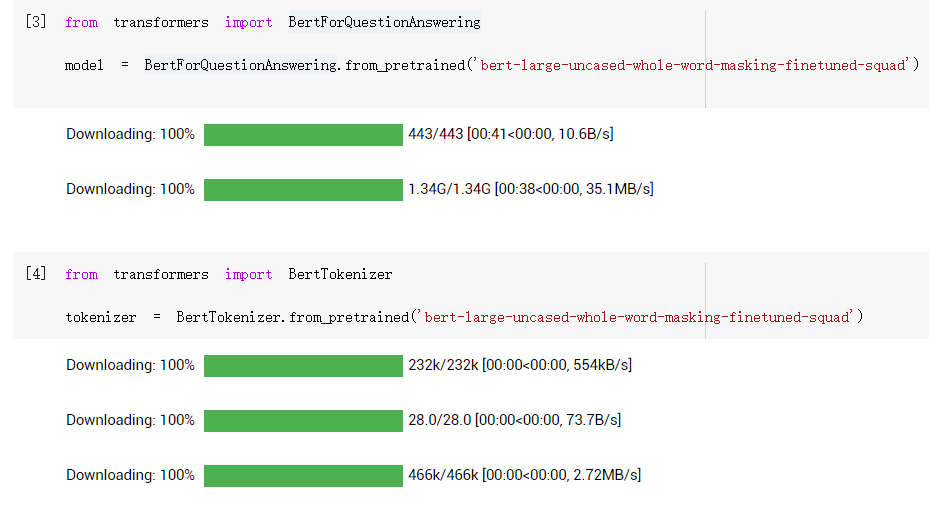
在获取输出嵌入和“开始”权重之间的点积之后，我们应用softmax激活来生成所有单词的概率分布。我们选择的是概率最高的单词作为起始tokens。

我们对结束token重复这个过程——我们有一个独立的权值向量。



## 2.3 简单的一个QA

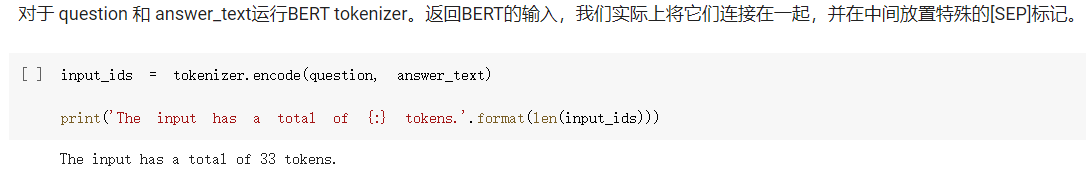
加载Bert自带的预训练模型：



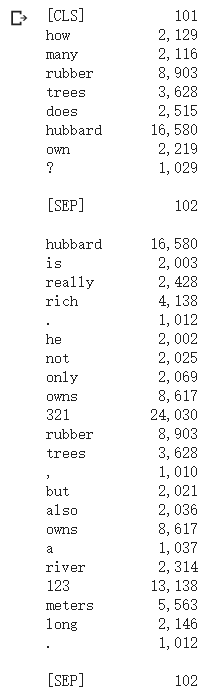
首先定义一个**Question Answering**如下：



期待的答案当然是321



为了准确地查看标记器(tokenizer)在做什么，让我们打印出带有id的标记。

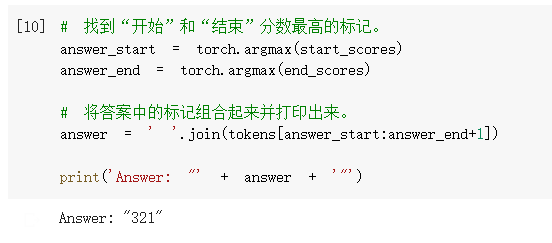


我们拼接question 和 answer\_text在一起，但BERT仍然需要一种方法来区分它们。bert有两个特殊的"Segment" 嵌入。一个是A，一个是B。在单词嵌入进入BERT层之前，段A嵌入需要添加到question标记中，段B嵌入需要添加到每个answer\_text标记中。

这些添加的内容由transformers 库为我们处理，我们所需要做的就是为每个令牌指定一个’0’或’1’。



以下代码实现查看最有可能的开始词和结束词来突出答案。

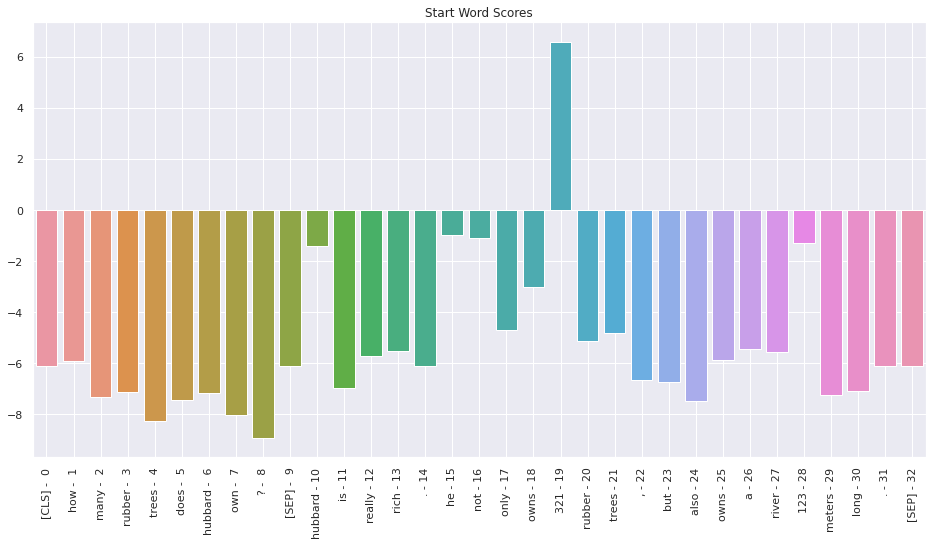


与期待答案一致。

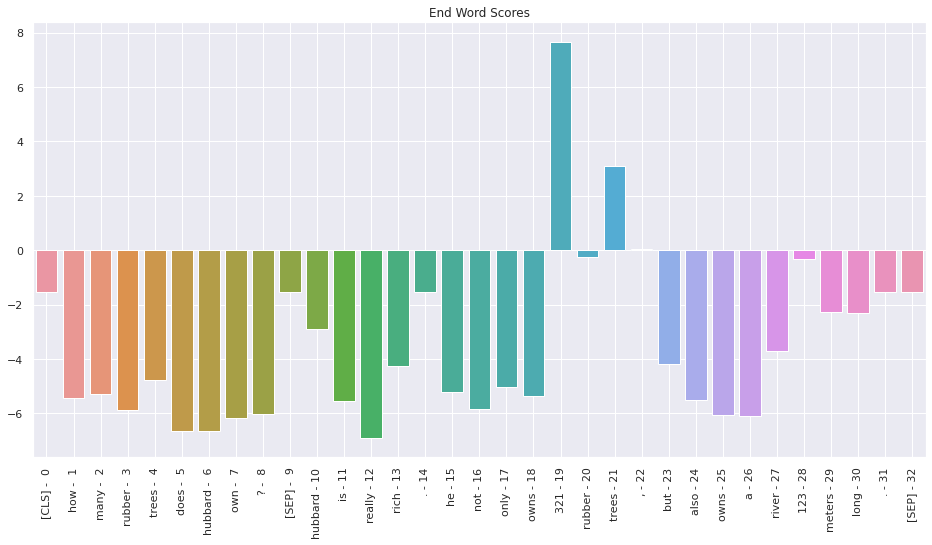
## 2.4 可视化单词分数

以2.3的例子为例，展示可视化的过程

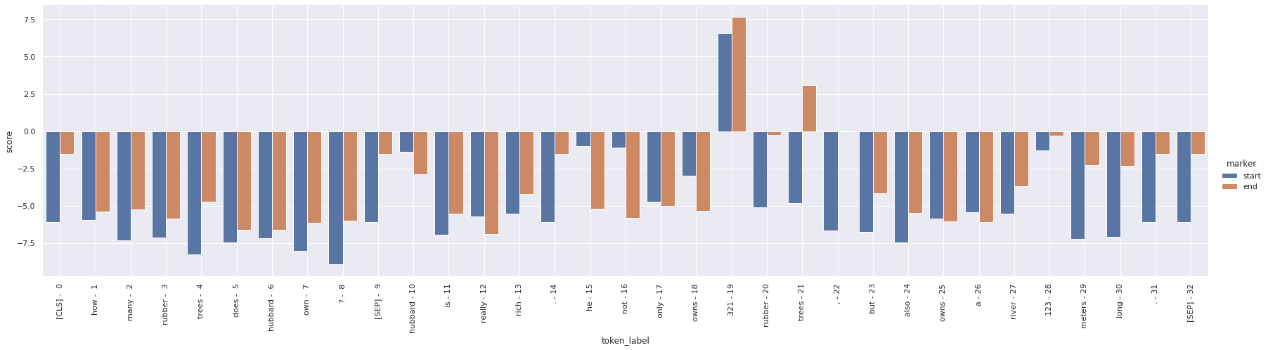
### 2.4.1 显示每个输入单词作为“start”单词的得分



### 2.4.2 显示作为“end”单词的每个输入单词的得分。

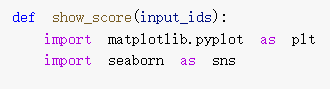


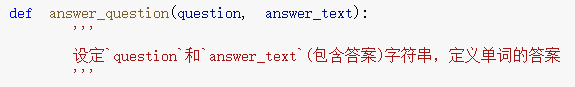
### 2.4.3 合并start和end



## 2.5 汇总画图和QA模型

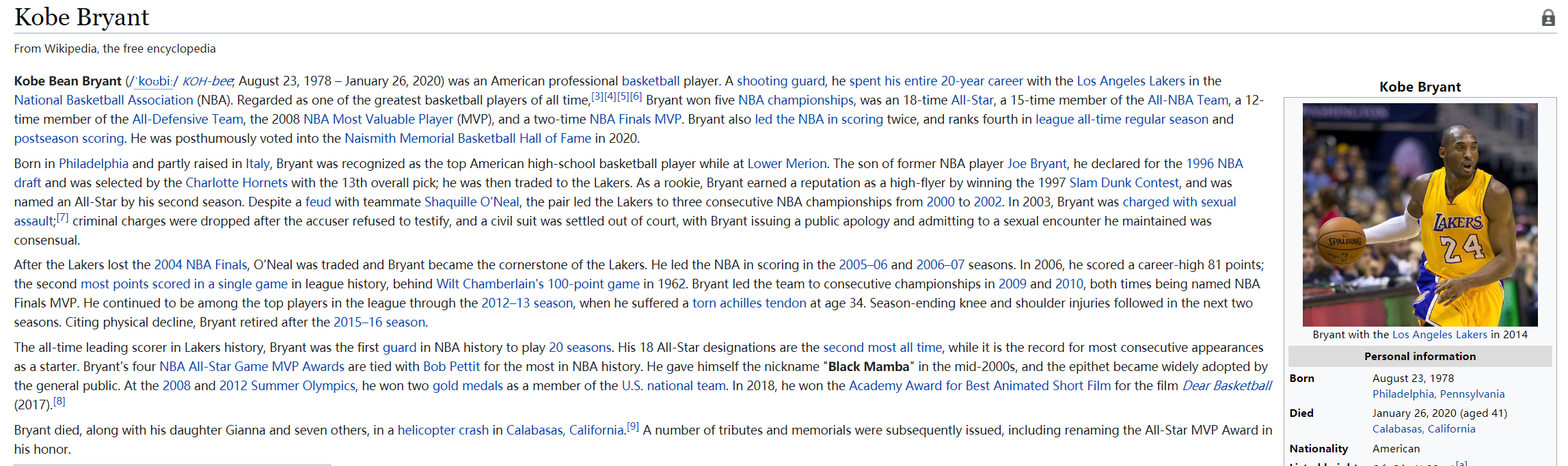
详情见附件代码。

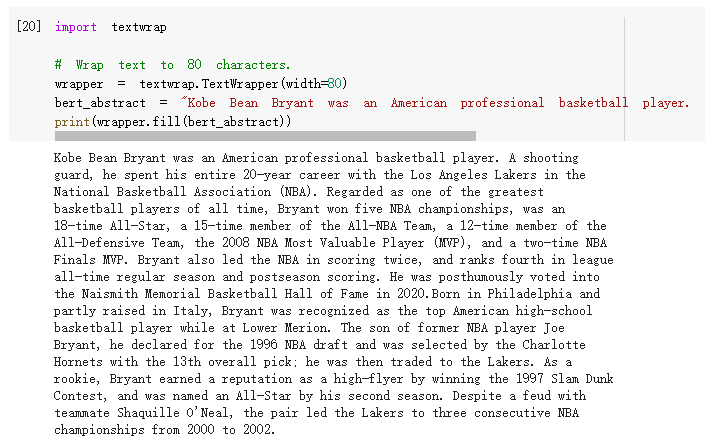




## 2.6 长文本的QA例子

我们将英文wiki网关于科比的信息获取下来，如下：

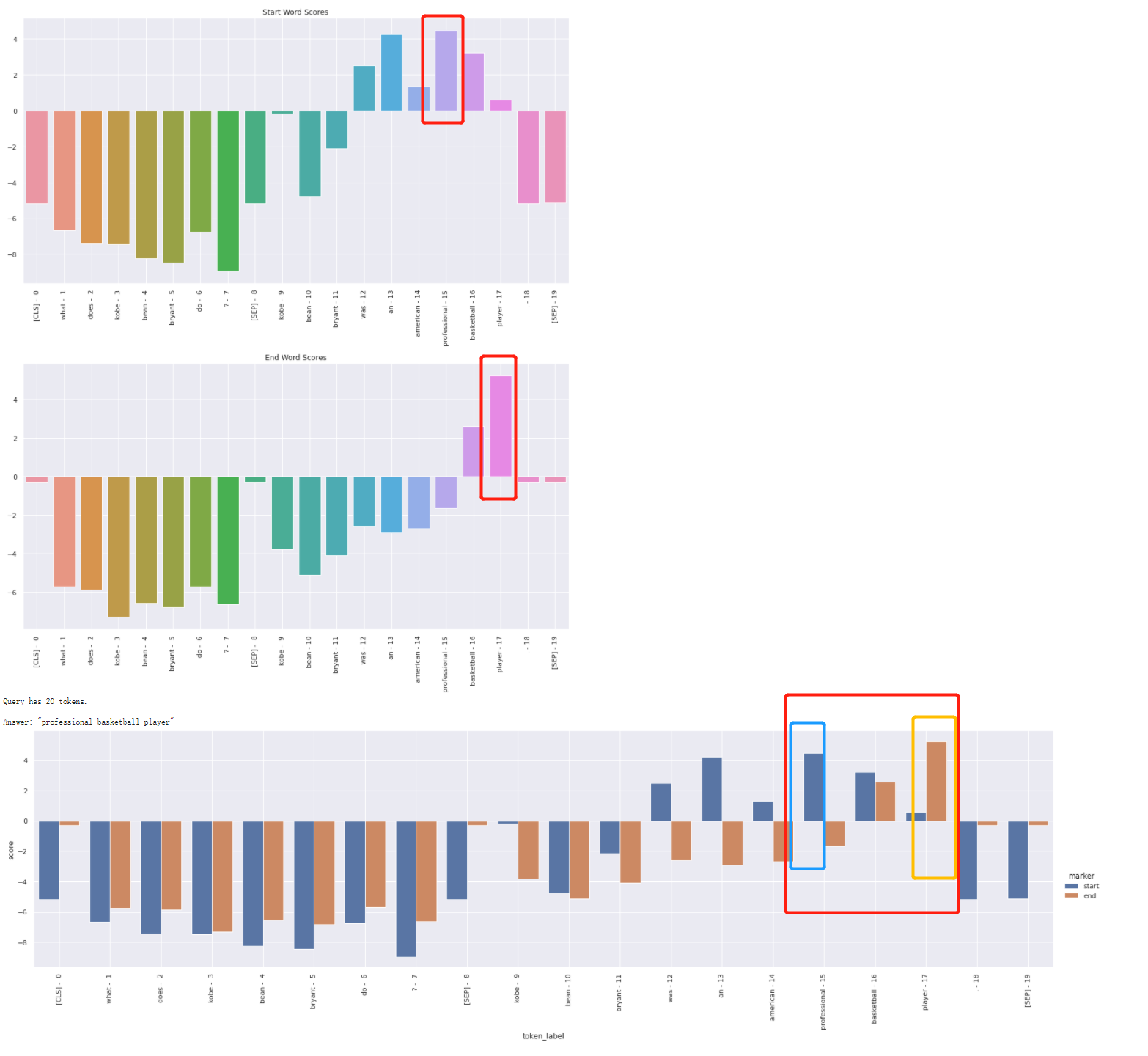




向模型提出了以下问题并都得到了正确答案：

|  |  |  |
| --- | --- | --- |
| 问题 | 答案 | 判断 |
| How many NBA championships  has Bryant won? | Answer: "five" | 正确 |
| How many times has Bryant  been an All-Star? | Answer: "18" | 正确 |
| How many times has kobe  been on the all-defensive team? | Answer: "12" | 正确 |
| what does Kobe Bean Bryant do? | Answer:  "an american professional basketball player" | 正确 |

给出最后一个问题的可视化过程：



# 3. 大规模数据集的尝试

## 3.1 数据来源及展示

①[HUNG-YI LEE (李宏毅)](https://speech.ee.ntu.edu.tw/~hylee/index.html)的HW7问答，数据集(包括训练集、验证集和测试集[文章+问题+答案])

* Training set: 26935 QA pairs
* Dev set: 3523 QA pairs
* Test set: 3492 QA pairs
* {train/dev/test}\_questions:
  + List of dicts with the following keys:
    - id (int)
    - paragraph\_id (int)
    - question\_text (string)
    - answer\_text (string)
    - answer\_start (int)
    - answer\_end (int)
* {train/dev/test}\_paragraphs:
  + List of strings
  + paragraph\_ids in questions correspond to indexs in paragraphs
  + A paragraph may be used by several questions

②Json格式

训练集（hw7\_train.json）和验证集（hw7\_dev.json）的格式如下表

|  |  |
| --- | --- |
| questions | "id": 0,              "paragraph\_id": 538,              "question\_text": "哪一個訓練中心的設備被使用來訓練網絡城與媒體城勞工的未來知識?",              "answer\_text": "杜拜知識村",              "answer\_start": 312,              "answer\_end": 316              "id": 2546,              "paragraph\_id": 764,              "question\_text": "瑞士近衛隊的制服代表著哪一個時期的風格?",              "answer\_text": "文藝復興時期",              "answer\_start": 71,              "answer\_end": 76 |
|  | …… |
| paragraphs |  |

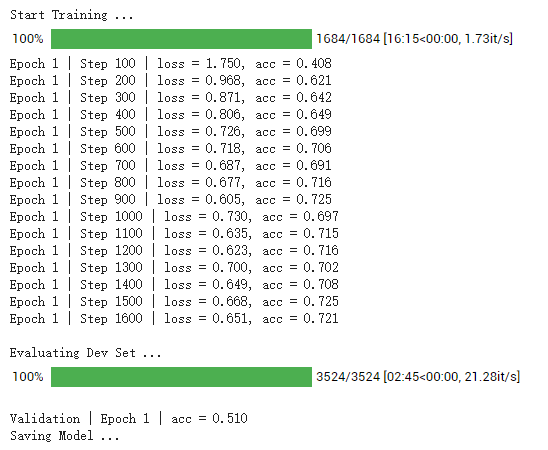
测试集（hw7\_test.json）格式如下表：

|  |  |
| --- | --- |
| questions | "id": 0,              "paragraph\_id": 792,              "question\_text": "士官長的頭盔上會有何裝飾物?",              "answer\_text": null,              "answer\_start": null,              "answer\_end": null |
| paragraphs |  |

## 3.2 程序结果及展示

### 3.2.1 训练模型并保存

如下图，训练出的模型在验证集上有0.721的accuracy。



### 3.2.2 测试集结果展示

然后我们将训练好的模型，应用于测试集上。以下为相应代码：

|  |
| --- |
| print("Evaluating Test Set ...")  result = []  model.eval()  with torch.no\_grad():      for data in tqdm(test\_loader):          output = model(input\_ids=data[0].squeeze(dim=0).to(device), token\_type\_ids=data[1].squeeze(dim=0).to(device),                         attention\_mask=data[2].squeeze(dim=0).to(device))          result.append(evaluate(data, output))  result\_file = "result.csv"  with open(result\_file, 'w') as f:      f.write("ID,Answer\n")  for i, test\_question in enumerate(test\_questions):          f.write(f"{test\_question['id']},{result[i].replace(',','')}\n")  print(f"Completed! Result is in {result\_file}") |

在运行以上代码后，生成结果文件（result.csv）。

如下图，留空的为找不到相应答案，其余的部分预测答案与人工作答相差无异。



# 附录

|  |
| --- |
| 深入Bert实战(Pytorch)----问答 fine-Tuning\_名字填充中的博客-CSDN博客  <https://blog.csdn.net/qq_42388742/article/details/113575739> |
| 【NLP-2019】解读BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding - 忆凡人生 - 博客园  <https://www.cnblogs.com/yifanrensheng/p/13510881.html> |
| Pytorch-Bert预训练模型的使用（调用transformers） - 最咸的鱼 - 博客园  <https://www.cnblogs.com/cxq1126/p/13517394.html>  GitHub - jessevig/bertviz: Tool for visualizing attention in the Transformer model (BERT, GPT-2, Albert, XLNet, RoBERTa, CTRL, etc.)  <https://github.com/jessevig/bertviz> |
| ***BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*** |