如何使用 Transformers 和 Tokenizers 从头开始训练新的语言模型

Denis Rothman。Denis Rothman 参考 Hugging Face 的笔记本，预训练了一个 transformer 模型。接下来的步骤是构建更大的数据集并测试多个 transformer 模型。

推荐理解这个笔记本。GPT-3 引擎的出现提供了一个可以超越许多训练过的 transformer 模型的 API。然而，要让 transformer 知道如何处理输入数据集，理解它们的训练过程是至关重要的。

这个笔记本中的 Transformer 模型名为 ***KantaiBERT***。***KantaiBERT*** 是一个以 RoBERTa Transformer 方式训练的，采用 DistilBERT 架构的模型。数据集是从 Jiumo Search 下载的Pride+and+Prejudice,To+Kill+A+Mockingbird,THE CATCHER IN THE RYE三本书编译而成的。

KantaiBERT 使用一个具有 8400 万参数的小模型进行预训练，采用了与 DistilBERT 相同的层数和头数，即 6 层、768 隐藏单元和 12 个注意力头。然后，***KantaiBERT*** 被微调用于下游的掩码语言建模任务。

Hugging Face 原始参考和注释：

笔记本版本（参考博客文章的原始链接 link）。

步骤1：加载数据集

```
from IPython.display import Image          # 这是用于在笔记本中渲染图像的
```

步骤1：加载数据集

```
#@title 步骤1：加载数据集
#1.使用 Colab 文件管理器加载 kant.txt
#2.从 GitHub 下载文件
#!curl -L https://github.com/qian-qiang/Good-Study-Day-Day-Up/tree/main/%E5%A4%A7%E6%A8%A1%E5%9E%8B%E5%AD%A6%E4%B9%A0/Book/others/kant.txt --outpu
```

步骤 2：安装 Hugging Face 的 Transformers

步骤 2：2024 年 6 月更新：安装 Hugging Face Transformers

```
#@title 步骤 2：2024 年 6 月更新：安装 Hugging Face Transformers
'''
# 这里我们不需要 TensorFlow
!pip uninstall -y tensorflow
# 从 master 分支安装 `transformers`
!pip install git+https://github.com/huggingface/transformers
!pip list | grep -E 'transformers|tokenizers'
# 本笔记本更新时的 transformers 版本 --- 2.9.1
# 本笔记本更新时的 tokenizers 版本 --- 0.7.0

'''
```

```
'\n# 这里我们不需要 TensorFlow\n!pip uninstall -y tensorflow\n# 从 master 分支安装 `transfor
mers`\n!pip install git+https://github.com/huggingface/transformers\n!pip list | grep -E 'tr
ansformers|tokenizers'\n# 本笔记本更新时的 transformers 版本 --- 2.9.1\n# 本笔记本更新时的 t
```

2023年6月更新来自 Hugging Face 问题 22816：

https://github.com/huggingface/transformers/issues/22816

“PartialState 导入作为依赖项在昨天添加到 transformers 的开发分支中。PartialState 是在 accelerate 0.17.0 版本中添加的，因此对于 transformers 的开发分支，需要 accelerate >= 0.17.0。

降级 transformers 版本可以移除导入 PartialState 的代码。”

Denis Rothman: 以下代码单元格导入了最新版本的 Hugging Face transformers，而无需降级。

为了适应 Hugging Face 的升级，使用 Google Colab Pro 激活了 GPU 加速器，配备了以下 NVIDIA GPU： GPU 名称：NVIDIA A100-SXM4-40GB

```
!pip install transformers
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.41.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.14.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.23.3)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.25.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (24.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2024.5.15)
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.31.0)
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.19.1)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.66.4)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.23.0->transformers) (20
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.23.0->transfo
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2024.6.2)
```

步骤 3：训练一个 tokenizer

步骤3：训练一个分词器

```
#@title 步骤3：训练一个分词器
%%time
from pathlib import Path
from tokenizers import ByteLevelBPETokenizer

# 获取所有 .txt 文件的路径
paths = [str(x) for x in Path(".").glob("**/*.txt")]

# 初始化分词器
tokenizer = ByteLevelBPETokenizer()

# 自定义训练参数
tokenizer.train(files=paths, vocab_size=52_000, min_frequency=2, special_tokens=[
        "<s>",
        "<pad>",
        "</s>",
        "<unk>",
        "<mask>",
])
```

```
CPU times: user 1min 21s, sys: 1.3 s, total: 1min 22s
Wall time: 52 s
```

步骤4：将文件保存到磁盘

步骤4：将文件保存到磁盘

```
#@title 步骤4：将文件保存到磁盘
import os

# 指定保存分词器的目录
token_dir = '/content/KantaiBERT'

# 如果目录不存在，则创建它
if not os.path.exists(token_dir):
        os.makedirs(token_dir)

# 保存分词器
tokenizer.save_model(token_dir)
```

```
['/content/KantaiBERT/vocab.json', '/content/KantaiBERT/merges.txt']
```

步骤5：加载训练好的分词器文件

步骤5 加载训练好的分词器文件

```
#@title 步骤5 加载训练好的分词器文件
from tokenizers.implementations import ByteLevelBPETokenizer
from tokenizers.processors import BertProcessing

tokenizer = ByteLevelBPETokenizer(
        "./KantaiBERT/vocab.json",
        "./KantaiBERT/merges.txt",
)
```

```
#然后使用词元分析器对序列进行编码
tokenizer.encode("The  Critique  of  Pure  Reason.").tokens
```

```
['The', 'ĠC', 'rit', 'ique', 'Ġof', 'ĠPure', 'ĠReason', '.']
```

```
#还可以看看该序列的词元数量：
tokenizer.encode("The  Critique  of  Pure  Reason.")
```

```
Encoding(num_tokens=8, attributes=[ids, type_ids, tokens, offsets, attention_mask, special_tokens_mask, overflowing])
```

```
#然后添加开始和结束词元
tokenizer._tokenizer.post_processor  =  BertProcessing(
        ("</s>",  tokenizer.token_to_id("</s>")),
        ("<s>",  tokenizer.token_to_id("<s>")),
)
tokenizer.enable_truncation(max_length=512)
```

步骤 6：检查资源约束：GPU 和 NVIDIA

步骤 6：检查资源约束：GPU 和 NVIDIA

```
#@title  步骤  6：检查资源约束：GPU  和  NVIDIA
!nvidia-smi
```

```
Thu Jun 13 08:45:39 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |                    0 |
| N/A   58C    P0              29W /  70W |   9799MiB / 15360MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
+---------------------------------------------------------------------------------------+
```

检查 PyTorch 是否能够使用 CUDA

```
#@title  检查  PyTorch  是否能够使用  CUDA
import  torch
torch.cuda.is_available()
```

```
True
```

步骤 7：定义模型配置

步骤 7：定义模型配置

```
#@title  步骤  7：定义模型配置
#词表大小设置为52000,包含12个注意力头和6层：
from  transformers  import  RobertaConfig

config  =  RobertaConfig(
        vocab_size=52_000,
        max_position_embeddings=514,
        num_attention_heads=12,
        num_hidden_layers=6,
        type_vocab_size=1,
)

print(config)
```

```
RobertaConfig {
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
```

```
  "eos_token_id": 2,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 514,
  "model_type": "roberta",
  "num_attention_heads": 12,
  "num_hidden_layers": 6,
  "pad_token_id": 1,
  "position_embedding_type": "absolute",
  "transformers_version": "4.41.2",
  "type_vocab_size": 1,
  "use_cache": true,
  "vocab_size": 52000
}
```


步骤 8：在 transformers 中重新加载分词器

步骤 8：在 transformers 中重新加载分词器

```
#@title 步骤 8：在 transformers 中重新加载分词器
from transformers import RobertaTokenizer
tokenizer = RobertaTokenizer.from_pretrained("./KantaiBERT", max_length=512)
```

步骤 9：从头开始初始化模型

步骤 9：从头开始初始化模型

```
#@title 步骤 9：从头开始初始化模型
from transformers import RobertaForMaskedLM

model = RobertaForMaskedLM(config=config)
print(model)
```

```
RobertaForMaskedLM(
  (roberta): RobertaModel(
    (embeddings): RobertaEmbeddings(
      (word_embeddings): Embedding(52000, 768, padding_idx=1)
      (position_embeddings): Embedding(514, 768, padding_idx=1)
      (token_type_embeddings): Embedding(1, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): RobertaEncoder(
      (layer): ModuleList(
        (0-5): 6 x RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): RobertaSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): RobertaIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): RobertaOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
  )
  (lm_head): RobertaLMHead(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (decoder): Linear(in_features=768, out_features=52000, bias=True)
```

```
    )
  )
```

探索模型参数

```
print(model.num_parameters())
#  =>  84,095,008  parameters
```

```
83504416
```

探索模型参数

```
#@title  探索模型参数
LP=list(model.parameters())
lp=len(LP)
print(lp)
for  p  in  range(0,lp):
    print(LP[p])
```

```
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.], requires_grad=True)
Parameter containing:
tensor([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       requires_grad=True)
```

计算模型参数数量

```
#@title  计算模型参数数量
np=0
for  p  in  range(0,lp):#number  of  tensors
    PL2=True
    try:
        L2=len(LP[p][0])  #check  if  2D
    except:
        L2=1                          #not  2D  but  1D
        PL2=False
    L1=len(LP[p])
    L3=L1*L2
    np+=L3                          #  number  of  parameters  per  tensor
    if  PL2==True:
        print(p,L1,L2,L3)    #  displaying  the  sizes  of  the  parameters
    if  PL2==False:
        print(p,L1,L3)    #  displaying  the  sizes  of  the  parameters

print(np)                           #  total  number  of  parameters
```

```
49 768 3072 2359296
50 768 768
51 768 768
52 768 768
53 768 768 589824
54 768 768
55 768 768 589824
56 768 768
57 768 768 589824
58 768 768
59 768 768 589824
60 768 768
61 768 768
62 768 768
63 3072 768 2359296
64 3072 3072
65 768 3072 2359296
66 768 768
67 768 768
68 768 768
69 768 768 589824
70 768 768
71 768 768 589824
72 768 768
73 768 768 589824
74 768 768
75 768 768 589824
76 768 768
77 768 768
78 768 768
79 3072 768 2359296
80 3072 3072
81 768 3072 2359296
82 768 768
83 768 768
84 768 768
85 768 768 589824
86 768 768
87 768 768 589824
88 768 768
89 768 768 589824
90 768 768
91 768 768 589824
92 768 768
93 768 768
94 768 768
95 3072 768 2359296
96 3072 3072
97 768 3072 2359296
98 768 768
99 768 768
100 768 768
101 52000 52000
102 768 768 589824
103 768 768
104 768 768
105 768 768
83504416
```

步骤 10：构建数据集

步骤 10：构建数据集

```
#@title  步骤  10：构建数据集
%%time
from  transformers  import  LineByLineTextDataset

dataset  =  LineByLineTextDataset(
        tokenizer=tokenizer,
        file_path="./kant.txt",
        block_size=128,
)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/data/datasets/language_modeling.py:119: FutureWarning: This dataset will be removed from th
  warnings.warn(
CPU times: user 1min 55s, sys: 738 ms, total: 1min 56s
Wall time: 1min 59s
```

步骤 11：定义数据收集器

步骤 11：定义数据收集器

```
#@title  步骤  11：定义数据收集器
from  transformers  import  DataCollatorForLanguageModeling

data_collator  =  DataCollatorForLanguageModeling(
        tokenizer=tokenizer,  mlm=True,  mlm_probability=0.15
)
```

步骤 12：初始化训练器

```
#如果下步骤12初始化训练器失败提示accelerate版本太低就运行此代码，并从新连接
#!pip  install  accelerate

#import  accelerate

#print(accelerate.__version__)
```

初始化训练器

```
#@title  初始化训练器

from  transformers  import  Trainer,  TrainingArguments

training_args  =  TrainingArguments(
        output_dir="./KantaiBERT",
        overwrite_output_dir=True,
        num_train_epochs=1,
        per_device_train_batch_size=64,
        save_steps=10_000,
        save_total_limit=2,
)

trainer  =  Trainer(
        model=model,
        args=training_args,
        data_collator=data_collator,
        train_dataset=dataset,
)
```

步骤 13：预训练模型

步骤 13：预训练模型

```
#@title  步骤  13：预训练模型
%%time
trainer.train()
```

[48/48 00:59, Epoch 1/1]

Step	Training Loss

```
CPU times: user 58.7 s, sys: 778 ms, total: 59.5 s
Wall time: 1min 1s
TrainOutput(global_step=48, training_loss=9.308392206827799, metrics={'train_runtime':
60.6015, 'train_samples_per_second': 49.883, 'train_steps_per_second': 0.792, 'total_flos':
```

步骤 14：将最终模型（包括分词器和配置）保存到磁盘

步骤 14：将最终模型（包括分词器和配置）保存到磁盘

```
#@title 步骤 14：将最终模型（包括分词器和配置）保存到磁盘
trainer.save_model("./KantaiBERT")
```

步骤 15：使用 FillMaskPipeline 进行语言建模

步骤 15：使用 FillMaskPipeline 进行语言建模

```
#@title 步骤 15：使用 FillMaskPipeline 进行语言建模
from transformers import pipeline

fill_mask = pipeline(
        "fill-mask",
        model="./KantaiBERT",
        tokenizer="./KantaiBERT"
)


fill_mask("I don t currently have the time for giving a proper long <mask>.")
```

```
[{'score': 0.018490975722670555,
  'token': 262,
  'token_str': '.',
  'sequence': 'I don t currently have the time for giving a proper long..'},
 {'score': 0.01700388640165329,
  'token': 280,
  'token_str': ' I',
  'sequence': 'I don t currently have the time for giving a proper long I.'},
 {'score': 0.004652389325201511,
  'token': 18,
  'token_str': '.',
  'sequence': 'I don t currently have the time for giving a proper long..'},
 {'score': 0.003358564805239439,
  'token': 278,
  'token_str': ' :',
  'sequence': 'I don t currently have the time for giving a proper long :.'},
 {'score': 0.002129371277987957,
  'token': 288,
  'token_str': '..',
  'sequence': 'I don t currently have the time for giving a proper long...'}]
```