transformers库中加载和保存模型的方法和源码简单分析

6 分钟前

武辰
数学话题下的优秀答主
+ 关注

加载模型

以一个bert模型为例介绍加载模型的方式。该bert模型简称“rbt3”，在huggingface的网址为：huggingface.co/hfl/rbt3

有多种方式来加载模型。这里介绍常用的两种途径。

使用模型名字的方法

第一种方式，直接使用模型的简称来加载。

```
model = AutoModel.from_pretrained("hfl/rbt3")
```

或者

```
model = AutoModel.from_pretrained("hfl/rbt3",
cache_dir='D:/Users/Desktop/aigc/cache/rbt/')
```

代码会尝试从 Hugging Face的模型仓库中下载并加载模型 'hfl/rbt3'。参数 cache_dir 指定了一个本地缓存目录，用于存放下载的模型文件。如果模型已经被下载到指定的 cache_dir，函数将会使用本地的文件，避免重复下载。如果没有在本地找到，则会从Hugging Face的在线模型仓库中下载模型。

如果你去cache_dir里面找rbt3模型，可能会发现模型结构长这样：

真正的模型文件在blobs路径里。snapshots里面是快捷方式。我以前踩过一个坑——由于linux服务器无法连接huggingface，我先在windows下载好模型，然后把cache_dir里的文件夹原封不动地传到linux服务器里，发现并不能加载模型。解决方法是，依照windows里的快捷方式，在linux里加入软链接。

传入模型路径的方法

第二种方式，传入模型在本地的路径。

```
model = AutoModel.from_pretrained('D:/Users/D
```

这行代码会从本地文件系统的给定路径加载模型。路径 "D:/Users/Desktop/aigc/rbt/" 应该包含一个已经下载好的模型，这个目录里通常包含如 config.json 和预训练模型的权重文件，可能是 pytorch_model.bin，以及其他必要的文件，如分词器文件。

'D:/Users/Desktop/aigc/rbt/'路径下的结构如下图所示：

这些文件是如何获取的呢？通过执行 git clone 命令，将 Hugging Face 的存储库克隆至本地。

保存模型

通过save_pretrained()保存整个模型

save_pretrained方法 是保存预训练模型最常用的方法。这将保存模型的权重和配置到一个文件夹中，可以在之后使用from_pretrained方法加载。

```
model = AutoModel.from_pretrained("hfl/rbt3")
model.save_pretrained(save_directory)
```

进入save_directory路径，可以看到两个文件，一个是配置文件config.json，一个是模型的权重文件。

配置文件除了config.json，可能还会有其他配置文件。由于本例子中的模型比较简单，所以一个配置文件就能存储必要的信息。

模型权重文件默认是pytorch_model.bin。但是在一些transformers版本中，可能保存的是model.safetensors文件。

关于safetensors，以下是GPT的答案。

Hugging Face 在 2022 年推出了一种新的文件格式 “safetensors”，它是为了存储和分享深度学习模型权重提供的一种替代方法。“safetensors” 的设计目的是为了改善模型文件的兼容性、安全性和效率。下面详细解释这两种文件格式的差异：

bin 文件:

- 这是传统的二进制文件格式，通常由 PyTorch 和其他深度学习框架用来保存模型权重（state_dict）。
- .bin 文件通常与 PyTorch 的 torch.save 方法一起使用，它使用了 Python 的 pickle 序列化协议。
- pickle 可以容易地受到版本冲突和不兼容的影响，因为它保存了对象的全状态，包括 Python 环境等细节，可能导致在不同环境中反序列化出错。
- .bin 文件在保存和加载时不进行文件内容的完整性校验。

safetensors 文件:

- “safetensors” 是由 Hugging Face 推出的一种新文件格式，目的是提供更加高效、安全且框架无关的方式来保存和加载模型权重。
- 这种格式设计为与 PyTorch、TensorFlow、JAX 等深度学习框架的版本和实现无关。
- “safetensors” 文件内部使用了更强大的版本控制，存储保存的张量（tensors）时可以包含更多元数据，有助于在模型间传递信息，如张量的形状、数据类型等。
- 它还具有一定的完整性校验特性，能够校验文件内容是否完整且未被篡改。
- “safetensors” 文件格式还支持更高效的压缩技术，为模型降低了存储需求。

在 Hugging Face 的 transformers 库中，通常保存模型的权重时，默认使用的是 .bin 文件格式。然而，如果想利用 “safetensors” 文件格式的优势，库也提供了相应的接口支持。

当使用 “safetensors” 格式的时候，相对于 .bin 格式，你可以期待获得更好的跨版本和跨框架的兼容性和更重的安全保证。如果你需要在不同深度学习框架之间共享模型权重，或者担心文件完整性问题，使用 “safetensors” 可能是一个更好的选择。

在训练中保存checkpoint

如何在训练过程中有效地保存模型的权重?

通过合适地配置 Trainer 类，可以实现在训练的关键节点自动保存模型状态。当调用 trainer.train() 方法并设置了保存策略时，模型的状态将按照指定的策略被保留。以下面的代码为例，在每个训练周期（epoch）完成后，模型权重将被自动保存：

```
from transformers import TrainingArguments


train_args = TrainingArguments(
    output_dir="./checkpoints",
    per_device_train_batch_size=64,
    per_device_eval_batch_size=128,
    logging_steps=500,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    save_total_limit=3,
    learning_rate=2e-5,
    weight_decay=0.01,
    metric_for_best_model="f1",
    load_best_model_at_end=True
)
```

模型在transformers库源码中的trainer.py文件中的_save()方法执行保存操作。

该方法的源码如下:

```
def _save(self, output_dir: Optional[str] = N
    # If we are executing this function, we a
    output_dir = output_dir if output_dir is
    os.makedirs(output_dir, exist_ok=True)
    logger.info(f"Saving model checkpoint to


    supported_classes = (PreTrainedModel,) if
    # Save a trained model and configuration
    # They can then be reloaded using `from_p
    if not isinstance(self.model, supported_c
        if state_dict is None:
            state_dict = self.model.state_dic


        if isinstance(unwrap_model(self.model
            unwrap_model(self.model).save_pre
                output_dir, state_dict=state_
            )
        else:
            logger.info("Trainer.model is not
            if self.args.save_safetensors:
                safetensors.torch.save_file(s
            else:
                torch.save(state_dict, os.pat
    else:
        self.model.save_pretrained(
            output_dir, state_dict=state_dict
        )
```

这份源代码有几个值得说的地方。

（1）定义一个包含支持保存的模型类的元组

首先看这一行：

```
supported_classes = (PreTrainedModel,) if not
```

这定义了一个包含支持保存的模型类的元组。默认情况下只包含 PreTrainedModel 类。如果 PEFT（参数高效微调框架）可用，也会包含 PeftModel。

> PreTrainedModel 是 Hugging Face transformers 库中实现的一个 Python 类，它提供了预训练语言模型通用的基础架构和接口。这个类封装了加载预训练参数、保存模型、推理和微调模型等功能。
> PreTrainedModel 类在 Hugging Face transformers 库中提供了预训练模型的通用基础设施和共享接口，使得在下游任务中重用、微调和部署预训练模型变得非常简单。它是使用这个库时最重要和常用的组件之一。

（2）判断需要保存的模型是否在元组里

```
if not isinstance(self.model, supported_class
```

在本例中，self.model的类型是BertForSequenceClassification，它属于PreTrainedModel类。

（3）调用save_pretrained

```
self.model.save_pretrained(output_dir,
state_dict=state_dict,
safe_serialization=self.args.save_safetensors
```

这一行代码调用了save_pretrained，经过这一行代码后，对应的checkpoints文件夹将会保存config.json文件和模型权重文件（bin或者safetensors）

（4）保存分词器

```
if self.tokenizer is not None:
            self.tokenizer.save_pretrained(ou
```

如果训练器有分词器，则分词器也会被一同保存到指定的目录。

（5）保存训练参数

```
torch.save(self.args, os.path.join(output_dir
```

我们保存训练参数到输出目录，这是一个好的实践，因为它可以帮助你记住训练时使用的具体参数（这些参数用于配置训练过程的各个方面，如批次大小、学习率等）。

这一行将TrainingArguments保存在training_args.bin文件中。

上面说到_save()保存了三个文件：config.json/pytorch_model.bin/training_args.bin，事实上，在源码中，不仅仅在_save()方法中保存文件。**_save_checkpoint()方法调用了save_model()方法，而save_model()方法调用了_save()方法，且**save_model()方法中除了调用_save()方法，似乎没有其他保存到本地的操作。**另外，**_save()方法中会调用save_pretrained()方法。

调用了多个不同的方法，比如_save_checkpoint()方法中也保存了一些文件：

- optimizer.pt: 此文件是 PyTorch 序列化的优化器状态的二进制文件，包括所有的优化器参数（学习率等），累积的梯度信息等。在训练过程中保存优化器状态可以用于继续训练或调试。

以下内容由GPT生成：

> 在深度学习中，optimizer.pt 文件包含优化器的状态，这通常指的是用来更新模型权重的内部变量的状态。对于不包含优化器状态的基本梯度下降方法来说，没有累积梯度（也称为动量）信息保存。然而，对于像 Adam、SGD with Momentum（带动量的随机梯度下降）或 RMSprop 等更复杂的优化方法，累积的梯度信息（或者说历史梯度信息）是非常重要的优化器状态的一部分。
> 累积梯度信息的作用包括：
> 平滑梯度：累积的梯度信息通常用于计算当前更新步骤的平滑或指数加权移动平均，这致力于减少梯度的方差，从而使得优化过程更加稳定。
> 加速收敛：通过引入先前梯度的信息，优化算法可能在某些方向上加速，这对于加速模型的收敛和提高训练效率至关重要。
> 逃离局部最小值/鞍点：累积梯度信息可能有助于优化器从局部最小值或鞍点中逃脱，这在高维空间中尤为重要。
> 自适应学习率：某些优化算法（如 Adam）使用过去梯度的信息来自动调节每个参数的学习率，这有助于在不引入额外超参数的情况下处理不同参数的不同学习率需求。
> 当你保存优化器状态（如使用 torch.save(optimizer.state_dict(), 'optimizer.pt')），你实际上是在保存这些重要的内部变量，包括梯度的累积值、更新步数、学习率等，这些都是在训练期间对每个权重更新至关重要的信息。
> 正是由于这些原因，如果你需要暂停训练并在未来某一时间点恢复，保存优化器的状态是必要的。这样做可以确保训练过程的连续性，使得优化器能够像暂停时一样拥有相同的内部状态，从而继续进行模型训练而不会有任何状态丢失所导致的影响。

- rng_state.pth: 该文件包含随机数生成器的状态信息。这对于确保训练的再现性很重要，因为它允许你从相同的随机状态重新开始训练。
- scheduler.pt: 存储了学习率调度器状态的二进制文件。调度器负责在训练过程中调整学习率，保存其状态允许在训练过程中准确地重建学习率进度。
- trainer_state.json: 这个 JSON 文件包含了 Trainer 的状态，例如当前 epoch、全局步数、最佳模型的得分等信息。它有助于了解训练进度和性能。

综上，每个checkpoints一共包含7个文件。这些文件为恢复训练状态提供了完整的上下文，包括模型权重、训练配置、优化器和调度器状态、随机数生成器状态等。通过这些文件可以在训练被中断后，从同一点重新开始训练过程。下图显示了七个文件的基本信息：

编辑于 2024-06-14 22:33 · IP 属地广东

评论 8

写评论

千城
我以前踩过一个坑——由于linux服务器无法连接huggingface，我先在windows下载好模型，然后把...
02-01 · IP 属地湖南 回复 喜欢

抽空吃个早餐 ▸ 你好
放到第代码首行位置
05-23 · IP 属地上海 喜欢

你大爷还是大爷
我也遇到了同样的问题，请问解决了吗
05-23 · IP 属地广东 喜欢