《机器学习编程实践》课程——DAY 6

课程内容

- **直接使用 hugging face 下载模型和数据**
- **修改 hugging face 封装内部框架**

一、 直接使用 hugging face

模型权重的下载和加载——不推荐手动下载、推荐使用**代码**下载

1 hugging face 构成

- **Models**
- **Datasets**
- **Community** ——包含各种教程帖
- **Docs** ——包含 pytorch 各种库

2 pytorch 简单库介绍

每个库都会有官方介绍和教程

2.1 Safetensors

将 tensor 类型数据转换成 safetensor 类型，使得数据更安全
存储为 safetensor 的模型的权重都可以非常方便地进行加载

2.2 Datasets

数据集的调度过程
作用类似自己写的"train_datasets"，之后直接对数据进行加载即可

Docs-Datasets-TUTORIALS-Load a dataset from the Hub ——介绍了加载数据集的简单教程：

- **load_dataset_builder**: 用于在下载数据集前检查其具体属性，包括 description 和 features
- **load_dataset**: 用于加载数据集

- **get_dataset_split_names**: 用于划分数据集的特定子集，如训练和测试

  Docs-Datasets-TUTORIALS-process ——介绍了数据预处理的简单教程:

- **map 函数**：对整个数据集进行标记（dataset 中已经包含这个方法）

2.3 Diffusers

文本到图像的生成模型 diffusion 的封装，封装成了 pipeline

Docs-Diffusers-API-pipelines ——各种模型具体的调用方法，找到 example 可以找到具体使用案例

2.4 Transformers

负责整体的训练以及以语言模型为基础的东西

Docs-Transformers-API：包含各种常见模型，包括 GPT 系列

2.5 PEFT

高效微调——在大模型微调时加入此组件来节省参数

最主要的方法：

- **prompt-based 方法**
- **lora 方法**

  lora：模型微调之后的权重等于原始权重加上两个小矩阵相乘的权重；在训练过程中给只需要保留两个矩阵的权重，不需要额外保留真实模型的权重

2.6 Accelerate

并行计算，可以把单卡代码修改成多卡

二、 修改库

以一个简单的例子为例：

1 加载模型

```
from transformers import T5Tokenizer, T5Config
from T5 import T5ForConditionalGeneration

# 加载T5模型
```

```
tokenizer = T5Tokenizer.from_pretrained(model_name, cache_dir=cache_dir)
model = T5ForConditionalGeneration.from_pretrained(model_name,
    cache_dir=cache_dir, n_prompt_tokens=n_prompt_tokens)
```

2 添加模型的参数

```
# 将不同的参数定义为不同的类，如：模型相关参数、数据训练相关参数
class ModelArguments:
   """
   Arguments pertaining to which model/config/tokenizer we are going to
       fine-tune from.
   """
   # 这里就定义了三个参数
   model_name_or_path: str = field(
      default=model_name,
      metadata={"help": "Path to pretrained model or model identifier from
          huggingface.co/models"}
   )
   tokenizer_name: Optional[str] = field(
      default=model_name, metadata={"help": "Pretrained tokenizer name or
          path if not the same as model_name"}
   )
   cache_dir: Optional[str] = field(
      default=cache_dir, metadata={"help": "Where do you want to store the
          pretrained models downloaded from s3"}
   )


class DataTrainingArguments:
   """
   Arguments pertaining to what data we are going to input our model for
       training and eval.
   """
   # 训练数据集的路径
   train_file_path: Optional[str] = field(
      default='train_data.pt',
      metadata={"help": "Path for cached train dataset"},
   )
   # 验证数据集的路径
```

```
    valid_file_path: Optional[str] = field(
        default='valid_data.pt',
        metadata={"help": "Path for cached valid dataset"},
    )
    # 源文本的最大长度
    max_len: Optional[int] = field(
        default=32,
        metadata={"help": "Max input length for the source text"},
    )
    # 目标文本的最大长度
    target_max_len: Optional[int] = field(
        default=16,
        metadata={"help": "Max input length for the target text"},
    )
```

3 主体架构

```
# 最终参数, ModelArguments为原有参数
parser = HfArgumentParser((ModelArguments, DataTrainingArguments,
    TrainingArguments))

prompt_model = prompt_T5_model(n_prompt_tokens=n_prompt_tokens,
    model=model, config=config)

trainer = Trainer(
    model=prompt_model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=valid_dataset,
    data_collator=T2TDataCollator(),
)
```