05

# 基础组件之Datasets

（1）Datasets简介

（2）Datasets基本使用

（3）Datasets加载本地数据集

（4）Datasets + DataCollator模型微调代码优化

# 基础组件之Datasets

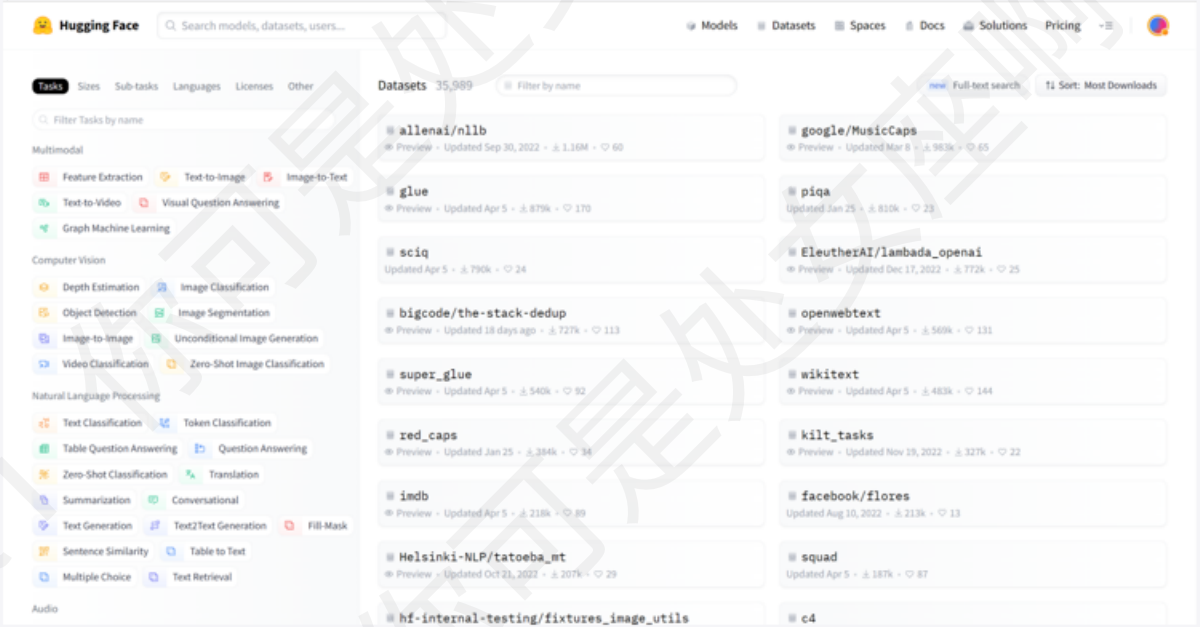
## Datasets

- **简介**
  - datasets库是一个非常简单易用的数据集加载库，可以方便快捷的从本地或者HuggingFace Hub加载数据集
- **公开数据集地址**
  - https://huggingface.co/datasets
- **文档地址**
  - https://huggingface.co/docs/datasets/index

# 基础组件之Datasets

## Datasets基本使用

- 加载在线数据集（load_dataset）
- 加载数据集某一项任务（load_dataset）
- 按照数据集划分进行加载（load_dataset）
- 查看数据集（index and slice）
- 数据集划分（train_test_split）
- 数据选取与过滤（select and filter）
- 数据映射（map）
- 保存与加载（save_to_disk / load_from_disk）



# 基础组件之Datasets

## Datasets加载本地数据

- **直接加载文件作为数据集**
  - CSV、JSON
- **加载文件夹内全部文件作为数据集**
- **通过预先加载的其他格式转换加载数据集**
  - dict、pandas、list
- **通过自定义加载脚本加载数据集**
  - def _info(self)
  - def _split_generators(self, dl_manager)
  - def _generate_examples(self, filepath)

```
class CMRC2018TRIAL(datasets.GeneratorBasedBuilder):

    def _info(self) -> DatasetInfo:
        """
            info方法, 定义数据集的信息,这里要对数据的字段进行定义
        :return:
        """
        return datasets.DatasetInfo(…

    def _split_generators(self, dl_manager: DownloadManager):
        """
            返回datasets.SplitGenerator
            涉及两个参数: name和gen_kwargs
            name: 指定数据集的划分
            gen_kwargs: 指定要读取的文件的路径, 与_generate_examples的入参数一致
        :param dl_manager:
        :return: [ datasets.SplitGenerator ]
        """
        return [datasets.SplitGenerator(name=datasets.Split.TRAIN,
                                         gen_kwargs={"filepath": "./cmrc2018_trial.json"})]

    def _generate_examples(self, filepath):
        """
            生成具体的样本, 使用yield
            需要额外指定key, id从0开始自增就可以
        :param filepath:
        :return:
        """
        # Yields (key, example) tuples from the dataset
        with open(filepath, encoding="utf-8") as f:…
```

# 基础组件之Model

## 模型微调代码优化

- **任务类型**
  - 文本分类
- **使用模型**
  - hfl/rbt3
- **优化内容**
  - Datasets数据集加载
  - DataCollatorWithPadding

### Step2 加载数据集

```
import torch
from datasets import load_dataset
from transformers import  DataCollatorWithPadding

tokenizer = AutoTokenizer.from_pretrained("hfl/rbt3")
```

```
dataset = load_dataset("csv", data_files="./04-datasets/ChnSentiCorp_htl_all.csv", split='train')
dataset = dataset.filter(lambda x: x["review"] is not None)
datasets = dataset.train_test_split(test_size=0.1)
datasets
```

*输出已折叠 ...*

### Step3 数据预处理

```
def process_function(examples):
    tokenized_examples = tokenizer(examples["review"], max_length=128, truncation=True)
    tokenized_examples["labels"] = examples["label"]
    return tokenized_examples

tokenized_datasets = datasets.map(process_function, batched=True, remove_columns=dataset.column_names)
tokenized_datasets
```