Huggingface 的介绍，使用

（最强 Huggingface 入门手册）

1.Huggingface 的简介

Huggingface 即是网站名也是其公司名，随着 transformer 浪潮，Huggingface 逐步收纳了众多最前沿的模型和数据集等有趣的工作，与 transformers 库结合，可以快速使用学习这些模型。目前提到 NLP 必然绕不开 Huggingface。

2. Huggingface 的具体介绍

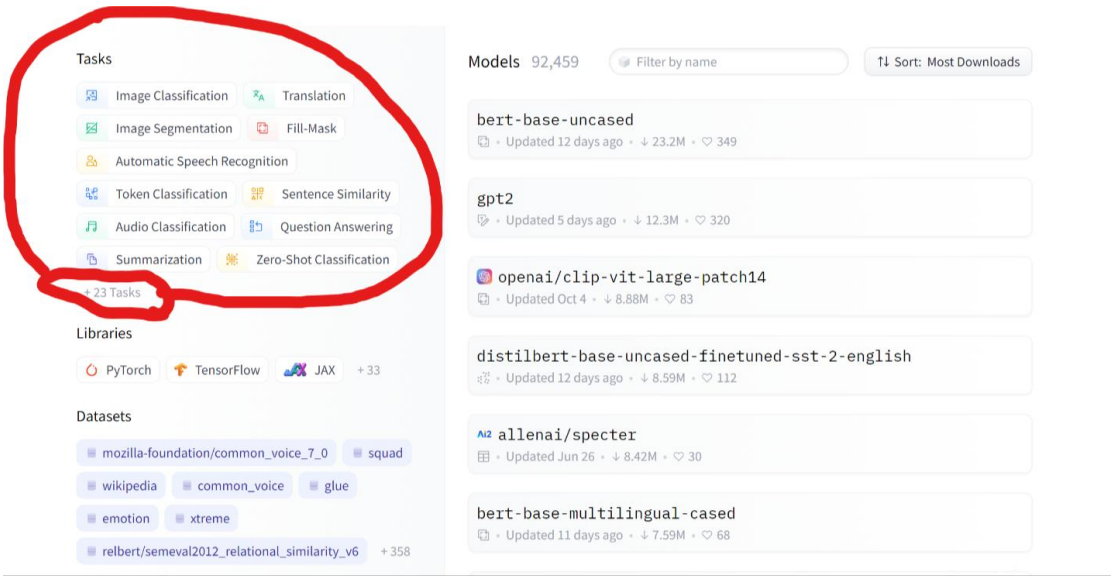
进入 Huggingface 网站,如下图所示。

其主要包含：

Models（模型），包括各种处理 CV 和 NLP 等任务的模型，上面模型都是可以免费获得
Datasets（数据集），包括很多数据集
Spaces（分享空间），包括社区空间下最新的一些有意思的分享，可以理解为 huggingface 朋友圈
Docs（文档，各种模型算法文档），包括各种模型算法等说明使用文档
Solutions（解决方案，体验等），包括 others
Pricing（dddd），懂的都懂

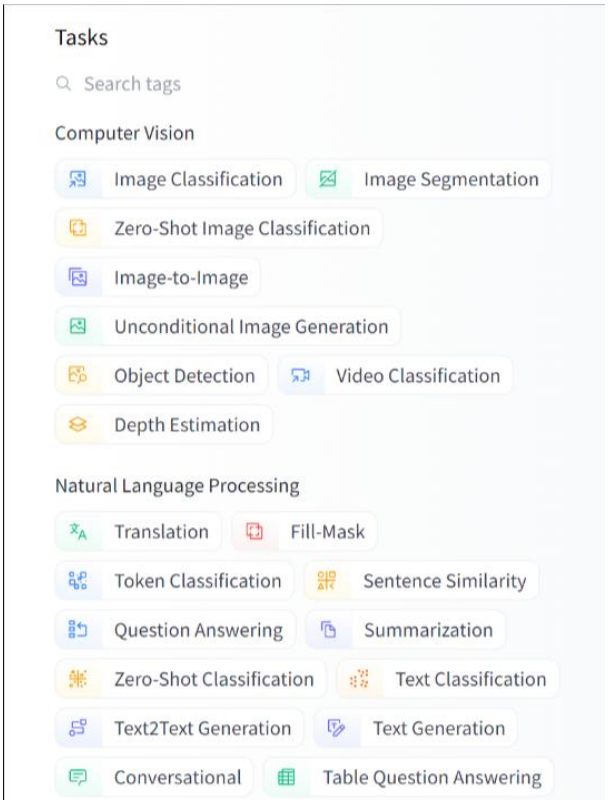
3.Huggingface 的 Models

点开 Models。可以看到下图的任务，再点开+23 Tasks。



可以看到下图所有的任务。

其中，主要包括计算机视觉、自然语言处理、语音处理、多模态、表格处理、强化学习。



展开介绍：

Computer Vision（计算机视觉任务）：包括 Image Classification（图像分类），Image Segmentation（图像分割）、zero-Shot Image Classification（零样本图像分类）、Image-to-Image（图像到图像的任务）、Unconditional Image Generation（无条件图像生成）、Object Detection(目标检测)、Video Classification（视频分类）、Depth Estimation(深度估计，估计拍摄者距离图像各处的距离)

Natural Language Processing（自然语言处理）：包括 Translation（机器翻译）、Fill-Mask(填充掩码，预测句子中被遮掩的词)、Token Classification（词分类）、Sentence Similarity（句子相似度）、Question Answering（问答系统），Summarization（总结，缩句）、Zero-Shot Classification (零样本分类)、Text Classification (文本分类)、Text2Text (文本到文本的生成)、Text Generation（文本生成）、Conversational（聊天）、Table Question Answer（表问答，1.预测表格中被遮掩单词 2.数字推理，判断句子是否被表格数据支持）

Audio（语音）：Automatic Speech Recognition（语音识别）、Audio Classification（语音分类）、Text-to-Speech（文本到语音的生成）、Audio-to-Audio（语音到语音的生成）、Voice Activity Detection（声音检测、检测识别出需要的声音部分）

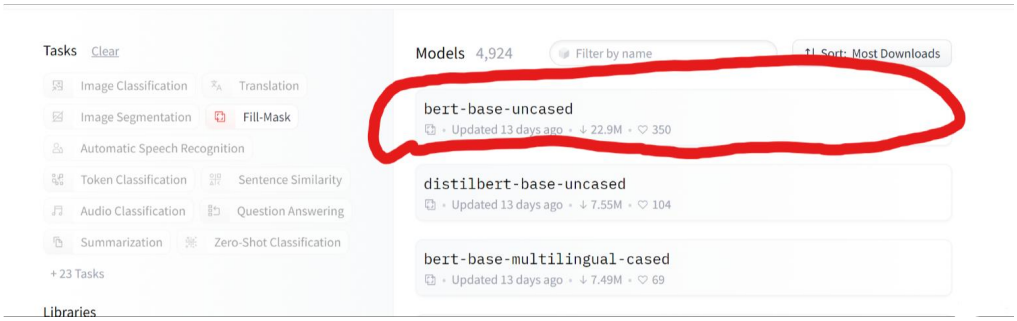
Multimodal（多模态）：Feature Extraction（特征提取）、Text-to-Image（文本到图像）、Visual Question Answering（视觉问答）、Image2Text（图像到文本）、Document Question Answering（文档问答）

Tabular（表格）：Tabular Classification（表分类）、Tabular Regression（表回归）

Reinforcement Learning（强化学习）：Reinforcement Learning（强化学习）、Robotics（机器人）

模型的使用

一般来说，页面上会给出模型的介绍。例如，我们打开其中一个 fill-mask 任务下下载最多的模型 bert-base-uncased。



可以看到模型描述：

Model description

BERT is a transformers model pretrained on a large corpus of English data in a self-supervised fashion. This means it was pretrained on the raw texts only, with no humans labeling them in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those texts. More precisely, it was pretrained with two objectives:

- Masked language modeling (MLM): taking a sentence, the model randomly masks 15% of the words in the input then run the entire masked sentence through the model and has to predict the masked words. This is different from traditional recurrent neural networks (RNNs) that usually see the words one after the other, or from autoregressive models like GPT which internally masks the future tokens. It allows the model to learn a bidirectional representation of the sentence.
- Next sentence prediction (NSP): the models concatenates two masked sentences as inputs during pretraining. Sometimes they correspond to sentences that were next to each other in the original text, sometimes not. The model then has to predict if the two sentences were following each other or not.

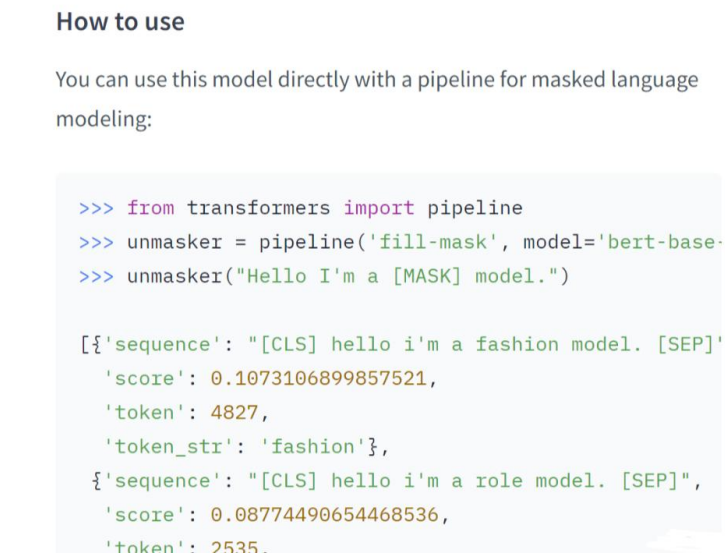
使用方法-1

需要提前安装 transformers 库，可以直接 pip install transformers 安装。还有 Pytorch 或 TensorFlow 库，读者自行下载。
下载完后可以使用 pipeline 直接简单的使用这些模型。第一次执行时 pipeline 会加载模型，模型会自动下载到本地，可以直接用。
第一个参数是任务类型，第二个是具体模型名字。

```
from transformers import pipeline
unmasker = pipeline('fill-mask',model='bert-base-uncased') unmasker("Hello I'm a [MASK] model.")
```

模型下载在本地的地址：
C:\Users\【自己的用户名】\.cache\huggingface\hub 、

当然，不同模型使用方法略有区别，直接通过页面学习或文档学习最好



可以自定义加载输入分词器:使用 AutoTokenizer

```
from transformers import AutoTokenizer #下面这种方式可以自动加载 bert-base-uncased 中使用的分词器
tokenizer=AutoTokenizer.from_pretrained("bert-base-uncased")
```

可以自定义加载模型结构:使用 AutoModel

不包括输入分词器和输出部分！！！

```
from transformers import AutoModel

#下面这种方式可以自动加载 bert-base-uncased 中使用的模型，没有最后的全连接输出层和 softmax

model=AutoModel.from_pretrained("bert-base-uncased")
```

可以自定义加载模型和输出部分:使用 AutoModelForSequenceClassification 等

```
from transformers import AutoModelForSequenceClassification

#下面这种方式可以自动加载 bert-base-uncased 中使用的模型（包括了输出部分），有最后的全连接输出层
```

```
model=AutoModel.AutoModelForSequenceClassification("bert-base-uncased")
```

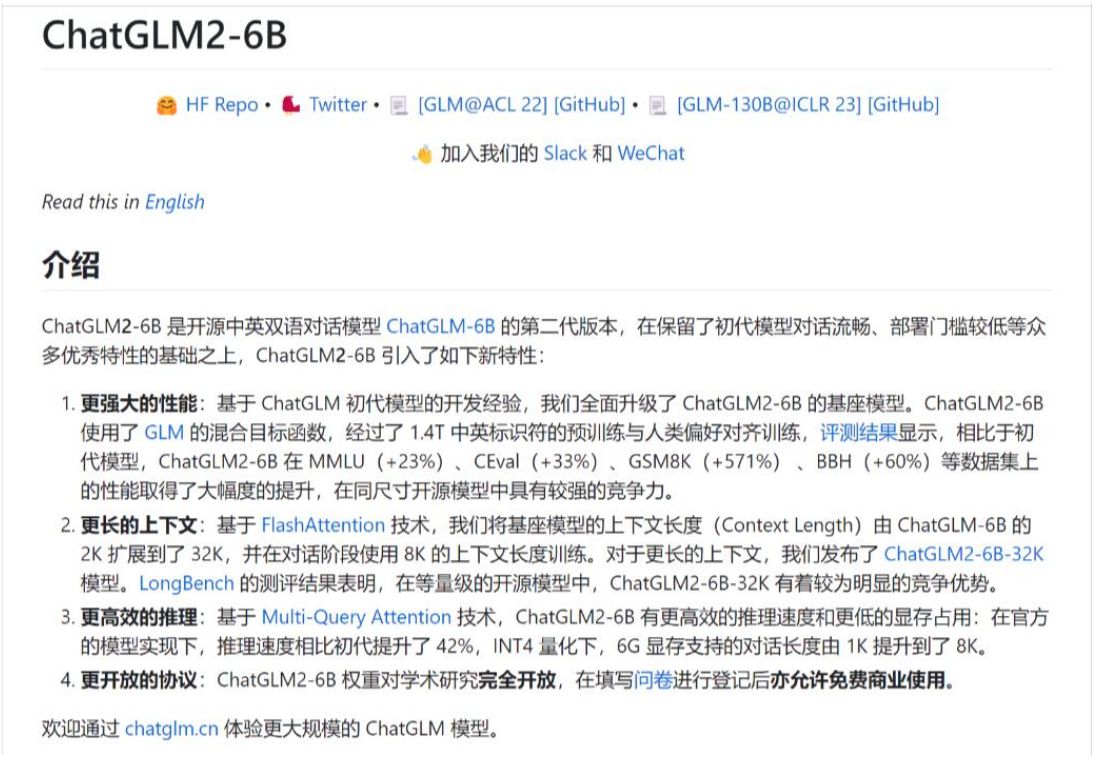
模型保存

```
model.save_pretrained("./")#保持到当前目录
```

1 个简单流程例子

```
input=['The first sentence!','The second sentence!']

from transformers import AutoTokenizer

tokenizer=AutoTokenizer.from_pretrained("distilbert-base-uncased-finetuned-sst-2-english")#https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english

input=tokenizer(input, padding=True, truncation=True, return_tensors='pt')#padding='max_length'

from transformers import AutoModelForSequenceClassification

model=AutoModel.AutoModelForSequenceClassification("distilbert-base-uncased-finetuned-sst-2-english")

print(model)

output=model(**input)

print(output.logits.shape)

import torch

predictions=torch.nn.functional.softmax(output.logits, dim=1)

print(predictions)

print(model.config.id2label)
```

使用方法-2



ChatGLM2-6B

HF Repo • Twitter • [GLM@ACL 22] [GitHub] • [GLM-130B@ICLR 23] [GitHub]

加入我们的 Slack 和 WeChat

Read this in English

介绍

ChatGLM2-6B 是开源中英双语对话模型 ChatGLM-6B 的第二代版本，在保留了初代模型对话流畅、部署门槛较低等众多优秀特性的基础之上，ChatGLM2-6B 引入了如下新特性：

1. **更强大的性能**：基于 ChatGLM 初代模型的开发经验，我们全面升级了 ChatGLM2-6B 的基座模型。ChatGLM2-6B 使用了 GLM 的混合目标函数，经过了 1.4T 中英标识符的预训练与人类偏好对齐训练，评测结果显示，相比于初代模型，ChatGLM2-6B 在 MMLU（+23%）、CEval（+33%）、GSM8K（+571%）、BBH（+60%）等数据集上的性能取得了大幅度的提升，在同尺寸开源模型中具有较强的竞争力。
2. **更长的上下文**：基于 FlashAttention 技术，我们将基座模型的上下文长度（Context Length）由 ChatGLM-6B 的 2K 扩展到了 32K，并在对话阶段使用 8K 的上下文长度训练。对于更长的上下文，我们发布了 ChatGLM2-6B-32K 模型。LongBench 的测评结果表明，在等量级的开源模型中，ChatGLM2-6B-32K 有着较为明显的竞争优势。
3. **更高效的推理**：基于 Multi-Query Attention 技术，ChatGLM2-6B 有更高效的推理速度和更低的显存占用：在官方的模型实现下，推理速度相比初代提升了 42%，INT4 量化下，6G 显存支持的对话长度由 1K 提升到了 8K。
4. **更开放的协议**：ChatGLM2-6B 权重对学术研究**完全开放**，在填写问卷进行登记后**亦允许免费商业使用。**

欢迎通过 chatglm.cn 体验更大规模的 ChatGLM 模型。

下面以 ChatGLM2-6B 为例（见上图），先在 github 上 git 下 ChatGLM2-6B 除模型外的相关文件

```
git clone https://github.com/THUDM/ChatGLM2-6B.git cd ChatGLM2-6B
```

再安装好相关依赖

```
pip install -r requirements.txt
```

接下来可以类似刚才的方法一直接执行下面代码，会在网上自动下载模型文件

```
>>> from transformers import AutoTokenizer, AutoModel

>>> tokenizer = AutoTokenizer.from_pretrained("THUDM/chatglm2-6b", trust_remote_code=True)

>>> model = AutoModel.from_pretrained("THUDM/chatglm2-6b", trust_remote_code=True, device='cuda')

>>> model = model.eval()

>>> response, history = model.chat(tokenizer, "你好", history=[])

>>> print(response)

你好 !我是人工智能助手 ChatGLM2-6B,很高兴见到你,欢迎问我任何问题。

>>> response, history = model.chat(tokenizer, "晚上睡不着应该怎么办", history=history)

>>> print(response)
```

晚上睡不着可能会让你感到焦虑或不舒服,但以下是一些可以帮助你入睡的方法:

1. 制定规律的睡眠时间表:保持规律的睡眠时间表可以帮助你建立健康的睡眠习惯,使你更容易入睡。尽量在每天的相同时间上床,并在同一时间起床。

2. 创造一个舒适的睡眠环境:确保睡眠环境舒适,安静,黑暗且温度适宜。可以使用舒适的床上用品,并保持房间通风。

3. 放松身心:在睡前做些放松的活动,例如泡个热水澡,听些轻柔的音乐,阅读一些有趣的书籍等,有助于缓解紧张和焦虑,使你更容易入睡。

4. 避免饮用含有咖啡因的饮料:咖啡因是一种刺激性物质,会影响你的睡眠质量。尽量避免在睡前饮用含有咖啡因的饮料,例如咖啡,茶和可乐。

5. 避免在床上做与睡眠无关的事情:在床上做些与睡眠无关的事情,例如看电影,玩游戏或工作等,可能会干扰你的睡眠。

6. 尝试呼吸技巧:深呼吸是一种放松技巧,可以帮助你缓解紧张和焦虑,使你更容易入睡。试着慢慢吸气,保持几秒钟,然后缓慢呼气。

如果这些方法无法帮助你入睡,你可以考虑咨询医生或睡眠专家,寻求进一步的建议。

也可以方法二，找到 huggingface 上 ChatGLM2-6B 模型地址，直接 git

```
git clone https://huggingface.co/THUDM/chatglm2-6b
```

然后打开刚才的 ChatGLM2-6B 里的 web_demo.py，修改里面的模型和 AutoTokenizer 目录，为刚才 git 模型的目录，例如我在 ChatGLM2-6B 里新建了一个 model，在 model 目录下 git 模型的，所以我的目录修改为下图

```
tokenizer = AutoTokenizer.from_pretrained("model/chatglm2-6b", trust_remote_code=True)
model = AutoModel.from_pretrained("model/chatglm2-6b", trust_remote_code=True)
```


最后，在终端直接执行下面代码

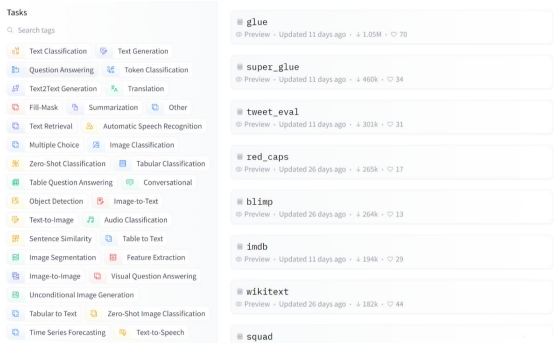
```
python web_demo.py
```

点击下图链接，即可使用 ChatGLM2-6B

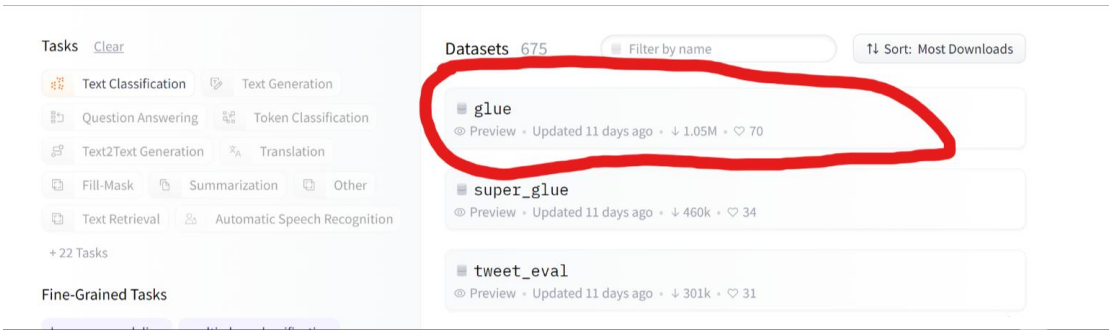
```
(base) /mnt/workspace/ChatGLM2-6B> python web_demo.py

================================BUG REPORT================================
Welcome to bitsandbytes. For bug reports, please run

python -m bitsandbytes

 and submit this information together with your error trace to: https://github.com/TimDettmers/bitsandbytes/issues
==========================================================================
/home/pai/lib/python3.9/site-packages/bitsandbytes/cuda_setup/main.py:147: UserWarning: /home/pai did not contain ['libcudart.so', 'libcudart.so.11.0', 'libcudart.so.12.0'] as expected! Searching further paths...
  warn(msg)
/home/pai/lib/python3.9/site-packages/bitsandbytes/cuda_setup/main.py:147: UserWarning: WARNING: The following directories listed in your path were found to be non-existent: {PosixPath('/usr/local/cuda/extras/CUPTI/lib64'), PosixPath('/usr/local/nvidia/lib64'), PosixPath('/usr/local/nvidia/lib')}
  warn(msg)
CUDA SETUP: CUDA runtime path found: /usr/local/cuda/lib64/libcudart.so
CUDA SETUP: Highest compute capability among GPUs detected: 7.0
CUDA SETUP: Detected CUDA version 113
/home/pai/lib/python3.9/site-packages/bitsandbytes/cuda_setup/main.py:147: UserWarning: WARNING: Compute capability < 7.5 detected! Only slow 8-bit matmul is supported for your GPU!
  warn(msg)
CUDA SETUP: Loading binary /home/pai/lib/python3.9/site-packages/bitsandbytes/libbitsandbytes_cuda113_nocublaslt.so...
Loading checkpoint shards: 100%|                                                | 7/7 [01:09<00:00,  9.92s/it]
/mnt/workspace/ChatGLM2-6B/web_demo.py:89: GradioDeprecationWarning: The `style` method is deprecated. Please set these arguments in the constructor instead.
  user_input = gr.Textbox(show_label=False, placeholder="Input...", lines=10).style(
Running on local URL:  http://127.0.0.1:7860

To create a public link, set `share=True` in `launch()`.
```

4.Huggingface 的 dataset

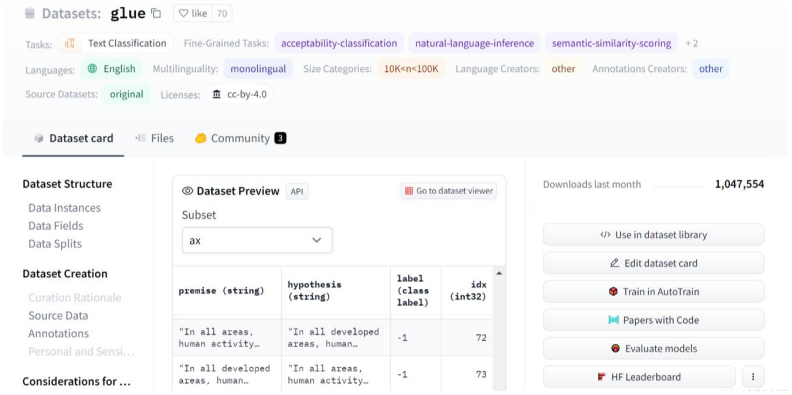
可以看到有如下任务的数据集。读者可自行打开学习



例如，我们打开 Text Classification 任务的 glue 数据集



可以看到下图，里面会有数据集的介绍、相关信息和下载方式，读者自行查看。



加粗样式

导入数据集方法（提前 pip install datasets）

```
from datasets import load_dataset

datasets=load_dataset("glue","mrpc")#glue 下还有其他数据集

print(datasets)

train_data=datasets['train']

print(train_data[0])
```

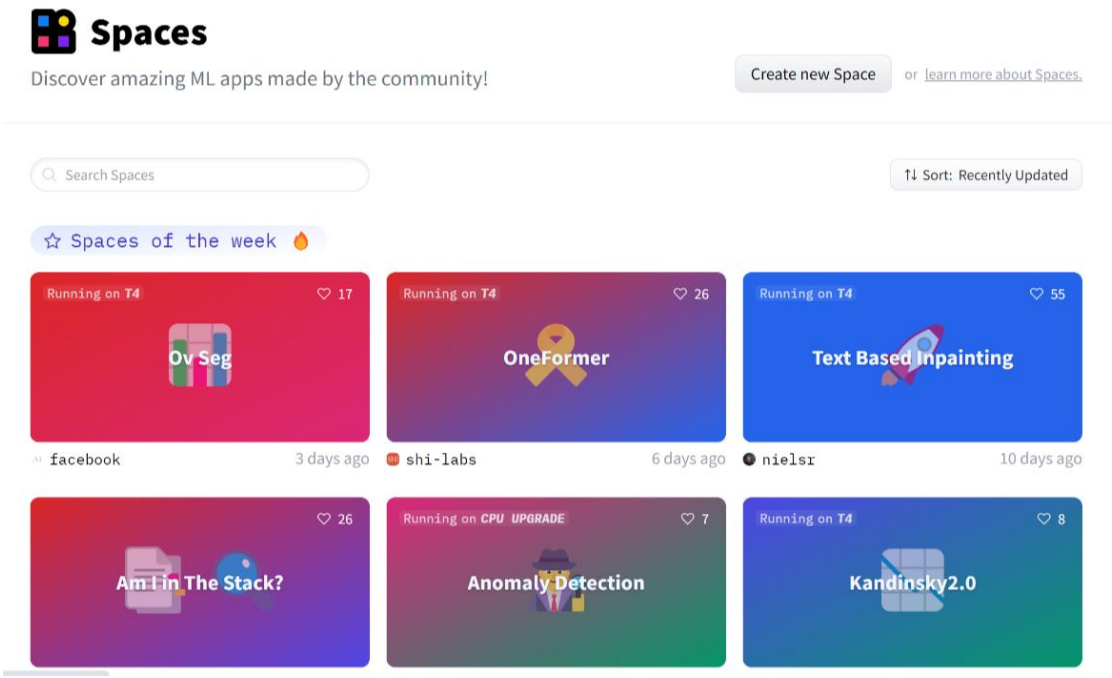
有了数据后训练模型方法

```
#下面给出 bert-base-uncased 的例子，实现对两个句子的相似度计算

#导入 tokenizer

from transformers import AutoTokenizer

tokenizer=AutoTokenizer.from_pretrained("bert-base-uncased")#https://huggingface.co/bert-base-uncased

#input=tokenizer('The first sentence!','The second sentence!')#测试

#print(tokenizer.convert_ids_to_tokens(input["input_id"]))

#实际使用 tokenizer 的方法，得到 tokenizer_data

def tokenize_function(example):

        return tokenizer(example["sentence1"],example["sentence2"],truncation=True)

from datasets import load_dataset
```

```
datasets=load_dataset("glue","mrpc")

tokenizer_data=datasets.map(tokenize_function,batched=True)

print(tokenizer_data)

#训练参数

from transformers import TrainingArguments

training_arg=TrainingArguments("test-trainer")#训练参数，可以自己去改，参数意思参考 https://huggingface.co/docs/transformers/main_classes/trainer#transformers.TrainingArguments

print(training_arg)#看下默认值

#导入模型

from transformers import AutoModelForSequenceClassification

model=AutoModelForSequenceClassification.from_pre_trained("bert-base-uncased",num_labels=2)#num_labels 自己定义了，所以不会导入输出层

#导入数据处理的一个东西 DataCollatorWithPadding，变成一个一个 batch

from transformers import DataCollatorWithPadding

data_collator=DataCollatorWithPadding(tokenizer=tokenizer)

#导入训练器，进行训练,API : https://huggingface.co/docs/transformers/main_classes/trainer#transformers.Trainer

from transformers import Trainer

trainer=Trainer(model,training_arg,train_dataset=tokenizer_data["train"],eval_dataset=tokenizer_data["validation"],data_collator=data_collator,tokenizer=tokenizer)

trainer.train()
```

5.Huggingface 的 spaces

点开如下图所示。里面有些近些天有趣的东西火热的 apps。



比如，我看到一个有趣的扩散模型的 app。。打开后发现其可以实现许多多模态之间的转换。