# 源码部署

## 系统要求

**系统**

Ubuntu x86\_64 系统 >=18.04

python >= 3.8

**安装 python 依赖**

pip3 install -r requirements.txt

**安装torch**

#CPU版本

pip3 install torch --index-url https://download.pytorch.org/whl/cpu

或

#GPU版本

pip3 install torch --index-url https://download.pytorch.org/whl/cu118

**⽀持 CPU 运⾏， 如需使⽤ GPU 运⾏，需硬件配备英伟达显卡**

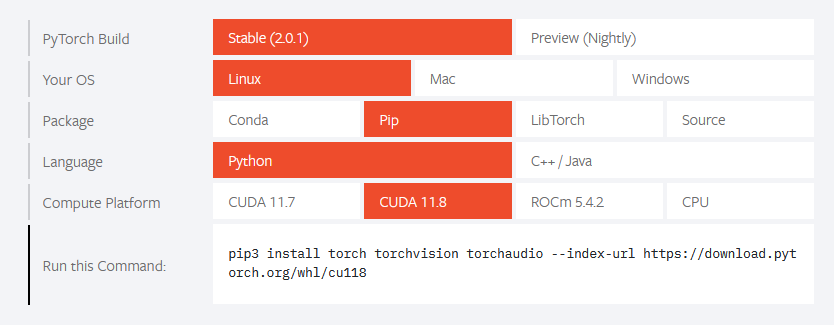
nvidia显卡驱动 > 525.89.02

cuda版本 > 11.8

torch==2.0.0+cu118

或根据具体硬件配置参照torch官方网站建议版本安装

<https://pytorch.org/>



其他硬件版本

<https://pytorch.org/get-started/previous-versions/>

**配置工程是否使用GPU**

打开工程文件中的config/config.py

修改

CONF.GPU= True

或在服务启动时传入device参数，具体见启动服务说明。

## 模型文件

由于依赖的大模型文件较大，建议从官方网站直接下载

### text2vec

安装huggingface下载器

git lfs install

下载文件

git clone <https://huggingface.co/GanymedeNil/text2vec-large-chinese>

或

直接打开页面手动下载

<https://huggingface.co/GanymedeNil/text2vec-large-chinese>

### 3 配置模型文件

打开工程文件中的config/config.py

修改

CONF.embeding\_model\_path = [text2vec的文件夹地址]

## 启动服务

进入工程文件夹

**GPU/CPU模式**

通过修改config/config.py文件中的CONF.GPU=True/False 来设置GPU和CPU模式

如果启动时指定device参数，则以device参数传递值为准

运行以下命令启动服务

python3 file\_classify/service.py [--port=] [--device=]

该服务监听本地[port]端口,接收客户端上传的文件并返回分析后的表格格式数据。默认在8000端口监听。如果未指定device参数，以CONF.GPU的配置为准。

启动参数

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **是否必须** | **类型** | **说明** |
| port | 否 | int | 默认在8000端口监听，可根据需要修改端口 |
| device | 否 | string | ‘GPU’/‘CPU’ 选择模型在GPU还是CPU加载  如果不设置该参数，默认使用config文件中的配置项 |

## 调用接口

**客户端脚本**

识别文件中的表格格式：该接口上传[file\_path]指定的本地文件到[ip]:[port]的识别服务，并返回识别后的表格格式数据，默认访问本地8000端口。

进入工程文件夹执行

python3 file\_classify/client.py file\_classify --file\_path=data/input/支付宝1.pdf [--ip=127.0.0.1] [--port=8000]

接口名**file\_classify**

调用参数

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **必须** | **类型** | **说明** |
| file\_path | 是 | string | 要识别的文件的本地路径。 |
| ip | 否 | string | 服务所在的ip地址，默认为本机地址127.0.0.1 |
| port | 否 | int | 默认访问8000端口，可根据需要修改 |

返回json文本

**HTTP协议**

**POST**

**接口 <http://[url:port]>/file\_classify/**

参数

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **必须** | **类型** | **说明** |
| info | 是 | string | 待识别的文件名 |
| data | 是 | string (base64) | 经过base64编码的文件二进制数据 |

返回json文本

**返回值**

返回值为json格式的文本

正确返回：

{

"agent\_type": "bank",

"file\_format": "excel",

"stamp\_layer": false,

"digital\_certificate": false,

"imformation": "",

"code": 400,

"success": frue,

"image":[],

"file":""

}

错误返回：

{

"code":402,

"success":false

}

**输出格式**

|  |  |  |
| --- | --- | --- |
| **字段** | **说明** | **输出内容** |
| success | 执行是否成功 | true：执行成功  false：执行失败 |
| code | 报错码 | 402：无效文件 |
| data.agent\_type | 发布机构类型 | bank：银行  wechat：微信  alipay：支付宝 |
| data.file\_format | 文件格式 | excel：excel格式  TextPDF：文本PDF文件  ImagePDF：图像PDF文件 |
| data.stamp\_layer | 印章图层是否已处理 | true：有印章图层  false：未识别印章图层 |
| data.digital\_certificate | 是否有数字证书 | true：包含数字证书  false：未识别数字证书 |
| data.imformation | 数字证书信息 | 持有人，颁发者、是否篡改；没有则为”” |
| data.image | 印章图片 | 列表,一个pdf印章可能有多张图片叠加而成，有几个返回几个。  base64编码 |
| flie | 去除电子印章后的文件，保持原来的文件格式； | Base64编码 |

**code 说明**

|  |  |  |
| --- | --- | --- |
| **代码类型** | **信息** | **说明** |
| 400 | Success | 执行成功 |
| 402 | Type error | 文件格式不符合要求 |
| -1 | unknow | 未知错误 |

注：使用客户端脚本执行时，为测试方便，不在终端打印flie和image字段，如有返回的转换后文件和印章png，则在file\_path文件所在的文件夹下，生成

去印章转换后的文件：[原文件名]\_trans.pdf

印章png文件：[原文件名]\_image1.png [原文件名]\_image2.png ...[如有多个]