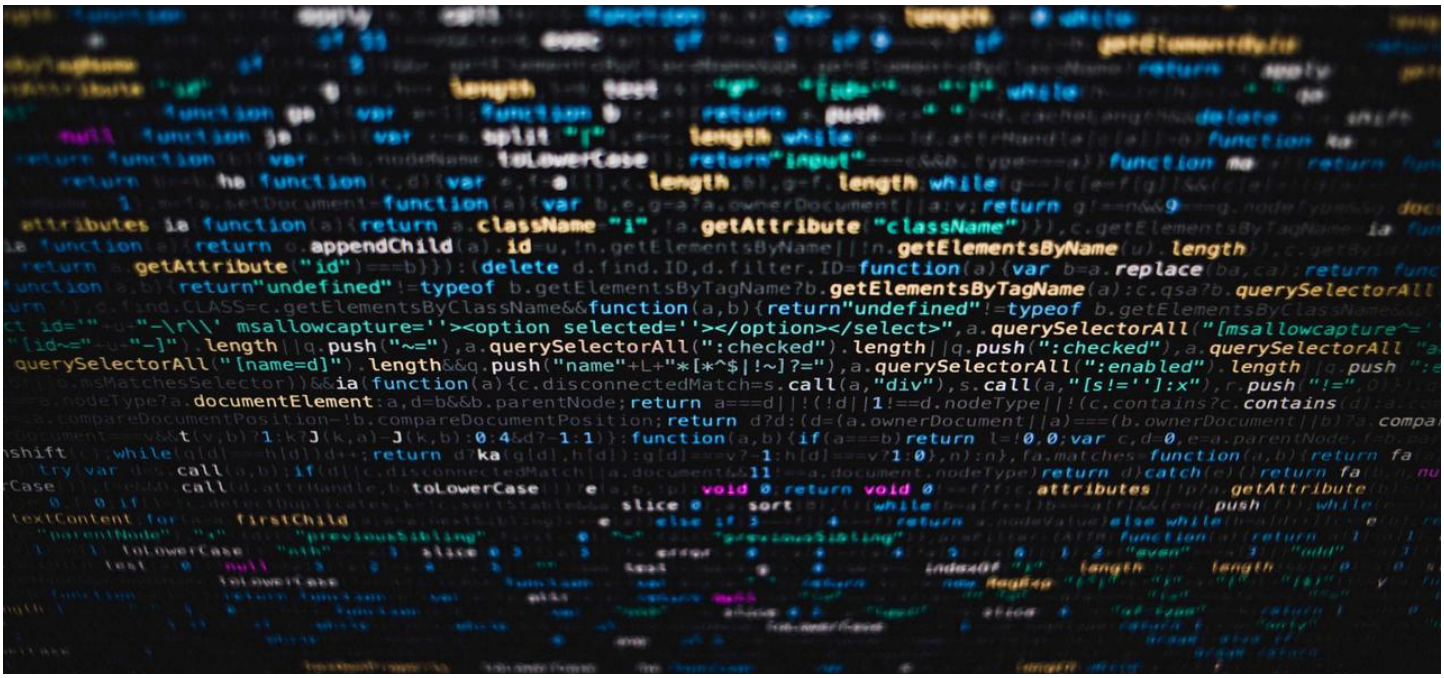
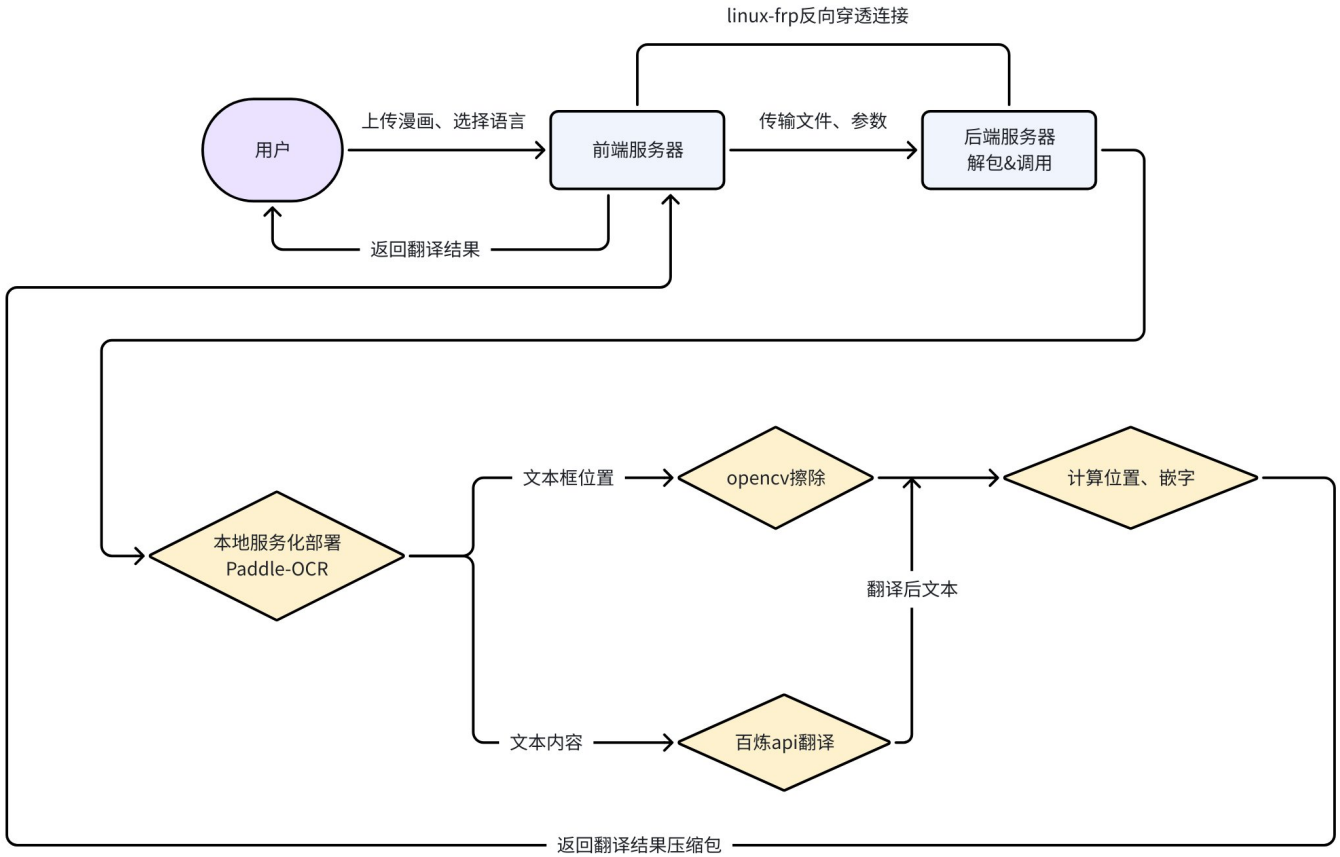


Manga translation

项目流程简介

上传漫画，服务器转发给计算端，计算端做ocr识别、擦除、嵌字，调用百炼api做翻译，把结果返回给服务器，用户可下载。



项目地址：https://github.com/Le1zyCatt/manga_translate

关于paddle-ocr的服务化部署

1. 运行服务

参照文档：**https://www.paddleocr.ai/latest/version3.x/deployment/serving.html#12**

```bash
conda activate paddle-ocr
paddlex --install serving
paddlex --serve --pipeline OCR #这一步会下载所有paddle模型，不过都很小，别急
```

运行完毕后会出现：

```bash
INFO:     Started server process [51689]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8080 (Press CTRL+C to quit)
```

默认8080端口。上面那个*--pipeline*参数可以指定配置文件（产线名称）。

> ⚠ **注意**

根据paddle-ocr官方文档：使用教程 - PaddleOCR 文档

PP-OCRv5有五个模块：

如果直接使用默认调用方式，则五个模块均会使用。但从输出的json来看，完全无法还原出原来的文本框在哪里。**这对于漫画翻译场景非常致命。**具体原因参考GPT回答：

ChatGPT

```
🟥 为什么 PaddleX 的坐标不能反推？
因为 PaddleX（paddlex serve OCR pipeline）会在内部做很多预处理，但 不会把这些预处理的参数暴露出来：
PaddleX OCR pipeline 内部操作包括：
自动旋转（angle classify）
文档预处理（doc unwarp / crop）
resize（通常是 960 或 1024）
padding
图像矩阵变换（perspective transform 等）
normalization
这些都会改变文本框的坐标。
但 PaddleX 返回的只有：
dt_polys
translated_texts
orientation
🚫 没有返回：
原图 → 处理图像的缩放因子
旋转角度（有 angle 但不是最终矩阵）
unwarp 的透视矩阵
padding 大小
所以根本没法逆变换回原图坐标。
👉 这是我说的“无法恢复文本框坐标”。

```

于是翻阅文档，经查阅得知paddlex的服务化部署可使用自定义yaml配置文件，以达到禁用特定模块的效果。禁用特定模块理论上可以得到能够inpaint的json文件。

首先导出默认配置：

```bash
paddlex --get_pipeline_config OCR
```

这会在当前目录下生成一个 `OCR.yaml` 文件：

OCR.yaml

```yaml
pipeline_name: OCR

text_type: general

use_doc_preprocessor: True
use_textline_orientation: True

SubPipelines:
  DocPreprocessor:
    pipeline_name: doc_preprocessor
    use_doc_orientation_classify: True
    use_doc_unwarping: True
    SubModules:
      DocOrientationClassify:
        module_name: doc_text_orientation
        model_name: PP-LCNet_x1_0_doc_ori
        model_dir: null
      DocUnwarping:
        module_name: image_unwarping
        model_name: UVDoc
        model_dir: null

SubModules:
  TextDetection:
    module_name: text_detection
    model_name: PP-OCRv5_server_det
    model_dir: null
    limit_side_len: 64
    limit_type: min
    max_side_limit: 4000
    thresh: 0.3
```

```
    box_thresh: 0.6
    unclip_ratio: 1.5
  TextLineOrientation:
    module_name: textline_orientation
    model_name: PP-LCNet_x1_0_textline_ori
    model_dir: null
    batch_size: 6
  TextRecognition:
    module_name: text_recognition
    model_name: PP-OCRv5_server_rec
    model_dir: null
    batch_size: 6
    score_thresh: 0.0
```

这就是默认的paddle-ocr服务化部署的配置了。经过试验可以知道：

`use_doc_orientation_classify: True`
`use_doc_unwarping: True`

这两个参数分别对应前两个模块，会导致出现 `angle` 参数，且导致定位文本框的一些参数在最终的输出json文件里无法获取。因此将 `True` 改为 `False` 以禁用。

根据PaddleOCR 与 PaddleX - PaddleOCR 文档，我们可以将修改后的yaml文件传入paddlex的启动参数：

bash

```
paddlex --serve --pipeline OCR --paddlex_config ./OCR.yaml
```

现在就启动了。识别的结果可以定位文本框并满足要求。

2. 调用

utils/paddle_ocr.py

```
import base64
import json
import requests
from pathlib import Path

# PaddleOCR 服务地址
OCR_URL = "http://localhost:8080/ocr"

def image_to_base64(file_path):
    """将图片文件转换为 Base64 编码"""
    with open(file_path, "rb") as f:
```

```
        return base64.b64encode(f.read()).decode('utf-8')

def ocr_image(file_base64, visualize=False):
    """调用 PaddleOCR 服务进行识别"""
    payload = {
        "file": file_base64,
        "fileType": 1,  # 1表示图像
        "visualize": visualize
    }
    headers = {"Content-Type": "application/json"}
    response = requests.post(OCR_URL, data=json.dumps(payload),
headers=headers)
    if response.status_code == 200:
        result = response.json()
        if result.get("errorCode", 1) == 0:
            return result["result"]  # 返回识别结果 JSON
        else:
            raise RuntimeError(f"OCR 服务错误: {result.get('errorMsg')}")
    else:
        raise RuntimeError(f"OCR 请求失败，HTTP 状态码: {response.status_code}")

def extract_text(ocr_result):
    """从 PaddleOCR 返回结果中提取文字"""
    texts = []
    for page in ocr_result.get("ocrResults", []):
        pruned = page.get("prunedResult", {})
        for item in pruned.get("res", []):
            text = item.get("text")
            if text:
                texts.append(text)
    return texts

def process_image_sequence(image_paths):
    """批量处理图片序列"""
    all_results = []
    for img_path in image_paths:
        if Path(img_path).exists():
            file_base64 = image_to_base64(img_path)
        else:
            file_base64 = img_path  # 已经是Base64
        ocr_result = ocr_image(file_base64)
        texts = extract_text(ocr_result)
        all_results.append({
            "image": img_path,
            "texts": texts,
            "raw_result": ocr_result
        })
```

```
    return all_results,ocr_result

if __name__ == "__main__":
    # 测试图片序列（可以是文件路径，也可以是 Base64）
    image_seq = ["./test_data/jap.jpg", "./test_data/chi.jpg"]
    results,ori_results = process_image_sequence(image_seq)
    # print(ori_results)
    print(json.dumps(ori_results, indent=2, ensure_ascii=False))
```

关于消息转发的步骤

1. 从公网访问

直接访问http://8.134.219.231:8501/

2. 在公网服务器上的配置

使用frp包：

代码块

```
#下载
wget https://github.com/fatedier/frp/releases/download/v0.59.0/frp_0.59.0_linux_amd64.tar.gz
tar -xzf frp_0.59.0_linux_amd64.tar.gz

cd frp_0.59.0_linux_amd64

#在目录下创建ini配置文件
nano frps.ini
```

之后，写入内容：

代码块

```
[common]
bind_port = 7000
token =114514
```

含义是：服务器监听的主控制端口是7000，frpc（client端）会从7000连接frps（server端）。需要的密码是114514。

3. 在计算服务器（我的电脑）上的配置

同样的，使用frp包：

代码块

```
#下载
wget https://github.com/fatedier/frp/releases/download/v0.59.0/frp_0.59.0_linux_amd64.tar.gz
tar -xzf frp_0.59.0_linux_amd64.tar.gz

cd frp_0.59.0_linux_amd64

#在目录下创建ini配置文件
nano frps.ini
```

之后，写入内容：

代码块

```
[common]
server_addr = 113.45.231.102
server_port = 7000
token = 114514

[proxy_to_local]
type = tcp
local_ip = 127.0.0.1
local_port = 5000
remote_port = 8000
```

含义是：通过*server_port*连接*server_addr*，使用密码114514。监听远程服务器的*remote_port*端口，并将本地（*local_ip*）的*local_port*端口的消息转发过去。

配置完毕之后，就可以从公网访问本地服务了。

如何在服务器部署服务

一、部署目标

- 假设我们已经在本地电脑跑通了一个基于Python的 `streamlit` 库的前端功能网页，代码写在 `app.py` 中

代码块

```python
import streamlit as st
import requests
import io

# 约定将请求数据发送到服务端的8000端口
BACKEND_URL = 'http://x.xxx.xxx.xxx:8000'

# ...
```

- 此时该代码只能在本地主机运行访问

代码块

```
cd local_app_py_dir
streamlit run app.py
```

- 但是我们想将其部署到服务器上，使得其它主机能通过特定网址访问该网页服务，此处我们通过远程云服务器部署该网页

二、远程连接

- 为了部署，我们应当从某处打开服务器的Linux系统命令行，如果是远程云服务器则可通过本地VSCode连接，参考VSCode配置SSH连接远程服务器
- 连接到服务器后，创建一个目录来存放所需相关文件，我们先将 `app.py` 拷贝到此处

代码块

```
mkdir app_py_dir
cd app_py_dir
touch app.py
vim app.py # 或从VSCode打开，反正把代码粘贴进去即可
```

三、虚拟环境

- 由于我们的网页部署依赖于 `streamlit` 和 `requests` 库，故需先借助 `venv` 创建Python虚拟环境（在前文创建好的路径下创建即可）

```
代码块
1  apt-get update && apt-get install python3-venv # 安装venv工具
2  python3 -m venv venv # 在当前目录下创建一个名为venv（指令的最后一项指定名称）的虚拟环境文件夹，内部会用于存放后续安装的库
3  source venv/bin/activate # 激活虚拟环境
```

- 然后在当前虚拟环境下安装所需的库

```
代码块
1  pip install --upgrade pip
2  pip install streamlit requests
```

四、服务文件

- 在 `/etc/systemd/system/` 目录处创建 `.service` 文件

```
代码块
1  cd /etc/systemd/system/
2  touch app_py.service
```

- 在 `app_py.service` 文件内写入如下内容

```
代码块
1  [Unit]
2  Description=My Own Streamlit Frontend
3  After=network.target
4
5  [Service]
6  User=root
7  WorkingDirectory=/root/app_py_dir
8  # 逐一此处使用虚拟环境中的streamlit绝对路径
9  ExecStart=/root/app_py_dir/venv/bin/streamlit run app.py --server.port 8501 --server.address 0.0.0.0
10 Restart=always
11 RestartSec=5
```

```
12
13  [Install]
14  WantedBy=multi-user.target
```

- 然后执行以下命令便可基本实现服务的持久化部署运行

代码块

```
1   systemctl daemon-reload # 重载配置
2   systemctl start app_py # 启动app_py.service
3   systemctl enable app_py # 设置开机自启app_py.service
```

- 检测服务运行状态以及其是否配置成功

代码块

```
1   sudo systemctl status app_py # 检查app_py.service的详细运行状态
2   sudo systemctl is-enabled app_py # 检查app_py.service是否已设为开机自启
```

五、外部访问

- 对于 `streamlit` 库，其部署的网页默认通过 `8501` 端口访问，服务器需要开放该端口（若是租借的云服务器，一般可在安全组设置中开放该端口）才能使得网页被正常访问
- 确保一切无误后，就可从其它主机上通过 `http://服务器公网IP:端口` 的方式访问

代码块

```
1   curl -4 icanhazip.com # 获取服务器公网IP
```

IMG_0472.MOV
112.60MB

About Ncatbot（乱入）

1. Install

```bash
sudo su
cd ~
python3 venv -m .ncatbot
source ~/.ncatbot/bin/activate
pip install ncatbot -U -i https://mirrors.aliyun.com/pypi/simple
```

Make sure you use `sudo` !!!

Also run `main.py` in `sudo`.