滑板车背诵_OCR部署说明

一. 项目需求

1. 实现中英文文档识别
2. 中英文内容分别识别

中英文识别，单独的中文以及英文检测识别均使用当前的ocr模型。前后端可以针对识别到的中英文信息进行区分，以适应不同的识别任务

二. 第三方库说明

以下三个第三方库都随文件一并发送。

均包含共用的include和区分arm_v7和arm_v8版本的动态库文件夹lib。

1. opencv2： 图片处理的第三方库
2. Openssl：用于模型解密的第三方库
3. paddle_lite：移动端模型推理框架

随文件一并发送的还有：代码（code）, 加密后的模型（model）, 以及一个label.txt (label)

三. 前端调用接口

本地安卓测试使用机型：荣耀30s=====>验证通过

setp1: 模型初始化

调用时机：用户进入当前功能时立即调用。该部分测试耗时：0.5s左右

前端调用：ocr_infer.h 中的 ppredictor::OCR_PPredictor*ocr_init(det_encodeModel_path,rec_ch_encodeModel_path,cls_model_path)

> Paramaters:
>
> 1) det_encodeModel_path: 加密后的检测模型路径：文件名为det.key
>
> 2) rec_ch_encodeModel_path: 加密后的中英文识别模型路径：文件名为rec_ch.key
>
> 3) cls_model_path：加密后的文字方向识别模型路径：文件名为cls.key
>
> Return:
>
> ocr_predictor: 解释器指针

step2: 文档OCR

该部分将文档检测，文本识别，文本方向识别统一在一个方法内，以便调用和部署

调用时机：用户针对传入的当前文档图片使用当前功能时调用。本地测试：检测耗时0.8-1s，单个边框识别耗时：0.01秒左右。

前端直接调用（5.19已更新）：

ocr_infer.h中的

std::vector<ppredictor::OCRPredictResult> doc_infer(original, ocr_predictor, label_ch_path)

> Paramaters:
>
> 1) original: 需要检测识别的文档图片：cv::Mat类型
>
> 2) ocr_predictor: step1返回值
>
> 3) label_ch_path：中文字符文档，文件名为：ch_label.txt
>
> Return:
>
> results: 识别内容。该变量为vector向量，其中元素为ppredictor::OCRPredictResult结构体

5.19更新为：

string doc_infer(cv::Mat &original, ppredictor::OCR_PPredictor *ocr_predictor, string label_ch_path)

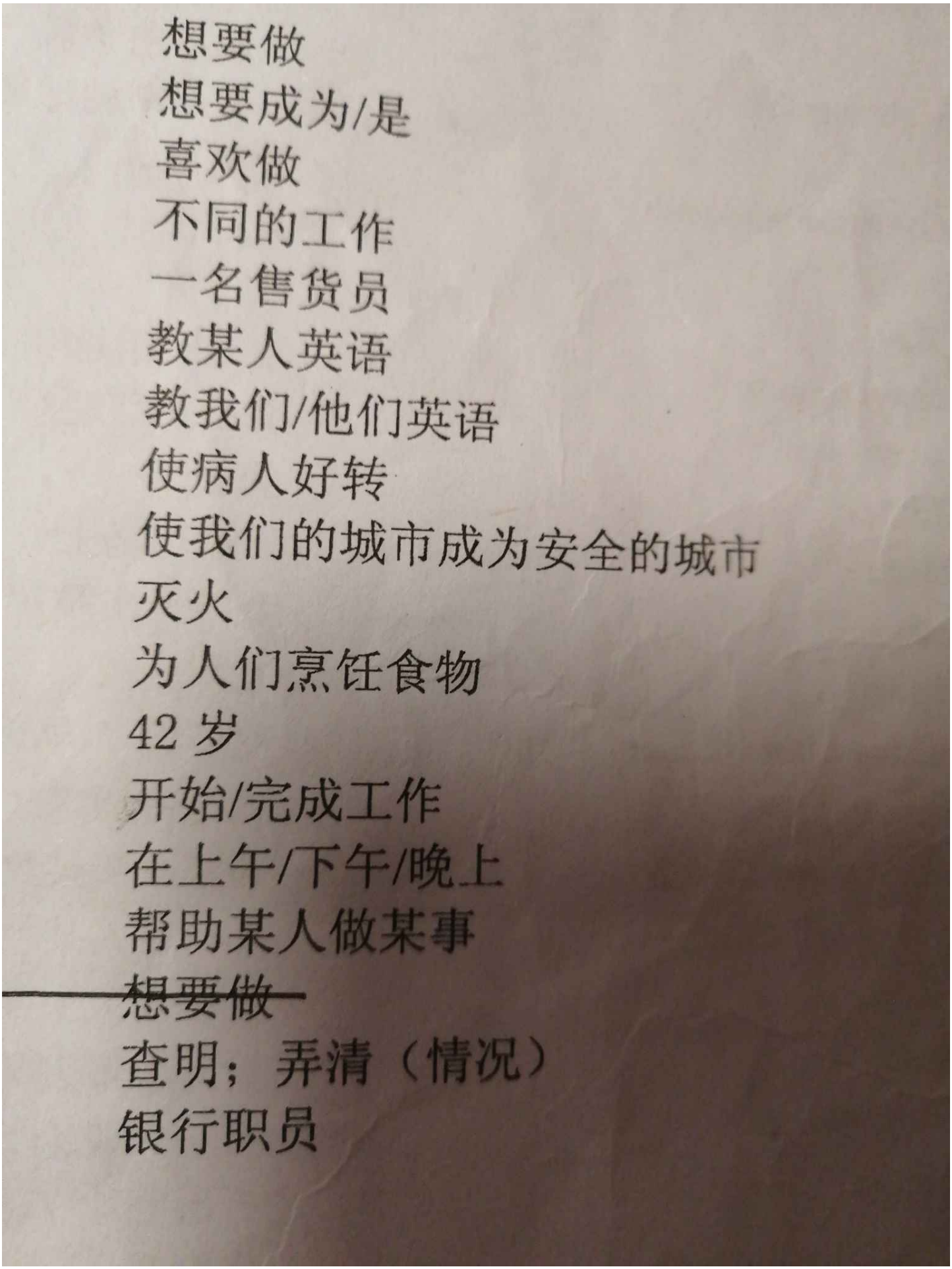
将返回值更新为string。每张图片返回一个string字符串，该字符串每行内容为检测到的一个边框中置信度大于0.5的字符内容，每行字符用“\n”分隔，前端直接获取展示即可。使用代码块为：

```
string text_result = "";
for(ppredictor::OCRPredictResult ocr_result: ocr_results){
    LOGE("字符内容=|%s|, 置信度分数=%f", ocr_result.word_text.c_str(), ocr_result.
    if(ocr_result.score>0.5){
        text_result.append(ocr_result.word_text.c_str());
        text_result.append("\n");
    }
}
```

text_result即为返回值。

Example：

用户输入图片为：



调用该方法直接获取的结果为：

> 想要做

> 想要成为/是
>
> 喜欢做
>
> 不同的工作
>
> 一名售货员
>
> 教某人英语
>
> 教我们/他们英语
>
> 使病人好转
>
> 使我们的城市成为安全的城市
>
> 灭火
>
> 为人们烹饪食物
>
> 42岁
>
> 开始/完成工作
>
> 在上午/下午/晚上
>
> 帮助某人做某事
>
> 想要做一
>
> 查明；弄清 (情况)
>
> 银行职员

step2中results说明

每张图片返回一个results

该reuslts为一个vector容器，其中每个元素表示一个边框，即检测到文档图片中的某个文字片段

Results 的每个元素是一个ppredictor::OCRPredictResult结构体

也就是说，输入一张图片，输出一个列表，该列表中的元素为一个结构体

因此前端只需要依次遍历该vector并从每个元素中取出word_text值即可。有必要的话，可以在取值前判定其score值。

该结构体定义为：

```
struct OCRPredictResult {
    std::vector<int> word_index;
    std::string word_text;
    int isChinese = -1;
    std::vector <std::vector<int>> points;
    std::vector<int> upper_lift_corner{0, 0};
    float score;
```

```
        float cls_score;
        int cls_label=-1;
    };
```

其中与前端使用相关联的是：

> word_text：该边框内含的字符内容
>
> isChinese：中英文数字的说明：如果该边框中字符包含汉字则为1===>该部分还需根据需求进行更改
>
> 后续需要的话，我可以区分每个字符的中英文以及数字属性，方便后端判断
>
> Score: 边框中字符的置信度：一般置信度>0.9则识别较为准确，置信度<0.6的结果可以摈弃，因为有手写等图片影响识别。
>
> Points: 该边框的位置坐标信息, 数据组织形式：[[左上], [右上], [右下], [左下]]

step3: 内存释放

调用时机： 用户执行完当前功能时调用，若用户未执行完当前功能而调用该模块则导致野指针异常而崩溃。假设用户当前时间段，需要执行多次该功能，执行多张图片，则等待本轮执行完调用。

建议：可在用户退出该功能页面时释放，或者用户退出该功能2秒后释放。以防止多次创建解释器而增加运行时间

前端直接调ocr_infer.h中：released(ppredictor::OCR_PPredictor *orc_predictor)

> Parameters:
>
> orc_predictor: step2中返回值

四：后续发展

1. 当前模型部署说明

由于之前大量时间花费在曲谱业务以及安全加密业务上，导致该任务的时间相对紧张。

前两周针对大模型进行了一系列的工作：模型调研，模型训练，模型裁剪以及训练。结果证明：大模型针对弯曲文本有一定的效果，但是模型较大，部署起来相对麻烦，且模型裁剪后效果较差。还需要进行更新则时间上来不及。

因此使用之前熟悉的paddleOCR做先行部署，后续更新

2. 当前模型能解决的问题

- 当前模型可针对不同方向的文本以及包含中英文，简单数学符号，数字信息，以及数字序号的文本内容进行检测和识别。

- 当前模型针对相对平整，清晰，手写内容或工整或较少的文本内容的检测效果较好。中英文识别准确率则能达到96%
- 当前模型针对短文本的检测效果较好

3. 当前模型存在的问题---后续更新

- 现阶段模型针对长文本的预测效果较弱
- 针对挨得较紧的长文本预测效果较差
- 弯曲文本预测效果较差
- 英文偶尔会出现空格丢失的问题
- 包含拼音的字符排序

以上问题由于时间匆忙，本次上线暂未深入处理，后续更新