文字识别

实践教程

腾讯云

文档目录

实践教程
用腾讯云文字识别实现企业资质证书识别

最近更新时间：2025-06-17 17:32:12

企业经营活动中，资质证书是证明企业生产能力的必要证件，也是企业入驻各类平台、组织项目申报等必须提交的，这里面包括营业执照、税务登记证、生产许可证、高新技术企业认定证书等等。在日常工作中，以平台类企业入驻为例，要求企业上传对应的资质证书然后进行审核，但由于企业资质证书种类繁多，各行各业的资质证书都有差异，没有统一的板式，通过人工审核工作量巨大且很容易出错。

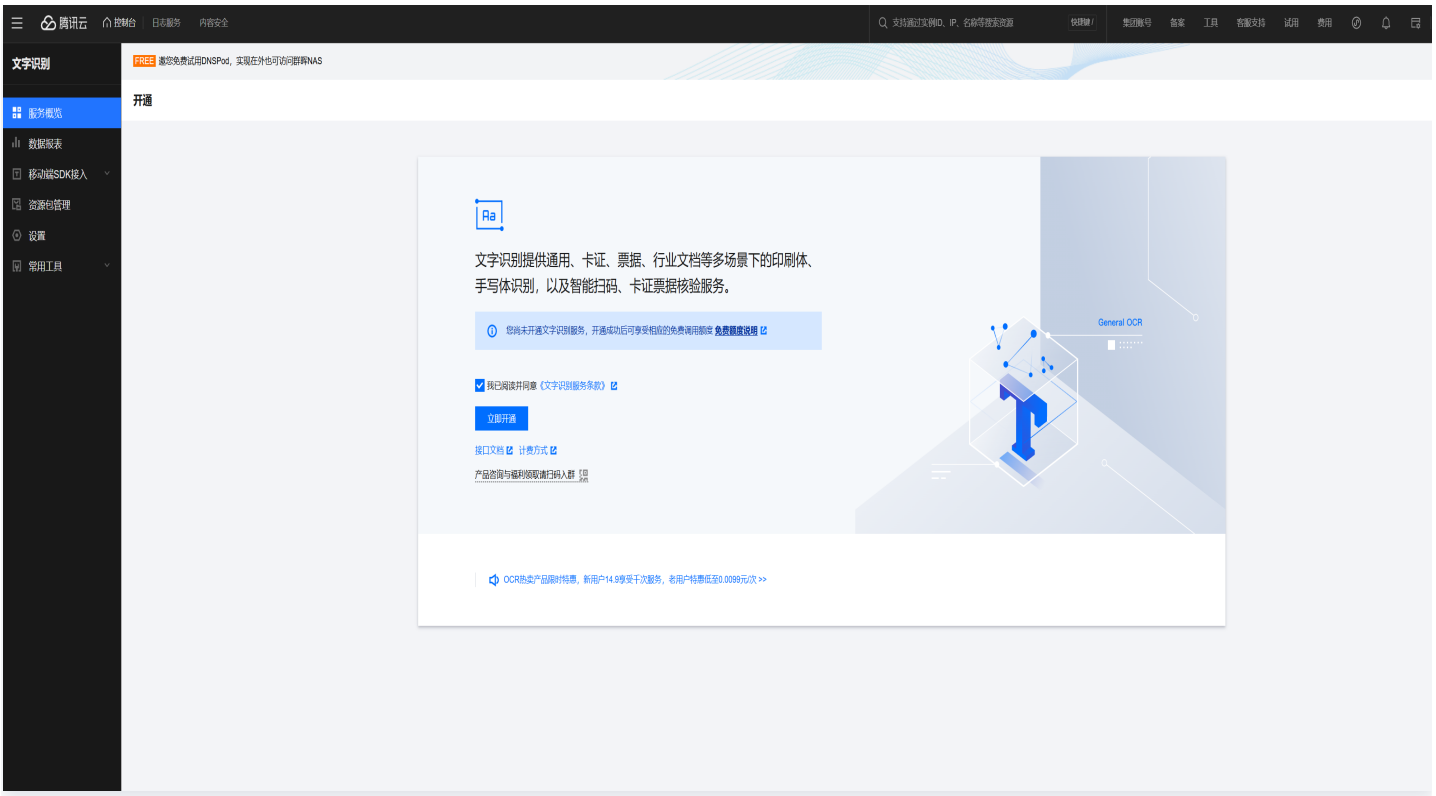
那么，有没有更智能化的方式让资质审核流程更加快捷和高效呢？搜索了国内外的文字识别产品，**发现腾讯云文字识别新推出了文档智能识别能力，能够识别并提取各类证照、票据、表单、合同等结构化场景的字段信息。**深入了解后，发现这个接口能力刚好和我们要解决的企业资质自动化审核问题完美契合。

接下来，将详细讲述我是如何使用文档智能识别能力，完成资质证书标题、企业名称、许可证编号、注册地址、企业负责人等信息的自动获取。

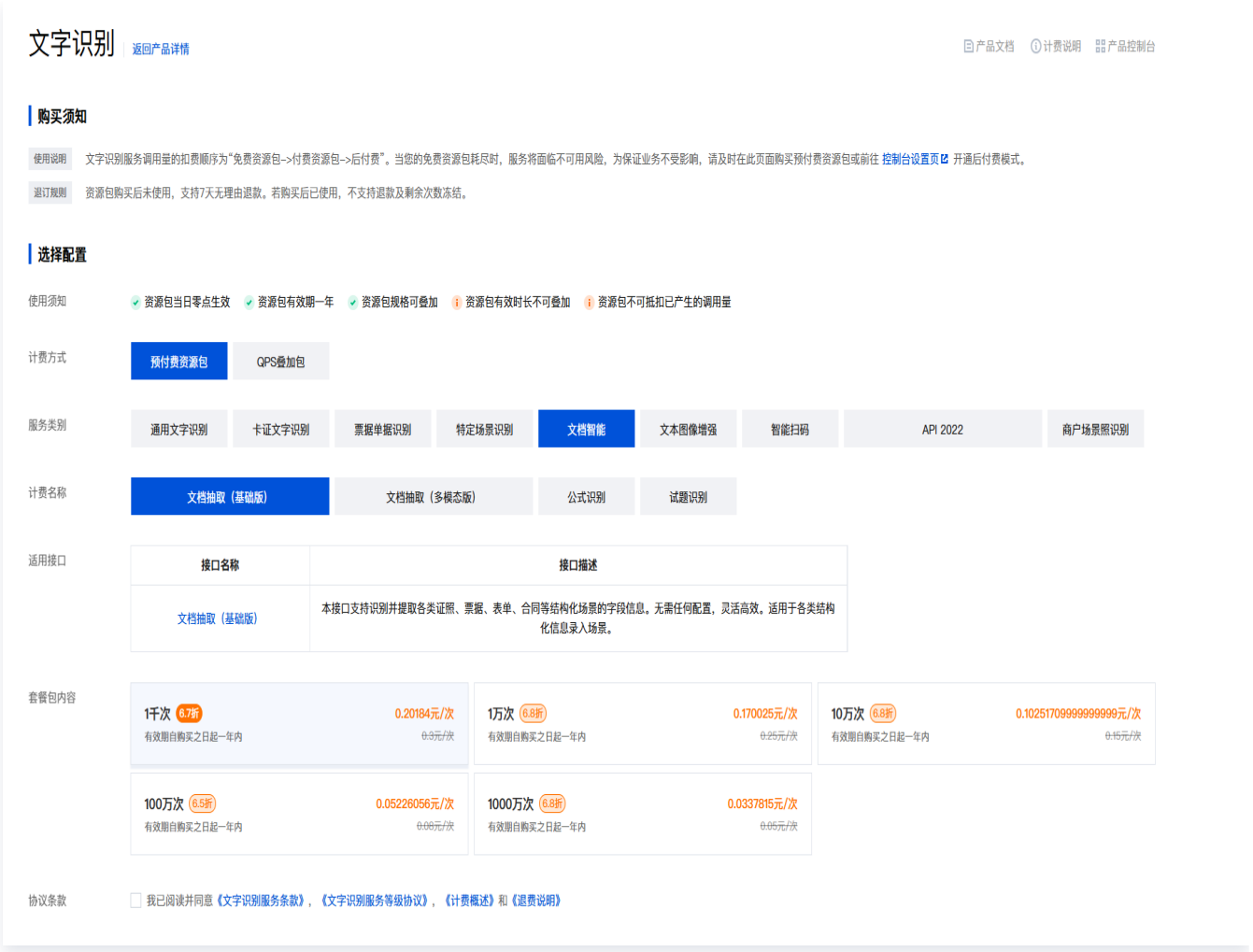
准备工作

为了使用腾讯云文档智能识别能力，首先需要进行一些准备工作。

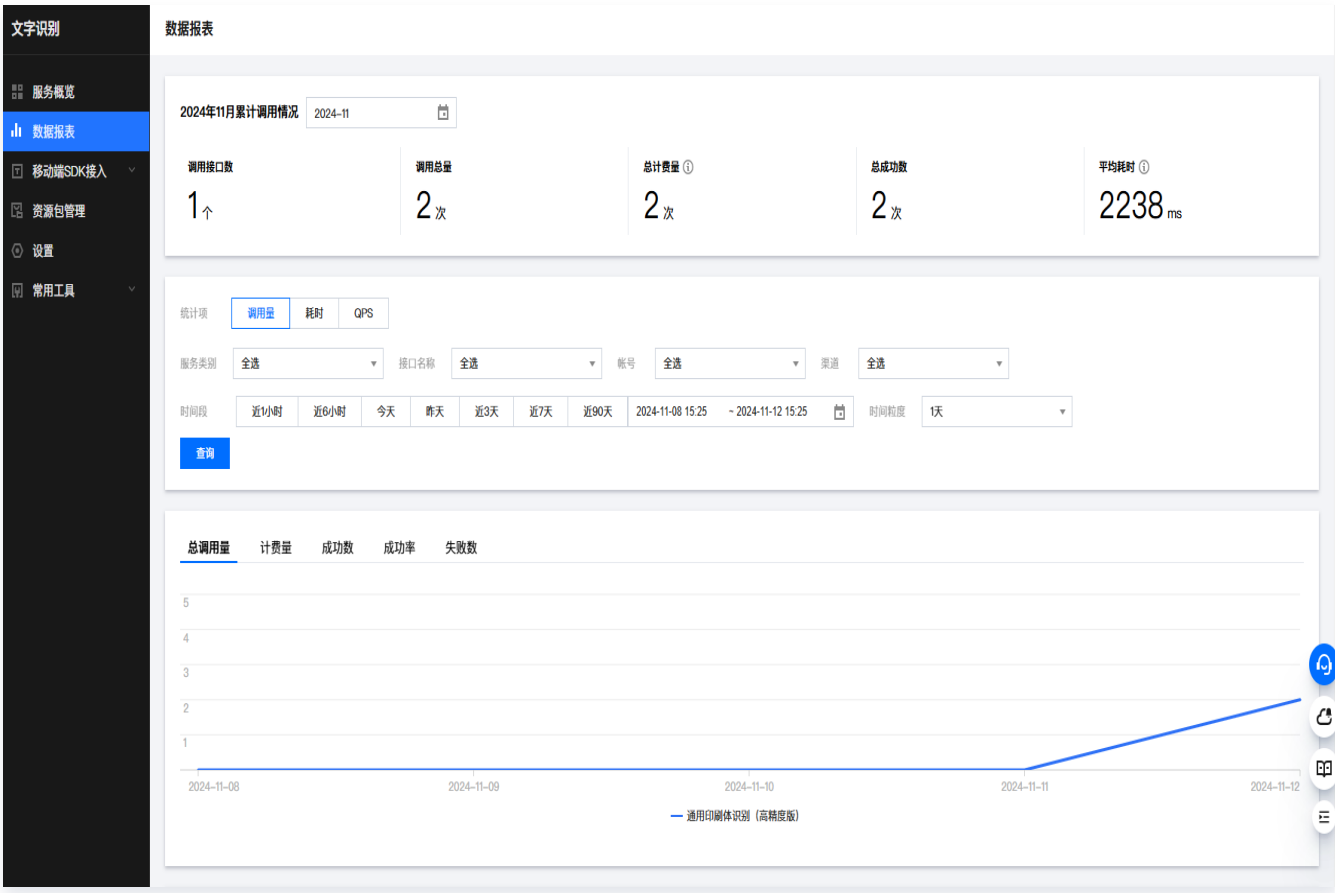
1. 腾讯云文字识别提供了丰富的demo体验，可以提前体验一下文档智能识别效果（文档智能提供基础版与多模态版选择）：腾讯云文字识别 demo 体验 。
2. 在使用腾讯云文字识别之前需要开通文字识别服务。打开腾讯云 OCR 控制台 页面，我们成功开通了文字识别服务。



3. 服务开通成功后，腾讯云文字识别赠送了免费的资源包，其中文档抽取（多模态版）有1000次的免费额度，可以在 资源包管理页面 查看资源包使用情况。
4. 当免费资源包用尽后，我们先根据使用情况购买了部分预付费资源包，后来又开通了后付费，保证业务可以持续正常调用接口。
    - 我们首先评估了业务的请求量级，于是在文字识别 购买页 购买了文档抽取（多模态版）100万次的资源包，资源包购买的越多优惠越大。可以在 资源包管理页面 中查看资源包的具体使用情况。



○ 然后在 数据报表页面 可以看到接口的调用量。



- 最后我们在 设置页面 开通了后付费服务，这样就不用担心资源包耗尽导致调用接口失败了。需要注意后付费设置每月只能变更一次。

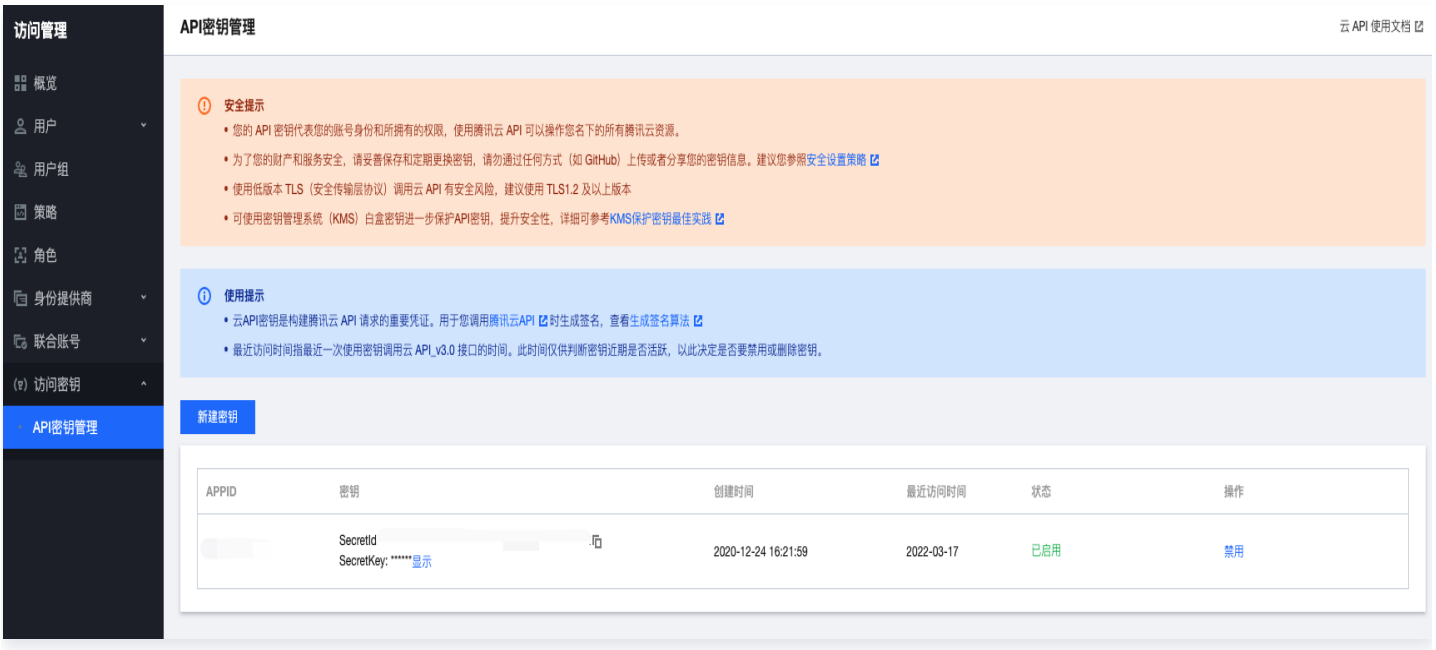
开发流程

通过下面几个步骤就可以正式使用文档抽取（多模态版）能力了。

- 获取个人密钥
- 文档抽取（多模态版）API 文档
- 体验在线调用
- 使用集成腾讯云 OCR 的 SDK
- 查询调用量

获取个人密钥

首先，我们需要获取个人 API 密钥，用于接口的调用。打开腾讯云访问管理的 API 密钥管理页面，可以创建个人密钥。

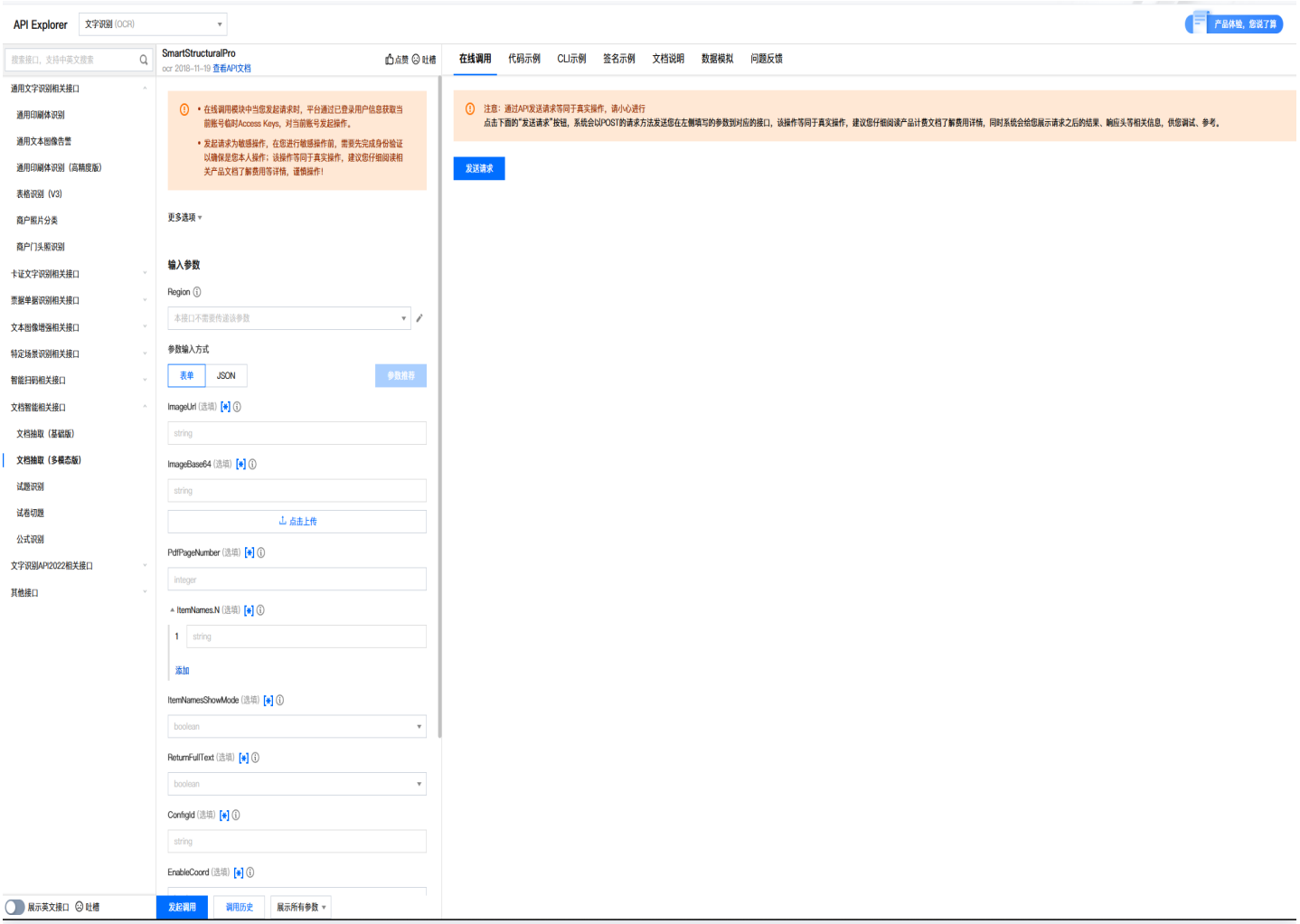


文档抽取（多模态版）API 文档

查看接口具体的使用说明，在文字识别的 API 文档 中可以查看文档抽取（多模态版）的输入参数、输出参数、错误码、示例等信息。

在线调试

腾讯云文字识别提供了在线调用 API Explorer 工具，方便我们可视化调用，并生成调用代码，可以直观的看到请求参数和返回参数。



使用 SDK 调用

接下来可以正式接入接口使用了，在文档抽取（多模态版）API 文档的最下方，提供了多个语言的开发工具集（SDK），SDK 的使用方法十分简单方便，我们可以根据自己需要的语言选择接入。

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python
- Tencent Cloud SDK 3.0 for Java
- Tencent Cloud SDK 3.0 for PHP
- Tencent Cloud SDK 3.0 for Go
- Tencent Cloud SDK 3.0 for NodeJS
- Tencent Cloud SDK 3.0 for .NET
- Tencent Cloud SDK 3.0 for C++
- Tencent Cloud SDK 3.0 for Ruby

命令行工具

- Tencent Cloud CLI 3.0

我们使用的开发语言是 GoLang。

1. 安装公共基础包

```
go get -v -u github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common
```

2. 安装文字识别对应的产品包

```
go get -v -u github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/ocr
```

3. 实现调用逻辑（仅为主要逻辑，非完整代码）

```
package main

import (
 "fmt"
 "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
 "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/errors"
 "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/profile"
 "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/regions"
 ocr "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/ocr/v20181119"
)

func main() {
 credential := common.NewCredential("secretId", "secretKey")
 client, _ := ocr.NewClient(credential, regions.Guangzhou,
profile.NewClientProfile())

 request := ocr.NewSmartStructuralOCRRequest()
 request.ImageUrl = common.StringPtr("xxxxx")
 response, err := client.SmartStructuralOCR(request)

 if _, ok := err.(*errors.TencentCloudSDKError); ok {
     fmt.Printf("An API error has returned: %s", err)
     return
 }
 if err != nil {
     panic(err)
 }
 fmt.Printf("%s\n", response.ToJsonString())
}
```

4. 文档抽取（多模态版）特定参数使用

我们的业务场景针对医疗资质的审核往往比较看重其中特定字段，例如类别、编号，地址、姓名，有效期等，在使用文档抽取（多模态版）接口识别医疗资质证书时，我们需要文档抽取（多模态版）接口返回这些特定字段，方便我们进一步的审核。

查阅了文档抽取（多模态版）接口文档后，我们发现可以自定义结构化功能需返回的字段名称，在请求时候传入对应的参数即可。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：SmartStructuralOCR。
Version	是	String	公共参数，本接口取值：2018-11-19。
Region	是	String	公共参数，详见产品支持的 地域列表，本接口仅支持其中的: ap-bangkok, ap-beijing, ap-guangzhou, ap-hongkong, ap-shanghai, na-toronto
ImageUrl	否	String	图片的 Url 地址。 支持的图片格式：PNG、JPG、JPEG，暂不支持 GIF 格式。 支持的图片大小：所下载图片经 Base64 编码后不超过 7M。图片下载时间不超过 3 秒。 图片存储于腾讯云的 Url 可保障更高的下载速度和稳定性，建议图片存储于腾讯云。 非腾讯云存储的 Url 速度和稳定性可能受一定影响。
ImageBase64	否	String	图片的 Base64 值。 支持的图片格式：PNG、JPG、JPEG，暂不支持 GIF 格式。 支持的图片大小：所下载图片经Base64编码后不超过 7M。图片下载时间不超过 3 秒。 图片的 ImageUrl、ImageBase64 必须提供一个，如果都提供，只使用 ImageUrl。
ItemNames.N	否	Array of String	自定义结构化功能需返回的字段名称，例： 若客户只想返回姓名、性别两个字段的识别结果，则输入 ItemNames=["姓名","性别"]

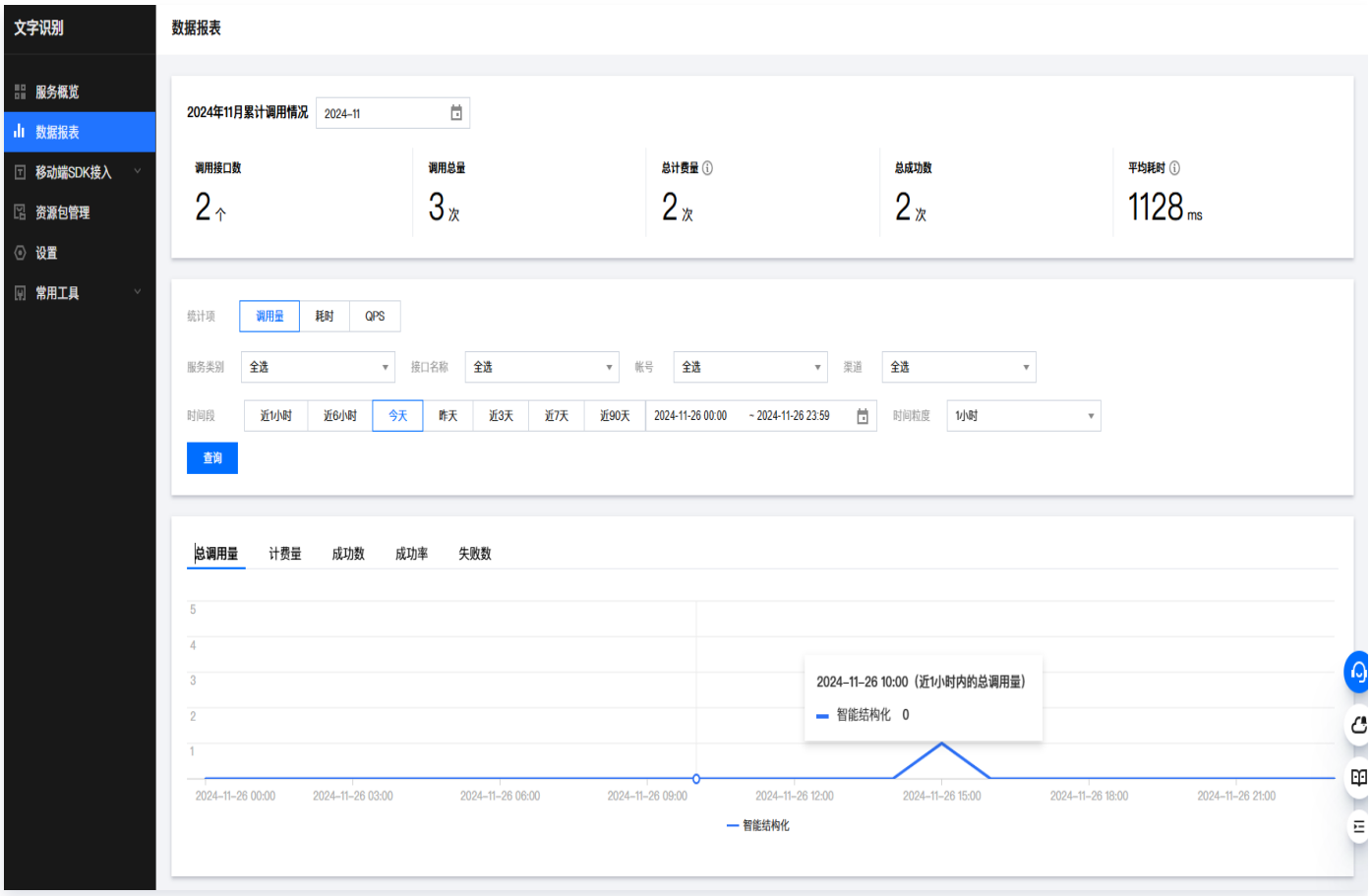
传入自定义参数，让文档抽取（多模态版）接口返回特定字段，包括：类别、编号，地址、姓名，有效期，调用逻辑如下（**仅为主要逻辑，非完整代码**）

```
package main

import (
    "fmt"
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
    "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/errors"
    "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/profile"
    "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/regions"
    ocr "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/ocr/v20181119"
)

func main() {
    credential := common.NewCredential("secretId", "secretKey")
    client, _ := ocr.NewClient(credential, regions.Guangzhou,
profile.NewClientProfile())

    request := ocr.SmartStructuralOCRRequest{
        ImageUrl: common.StringPtr("xxx") ,
        ItemNames: common.StringPtrs([]string{"类别","编号","地址","姓
名","有效期"}),
    }
    response, err := client.SmartStructuralOCR(request)

    if _, ok := err.(*errors.TencentCloudSDKError); ok {
        fmt.Printf("An API error has returned: %s", err)
        return
    }
    if err != nil {
        panic(err)
    }
    fmt.Printf("%s\n", response.ToJsonString())
}
```

5. 查询调用量

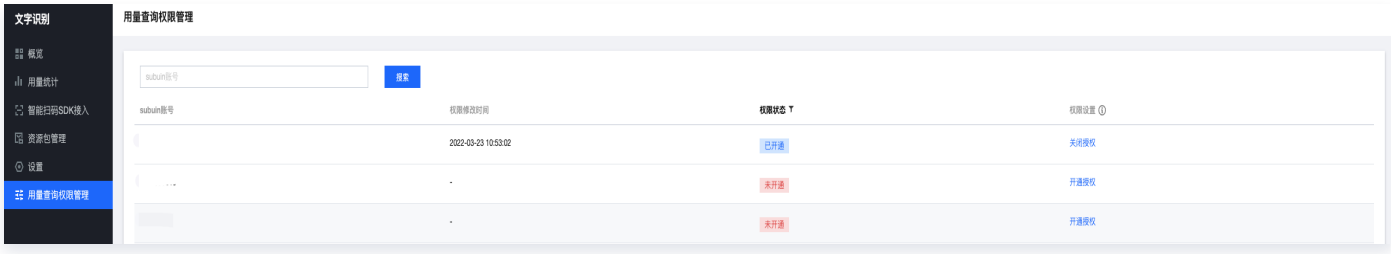
接口调用成功后，我们可以在 腾讯云文字识别控制台 查看接口的调用明细，包括调用量、成功量、失败量、错

误码等信息。



> **注意**
>
> 主账号登录后可以查看所有账号的调用量明细，子账号只能查询自己的调用量明细。

如果想要让子账号也有权限查看所有账号的调用明细，可以在 用量查询权限管理页面 给子账号赋权，这样子账号也可以查询到所有账号的调用量。



用腾讯云 AI 文本图像增强打造一个掌上扫描仪

最近更新时间：2024-12-26 11:11:02

在日常生活、工作中， 受限于拍照技术、拍摄条件等制约，得到的文本图像往往存在光照不均、角度倾斜、文字模糊等情况。这种低质量的文本图像不仅不利于保存和后续研究，也不利于光学字符识别。为了解决以上问题，特别调研了业内相关的产品，发现腾讯云 AI 的文本图像增强能力可以很好的打造一个掌上扫描仪。具体来说，软件底层采用计算机视觉技术，面向文本类图片场景提供图像处理服务，包括切边增强、弯曲矫正、阴影去除、摩尔纹去除等能力，可以有效优化文档类的图片质量，提升文字的清晰度，极大提高了低质量的文本图像的质量；用户操作方便只需要上传需要增强的文本图像，就可以自动处理图像，待图像处理完成后，用户就可以下载增强后的图片。
接下来，我将详细讲述掌上扫描仪的实现过程。

一、准备工作

为了使用腾讯云文本图像增强能力，我做了以下几个准备工作。

1.1 开通文本图像增强功能

在使用腾讯云文本图像增强之前，通过腾讯云官网开通 文本图像增强 服务。

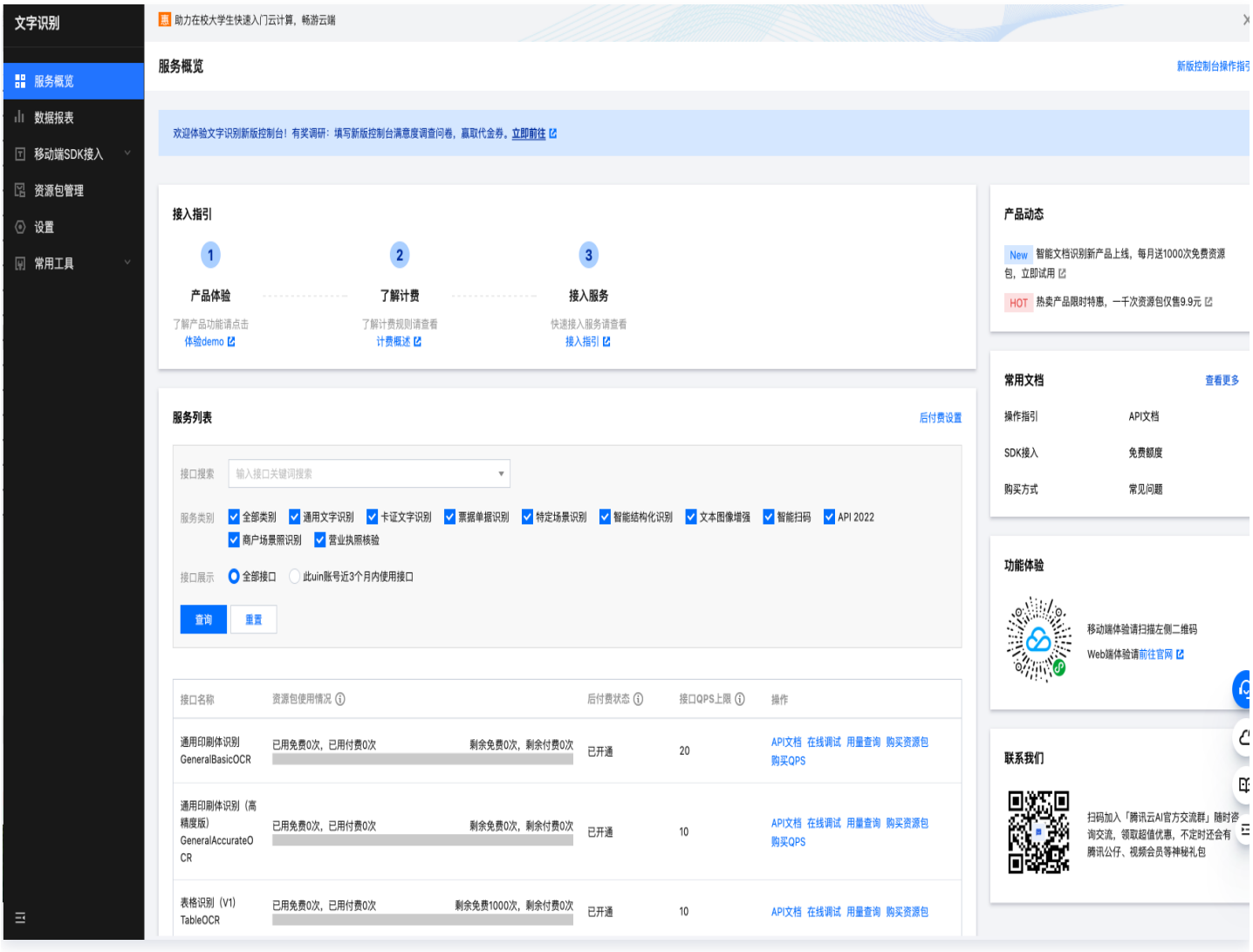
服务开通成功后，腾讯云 AI 文字识别赠送了免费的资源包，其中文本图像增强1000次免费额度，可以在 资源包管理页面 查看资源包使用情况。

营业执照识别月免费资源包	赠送	1000	0 (0%)	1000 (100%)	2022-09-01 00:00:00	2022-09-30 23:59:59	退款
智能结构化-1k次免费资源包	赠送	1000	0 (0%)	1000 (100%)	2022-09-01 00:00:00	2022-09-30 23:59:59	退款
行驶证/驾驶证识别月免费资源包	赠送	1000	0 (0%)	1000 (100%)	2022-09-01 00:00:00	2022-09-30 23:59:59	退款
车牌识别月免费资源包	赠送	1000	0 (0%)	1000 (100%)	2022-09-01 00:00:00	2022-09-30 23:59:59	退款
通用印刷体识别月免费资源包	赠送	1000	0 (0%)	1000 (100%)	2022-09-01 00:00:00	2022-09-30 23:59:59	退款
通用手写体识别月免费资源包	赠送	1000	0 (0%)	1000 (100%)	2022-09-01 00:00:00	2022-09-30 23:59:59	退款
身份证识别月免费资源包	赠送	1000	0 (0%)	1000 (100%)	2022-09-01 00:00:00	2022-09-30 23:59:59	退款
名片识别1000次月免费资源包	赠送	1000	0 (0%)	1000 (100%)	2022-09-01 00:00:00	2022-09-30 23:59:59	退款
医疗票据识别-1k次免费资源包	赠送	1000	0 (0%)	1000 (100%)	2022-09-01 00:00:00	2022-09-30 23:59:59	退款
文本图像增强-1k次免费资源包	赠送	1000	20 (2%)	980 (98%)	2022-09-01 00:00:00	2022-09-30 23:59:59	退款

通过使用我发现在设置页面开通了后付费服务，这样就不用担心资源包耗尽导致调用接口失败了，但是后付费设置每月只能变更一次。

1.2 控制台监控信息

经过使用，我了解到所有文字识别服务的使用情况都可以在 控制台 中查看使用信息，可以从下图看到统计出当前月份的调用情况、计费情况、成功数、成功率等。



二、操作流程

通过以下几个步骤，就可以使用腾讯云 AI 文字识别的图像增强功能制作掌上扫描仪。

- 获取个人密钥
- 查看图像增强 API 文档
- 使用腾讯云 AI 文字识别的图像增强功能制作掌上扫描仪
- 体验掌上扫描仪的效果

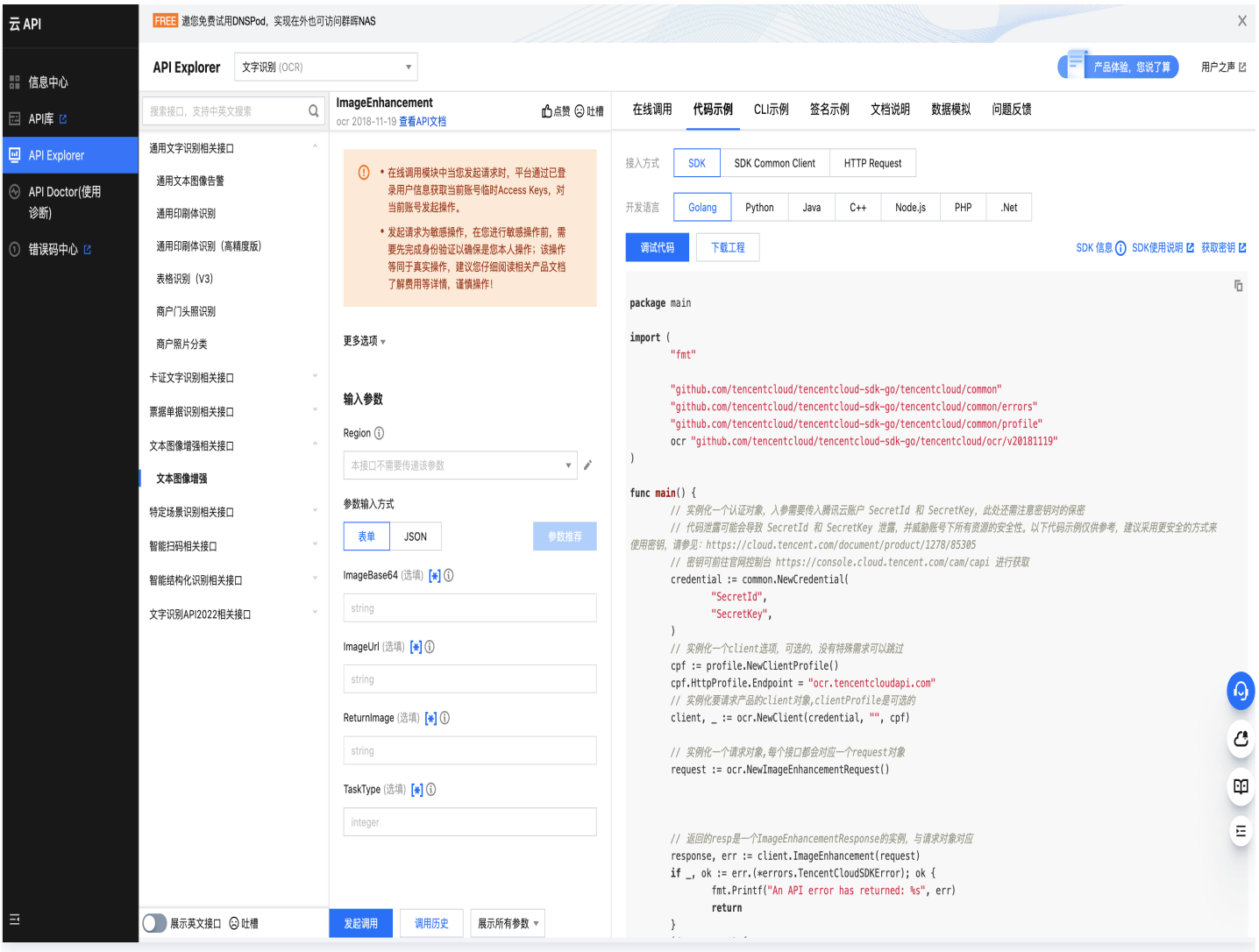
2.1 获取个人密钥

在腾讯云访问管理的 API 密钥管理页面，我们新建一个个人密钥。

复制生成的密钥，可以单击这里直达。

2.2 图像增强 API 接口说明

可以在 API Explorer 中选择文字图像增强—输入参数—选择需要的语言—即可生成对应语言的 API 调用代码。



2.3 使用腾讯云 AI 文字识别的图像增强功能制作掌上扫描仪

掌上扫描仪产品实现过程中主要分为以下几个步骤：

- 安装环境依赖的 SDK
- 调用图像增强接口

2.3.1 安装环境依赖的 SDK

```
#    安装公共基础包
go get -v -u github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common
#    安装对应的产品包
go get -v -u github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/cvm
#    一次性下载腾讯云所有产品的包
```

```
go get -v -u github.com/tencentcloud/tencentcloud-sdk-go
```

2.3.2 调用图像增强接口

```
package imageenhancement

import (
    "encoding/base64"
    "fmt"
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
    "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/errors"
    "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/profile"
    ocr "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/ocr/v20181119"
    "io/ioutil"
    "os"
    "testing"
)

//MainImageEnhancement 主函数
func MainImageEnhancement(imagesPath string) {
    //图片地址

    // 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey,此处还需注意
密钥对的保密
    // 密钥可前往https://console.cloud.tencent.com/cam/capi网站进行获取
    credential := common.NewCredential(
        //这里填入腾讯云账户密钥对
        "SecretId",
        "SecretKey",
    )
    // 实例化一个client选项，可选的，没有特殊需求可以跳过
    cpf := profile.NewClientProfile()
    cpf.HttpProfile.Endpoint = "ocr.tencentcloudapi.com"
    // 实例化要请求产品的client对象,clientProfile是可选的
    client, _ := ocr.NewClient(credential, "ap-guangzhou", cpf)
    //读取图片base64
    toBase64Str, err := imageToBase64(imagesPath)
    respImages, err := imageType208(*client, toBase64Str)
    if err != nil {
        return
```

```
    }
    err = writeFile("test.jpg", *respImages)
    if err != nil {
        return
    }
}

//imageType1 切边增强
func imageType1(client ocr.Client, toBase64Str string) (*string, error)
{
    // 实例化一个请求对象,每个接口都会对应一个request对象
    request := ocr.NewImageEnhancementRequest()
    request.ImageBase64 = common.StringPtr(toBase64Str)
    request.ReturnImage = common.StringPtr("preprocess")
    request.TaskType = common.Int64Ptr(1)

    // 返回的resp是一个ImageEnhancementResponse的实例，与请求对象对应
    response, err := client.ImageEnhancement(request)
    if _, ok := err.(*errors.TencentCloudSDKError); ok {
        fmt.Printf("An API error has returned: %s", err)
        return nil, err
    }
    if err != nil {
        return nil, err
    }
    return response.Response.Image, nil
}

//imageType2 弯曲矫正
func imageType2(client ocr.Client, toBase64Str string) (*string, error)
{
    // 实例化一个请求对象,每个接口都会对应一个request对象
    request := ocr.NewImageEnhancementRequest()
    request.ImageBase64 = common.StringPtr(toBase64Str)
    request.ReturnImage = common.StringPtr("preprocess")
    request.TaskType = common.Int64Ptr(2)

    // 返回的resp是一个ImageEnhancementResponse的实例，与请求对象对应
    response, err := client.ImageEnhancement(request)
    if _, ok := err.(*errors.TencentCloudSDKError); ok {
        fmt.Printf("An API error has returned: %s", err)
        return nil, err
    }
```

```
    if err != nil {
        return nil, err
    }
    return response.Response.Image, nil
}

//imageType202 黑白模式
func imageType202(client ocr.Client, toBase64Str string) (*string,
error) {
    // 实例化一个请求对象,每个接口都会对应一个request对象
    request := ocr.NewImageEnhancementRequest()
    request.ImageBase64 = common.StringPtr(toBase64Str)
    request.ReturnImage = common.StringPtr("preprocess")
    request.TaskType = common.Int64Ptr(202)

    // 返回的resp是一个ImageEnhancementResponse的实例，与请求对象对应
    response, err := client.ImageEnhancement(request)
    if _, ok := err.(*errors.TencentCloudSDKError); ok {
        fmt.Printf("An API error has returned: %s", err)
        return nil, err
    }
    if err != nil {
        return nil, err
    }
    return response.Response.Image, nil
}

... //这里省略重复的部分，可以扩展其他模式或者任意模式组合

//writeFile base64转image
func writeFile(path string, s string) error {
    //解析base64字符串
    dist, _ := base64.StdEncoding.DecodeString(s)
    //写入新文件
    f, _ := os.OpenFile(path, os.O_RDWR|os.O_CREATE, os.ModePerm)
    defer f.Close()
    _, err := f.Write(dist)
    return err
}

//imageToBase64 img转base64
func imageToBase64(filePath string) (string, error) {
    srcByte, err := ioutil.ReadFile(filePath)
```

```
    if err != nil {
        return "", err
    }
    res := base64.StdEncoding.EncodeToString(srcByte)
    return res, nil
}
```

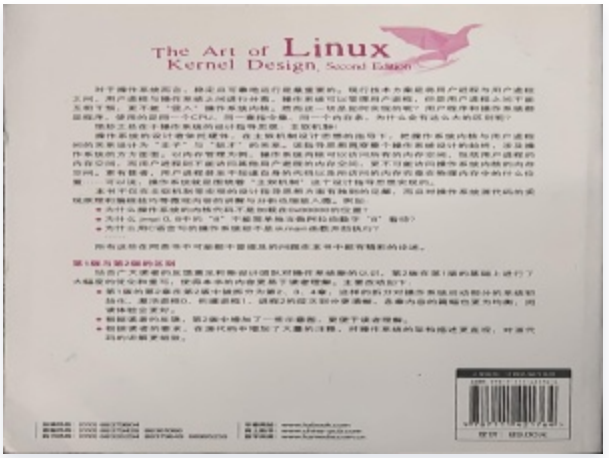
2.4 体验掌上扫描仪的效果

1）角度矫正

原始图片	切边增强后图片
	
	

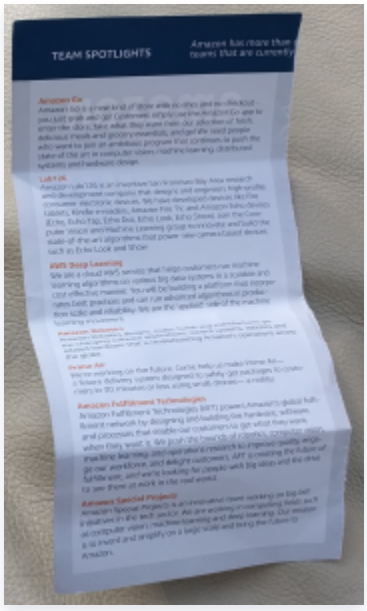
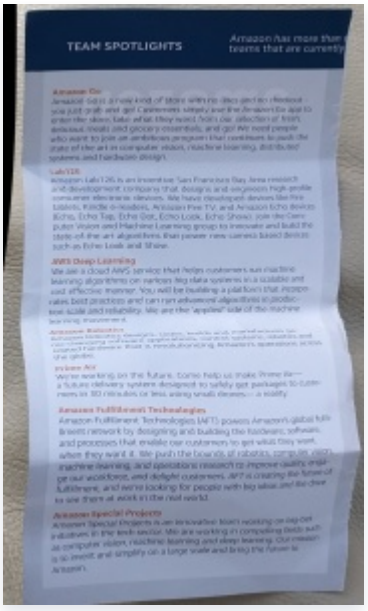

SOUTHERN
GATWICK AIRPORT SOUTH TERMINAL 51
CUSTOMER COPY
Keep this copy for your records
Sales - see receipt £40.00
Customer Present
ICC
PAN : CHASE VISA
PAN Seq No: 01
AID: A0000000031010
Merchant No: ****48571
TID: ****0532
Auth code: 035101
Date: 04/09/2017
Time: 18:24:49
Ref No: 5730F-66287-51LU2-66287
Please debit my account by amount £40.00
NO CARDHOLDER VERIFICATION
Please retain for your records
04/09/2017 18:25:02 55433 526017 51 5730

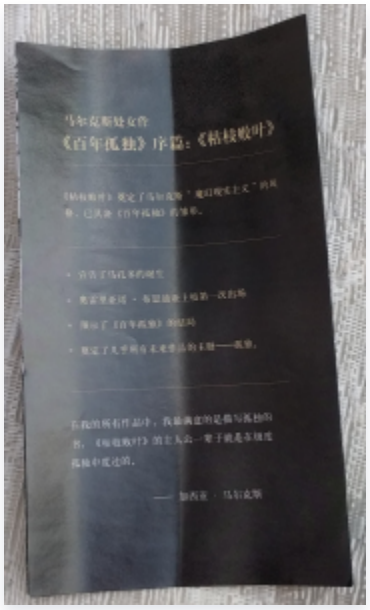

SOUTHERN
GATWICK AIRPORT SOUTH TERMINAL 51
CUSTOMER COPY
Keep this copy for your records
Sales - see receipt £40.00
Customer Present
ICC
PAN : CHASE VISA
PAN Seq No: 01
AID: A0000000031010
Merchant No: ****48571
TID: ****0532
Auth code: 035101
Date: 04/09/2017
Time: 18:24:49
Ref No: 5730F-66287-51LU2-66287
Please debit my account by amount £40.00
NO CARDHOLDER VERIFICATION
Please retain for your records
04/09/2017 18:25:02 55433 526017 51 5730


The Art of Linux
Kernel Design, Second Edition

经过角度矫正后，可以从上图中看出，角度矫正后的图片更加凸显文本内容，提高了文本图像的质量。

2）弯曲矫正

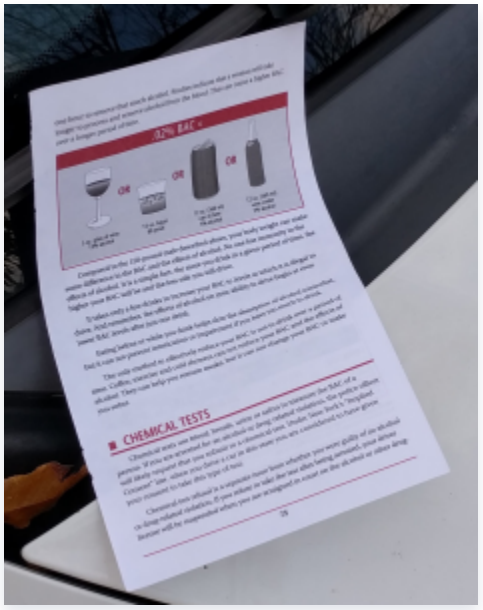
原始图片	矫正后图像
	
	

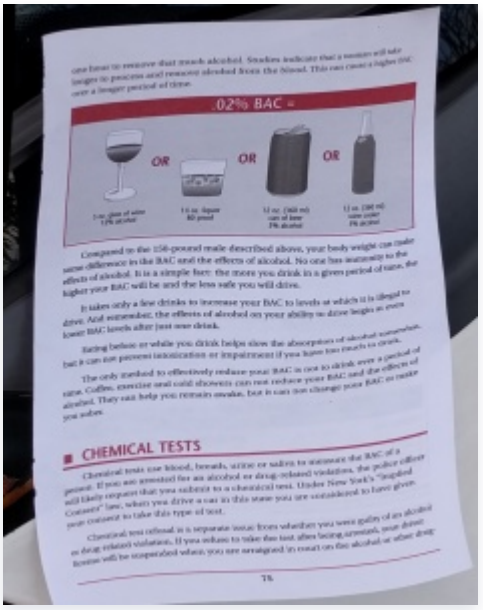

马尔克斯处女作
《百年孤独》序篇：《枯枝败叶》


马尔克斯处女作
《百年孤独》序篇：《枯枝败叶》


THE ART OF
FINDING FREEDOM
BY RAGHAVA LAKSHMINARAYA-


THE ART OF
FINDING FREEDOM
BY RAGHAVA LAKSHMINARAYA-





经过弯曲矫正后，可以从上图中看出，弯曲矫正后的图片文本更加清晰，提高了文本图像的质量。

3）去除摩尔纹

原始图片	去除摩尔纹后的图片

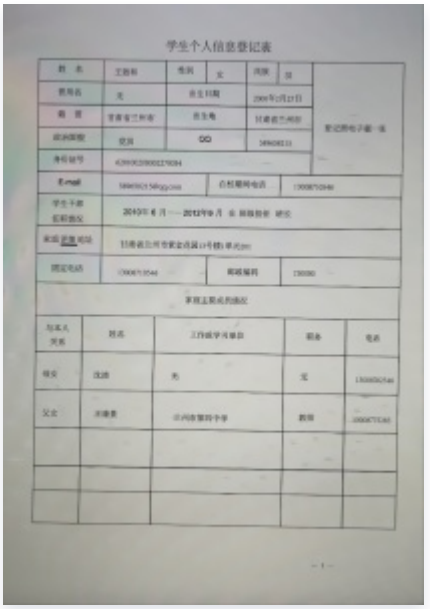

航空运输电子客票行程单
4985382968 6
VOID
VOID
CNY 2280.00
CNY 10.00
CNY 2340.00
2018-09-29


航空运输电子客票行程单
4985382968 6
VOID
VOID
CNY 2280.00
CNY 10.00
CNY 2340.00
2018-09-29


水草缠身自救法
01
02
03

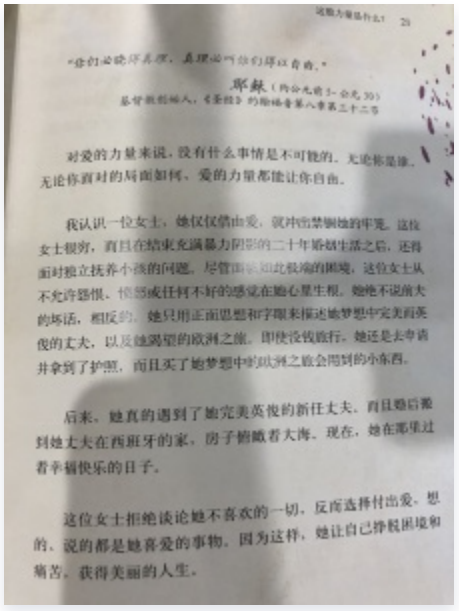
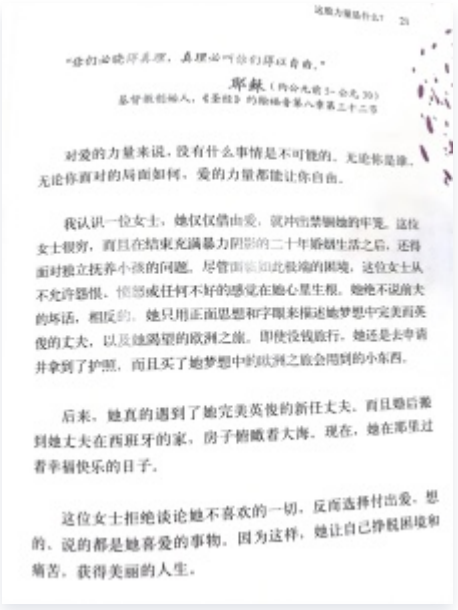

水草缠身自救法
01
02
03

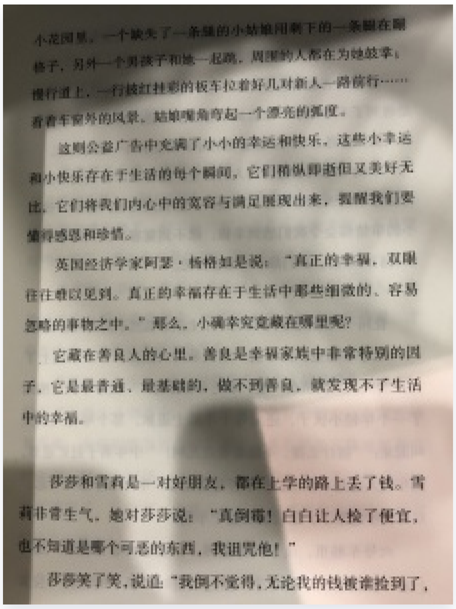

学生个人信息登记表


学生个人信息登记表

经过去除摩尔纹处理，很大程度的提高了文本图像的清晰度，排除了摩尔纹的干扰，提高了文本图像的质量。

4）去除阴影

原始图片	去除阴影后的图片
	



小花园里，一个缺失了一条腿的小姑娘用剩下的一条腿在跳格子，另外一个男孩子和她一起跳，周围的人都在为她鼓掌；慢行道上，一行披红挂彩的板车拉着好几对新人一路前行……看着车窗外的风景，姑娘嘴角弯起一个漂亮的弧度。

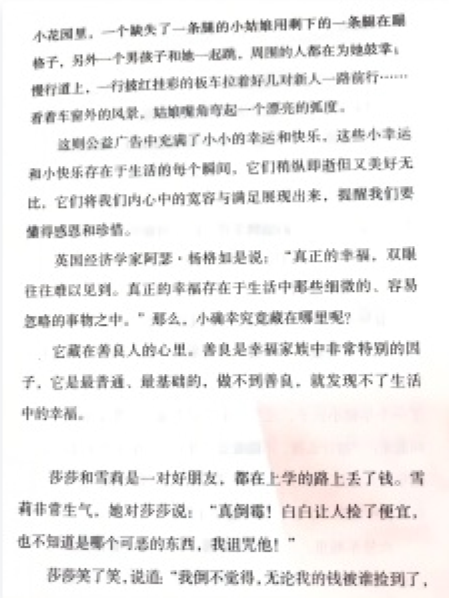
这则公益广告中充满了小小的幸运和快乐，这些小幸运和小快乐存在于生活的每个瞬间，它们稍纵即逝但又美好无比，它们将我们内心中的宽容与满足展现出来，提醒我们要懂得感恩和珍惜。

英国经济学家阿瑟·杨格如是说："真正的幸福，双眼往往难以见到，真正的幸福存在于生活中那些细微的、容易忽略的事物之中。"那么，小确幸究竟藏在哪里呢？

它藏在善良人的心里。善良是幸福家族中非常特别的因子，它是最普通、最基础的，做不到善良，就发现不了生活中的幸福。

莎莎和雪莉是一对好朋友，都在上学的路上丢了钱。雪莉非常生气，她对莎莎说："真倒霉！白白让人捡了便宜，也不知道是哪个可恶的东西，我诅咒他！"

莎莎笑了笑，说道："我倒不觉得，无论我的钱被谁捡到了，



小花园里，一个缺失了一条腿的小姑娘用剩下的一条腿在跳格子，另外一个男孩子和她一起跳，周围的人都在为她鼓掌；慢行道上，一行披红挂彩的板车拉着好几对新人一路前行……看着车窗外的风景，姑娘嘴角弯起一个漂亮的弧度。

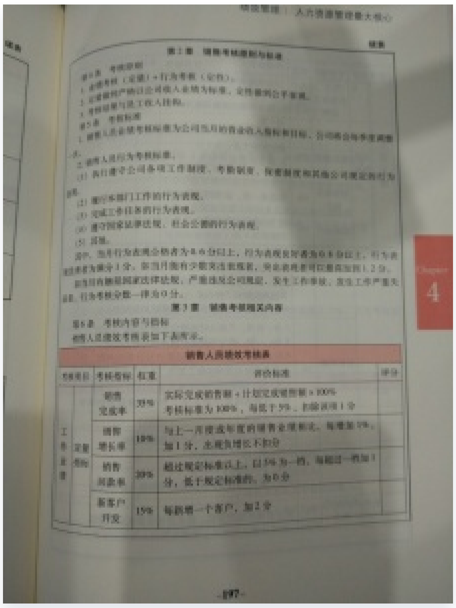
这则公益广告中充满了小小的幸运和快乐，这些小幸运和小快乐存在于生活的每个瞬间，它们稍纵即逝但又美好无比，它们将我们内心中的宽容与满足展现出来，提醒我们要懂得感恩和珍惜。

英国经济学家阿瑟·杨格如是说："真正的幸福，双眼往往难以见到，真正的幸福存在于生活中那些细微的、容易忽略的事物之中。"那么，小确幸究竟藏在哪里呢？

它藏在善良人的心里。善良是幸福家族中非常特别的因子，它是最普通、最基础的，做不到善良，就发现不了生活中的幸福。

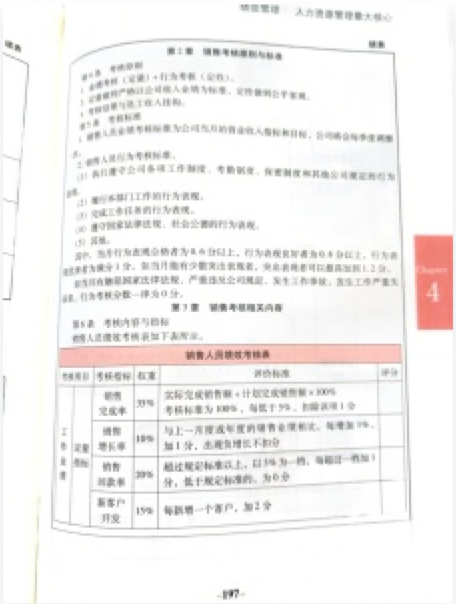
莎莎和雪莉是一对好朋友，都在上学的路上丢了钱。雪莉非常生气，她对莎莎说："真倒霉！白白让人捡了便宜，也不知道是哪个可恶的东西，我诅咒他！"

莎莎笑了笑，说道："我倒不觉得，无论我的钱被谁捡到了，



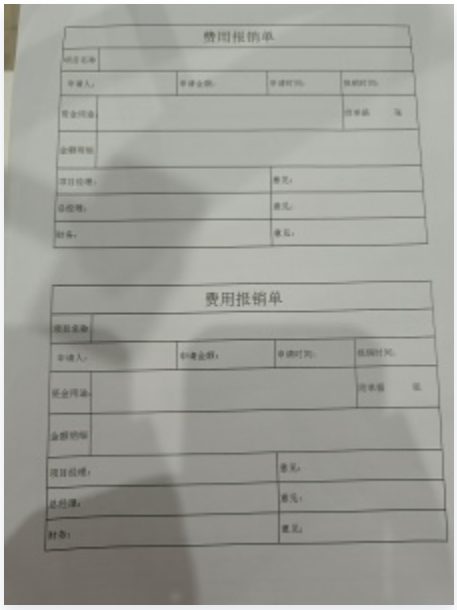
Chapter 4

-197-

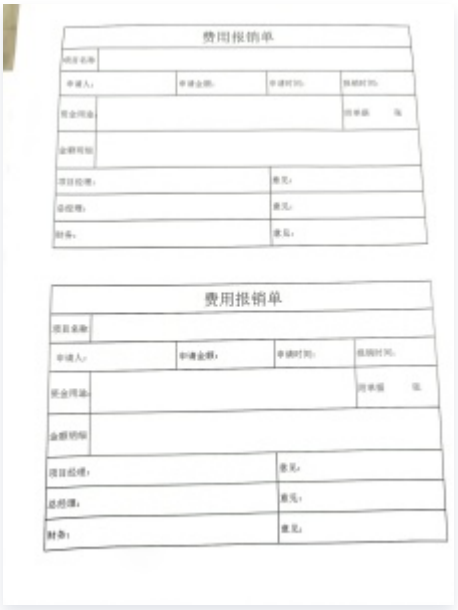


Chapter 4

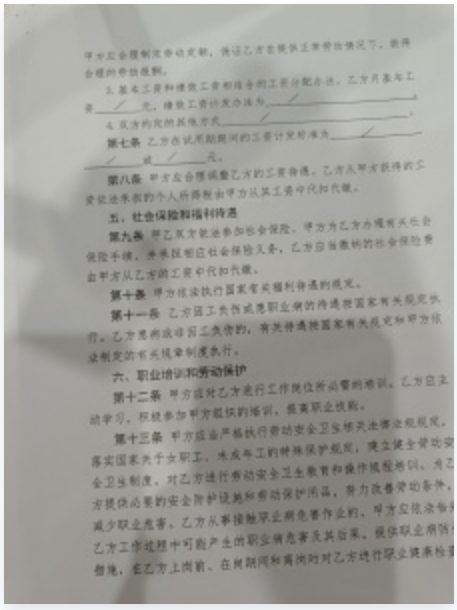
-197-


费用报销单

费用报销单


费用报销单

费用报销单


甲方应合理制定劳动定额，保证乙方在按执正常劳动情况下，获得合理的劳动报酬。

3. 基本工资和绩效工资相结合的工资分配办法。乙方月基本工资_____元，绩效工资计发办法为_____。

4. 双方约定的其他方式_____。

第七条 乙方在试用期期间的工资计发标准为_____元。

第八条 甲方应合理调整乙方的工资待遇，乙方从甲方获得的工资收法律规定的个人所得税由甲方从其工资中代扣代缴。

五、社会保险和福利待遇

第九条 甲乙双方依法参加社会保险，甲方为乙方办理有关社会保险手续，并承担相应社会保险义务，乙方应当缴纳的社会保险费由甲方从乙方的工资中代扣代缴。

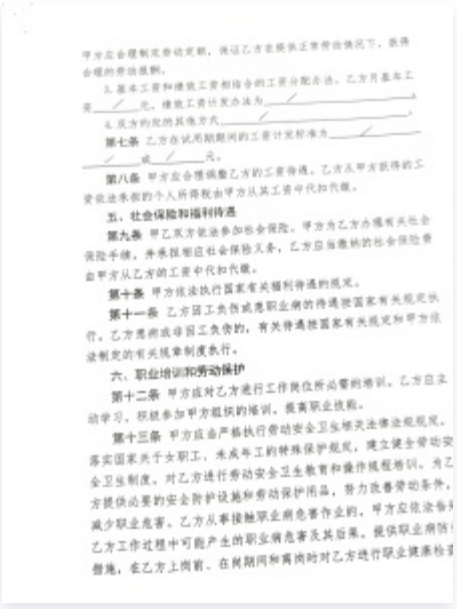
第十条 甲方依法执行国家有关福利待遇的规定。

第十一条 乙方因工负伤或患职业病的待遇按国家有关规定执行。乙方患病或非因工负伤的，有关待遇按国家有关规定和甲方依法制定的有关规章制度执行。

六、职业培训和劳动保护

第十二条 甲方应对乙方进行工作岗位所必需的培训。乙方应主动学习，积极参加甲方组织的培训，提高职业技能。

第十三条 甲方应由严格执行劳动安全卫生相关法律法规规定，落实国家关于女职工、未成年工的特殊保护规定，建立健全劳动安全卫生制度，对乙方进行劳动安全卫生教育和操作规程培训，为乙方提供必要的安全防护设施和劳动保护用品，努力改善劳动条件，减少职业危害。乙方从事接触职业病危害作业的，甲方应依法为乙方工作过程中可能产生的职业病危害及其后果，提供职业病防治措施，在乙方上岗前、在岗期间和离岗时对乙方进行职业健康检查


甲方应合理制定劳动定额，保证乙方在按执正常劳动情况下，获得合理的劳动报酬。

3. 基本工资和绩效工资相结合的工资分配办法。乙方月基本工资_____元，绩效工资计发办法为_____。

4. 双方约定的其他方式_____。

第七条 乙方在试用期期间的工资计发标准为_____元。

第八条 甲方应合理调整乙方的工资待遇，乙方从甲方获得的工资收法律规定的个人所得税由甲方从其工资中代扣代缴。

五、社会保险和福利待遇

第九条 甲乙双方依法参加社会保险，甲方为乙方办理有关社会保险手续，并承担相应社会保险义务，乙方应当缴纳的社会保险费由甲方从乙方的工资中代扣代缴。

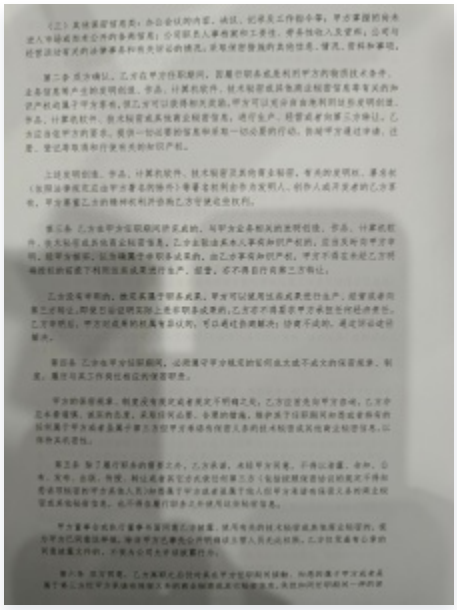
第十条 甲方依法执行国家有关福利待遇的规定。

第十一条 乙方因工负伤或患职业病的待遇按国家有关规定执行。乙方患病或非因工负伤的，有关待遇按国家有关规定和甲方依法制定的有关规章制度执行。

六、职业培训和劳动保护

第十二条 甲方应对乙方进行工作岗位所必需的培训。乙方应主动学习，积极参加甲方组织的培训，提高职业技能。

第十三条 甲方应由严格执行劳动安全卫生相关法律法规规定，落实国家关于女职工、未成年工的特殊保护规定，建立健全劳动安全卫生制度，对乙方进行劳动安全卫生教育和操作规程培训，为乙方提供必要的安全防护设施和劳动保护用品，努力改善劳动条件，减少职业危害。乙方从事接触职业病危害作业的，甲方应依法为乙方工作过程中可能产生的职业病危害及其后果，提供职业病防治措施，在乙方上岗前、在岗期间和离岗时对乙方进行职业健康检查



经过去除阴影处理，解决了因为环境因素对文本图像质量造成的影响，提高了文本图像的质量。

三、总结

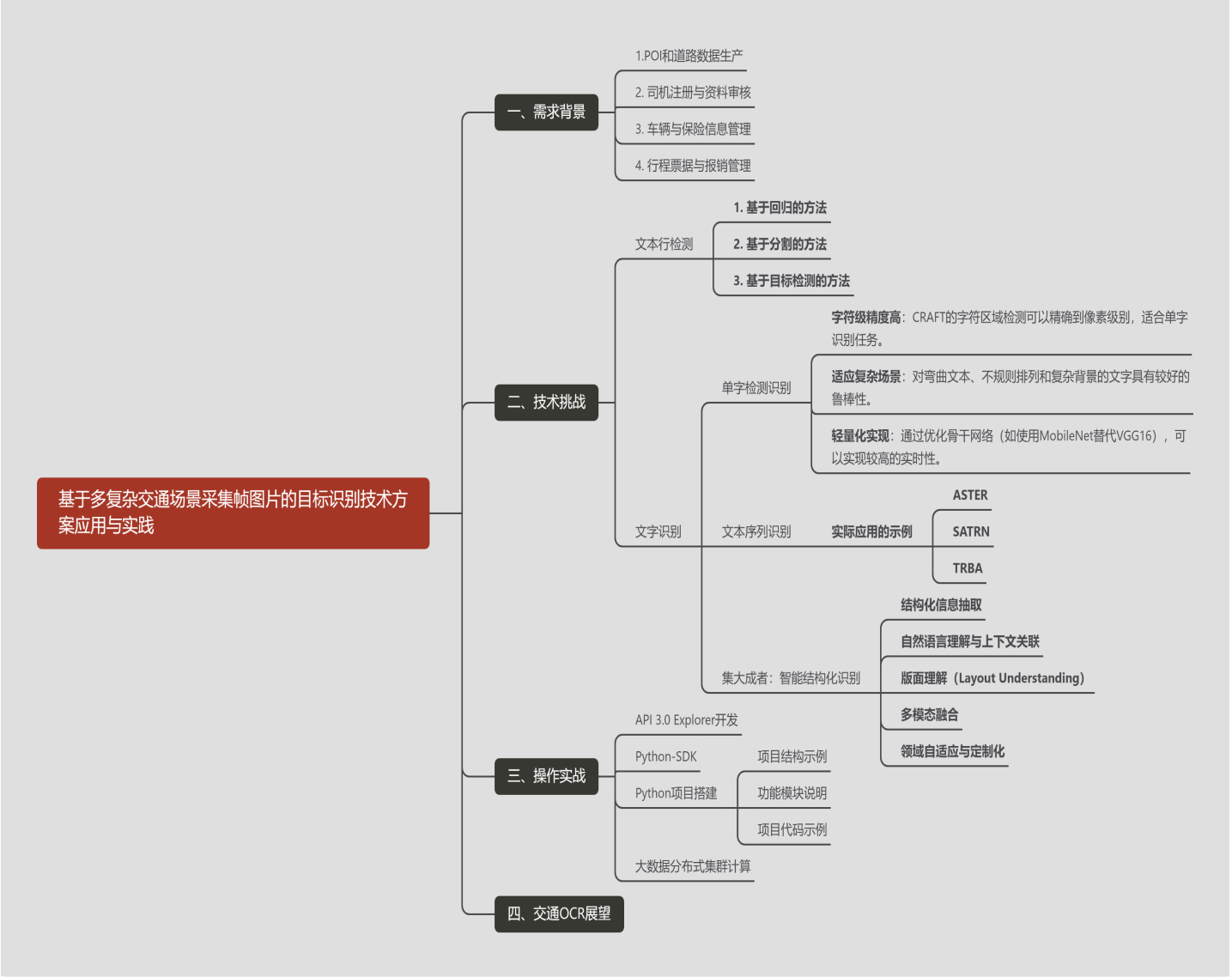
影响文本图像质量清晰程度有很多因素，室外光照度不均匀会造成图像灰度过于集中；摄像头获得的文本图像经过数/模转换，线路传输时都会产生噪声污染，文本图像质量不可避免降低，轻者表现为文本图像伴有噪点，难于看清文本图像细节；重者文本图像模糊不清，连大概文字轮廓都难以看清。因此，对图像进行分析处理之前，必须对图像进行改善。通过腾讯云AI的文本图像增强创造的掌上扫描仪解决了大部分文本图像不清晰的问题，提高了文本图像的质量。

基于多复杂交通场景采集帧图片的目标识别技术方案应用与实践

最近更新时间：2025-07-03 15:15:52

前言

本人曾有幸在一家大型地图公司任职数据挖掘工程师，几乎每日要处理 PB 级以上多复杂场景的交通数据，其中基于大数据的超量采集帧图片处理技术具有比较强的挑战性，也可以说 C 端产品很多优化细节以及用户体感好坏很大程度都取决于我们对于这些实时采集帧图片的处理和分析。
因此研究如何攻克基于大数据的超量图片技术处理和分析是我们一直以来探索和研究的目标，在探究的过程中我与团队尝试过相当多的解决方案最终都有不错的效果，那么如何不走弯，在短时间之内迅速研发出一套通用图片大数据处理挖掘方案，并有一定的可实现性与稳定性，就是本期我想开展的文章主题。
本文有较多行业术语和专业知识，博主尽量一一简化让读者易解，并通过真实案例 Demo 实践保证方案可行性。



一、需求背景

首先我想说明一点，尽管需求中数据格式与效果不尽相同，但是技术方案是可复用可延展的，因此不必考虑过多需求不一致的问题。这里我来简略介绍一下技术需求。

地图数据的丰富性和准确性决定了用户体验（C 端产品均如此），而传统的数据采集和分析制作，基本上都需要人工介入，再三确认上线，导致数据更新慢，加工成本高。因此我们需要通过对采集车高频的数据采集，运用图像算法能力，在 PB 集数目的采集图片中自动检测识别出各项地图要素的内容和位置，构建出实时更新的基础地图数据。(图来源高德地图)

高德的POI（Point Of Interest）数据　　POI数据的采集过程

这要求我们重点攻克场景文字识别技术在覆盖面、准确性和处理速度上的难题。在 POI 业务场景中，识别算法不仅需要尽可能全面地识别街边新开商铺的文字信息，还需确保识别结果的准确率达到99%以上，以支持 POI 名称的自动化生成；

在道路数据自动化处理场景中，识别算法需要精准捕捉道路标志牌上的细微变化，每天高效处理海量回传数据，以便及时更新道路限速、方向等关键信息。同时，由于采集设备和采集环境的多样性，高德场景文字识别算法常常需要应对极为复杂的图像条件，主要体现在以下几个方面：

1. **文字多样性**：文字语言、字体及排版形式丰富多变，例如商家招牌上的艺术字体、各类 LOGO 和多种排版样式。
2. **背景复杂性**：文字所在背景常常复杂多样，可能受到遮挡、光照不均或其他干扰因素的影响。
3. **图像来源多样性**：图像来源于低成本的众包设备，这些设备参数不一，成像质量参差不齐。图像可能存在倾斜、失焦、抖动等问题，进一步增加了识别难度。

除此之外，在网约车服务场景中也需要 文档智能 OCR 技术帮助平台和运营方高效处理大量与司机、车辆、行程相关的文档、票据和证件信息，以下都是需要处理的场景：

司机注册与资料审核

- 证件自动审核：在司机注册流程中，司机需上传驾驶证、行驶证、身份证、营运许可证等文件。文档智能 OCR 可自动识别证件信息，如姓名、证件号码、有效期、车牌号、车型和发证机构等，并将这些字段结构化录入系统。
- 多轮更新与验证：对于已注册的司机，每当证件信息发生变更（如新的保险单、车辆年检报告），司机可以直接上传新文件，OCR 将自动解析相关信息并更新后台数据库。由此实现司机资料的动态持续监控和自动化校对。

车辆与保险信息管理

- 车辆登记与年审信息录入：网约车平台对于车辆维护与合法性审查至关重要。文档智能 OCR 可将行驶证、车辆年检报告、保险单据中的车辆信息、保险到期日期、检测日期自动结构化提取并存档，以便系统定期提醒司机更新相关文件，确保车辆的合法合规状态。
- 保险理赔单据数字化：若车辆在营运中出现事故，需要提交保险理赔材料（如现场报案表格、维修清单、保险公司核损报告等）。通过 OCR，平台可自动提取事故地点、时间、赔偿金额、责任归属等信息，更快地进行理赔流程的记录、分析与追踪。

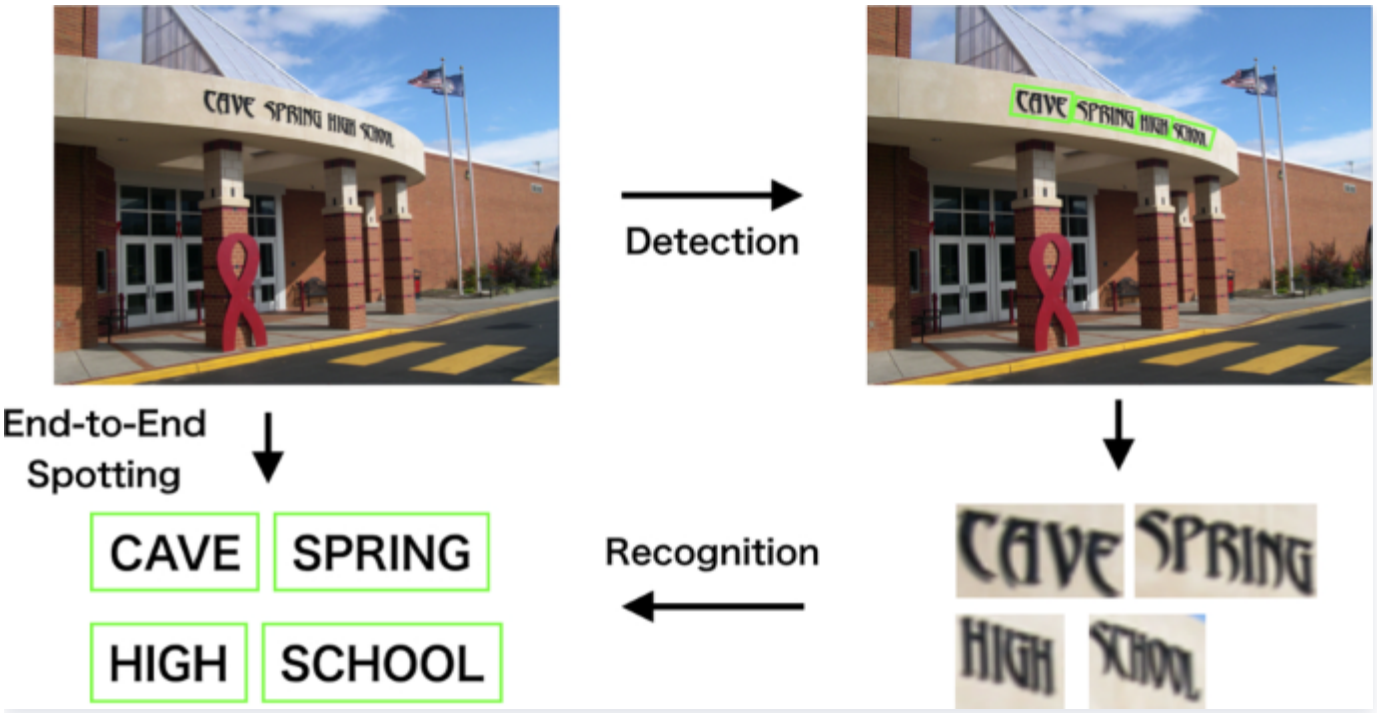
行程票据与报销管理

- 电子发票与支付凭证自动化处理：部分网约车平台可能需要定期为企业客户或商务出行用户提供发票报销服务。OCR 可快速识别发票编号、金额、开票时间、商户名称，将结果自动录入费用报销系统，从而简化后续财务操作。
- 停车费、路桥费票据解析：在部分场景下，司机或平台需要报销停车费、路桥费。文档智能 OCR 将停车票、过路费发票等票据中的收费金额、时间、地点自动解析并归档，让平台在财务结算或司机补贴审核时更高效。

总的来说，文档智能 OCR 在网约车场景中能够赋能平台实现从司机资质审核、车辆合规检查、保险理赔、费用报销到客服投诉处理的一系列流程自动化与智能化，大幅降低人工介入和出错率，为平台合规、风控、财务和客服等多层面运转提供高效率的数据基础。

二、技术挑战

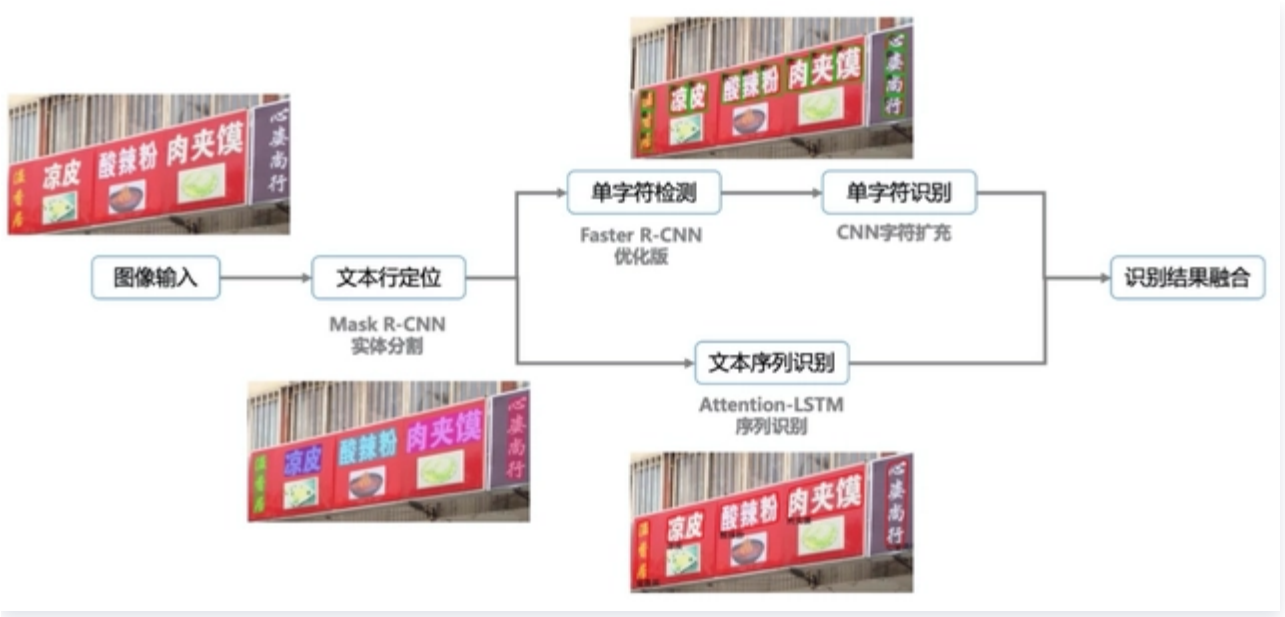
首先容许我介绍一下 STR 技术，场景文字识别（Scene Text Recognition，简称 STR）技术是一种专注于从自然场景图像中提取和识别文字信息的人工智能技术。它是计算机视觉与 自然语言处理 结合的典型应用，广泛应用于地图标注、文档处理、无人驾驶、增强现实和智能客服等领域。

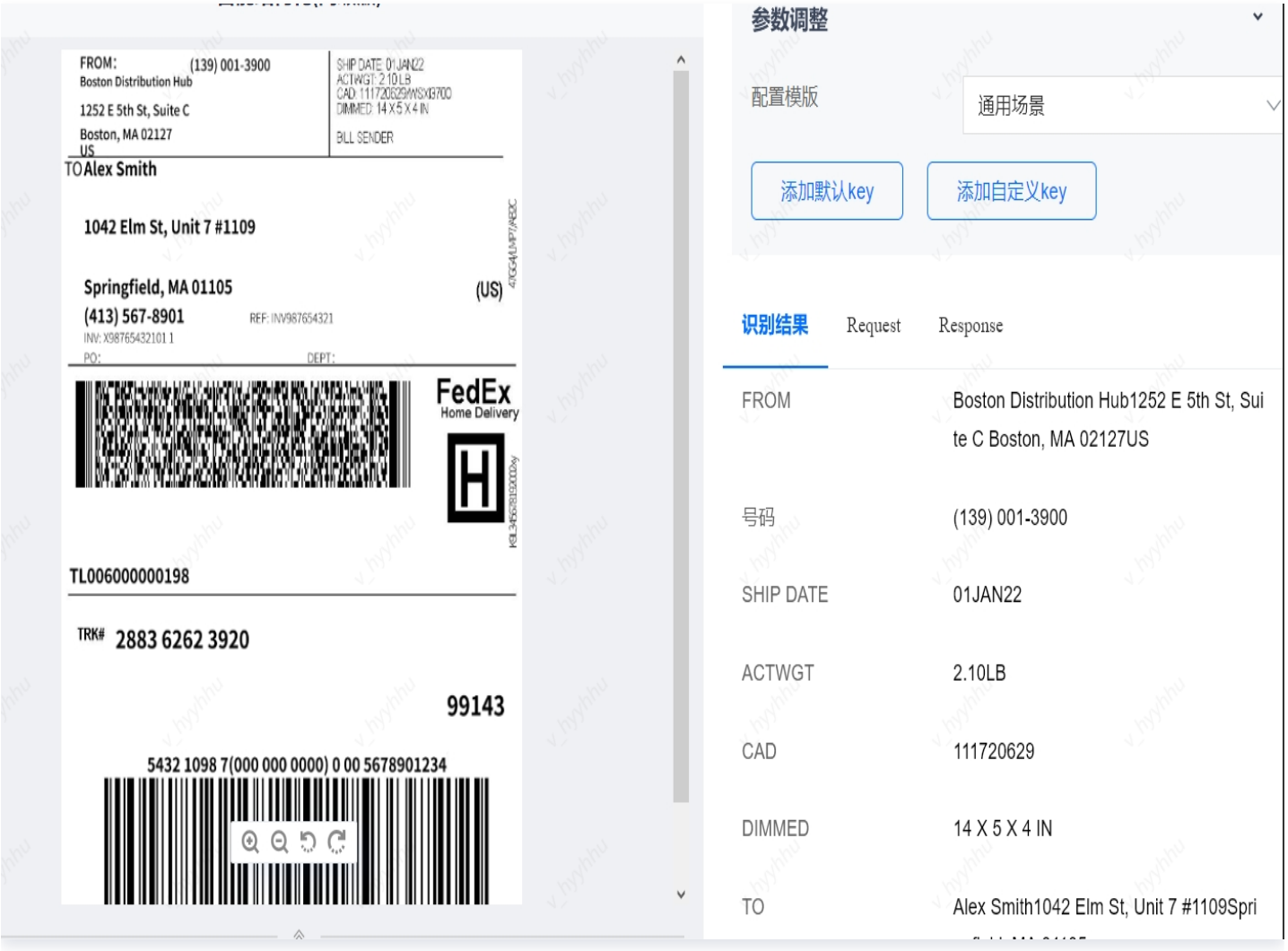


目前 STR 更多是强化端到端的 STR 模型，深度学习之前的 STR 我们暂且不提，最终效果期待通过一个模型同时完成文本行检测和文本识别的任务，这样可以快速实现一张图片的 POI 提取，但实际工作上面并没有这样轻松，二者之前还是存在很多策略去更精细化处理不同场景拍摄下的图片，End-to-End 框架的实现需要大量高质量的文本行及其识别结果的标注数据，而这种标注数据的成本非常高昂。
同时，合成的虚拟数据难以完全替代真实数据的效果。因此，将文本的检测与识别任务分离，分别优化两个独立的模型成为更高效的选择。

- **文本行检测模块**：负责定位文字区域，并生成文字的掩模，以纠正竖直、畸变、弯曲等失真问题。
- **序列识别模块**：对检测到的文字区域进行识别，提取完整的文字信息。然而，对于艺术字体或特殊排列的文本，这种模块的表现可能存在不足。
- **单字检测识别模块**：在序列识别效果欠佳的场景中，通过单字级别的检测与识别进行补充，提升整体识别精度和适应性。

通过这样的分模块设计，可以更好地应对复杂场景中的多样化文字问题，并优化各模块的性能。





文本行检测

目前主流有两种方法，一种是基于回归，第二种是分割：

1. 基于回归的方法

这类方法通过回归模型直接预测文字区域的位置和形状，适用于复杂场景中的文本行定位。
代表算法：EAST（Efficient and Accurate Scene Text Detector）

- **核心思路**：EAST直接回归每个像素点的文字区域边界框，省去了复杂的后处理步骤。
- 实现步骤：
  - 使用全卷积网络（Fully Convolutional Network, FCN）提取图像特征。
  - 通过像素点的几何属性（如旋转边界框或四边形）进行预测。
  - 后处理时，利用非极大值抑制（NMS）去除冗余检测框。

该扩展过程较复杂，实时性较差，而且拍摄图片很容易受遮挡物影响，所以实验初期就否定了该方法。

2. 基于分割的方法

分割方法将文本行检测视为语义分割问题，通过像素级分类来确定文字区域。
代表算法：PSENet（Progressive Scale Expansion Network）

- **核心思路**：逐步扩展文字的核心区域（text kernels）来获得完整的文本区域。
- 实现步骤：
  - 通过语义分割模型提取多个尺度的文字核。
  - 从小尺度核逐步扩展到大尺度，生成完整的文本区域。
  - 后处理时合并相邻文字核，形成完整的文本行。

同样是扩展过程较复杂，实时性较差，而且牌匾的文字间隔较大，因此也否定了该方法。

3. 基于目标检测的方法

这类方法借鉴目标检测技术（如 YOLO、Faster R-CNN），将文本行定位看作一个目标检测任务。
代表算法：Textboxes/Textboxes++

- **核心思路**：在 SSD（Single Shot MultiBox Detector）的基础上，调整检测框比例以适应长条形的文本区域。
- 实现步骤：
  - 使用卷积网络提取特征。
  - 根据不同尺度生成适配文字形状的检测框（宽高比更长）。
  - 后处理时通过 NMS 合并重叠的检测框。

该方法算是比较理想的方法，速度快，易于集成。Mask R-CNN 是实例分割任务的里程碑 Mask R-CNN 继承了 Faster R-CNN 的目标检测框架，先通过 Region Proposal Network（RPN）生成候选区域，再对每个候选区域精确回归边界框和掩模。这一机制非常适合文本行检测任务中复杂的文字布局（如交错排列、弯曲文字和密集文本行）。在场景文字中，尤其是广告牌或路边牌匾中，重叠的文本区域非常常见。实例分割能够准确区分这些重叠区域，而语义分割模型往往混淆这些区域。

值得一提的是 Transformer 的引入，其强大的全局建模能力也被应用于文本行定位。在一些老旧街区，街道密集地区，还是较难定位到关键的 POI 信息， Transformer 的加入带来了一定的新思路思考。

文字识别

在 POI 和道路数据的自动化生产过程中，文字识别的结果需要满足两方面的业务需求。一方面，要求尽可能完整地识别出文本行的内容；另一方面，算法需要能够区分出识别结果中准确率极高的部分。POI 名称错误会导致用户体验受损，甚至影响地图服务的可靠性。且不同于通常以单字为维度进行评估的文字识别标准，应该更加关注整个文本行的识别结果：

- **文本行识别全对率**：指文字内容和顺序完全正确的文本行在所有文本行中的占比，用于评估在 POI 名称、道路名称等整体文字识别任务中的表现。
- **文本行识别高置信占比**：指识别结果中高置信度（准确率大于99%）文本行的占比，用于衡量算法拆分高准确率部分的能力。

在实际业务中，单字识别错误可能导致整个文本行无法满足使用需求。例如，将“建设银行”错误识别为“建设行”，即便单字识别率较高，也会严重影响文本行的正确率。因此，整体的文本行准确性（全对率）更符合业务场景的评估要求。

因此要达成此目标，单字检测识别和序列识别结果融合是提高文字识别系统整体性能的重要策略，尤其是在场景文字识别任务中，通过融合两种方法的优势，既可以保证高精度，又能处理特殊场景下的复杂问题。

单字检测识别和序列识别各有优缺点：

- **单字检测识别**：逐字进行检测与识别，适合处理弯曲文本、复杂排列和艺术字体等场景，但在长文本行中，单字级别的结果可能缺乏上下文一致性。
- **序列识别**：基于序列建模，能够整体识别文本行内容，具有较强的上下文理解能力，但对复杂场景（如遮挡、字形变形）和局部噪声的鲁棒性较弱。

通过融合这两种方法，可以充分发挥它们的优势：

- **单字检测识别**弥补序列识别在细粒度局部处理上的不足。
- **序列识别**增强了**单字检测识**别在上下文一致性和长文本行处理上的表现。

单字检测识别

单字检测识别的主流算法是基于**深度学习的字符级检测和识别方法**，其中 CRAFT（Character Region Awareness for Text Detection）是最为经典且高效的单字检测算法之一。这种方法能够实现对复杂背景和不

规则排列字符的精确检测，同时为字符级别识别提供高质量的输入数据。CRAFT 通过检测**字符区域**和**字符间链接区域**，将字符级别的检测问题转化为像素级别的分割问题。能够直接检测单字区域，避免文字行检测中的误差传播。

算法特点

- **字符级精度高**：CRAFT 的字符区域检测可以精确到像素级别，适合单字识别任务。
- **适应复杂场景**：对弯曲文本、不规则排列和复杂背景的文字具有较好的鲁棒性。
- **轻量化实现**：通过优化骨干网络（如使用 MobileNet 替代 VGG16），可以实现较高的实时性。

当然还测试了相当一部分算法：

算法	优点	缺点	适用场景
CRAFT	像素级字符检测，高精度，适应复杂背景	后处理较复杂，对密集场景效果略逊	弯曲文字、复杂背景、多语言场景
PAN	高效轻量化，实时性强，适应密集场景	对大字符区域检测效果一般	边缘设备、实时检测场景
DBNet	可微分二值化，高质量分割，边界清晰	后处理依赖二值化分割图	小字符检测、文档扫描场景
TextSnake	弯曲文字适应性强，生成高质量单字检测框	对直线文字场景效率较低	弯曲文字、艺术字体
EAST	检测速度快，框架简单，扩展性强	单字检测效果有限	实时任务、文字行检测优化的单字检测场景

结合当前业务可以选择最优的算法计算。

文本序列识别

文本序列识别是 OCR 任务中核心环节之一，目前最主流、最高效的文本序列识别算法主要基于深度学习的序列建模方法，包括 **Attention 机制**和 **Transformer 结构**的应用。CRNN 经典算法就不提了，让我们聚焦于 Transformer-BERT 算法，Transformer 是目前文本序列识别领域的最新进展，凭借其强大的全局建模能力和并行计算优势，成为复杂场景文本识别的首选方法。将视觉特征转换为初步的文字序列，然后将文字序列输入到 BERT 模型。将 BERT 的语义特征与视觉特征融合，生成更精确的最终识别结果。

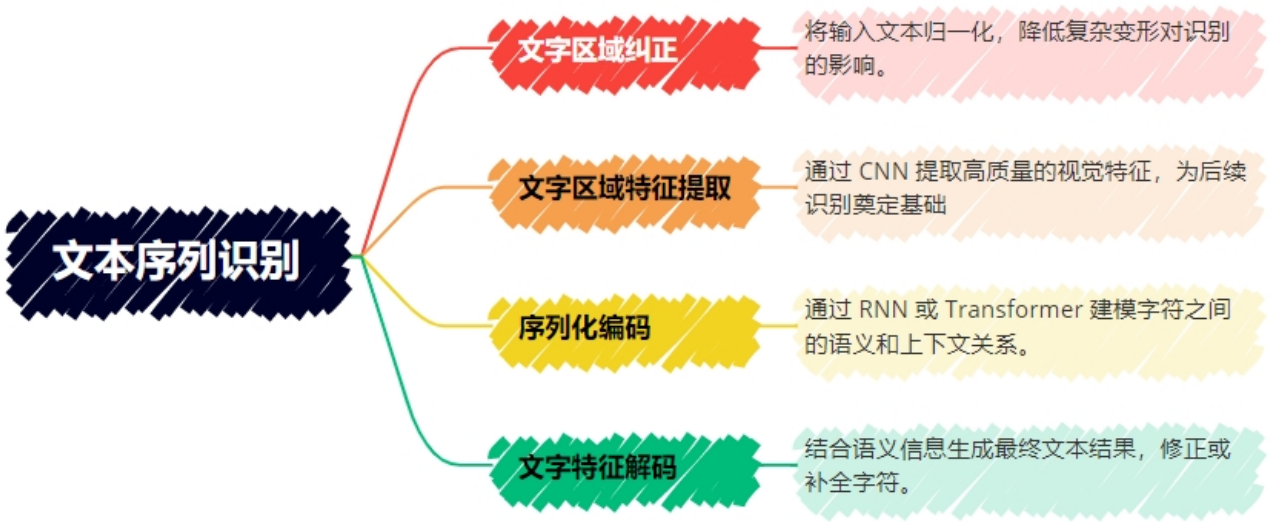
- **视觉特征提取**：使用 CNN、ResNet 或 Vision Transformer 提取文字图像的视觉特征。
- **文本序列建模**：将视觉特征转换为初步的文字序列，然后将文字序列输入到 BERT 模型。
- **融合模块**：将 BERT 的语义特征与视觉特征融合，生成更精确的最终识别结果。

可以说视觉 + BERT 融合模型这套文本识别框架可以算得上是目前最优的策略了。在实际应用中，由于被识别的目标主要以自然场景的短中文本为主，场景文本的几何畸变、扭曲、模糊程度极为严重。

将文本序列识别分解为四个子任务的优势在于各个环节的分工明确，互相配合，形成了端到端的高效识别流程：

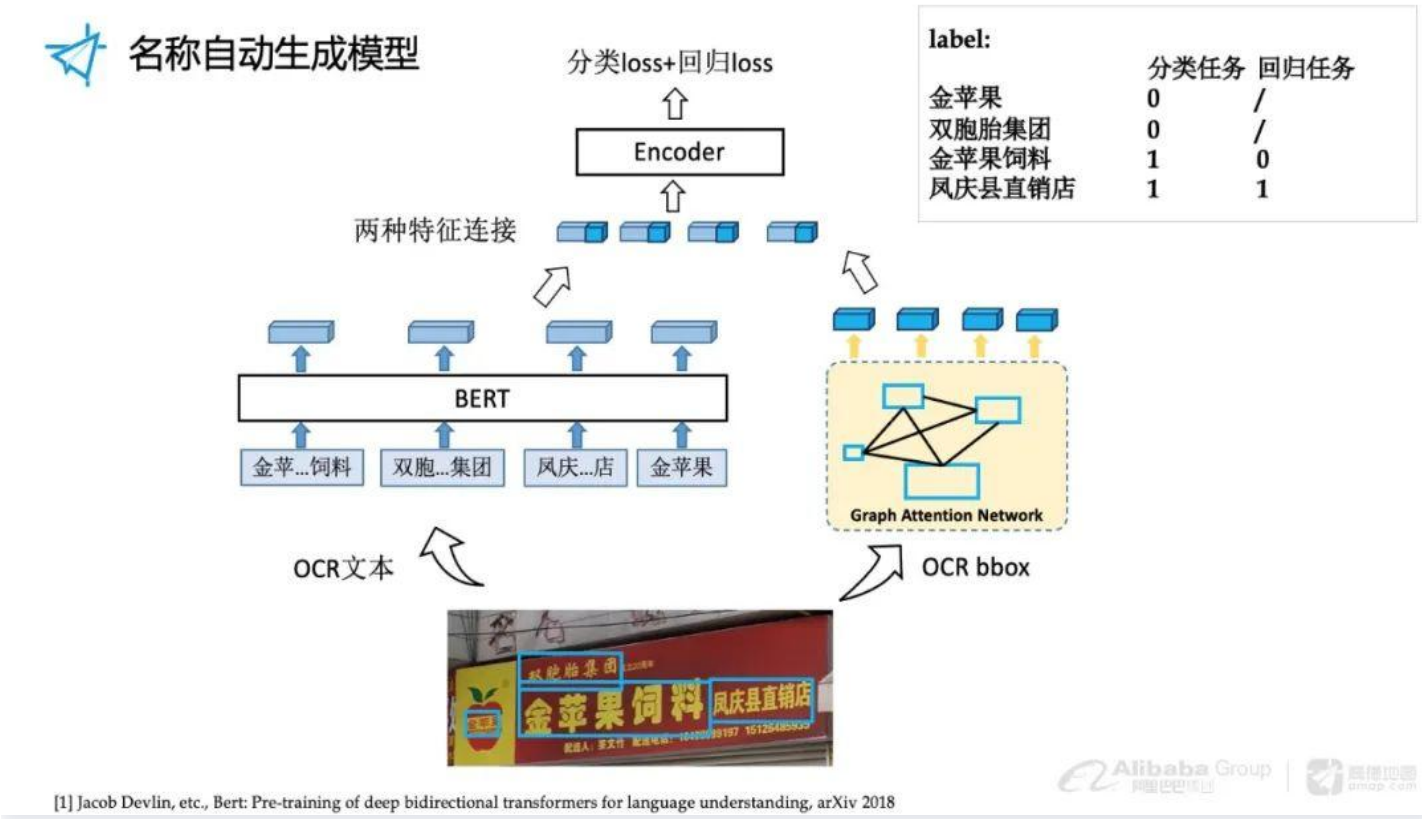
1. **文字区域纠正**：将输入文本归一化，降低复杂变形对识别的影响。
2. **文字区域特征提取**：通过 CNN 提取高质量的视觉特征，为后续识别奠定基础。

3. **序列化编码**：通过 RNN 或 Transformer 建模字符之间的语义和上下文关系。
4. **文字特征解码**：结合语义信息生成最终文本结果，修正或补全字符。这种分解方法尤其适用于**复杂场景文本识别**，如弯曲文本、艺术字体或不规则文本行的识别。通过**视觉特征**与**语义信息**的有效结合，算法在保证识别精度的同时，大大增强了对复杂布局和上下文信息的理解能力。



实际应用的示例

1. **ASTER**
   - 使用 STN 进行文字纠正 > CNN 提取特征 > RNN+Attention 编码与解码 > 输出识别结果。
   - 适合弯曲文本、不规则文本。
2. **SATRN**
   - 使用 Transformer 结构完成编码与解码 > 捕捉全局上下文关系 > 输出文本序列。
   - 适合复杂背景和长文本场景。
3. **TRBA（Transformer with Backbone Attention）**
   - 将视觉特征与语义建模结合，利用 Attention 提升字符级对齐效果。



集大成者：文档智能 OCR 识别

文档智能（Document AI）是指在传统光学字符识别（Optical Character Recognition, OCR）的基础上，结合自然语言处理（NLP）、计算机视觉（CV）、深度学习（DL）以及信息抽取（Information Extraction）等技术，将非结构化或半结构化的文档影像内容智能化地解析、提取并转换为可供机器直接理解和使用的结构化数据格式。相较于传统 OCR 仅能将图片中的文本转换为可编辑字符，文档智能 OCR 不仅关注文字的提取准确度，更强调文本在文档中的逻辑层次、语义关系和字段信息的抽取。

主要特点

- **结构化信息抽取**：不仅仅是识别字符，更能识别字段之间的层次关系。例如在发票、保险表单、合同、身份证件、银行对账单等文档中，能够自动识别“姓名”、“日期”、“金额”等特定字段并将它们提取出来。
- **自然语言理解与上下文关联**：利用自然语言处理技术对识别出的文本内容进行上下文理解。例如在表单中判断字段的类型与意义，或从长文本中检测关键词和相关实体。
- **版面理解（Layout Understanding）**：利用计算机视觉对文档版面结构进行分析与解析，包括段落、表格、列表、标题等特征的检测，从而更好地识别文档的逻辑组织形式。
- **多模态融合**：同时利用图像信息和文本语义信息。例如结合文档图像中的空间布局（如表格的行列分布）和文本行顺序与语义关联来确定数据项之间的关系。
- **领域自适应与定制化**：面向特定领域（如金融、医疗、保险）的文档进行模型微调，使模型在特定格式或特定语言风格下也能获得高精度的结构化提取。

文档智能是 OCR 技术的进化形态，它将图像文字识别从“读懂文字”扩展为“读懂文档”，是企业实现数字化转型、自动化处理业务流程和数据资产智能化管理的关键技术要素。下面我将通过腾讯云文档智能 OCR 进行演示：

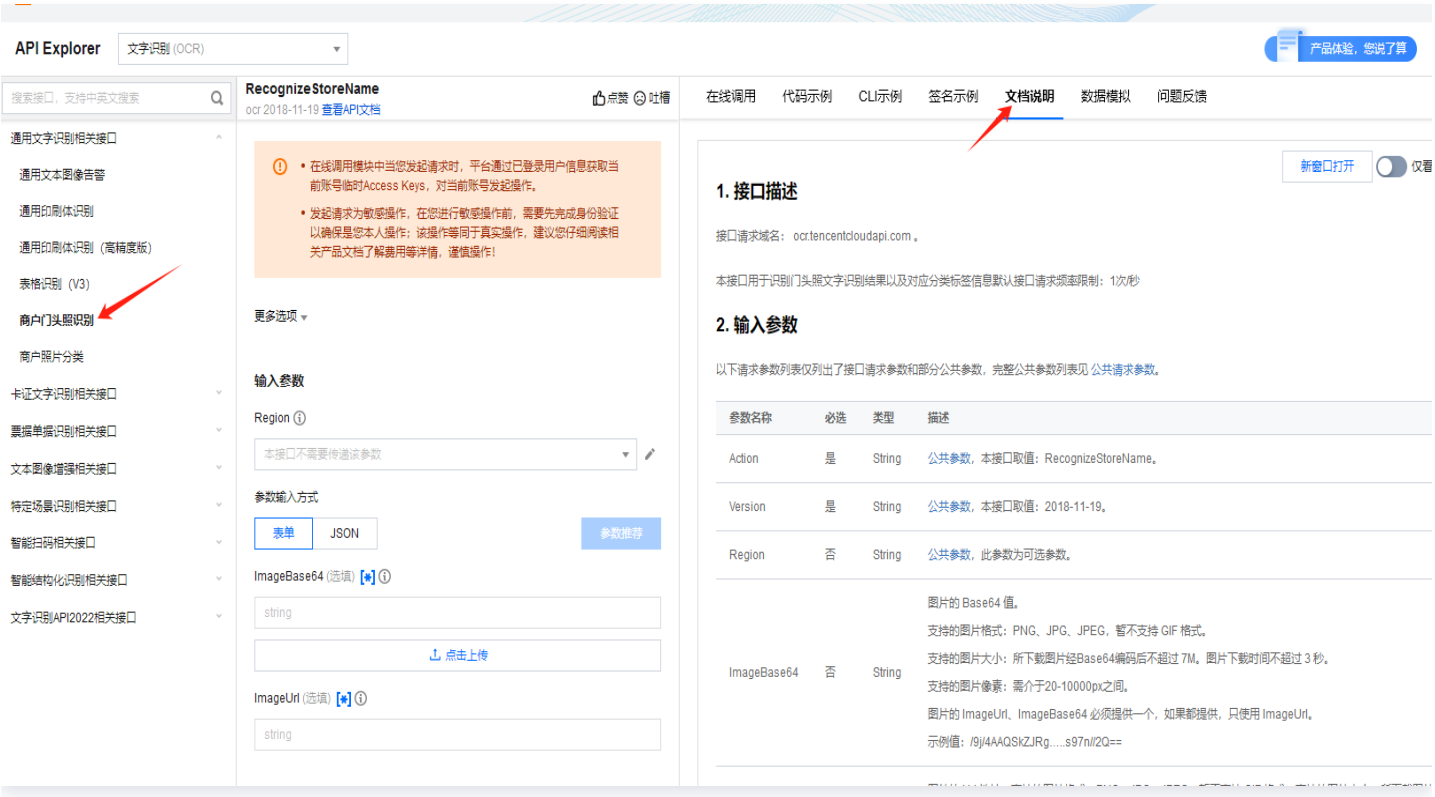
三、操作实战

那么说了这么多实战是抽象乏味，如果此时再去一步一步实践整个积累好几年技术迭代的目标识别项目，那必然会状况百出，因此在这里还是推荐使用腾讯云文档抽取（多模态版）DEMO 感受一下：商户门头照识别 Demo 。
目前市面上成熟的 OCR 智能结构化识别产品还是稀少，因总体设计十分复杂，但腾讯技术在微信图片识别以及多场景 OCR 技术积累十分深厚，倘若我们非一定要自研 OCR 产品，是完全可以利用腾讯云文档抽取（多模态版）产品去完成快速迭代开发的。通过以上 Demo 也可以了解到 API 的快速使用，也有详细的产品文档查阅，可以说十分方便。
本次实验操作依托腾讯云 OCR 第三方接口进行实践操作，先将整体架构完成部署，算法部分完全可以作为一个黑盒进行使用，后续如有自研需求可逐步替代，否则我还是十分推荐用第三方接口开发，节省大量人力物力。
首先我们需要开通服务：文字识别提供通用、卡证、票据、行业文档等多场景下的印刷体、手写体识别，以及智能扫码、卡证票据核验服务。

开通自动发放免费次数1000次，资源包 一年内有效：

API 3.0 Explorer 开发

如果是开发初学者话，有代码编写基础，对 HTTP 请求和 API 调用有一定的了解，可以通过此方式使用文字识别服务。 该方式能够实现在线调用、签名验证、SDK 代码生成和快速检索接口等能力：API Explorer：



针对每个参数都有详细文档解释，此处我再详细解释一遍：

- **Region**：HTTP 请求头：X-TC-Region。地域参数，用来标识希望操作哪个地域的数据。取值参考接口文档中输入参数章节关于公共参数 Region 的说明。**注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。**

- **ImageBase64**：图片的 Base64 值。
  - 支持的图片格式：PNG、JPG、JPEG，暂不支持 GIF 格式。
  - 支持的图片大小：所下载图片经 Base64 编码后不超过 7M。图片下载时间不超过 3 秒。
  - 支持的图片像素：需介于20 - 10000px之间。
  - 图片的 ImageUrl、ImageBase64 必须提供一个，如果都提供，只使用 ImageUrl。
  - 示例值：/9j/4AAQSkZJRg…..s97n//2Q==

为方便大家使用，此处展示图片转换为 Base64 编码的实现：

```
import base64

# 定义图片路径
image_path = 'your_image.jpg'

# 将图片转换为
Base64with open(image_path, 'rb') as image_file:
    # 读取图片内容并转换为Base64编码
    base64_encoded = base64.b64encode(image_file.read()).decode('utf-8')

# 打印Base64字符串
print("Base64 编码结果：")
print(base64_encoded)
```

原始图片：

接口相应结果为：


```
{
    "Response": {
      "Angle": -5.7148051261901855,
      "RequestId": "c3e51da5-80f5-480e-9749-518e2f130e26",
      "StoreInfo": [
      {
        "Name": "商店名称",
        "Rect": {
          "Height": 137,
          "Width": 280,
          "X": 491,
```

```
            "Y": 238
            },
            "Value": "蜜雪冰城"
            },
            {
            "Name": "商店名称",
            "Rect": {
              "Height": 153,
              "Width": 276,
              "X": 1144,
              "Y": 208
              },
              "Value": "蜜雪冰城"
              }
              ],
              "StoreLabel": [
              "标准门头照"
              ]
        }
 }
```


参数名称	类型	描述

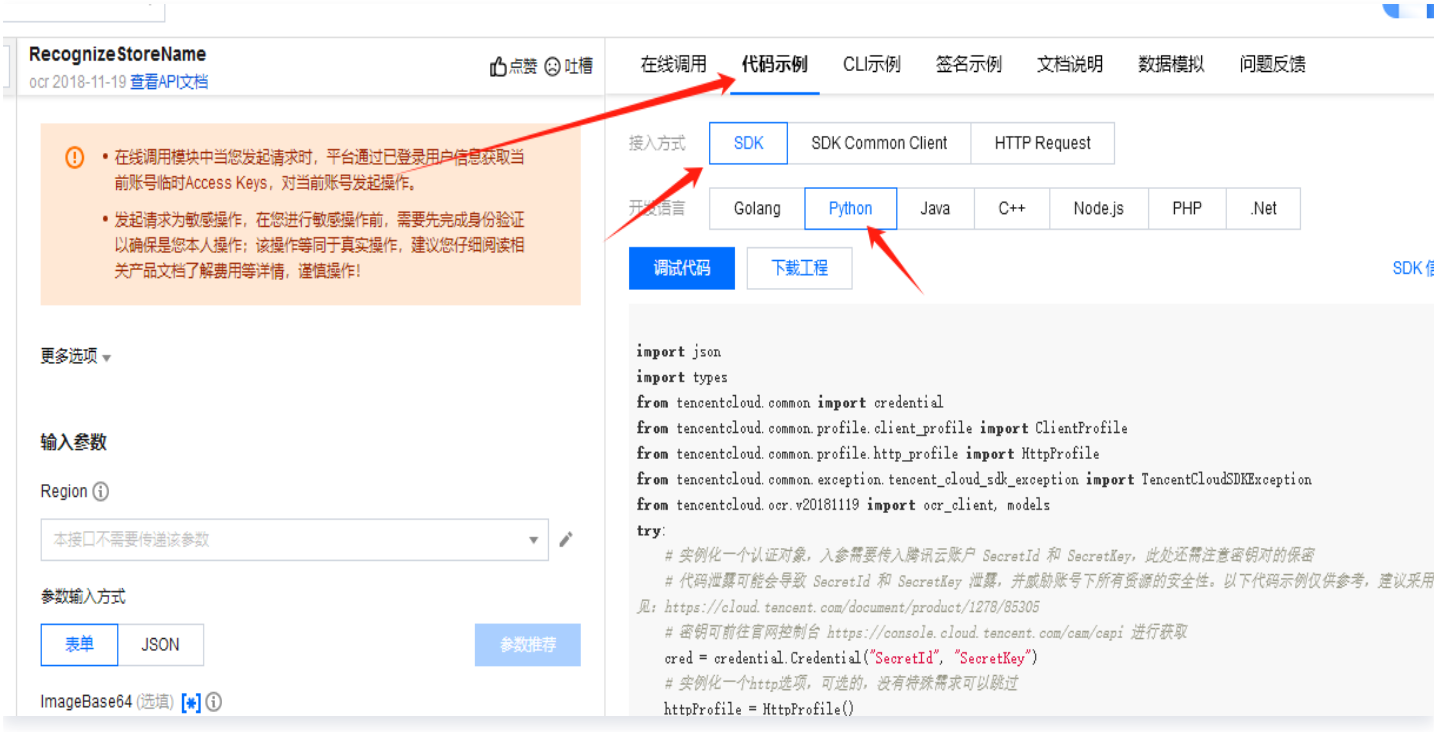
StoreInfo	Array of StoreInfo	门头照名称 示例值：
		{ "Name": "商店名称",
		"Rect": {
		"Height": 263,
		"Width": 1132,
		"X": 232,
		"Y": 366
		},
		"Value": "城市生活超市"
		}

Angle	Float	图片旋转角度（角度制），文本的水平方向为0°，顺时针为正，逆时针为负。 示例值：0.988696813583374
StoreLabel	Array of String	门头照标签 示例值： "标准门头照"。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

PythonSDK

根据 API Explorer 可以快速找到自己想要的 SDK 信息，例如我们用 Python 去集成该 API，首先安装 SDK 包：

```
pip install tencentcloud-sdk-python-ocr -i
https://pypi.tuna.tsinghua.edu.cn/simple
```



SecretId 和 SecretKey 加入到 application.yaml 里面，登录 API 密钥管理 可获取。

```
import json
import types
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from tencentcloud.common.exception.tencent_cloud_sdk_exception import
TencentCloudSDKException
from tencentcloud.ocr.v20181119 import ocr_client, models
try:
    # 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注
意密钥对的保密
    # 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全
性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见：
https://cloud.tencent.com/document/product/1278/85305
```

```
    # 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获
取
    cred = credential.Credential(SecretId, SecretKey)
    # 实例化一个http选项，可选的，没有特殊需求可以跳过
    httpProfile = HttpProfile()
    httpProfile.endpoint = "ocr.tencentcloudapi.com"

    # 实例化一个client选项，可选的，没有特殊需求可以跳过
    clientProfile = ClientProfile()
    clientProfile.httpProfile = httpProfile

    # 实例化要请求产品的client对象,clientProfile是可选的
    client = ocr_client.OcrClient(cred, "", clientProfile)
    # 实例化一个请求对象,每个接口都会对应一个request对象
    req = models.RecognizeStoreNameRequest()
    params = {
      "ImageBase64": base64_encoded
      }
    req.from_json_string(json.dumps(params))

    # 返回的resp是一个RecognizeStoreNameResponse的实例，与请求对象对应
    resp = client.RecognizeStoreName(req)
    # 输出json格式的字符串回包
    print(resp.to_json_string())

except TencentCloudSDKException as err:
    print(err)
```

调用返回结果：

{"StoreInfo": [{"Name": "商店名称", "Value": "蜜雪冰城", "Rect": {"X": 491, "Y": 238, "Width": 280, "Height": 137}}, {"Name": "商店名称", "Value": "蜜雪冰城", "Rect": {"X": 1144, "Y": 208, "Width": 276, "Height": 153}}], "Angle": -5.7148051261901855, "StoreLabel": ["标准门头照"], "RequestId": "12cb615e-7c08-41fe-97fb-68b859388bee"}

开发十分快捷简单。这里再分析大数据集群图片 OCR 处理开发。

Python 项目搭建

下面给出一个参考的 Python 项目工程化搭建示例方案，以满足在所示架构（包括服务层、POI 生产、司机注册与资料审核、车辆与保险信息管理、行程票据与报销管理、客服与投诉处理，以及下层的技术应用层如商户门头照识别、网约车行程单识别、通用票据识别、驾驶证/行驶证识别、机动车登记证书识别等）基础上，使用腾讯云文档抽取（多模态版）API 构建完整系统的需求。该项目会同步上传 GitHub 开源，欢迎一起维护。

项目结构示例

```
my_ocr_system/
├─ README.md
├─ requirements.txt
├─ setup.py
├─ config/
│   ├─ config.py              # 配置文件，如API密钥、模型参数、服务端点等
├─ src/|
├─ main.py                 # 系统主入口，可提供CLI/HTTP接口启动
│   ├─ __init__.py
│   ├─ common/
│   │   ├─ __init__.py
│   │   ├─ utils.py           # 工具函数，如图片base64编码、日志记录工具
│   │   ├─ ocr_client_factory.py # 封装OCR客户端实例化逻辑
│   ├─ services/              # 服务层逻辑（POI生产、司机审核、保险信息、行程报
销、客服投诉处理）
│   │   ├─ __init__.py
│   │   ├─ poi_service.py     # POI生产相关逻辑与对接
│   │   ├─ driver_service.py  # 司机注册与资料审核相关服务逻辑
│   │   ├─ vehicle_service.py # 车辆与保险信息管理逻辑
│   │   ├─ itinerary_service.py # 行程票据与报销管理逻辑
│   │   ├─ customer_service.py # 客服与投诉处理逻辑
│   ├─ applications/          # 技术应用层逻辑（OCR实际应用）
│   │   ├─ __init__.py
│   │   ├─ merchant_photo_recognition.py     # 商户门头照识别
│   │   ├─ ride_order_recognition.py         # 网约车行程单识别
│   │   ├─ general_invoice_recognition.py     # 通用票据识别
│   │   ├─ driver_license_recognition.py      # 驾驶证和行驶证识别
│   │   ├─ vehicle_registration_certificate.py # 机动车登记证书识别
│   │   └─ templates/          # 若使用自定义模板存放识别结构化定义文件
│   ├─ interfaces/            # 对外接口层，如HTTP接口（RESTful API）或 gRPC
接口
│   │   ├─ __init__.py|   │   ├─ api.py              # Flask/FastAPI等Web框架
对外提供API入口
│   │   ├─ router.py          # 路由定义
│   ├─ db/                    # 数据存储与访问层，可选，如使用数据库或缓存
│   │   ├─ __init__.py|   │   ├─ models.py           # 数据模型定义
│   │   ├─ dao.py             # 数据访问对象
│   └─ workflow/              # 若需编排复杂OCR流程的工作流逻辑放在此处
│       ├─ __init__.py
│       ├─ workflows.py
│       ├─ tasks.py
└─ tests/
    ├─ __init__.py
```

```
    ├─ test_merchant_photo_recognition.py
    ├─ test_driver_service.py
    ├─ ...   # 各类测试用例
```

功能模块说明

config/:

- `config.py` 中存放全局配置，如腾讯云 API 的密钥（通过更安全的方式加载，如环境变量或 密钥管理服务）、OCR 端点地址、日志级别、数据库连接信息、模板路径等。

common/:

- `utils.py`：存放通用工具函数，例如图片文件转 base64、日志工具、异常处理辅助函数。
- `ocr_client_factory.py`：负责根据配置和密钥，初始化腾讯云 OCR 的客户端实例（如 OcrClient），统一客户端初始化逻辑。

services/:

该层为业务服务层逻辑，如图示架构中的 POI 生产、司机资料审核、车辆保险信息管理、行程票据报销、客服与投诉处理等，这些逻辑中会调用 applications 层的 OCR 功能进行识别，并将结果处理后写入数据库或进一步操作。

- `poi_service.py`：涉及 POI 点位生产的业务逻辑，可调用商户门头照识别模块获得POI必要信息。
- `driver_service.py`：司机注册、资料审核等逻辑，可调用驾驶证识别模块来验证司机信息。
- `vehicle_service.py`：车辆与保险信息管理逻辑，可调用行驶证识别或保险相关发票识别模块辅助信息录入。
- `itinerary_service.py`：行程票据与报销管理逻辑，可调用通用票据识别或网约车行程单识别模块来提取报销所需的发票、单据信息。
- `customer_service.py`：客服与投诉流程中需要 OCR 识别一些证据类图片或记录时，可调用对应的 OCR 模块协助分类和信息提取。

applications/:

该层为技术应用层，与OCR核心功能直接交互。根据具体文档类型，调用腾讯云 OCR SDK 完成识别，并将结果以结构化数据格式返回给 services 层。

- `merchant_photo_recognition.py`：对商户门头照片进行 OCR 识别，提取店名、地址等信息。
- `ride_order_recognition.py`：网约车行程单识别。
- `general_invoice_recognition.py`：通用票据识别。
- `driver_license_recognition.py`：驾驶证和行驶证 OCR 识别。
- `vehicle_registration_certificate.py`：机动车登记证书识别。

若对这些识别结果需要特定的键值映射或模板定义，可在 templates/ 目录中维护 JSON 或 YAML 格式的模板文件，再在对应模块中加载模板并利用多模态模型做好键值对应。

interfaces/:

提供对外服务的统一接口层（API 层）。

- `api.py`：例如使用FastAPI/Flask框架定义 HTTP 接口，通过 RESTful 方式对外提供识别请求入口。外部系统可以调用这些 HTTP 接口传入图片，返回 JSON 格式的结构化识别结果。
- `router.py`：路由配置，将 URL 与对应的服务逻辑关联。

db/:

如果需要对识别结果或后续业务数据进行持久化存储，可在此层实现数据模型定义与数据访问逻辑（DAO 层）。

- `models.py`：定义 ORM 模型（如 SQLAlchemy）。
- `dao.py`：提供数据的 CRUD 接口函数，供services层调用。

workflow/:

若系统内有更复杂的流程编排（例如先识别门头照，拿到 POI 信息，再去调用其他 OCR 接口补全数据，最终整合为一条 POI 数据记录），可以在此定义工作流与任务调度逻辑。

tests/:

测试用例目录，用于放置单元测试和集成测试代码。

- 针对 `applications/` 和 `services/` 分别编写测试用例，以确保 OCR 识别逻辑和业务逻辑的正确性。

项目代码示例

因该项目会同步上传 GitHub 开源，故在本文中不作过多代码介绍，以 `common/ocr_client_factory.py` 为例：

```python
# common/ocr_client_factory.py
import os
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from config.config import Config

def get_ocr_client():
    secret_id = os.environ.get("TENCENT_SECRET_ID", Config.SECRET_ID)
    secret_key = os.environ.get("TENCENT_SECRET_KEY", Config.SECRET_KEY)
    endpoint = Config.OCR_ENDPOINT

    cred = credential.Credential(secret_id, secret_key)
    http_profile = HttpProfile()
    http_profile.endpoint = endpoint

    client_profile = ClientProfile(httpProfile=http_profile)
    from tencentcloud.ocr.v20181119 import ocr_client
    client = ocr_client.OcrClient(cred, "", client_profile)
    return client
```

`applications/merchant_photo_recognition.py`（商户门头照识别示例）：

```python
# applications/merchant_photo_recognition.py
import json
import base64
from common.ocr_client_factory import get_ocr_client
from tencentcloud.ocr.v20181119 import models
def recognize_merchant_store(photo_path):
    with open(photo_path, "rb") as f:
        image_data = f.read()
    image_base64 = base64.b64encode(image_data).decode()

    client = get_ocr_client()
    req = models.RecognizeStoreNameRequest()
    params = {
        "ImageBase64": image_base64
    }
    req.from_json_string(json.dumps(params))
    resp = client.RecognizeStoreName(req)
    return json.loads(resp.to_json_string())
```

services/poi_service.py (POI 生产逻辑示例):
调用 merchant_photo_recognition.py 获取门头识别信息，再进行后续处理。

```python
# services/poi_service.py
from applications.merchant_photo_recognition import
recognize_merchant_store
def create_poi_from_store_photo(photo_path):
      # 调用应用层的识别函数
      result = recognize_merchant_store(photo_path)
      store_name = result.get("StoreName", "")
      address = result.get("Address", "")
      # 根据需要将store_name, address等信息处理或存入数据库
      # ...
      return {"store_name": store_name, "address": address}
```

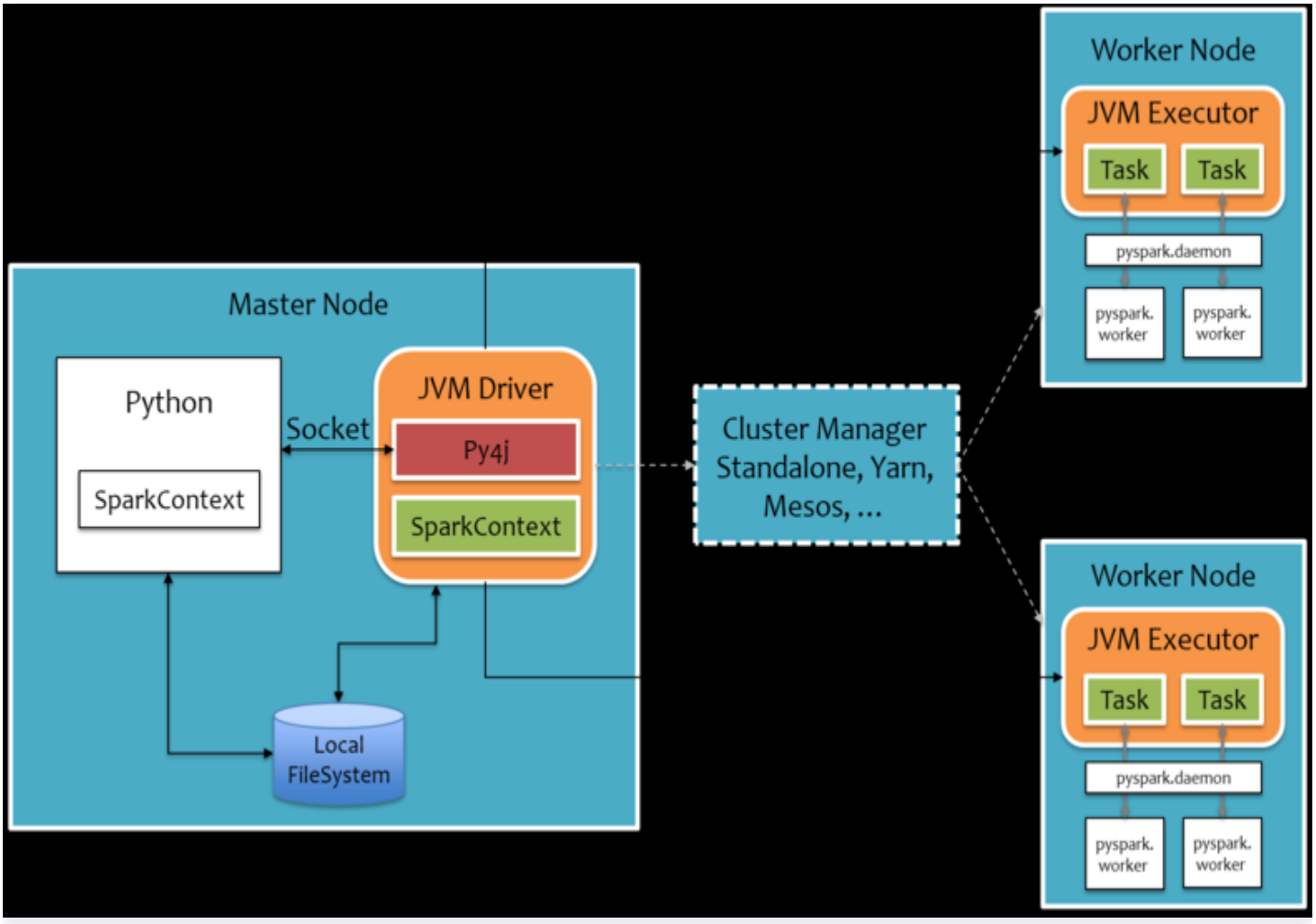
`interfaces/api.py` **（对外接口的简单示例，使用 Flask）：**

```python
# interfaces/api.py
from flask import Flask, request, jsonify
from services.poi_service import create_poi_from_store_photo

app = Flask(__name__)
```

```
@app.route('/recognize/merchant', methods=['POST'])
def recognize_merchant():
        # 接收图片文件上传或URL
        file = request.files['image']
        file_path = "/tmp/upload.jpg"
        file.save(file_path)

        result = create_poi_from_store_photo(file_path)
        return jsonify(result)
if __name__ == '__main__':
        app.run(host='0.0.0.0', port=8000)
```

大数据分布式集群计算

每个公司大数据集群架构不一致，Hadoop 集群推荐使用 PySpark 会更好处理，具体原理一图展示：



主要的计算提交到 Hadoop 分布式执行而不是在 PySpark 客户端节点下载处理，这是正确使用 PySpark 的关键。

```
import pandas as pd
import numpy as np
from pyspark.sql import HiveContext
from pyspark.sql import SparkSession
from pyspark.sql import SQLContext
from pyspark import SparkContext
import json
import types
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from tencentcloud.common.exception.tencent_cloud_sdk_exception import
TencentCloudSDKException
from tencentcloud.ocr.v20181119 import ocr_client, models

def get_sparksession():
      spark = SparkSession.builder \
          .master("yarn")\
          .appName("pyspark_TencentOCR") \
          .enableHiveSupport()\
          .getOrCreate()
      return spark
def TencentOCR(base64_encoded):
      cred = credential.Credential(SecretId, SecretKey)
      # 实例化一个http选项，可选的，没有特殊需求可以跳过
      httpProfile = HttpProfile()
      httpProfile.endpoint = "ocr.tencentcloudapi.com"

      # 实例化一个client选项，可选的，没有特殊需求可以跳过
      clientProfile = ClientProfile()
      clientProfile.httpProfile = httpProfile
      # 实例化要请求产品的client对象,clientProfile是可选的
      client = ocr_client.OcrClient(cred, "", clientProfile)

      # 实例化一个请求对象,每个接口都会对应一个request对象
      req = models.RecognizeStoreNameRequest()
      params = {
          "ImageBase64": base64_encoded
      }
      req.from_json_string(json.dumps(params))

      # 返回的resp是一个RecognizeStoreNameResponse的实例，与请求对象对应
      resp = client.RecognizeStoreName(req)
```

```
        # 输出json格式的字符串回包
        print(resp.to_json_string())

if __name__ == '__main__':
        spark=get_sparksession()
        sql_str="提取图片Base64"
        df_dataframe=get_stable_link(sql_str)
        df_test=TencentOCR(df_dataframe)
        spark.stop()
```

除此之外，开发任务已经就此结束，可以说十分高效方便，结合腾讯云每月的免费额度，如果没有强制技术自研内需是完全可以依赖的。

四、交通 OCR 展望

结合最新大模型技术与 OCR 技术的融合，路况状态信息的准确性会影响到用户在出行过程中的体验。传统的路况状态，主要依靠驾车用户的轨迹信息、用户上传、公共信息等要素通过技术算法处理后提供给用户。但在用户少、驾驶行为异常的道路上，这种方法难以完全保证路况状态信息的准确性。
如果我们能通过视频图像识别的算法，由视频图片观察到的路面状况，包括机动车数量、道路宽度和空旷度等因素来判断道路通行状态（畅通、缓行、拥堵），这将提高道路路况状态判断的准确性，从而提升地图用户的出行体验。
展望未来，随着大模型技术与 OCR 技术的深度融合，交通场景下的智能识别将进一步扩展到**路况状态分析、实时动态数据更新**等应用场景。通过算法提升对道路通行状态的准确判断，地图服务将更加智能化，提升 C 端用户出行体验的精准性和便捷性。

腾讯云 OCR 为何物？又是如何助力各行业实现“结构化”升级？

最近更新时间：2025-05-27 17:59:31

前言：OCR 技术，未来已来

在这个数字化和信息化高速发展的时代，各行各业都在寻求更高效的方式来管理数据。尤其是在文件处理领域，传统的人工方式已经无法满足快速、精确的需求。您有没有想过，电子票据、复杂的发票单据、跨境物流单证，甚至是各种行业的繁琐纸质材料，能不能像数字文本一样，精准读取、提取、分析？答案是：可以！这正是腾讯云文档智能 OCR 技术的魅力所在。
OCR（光学字符识别）技术本身已经发展了几十年，而“文档智能 OCR”则是它的升级版——不仅仅是识别字符，还能精准提取文档中的结构化数据，甚至识别和解析复杂的表格信息。今天，我们将深度探讨腾讯云文档智能 OCR 的强大功能，并分享它在交通、物流、金融、零售等行业中的应用与未来潜力。
腾讯云智能 OCR 的技术优势：高效、精准、智能

概述

腾讯云智能 OCR（Optical Character Recognition，光学字符识别），它是腾讯云提供的一项基于人工智能技术的服务，旨在帮助用户从图像或扫描文档中提取文字信息。通过对图片、扫描文档、手写文本等图像内容的分析，腾讯云智能 OCR 能够识别并提取其中的文字，转化为可编辑、可查询的数据。其 官方文档；想进一步了解的可进其官文进行学习研究。

主要功能

1. 通用文字识别：
   - 支持对图片中包含的文字进行识别，无论是打印体、手写体还是印刷文字。
   - 广泛应用于身份证、银行票据、合同文档、手写笔记等领域。
2. 文档文字识别：
   - 针对扫描文档、PDF文件、图片中的文字进行提取。
   - 适用于各种结构化与非结构化文档，如发票、合同、报表等。
3. **银行卡识别**：提取银行卡上的卡号、有效期、持卡人姓名等信息，广泛应用于金融和支付行业。
4. **身份证识别**：支持识别身份证正面和反面的信息，包括姓名、性别、民族、出生日期、身份证号码等。
5. **车牌识别**：支持对车牌号的精准识别，适用于交通管理、停车场等场景。
6. **票据识别**：识别各种票据和票卡的内容，包括发票、车票、机票、纸质单据等，帮助用户快速获取票据内容。
7. **手写文字识别**：可以识别手写文字，即便是潦草的手写体也有较高的识别准确率，适用于笔记、签名等场景。
8. **多语言支持**：腾讯云智能 OCR 支持多种语言的识别，包括中文、英文、日文、韩文等，适合全球化应用。

技术特点

1. **高识别精度：**腾讯云 OCR 采用深度学习和卷积神经网络（CNN）等前沿AI技术，能够实现高精度的文字识别，尤其在复杂背景和模糊图像中的识别效果较好。
2. **实时识别与处理：**提供高效的 API 接口，能够实现实时的图像文字识别，适用于需要快速处理和反馈的场景。
3. **支持批量处理：**对于大量文档或图像，腾讯云 OCR 支持批量上传和批量识别，能够有效提高工作效率。
4. **简单易用的 API 接口：**提供丰富的 API 接口，用户只需要将图像或文档上传，便能轻松获取识别结果，适合开发者集成到各种应用中。
5. **数据安全：**腾讯云提供企业级数据安全保障，对用户上传的图像和文档进行加密传输和存储，确保数据的隐私和安全。

应用场景

1. **金融行业：**在银行、保险、支付等场景中，OCR 可以帮助识别银行卡、身份证、支票、发票等各种金融文档，提高自动化处理能力。
2. **政府与公共服务：**在身份证识别、车牌识别、税务发票识别等领域，OCR技术能够提高处理效率，减少人工审核时间。
3. **智能办公：**助力文档数字化，扫描文档中的文字内容转换为可编辑的格式，便于存档、查询和处理。
4. **物流与交通：**通过车牌识别技术，可以实现智能停车、电子收费、交通违章检测等应用。
5. **教育与科研：**用于课堂笔记、手写试卷的识别，辅助教师和学生进行信息整理和学习。

优势与亮点

- **精准的文字识别能力：**无论是印刷体、手写体还是不同格式的文档，均能精准识别并转化为数字数据。
- **快速处理能力：**对图像的文字提取速度较快，适合实时应用场景。
- **多领域支持：**支持广泛的文档类型，包括发票、身份证、车牌等，具备跨行业适用性。
- **易于集成：**提供API服务，开发者可以轻松集成到自己的应用中，实现自动化的文字提取。
- **高安全性：**支持对用户数据的加密处理，确保隐私与安全。区别传统 OCR。

文档智能 OCR 与传统 OCR 的区别在于，传统 OCR 仅仅关注将图像转换为可编辑的文字，而文档智能 OCR 则不仅仅识别文字，还能智能分析出文档的结构、格式和关键数据。这意味着，对于一个复杂的发票、银行单据，或者跨境物流单据，文档智能 OCR 可以根据预设规则，提取出诸如金额、日期、收发方、税号等关键信息，而不需要人工干预。

腾讯云文档智能 OCR 具备以下几个显著优势：

1. **高效的文本识别能力：**支持多种语言和不同类型的文档，包括手写、印刷以及扫描件。无论是发票、合同还是银行单据，识别速度都很快，并且可以保证高准确度。
2. **智能数据结构化提取：**通过深度学习算法，OCR 不仅能识别字符，还能精准提取表格数据、栏目信息，甚至可以理解文档的层次结构，自动将信息按字段分类。
3. **跨行业适应性强：**无论是交通运输中的货单，还是零售行业的发票，腾讯云智能 OCR 能够根据行业特定需求进行深度定制，提升应用效果。

其他

1. **产品功能：**
   - 通过 OCR 识别图片中的文本、手写内容、印刷文档等，并进行结构化输出。
   - 广泛适用于不同场景，如文本识别、表格提取、单据识别等。
2. **功能体验：**
   - 提供在线体验，用户可以上传图片体验识别效果。
   - 提供 Demo 和 API 接口的说明文档，方便开发者快速接入使用。
3. **使用要求：**
   - 需注册腾讯云账号并开通 OCR 服务。
   - 支持多语言开发环境：Java、Python、PHP、Node.js、C++等。
   - 提供灵活的输出格式，如 JSON、TXT、Excel 等。

行业应用场景：腾讯云 OCR 技术的跨界魔力

接下来，我们具体看一下腾讯云文档智能 OCR 在几个典型行业中的应用案例，看看它如何改变了我们过去对数据处理的认知。

交通与物流：让文档处理更高效

想象一下，跨境物流运输中的货单和单据通常包含大量的文字和数据，传统处理方式可能需要人工逐一检查、录入。这不仅耗时，而且容易出错。而使用腾讯云智能 OCR 技术后，物流公司可以自动识别运输单据中的关键信息，例如货物种类、重量、发货日期、目的地等信息。

案例分析：

某国际物流公司在使用腾讯云文档智能 OCR 后，发现运输单据的处理速度提高了60%，并且人工错误率下降了75%。尤其在处理跨境运输单证时，OCR 能够准确地提取出不同语言、不同格式的关键信息，大大提高了跨境物流的效率。

金融行业：轻松提取金融数据，优化客户体验

金融行业的文档种类繁多，从银行对账单到保险单据，从贷款申请表到信用卡账单，传统的人工处理方式无法满足快速审批和高效运营的需求。腾讯云智能 OCR 技术通过精准提取金融文档中的关键信息，帮助银行和金融机构提升效率、减少人工成本。

案例分析：

某银行在将其贷款审批流程引入 OCR 技术后，审批速度从原来的平均7个工作日缩短至2个工作日。而且，客户只需上传相关文档，系统便能自动提取出其中的个人信息、贷款金额、期限、利率等数据，大大减少了客户的等待时间。

零售行业：精准识别商品信息，优化库存管理

在零售行业，OCR 技术不仅能识别商品条形码、价格标签，还能自动提取发票上的消费信息，助力商家提升库存管理与财务核算的效率。通过文档智能 OCR，零售商可以自动更新库存信息，跟踪商品的流转，优化商品上架与销售

策略。

案例分析：

某大型零售品牌通过腾讯云OCR技术成功实现了自动化库存更新，原本需要人工逐一核对的过程被自动化系统取代。通过精确的商品数据提取，该品牌不仅减少了库存错误，还提升了供应链管理效率。

更详细解析：腾讯云 OCR 文档的核心功能

腾讯云 OCR 主要有以下能力：

1. **通用 OCR**：快速识别图片上的普通文本。
2. **卡证识别**：识别身份证、驾驶证、银行卡等特定卡证内容。
3. **表格识别**：提取复杂表格中的数据并结构化输出。
4. 票据识别：提取发票、收据中的关键信息（如金额、日期等）。
5. **手写体识别**：识别手写文本，提升人工录入效率。

实现 OCR 的快速接入

以下是一个完整的实践示例，使用腾讯云 OCR API 接入示例，快速实现文本识别。

环境准备

1. 安装腾讯云 SDK：

```
pip install tencentcloud-sdk-python
```

2. 获取腾讯云的 SecretId 和 SecretKey。

代码示例

使用签名方法 v3 的公共参数

签名方法 v3 （有时也称作 TC3-HMAC-SHA256）相比签名方法 v1 （有些文档可能会简称签名方法），更安全，支持更大的请求包，支持 POST JSON 格式，性能有一定提升，推荐使用该签名方法计算签名。完整介绍详见 签名方法 v3。

注意：出于简化的目的，部分接口文档中的示例使用的是签名方法 v1 GET 请求，而不是更安全的签名方法 v3。

使用签名方法 v3 时，公共参数需要统一放到 HTTP Header 请求头部中，如下表所示：

参数名称	类型	必选	描述
Action	String	是	HTTP 请求头：X-TC-Action。操作的接口名称。取值参考接口文档输入参数章节关于公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
Region	String	-	HTTP 请求头：X-TC-Region。地域参数，用来标识希望操作哪个地域的数据。取值参考接口文档中输入参数章节关于公共参数 Region 的说明。**注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。**
Timestamp	Integer	是	HTTP 请求头：X-TC-Timestamp。当前 UNIX 时间戳，可记录发起 API 请求的时间。例如 1529223702。**注意：如果与服务器时间相差超过5分钟，会引起签名过期错误。**
Version	String	是	HTTP 请求头：X-TC-Version。操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
Authorization	String	是	HTTP 标准身份认证头部字段，例如： TC3-HMAC-SHA256 Credential=AKID***/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024 其中， - TC3-HMAC-SHA256：签名方法，目前固定取该值； - Credential：签名凭证，AKID*** 是 SecretId；Date 是 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致；service 为具体产品名，通常为域名前缀。例如，域名 cvm.tencentcloudapi.com 意味着产品名是 cvm。本产品取值为 ocr；tc3_request 为固定字符串； - SignedHeaders：参与签名计算的头部信息，content-type 和 host 为必选头部； - Signature：签名摘要，计算过程详见 文档。
Token	String	否	HTTP 请求头：X-TC-Token。即 安全凭证服务 所颁发的临时安全凭证中的 Token，使用时需要将 SecretId 和 SecretKey 的值替换为临时安全凭证中的 TmpSecretId 和 TmpSecretKey。使用长期密钥时不能设置此 Token 字段。
Language	String	否	HTTP 请求头：X-TC-Language。指定接口返回的语言，仅部分接口支持此参数。取值：zh-CN，en-US。zh-CN 返回中文，en-US 返回英文。

```python
import json
from tencentcloud.common import credential
from tencentcloud.ocr.v20181119 import ocr_client, models
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile

# 用户凭证信息
secret_id = "您的SecretId"
secret_key = "您的SecretKey"

# 配置API请求
cred = credential.Credential(secret_id, secret_key)
httpProfile = HttpProfile()
httpProfile.endpoint = "ocr.tencentcloudapi.com"
clientProfile = ClientProfile()
```

```
clientProfile.httpProfile = httpProfile

# 初始化OCR客户端
client = ocr_client.OcrClient(cred, "ap-guangzhou", clientProfile)

# 上传图片进行识别
req = models.GeneralBasicOCRRequest()
params = {
        "ImageUrl": "https://example.com/sample-image.jpg"
}
req.from_json_string(json.dumps(params))

# 发起请求并解析结果
resp = client.GeneralBasicOCR(req)
result = json.loads(resp.to_json_string())

print("识别结果：")
for text in result["TextDetections"]:
        print(f"文本内容：{text['DetectedText']}")
```

代码解析

如上这段代码我实现了调用腾讯云 OCR（光学字符识别）服务进行图片文字识别的功能，以下是逐行解析，希望能够帮助大家理解：

1. 导入必要的库
   - `json`：用于处理 JSON 数据的内置模块。
   - `tencentcloud.common.credential`：用于存储腾讯云的凭证信息（SecretId 和 SecretKey）。
   - `tencentcloud.ocr.v20181119`：包含 OCR 客户端和相关模型类。
   - `ClientProfile` 和 `HttpProfile`：用于配置客户端的请求参数，包括 API 地址和网络传输相关设置。
2. 用户凭证信息　secret_id = "您的SecretId"；secret_key = "您的SecretKey"。
3. 需要填写腾讯云账号的 SecretId 和 SecretKey，这些信息用于鉴权。
4. 配置 API 请求

```
cred = credential.Credential(secret_id, secret_key)。
```

   - 创建一个凭证对象 cred，用于后续的请求鉴权。

```
httpProfile = HttpProfile()
```

```
httpProfile.endpoint = "ocr.tencentcloudapi.com"
```

- `HttpProfile` 用于设置 API 请求的 URL 端点。这里指定了 OCR 服务的接口地址。

```
clientProfile = ClientProfile()
clientProfile.httpProfile = httpProfile
```

- `ClientProfile` 用于存储客户端配置，将前面创建的 httpProfile 绑定到客户端配置中。
- **初始化 OCR 客户端**

```
  client = ocr_client.OcrClient(cred, "ap-guangzhou",
clientProfile)
```

  - 创建 `OcrClient` 客户端对象，指定地域为 `"ap-guangzhou"`（华南地区），并传入凭证和客户端配置。
- **上传图片进行识别**

```
req = models.GeneralBasicOCRRequest()
params = {
      "ImageUrl": "https://example.com/sample-image.jpg"
      }
req.from_json_string(json.dumps(params))
```

  - `GeneralBasicOCRRequest` 是 OCR 通用接口的请求对象。
  - `params` 包含请求参数，这里通过 `ImageUrl` 提供了需要识别的图片的网络地址。
  - `from_json_string` 方法将 JSON 格式的参数加载到请求对象中。
- **发起请求并解析结果**

```
  resp = client.GeneralBasicOCR(req)
  result = json.loads(resp.to_json_string())。
```

  - `GeneralBasicOCR` 方法将请求发送给腾讯云 OCR 服务，返回结果 resp。
  - `to_json_string` 将响应对象转为 JSON 字符串，然后用 `json.loads` 解析为 Python 字典，方便后续操作。
- **输出识别结果**

```
print("识别结果：")
for text in result["TextDetections"]:
```

```
        print(f"文本内容：{text['DetectedText']}")
```

- 遍历 `result` 字典中的 `"TextDetections"` 列表，每个元素包含识别出的文本信息。
- 输出 `"DetectedText"` 字段，即识别的文字内容。

总结

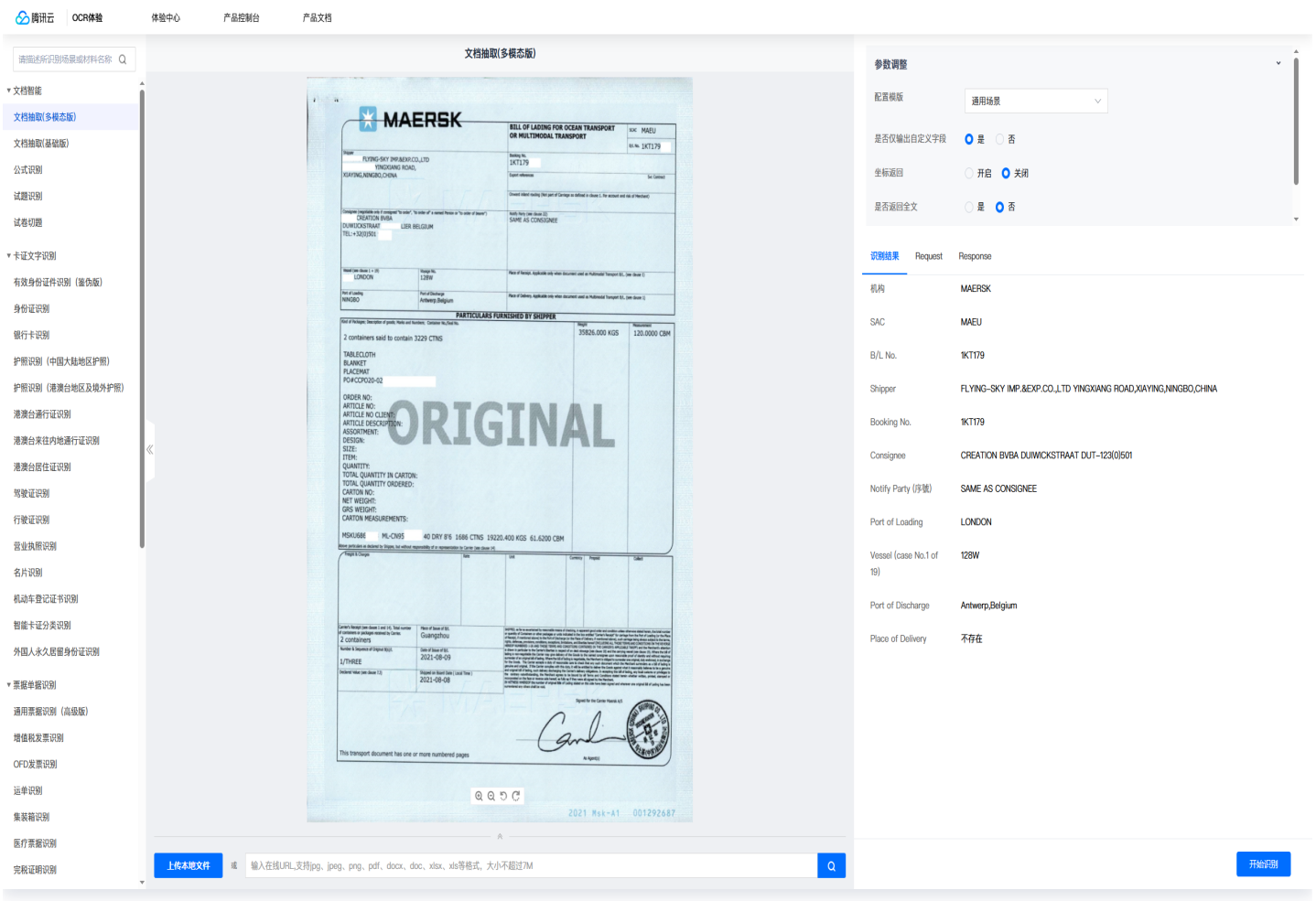
如上这段代码通过腾讯云 OCR 服务实现了图片中的文字识别功能，主要步骤包括：配置鉴权信息、初始化客户端、提交识别请求并解析响应结果。通过 ImageUrl 上传图片进行识别，返回的结果为 JSON 格式，解析后逐行输出识别到的文本。

输出示例

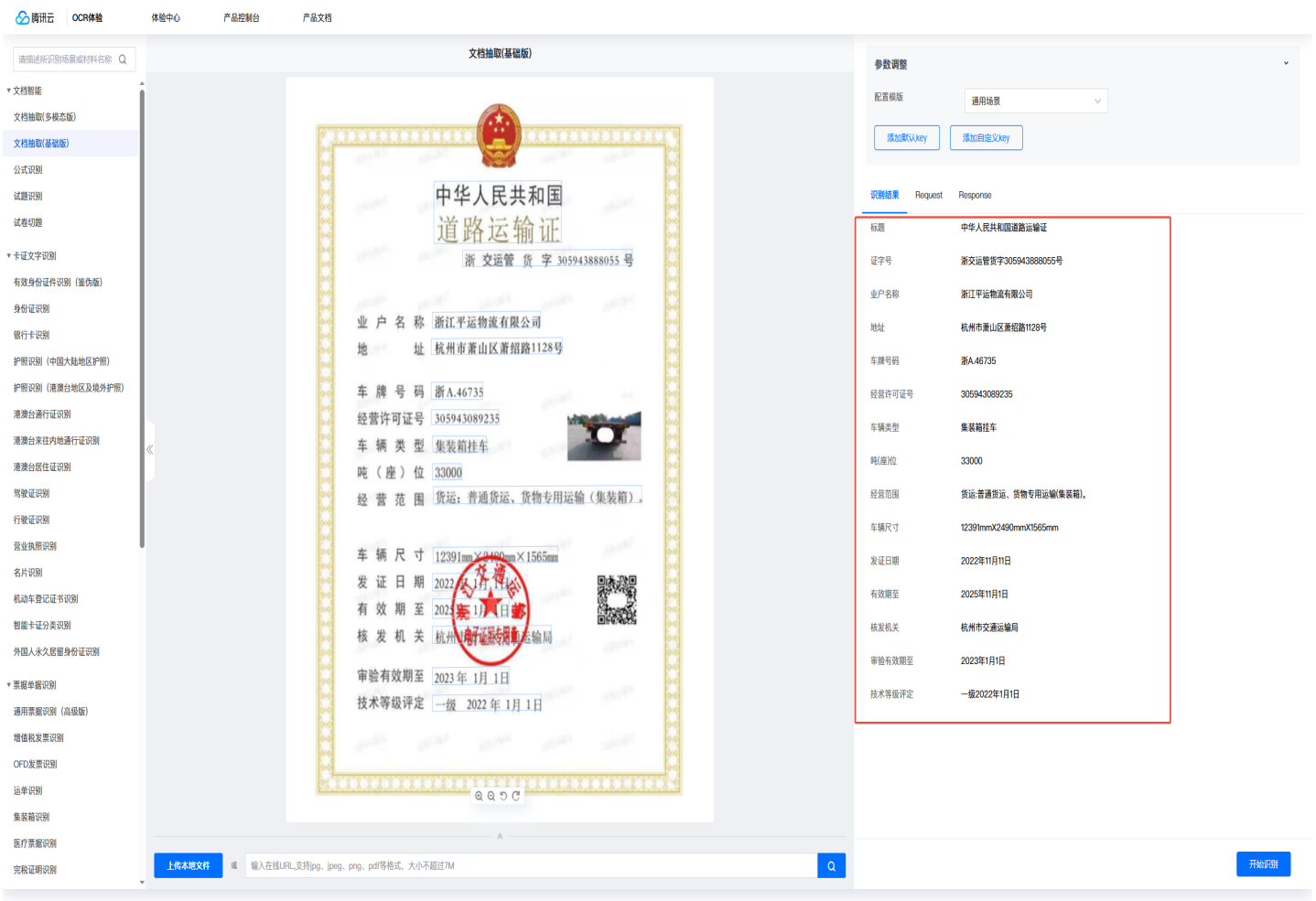
上传一张包含文字的图片，输出结果如下：

```
识别结果：
文本内容：腾讯云智能OCR
文本内容：让文本识别更高效、更智能！
```

同时，您也可以线上体验一下其官方集成好的 Demo，进行测试：
例如：



或者，您又可以体验，或自己上传体验识别。



同时欢迎感兴趣的同学亲身体验下：

- 产品官网/文档：文档智能 OCR 定制模板 OCR 自定义文字识别
- 产品 demo 体验：OCR Demo
- OCR 专项特惠：文字识别特惠活动文字识别购买文字识别选购

总结：未来的OCR，值得期待的“智慧”变革

最后，我想说：腾讯云文档智能 OCR 的强大能力不仅提升了工作效率，还为各行业带来了巨大的潜力。从交通物流到金融、零售行业，它已经不仅仅是一个工具，而是一个“智能助理”，通过自动化和数据结构化的方式，帮助企业解锁更多的商业价值。随着技术的不断发展和应用场景的扩展，未来的 OCR 将会在更多行业中崭露头角，成为数字化转型的重要推动力。

是不是很有趣呢？现在就让我们一起期待，OCR 技术在未来为我们的工作和生活带来更多的“智慧”吧！

点赞并分享您的想法！

如果您有关于 OCR 技术的更多想法，或者您也在某个行业中应用了腾讯云文档智能 OCR，欢迎留言和我分享！一起探讨更多智能科技的未来吧！

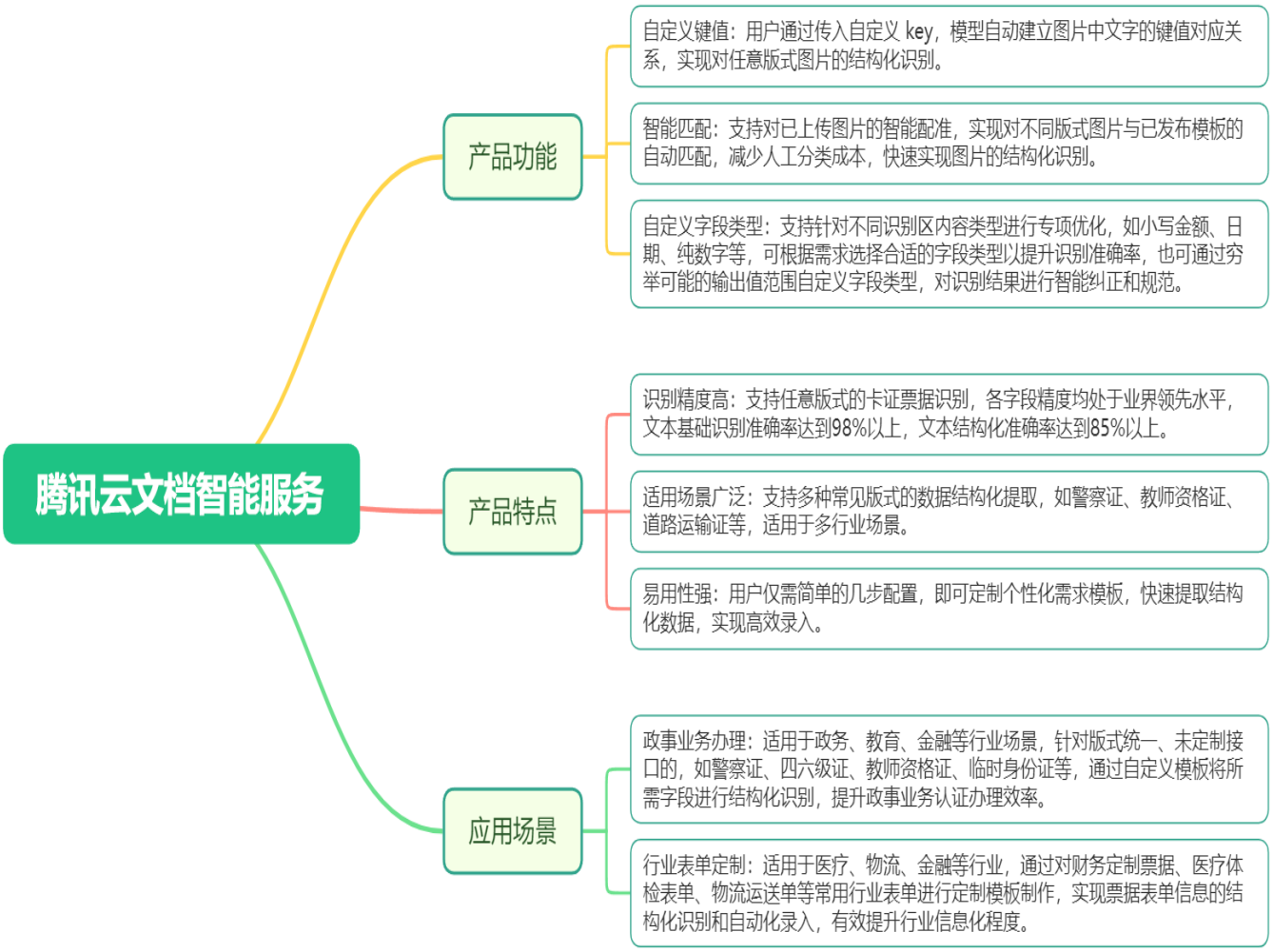
C#实战：使用腾讯云文档智能识别服务轻松提取物流送货单信息，解决仓储物流信息录入的效率问题

最近更新时间：2025-05-27 17:59:31

对于生产制造型企业来说，信息化建设是非常重要的环节，如何高效的解决信息过程中信息高效率的录入对于信息化建设的推进和人工效率的提升非常的有帮助，今天从物流环节给大家介绍一下如何通过腾讯云 文档智能 识别服务轻松提取送货单的信息，通过该功能可以减少物流人员信息的录入也可以大幅度减少人工录入出错后无法排查追溯原因的困境。本文通过 C#+WinForm 的方式给出一个实际的案例来实现，希望对大家能有所帮助！

一、文档智能服务介绍

文档智能（Document AI）融合了业界顶尖的深度学习技术、图像检测技术以及 OCR 大模型能力，实现了对任意版式结构化信息的精准抽取。无论是规范的固定卡证还是繁杂的物流单据，它都能轻松应对，实现智能化识别。此产品通过预学习构建键值对应关系，并支持客户定制模板，从而大幅提高数据提取与录入的效率。文档智能适用于众多场景，包括政务处理、票据核销、行业表单处理以及国际物流管理等。


腾讯云文档智能服务
产品功能
自定义键值：用户通过传入自定义 key，模型自动建立图片中文字的键值对应关系，实现对任意版式图片的结构化识别。
智能匹配：支持对已上传图片的智能配准，实现对不同版式图片与已发布模板的自动匹配，减少人工分类成本，快速实现图片的结构化识别。
自定义字段类型：支持针对不同识别区内容类型进行专项优化，如小写金额、日期、纯数字等，可根据需求选择合适的字段类型以提升识别准确率，也可通过穷举可能的输出值范围自定义字段类型，对识别结果进行智能纠正和规范。
产品特点
识别精度高：支持任意版式的卡证票据识别，各字段精度均处于业界领先水平，文本基础识别准确率达到98%以上，文本结构化准确率达到85%以上。
适用场景广泛：支持多种常见版式的数据结构化提取，如警察证、教师资格证、道路运输证等，适用于多行业场景。
易用性强：用户仅需简单的几步配置，即可定制个性化需求模板，快速提取结构化数据，实现高效录入。
应用场景
政事业务办理：适用于政务、教育、金融等行业场景，针对版式统一、未定制接口的，如警察证、四六级证、教师资格证、临时身份证等，通过自定义模板将所需字段进行结构化识别，提升政事业务认证办理效率。
行业表单定制：适用于医疗、物流、金融等行业，通过对财务定制票据、医疗体检表单、物流运送单等常用行业表单进行定制模板制作，实现票据表单信息的结构化识别和自动化录入，有效提升行业信息化程度。

二、开发完整流程

- 开通智能化服务
- 申请创建开发者密钥
- 创建C#项目编写代码集成 Demo

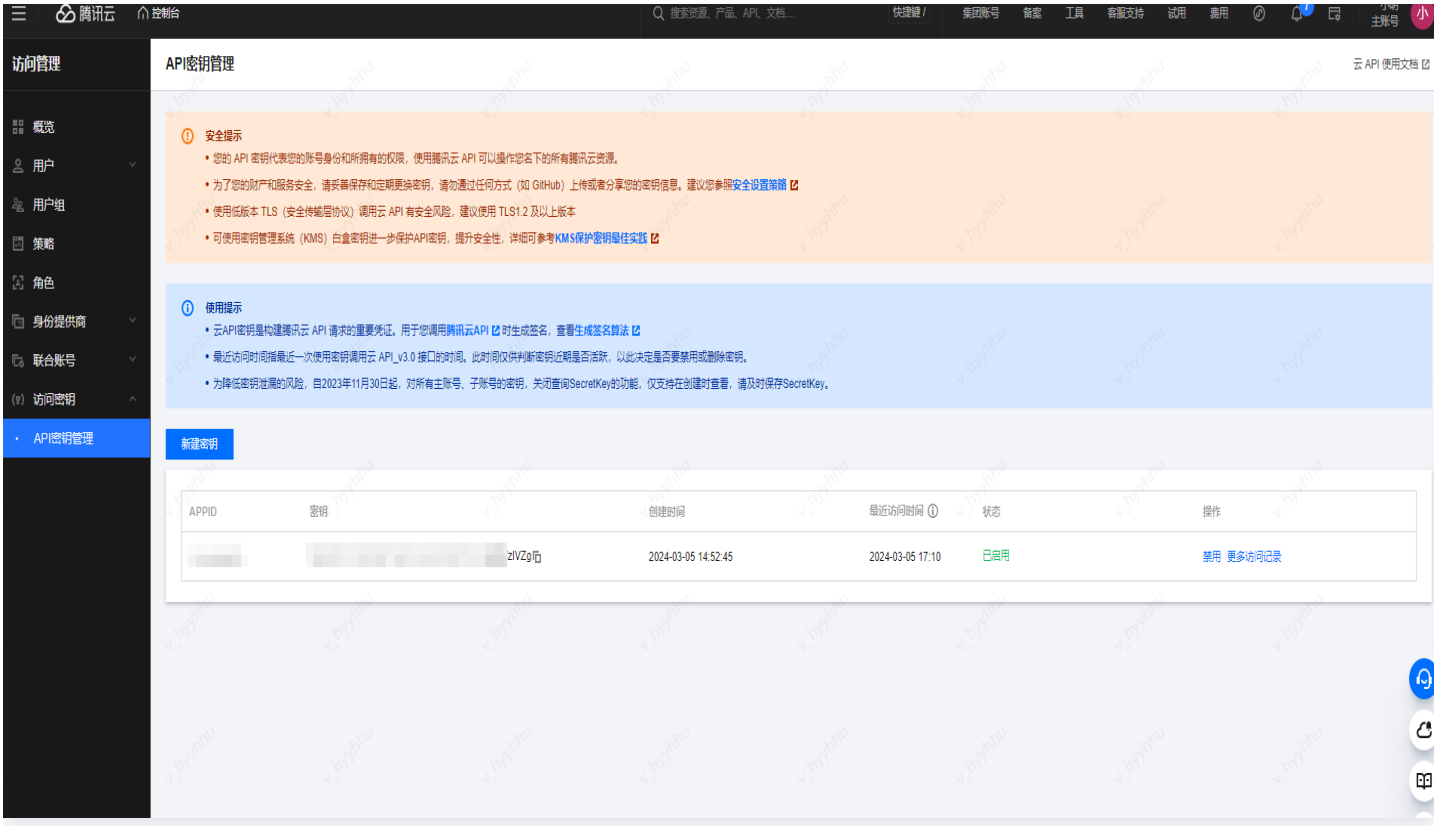
2.1 开通文档智能服务

首先需要使用自己的账户登录腾讯云官网，然后进入 文字识别 服务控制台，开通服务。

开通服务后可以通过控制台查看资源包，开通后默认有1000次的免费额度，方便大家本地开发测试，确认符合功能需求后后再去购买资源包，最后部署到生产环境使用。这个对于企业开发者还是非常有好的。

2.2 创建开发者密钥

因为需要本地调用接口集成开发，需要申请开发者密钥，然后创建开发者密钥，当然如果之前创建过的话可以忽略该步骤。创建成功之后如下：



特点注意：一定要妥善保护后自己的开发密钥，避免泄露，造成重大财产损失。

2.3 创建项目编写代码集成

首先从网络上找到一张送货单单据，具体如下图：

送货单

收货单位：	源悦餐饮连锁有限公司						送货单号：	NO:HT202402040010	
客户地址：	陕西省西安市高新技术产业开发区1000号						送货日期：	2024/2/4	
序号	订单号	客户物料	产品名称	型号/规格	单位	数量	单价	金额	备注
1		ABC001	蓝牙耳机	无线蓝牙连接，降噪功能，充电盒	个	10	45000	450000	
2		JKL004	移动电源	大容量电池，多个USB接口，便携	个	8	55000	440000	
3		PQR006	蓝牙音箱	无线蓝牙连接，高保真音质	个	6	27000	162000	
4		IJK047	智能马桶	加热座圈，冲洗功能，按摩清洁	个	9	47000	423000	
5		LMN048	智能电动滑板车	电动驱动，折叠设计，智能导航	辆	6	43000	258000	
6									
7									
8									
9									
10									
合计：	￥1,733,000.00								
注：	1、本单也用于购销合同，签字生效，如买方对产品质量存在异议时，可在收货三日内书面通知我方，否则视为产品全部合格。								
	2、本单签收后即可作为欠款凭证，货款未付清时，货物所有权依然归属供方所有，买方逾期支付货款的，应承担50%的违约金。								
	3、本货款结款方式为：月结。								
	4、双方已认真阅读以上事项并同意共同遵守，如有争议可协商解决，协商无果时可上诉供方所在的人民法院。								

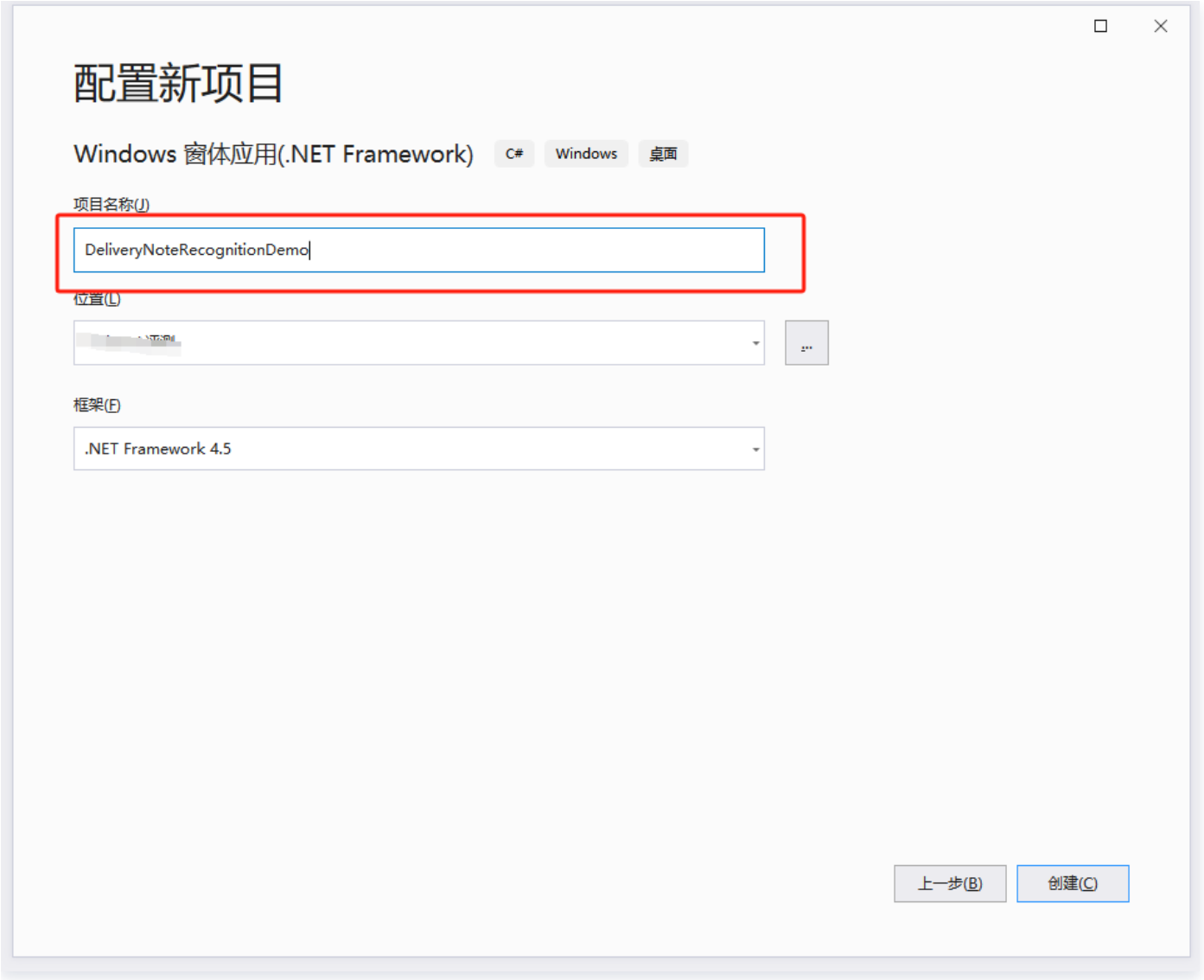
制单：　　业务：　　送货：　　客户签章：

本次实现的是通过该单据识别图片中的：

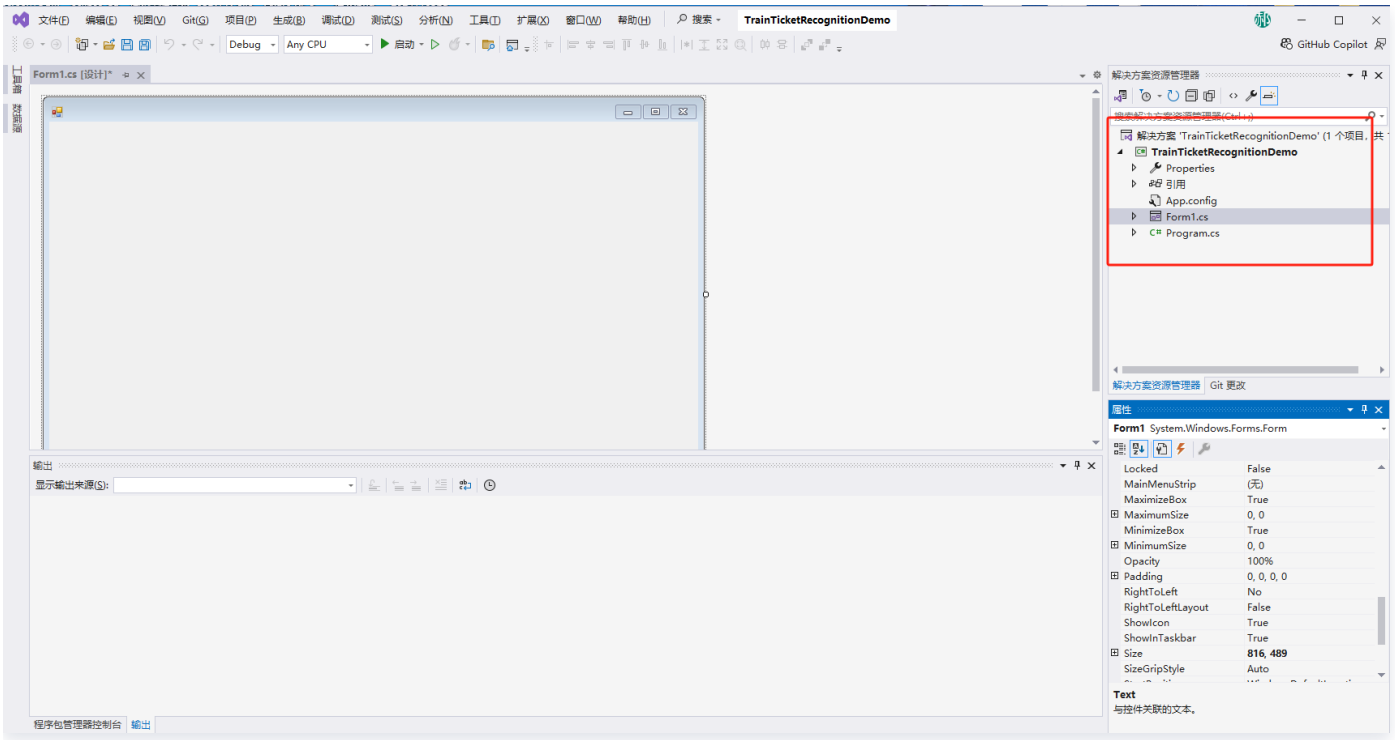
收货单位、送货单号、客户地址、送货日期、合计 五个字段。

首先打开 VS2022 创建一个WinForm项目，

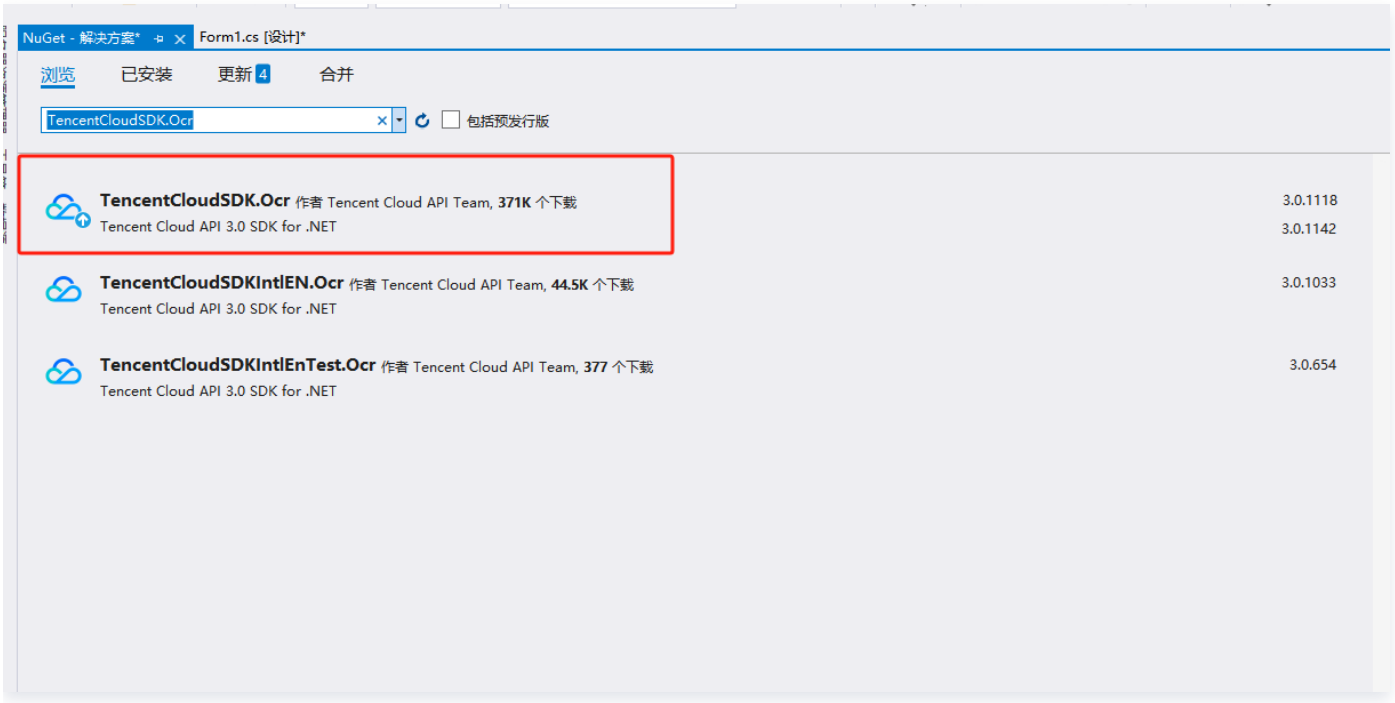
项目名称为 DeliveryNoteRecognitionDemo，具体创建如下图：



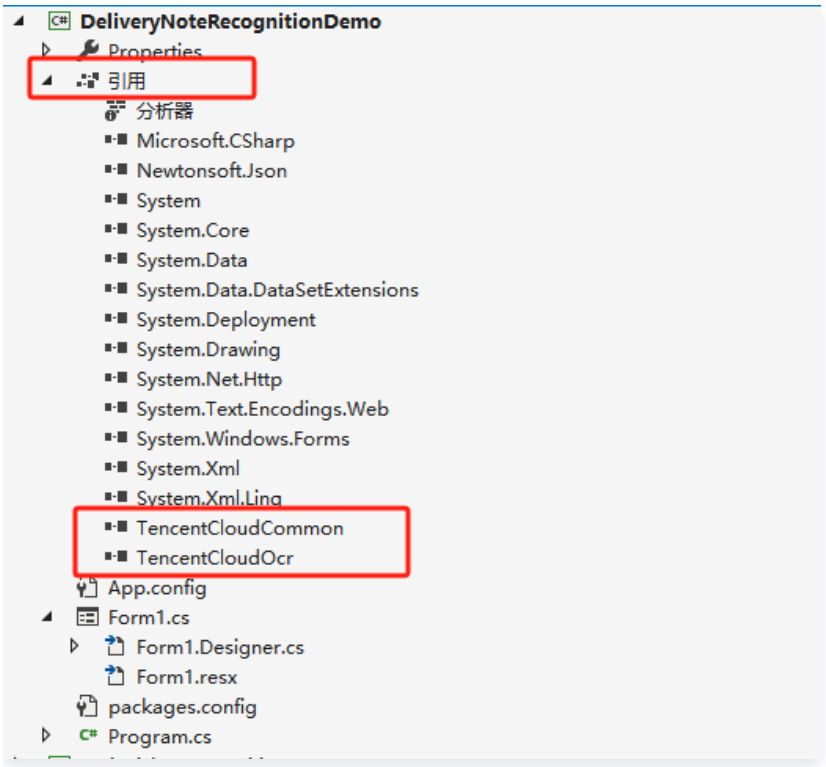
然后点击创建按钮来初始化项目。项目初始化如下：



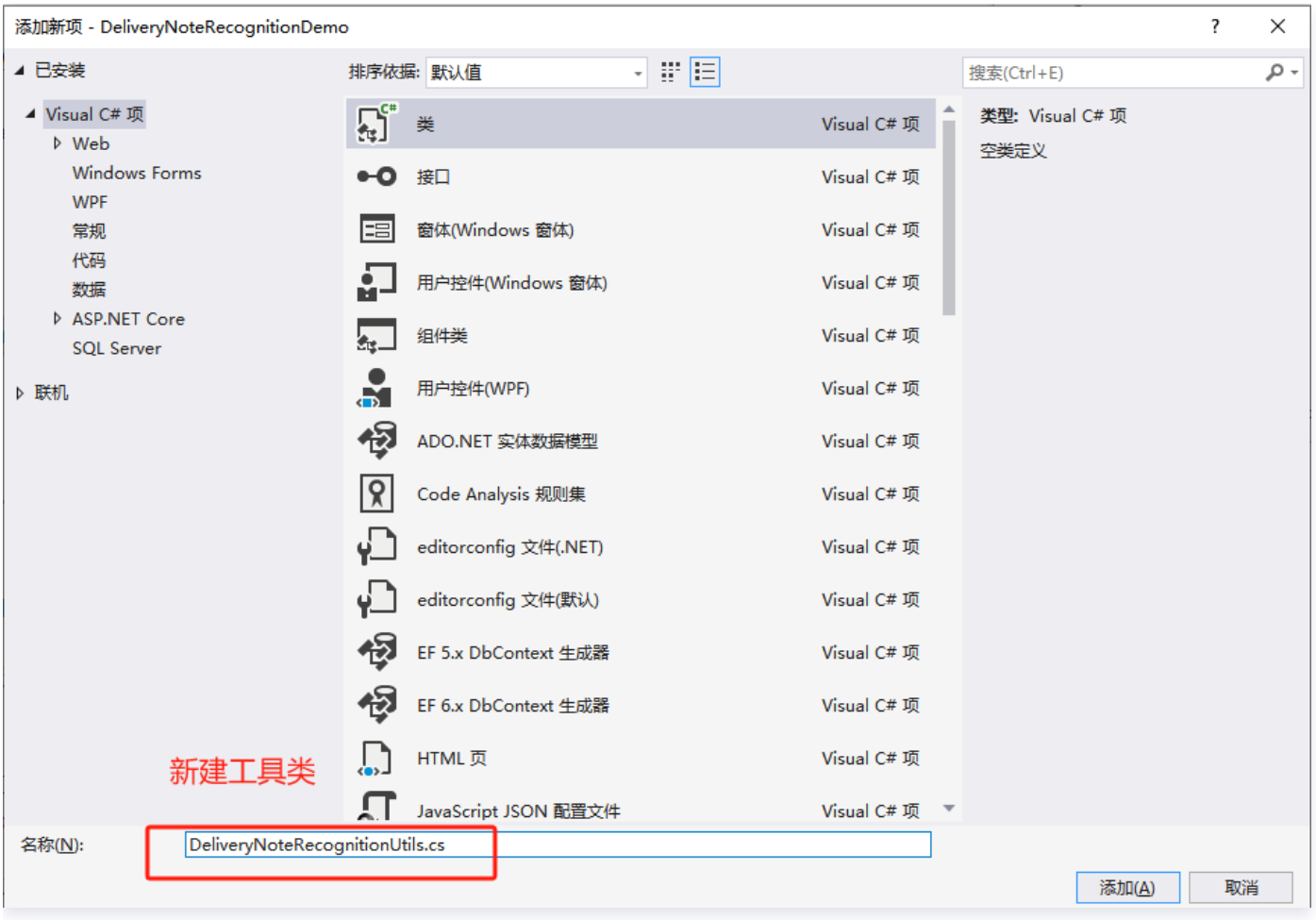
安装腾讯云文字识别的依赖包依赖包搜索 TencentCloudSDK.Ocr。



安装成功后如下：



接着创建一个送货单识别工具类 DeliveryNoteRecognitionUtils.cs。



具体代码如下：

```
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;using System;
using System.Collections.Generic;using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TencentCloud.Common;
using TencentCloud.Common.Profile;
using TencentCloud.Ocr.V20181119;
using TencentCloud.Ocr.V20181119.Models;

namespace DeliveryNoteRecognitionDemo
{
    /// <summary>
    /// 送货单识别工具类
    /// </summary>
    public class DeliveryNoteRecognitionUtils
    {
        /// <summary>
        ///
        /// </summary>
        /// <param name="imageUrl">图片URL</param>
        /// <returns></returns>
        public static DeliveryNoteRecognitionModel Get(string
imageUrl)
        {
            // 注意密钥妥善保存，避免泄露，可以放入配置文件或者数据库中
            Credential cred = new Credential
            {
                SecretId = "",
                SecretKey = ""
            };
            // 实例化一个client选项，可选的，没有特殊需求可以跳过
            ClientProfile clientProfile = new ClientProfile();
            // 实例化一个http选项，可选的，没有特殊需求可以跳过
            HttpProfile httpProfile = new HttpProfile();
            httpProfile.Endpoint = ("ocr.tencentcloudapi.com");
            clientProfile.HttpProfile = httpProfile;

            // 实例化要请求产品的client对象,clientProfile是可选的
            OcrClient client = new OcrClient(cred, "", clientProfile);
            // 实例化一个请求对象,每个接口都会对应一个request对象
            SmartStructuralProRequest req = new
SmartStructuralProRequest();
```

```
                req.ImageUrl = imageUrl;
                req.ItemNames = new string[] { "收货单位", "送货单号", "客户地
址", "送货日期", "合计" };
                DeliveryNoteRecognitionModel deliveryNoteRecognitionModel
= new DeliveryNoteRecognitionModel();
                // 返回的resp是一个SmartStructuralProResponse的实例，与请求对象
对应
                SmartStructuralProResponse resp =
client.SmartStructuralProSync(req);
                JObject jsonObject =
JObject.Parse(AbstractModel.ToJsonString(resp));
                deliveryNoteRecognitionModel.Consignee =
jsonObject["StructuralList"][0]["Groups"][0]["Lines"][0]["Value"]
["AutoContent"].ToString();
                deliveryNoteRecognitionModel.DeliveryNoteNumber =
jsonObject["StructuralList"][1]["Groups"][0]["Lines"][0]["Value"]
["AutoContent"].ToString();
                deliveryNoteRecognitionModel.CustomerAddress =
jsonObject["StructuralList"][2]["Groups"][0]["Lines"][0]["Value"]
["AutoContent"].ToString();
                deliveryNoteRecognitionModel.DeliveryDate =
jsonObject["StructuralList"][3]["Groups"][0]["Lines"][0]["Value"]
["AutoContent"].ToString();
                deliveryNoteRecognitionModel.TotalAmount =
jsonObject["StructuralList"][4]["Groups"][0]["Lines"][0]["Value"]
["AutoContent"].ToString();
                return deliveryNoteRecognitionModel;
                }
        }
}
```

说明：因为 json 结构比较复杂，目前不采用实体的方式获取字段属性，直接根据自定义的字段顺序进行获取，大家注意获取的顺序号。避免出现获取的字段对不上。

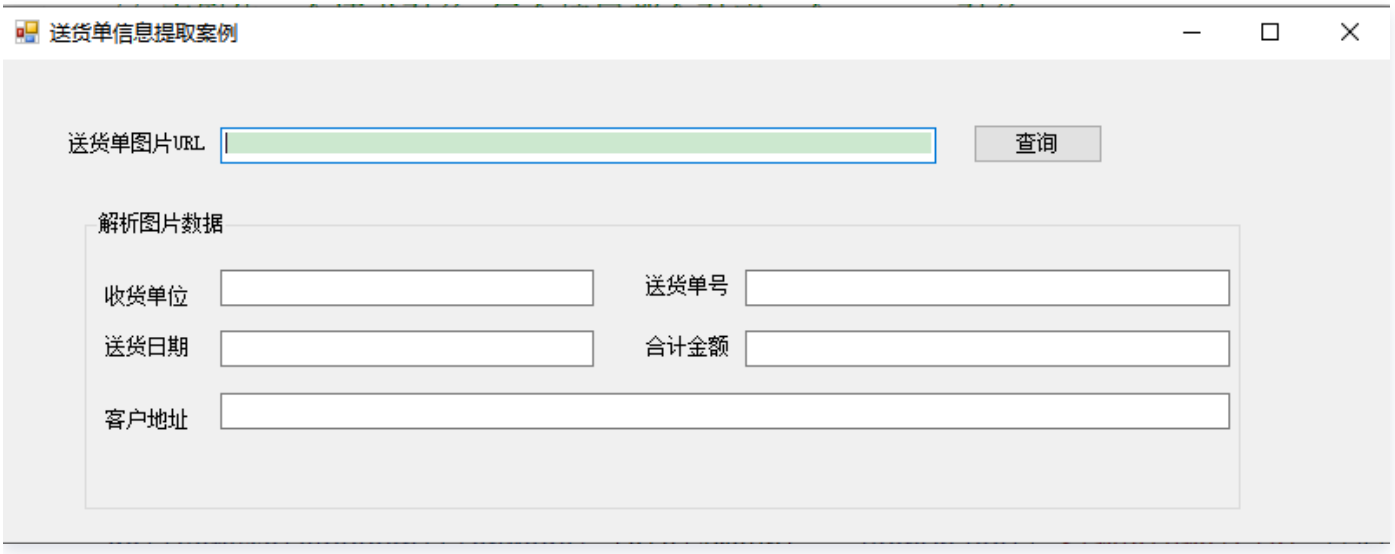
界面设计

这里增加一个图片地址的输入框和查询按钮，另外增加一个分组展示解析结果，具体后台代码如下：

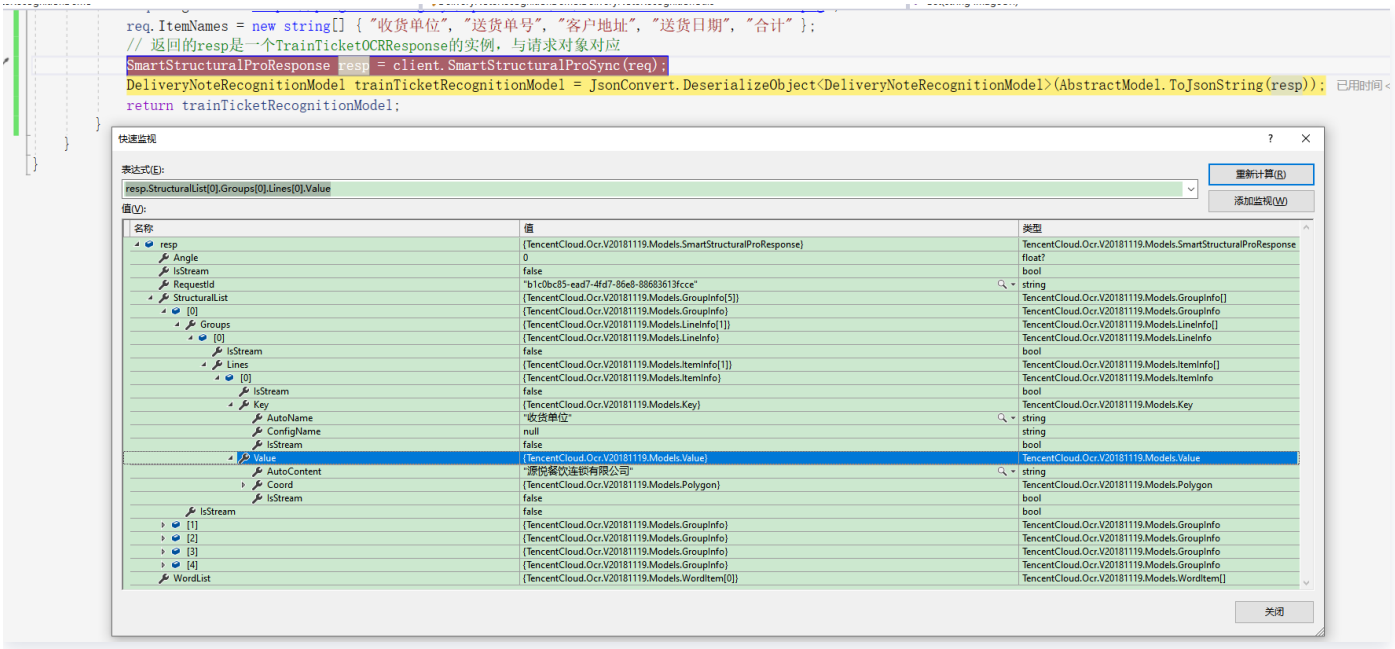
```
private void btnSearch_Click(object sender, EventArgs e)
      {
          tring url = txtImageUrl.Text;
          if (string.IsNullOrWhiteSpace(url))
          {
```

```
                MessageBox.Show("请输入查询送货单图片的URL");
            }
            else
            {
                DeliveryNoteRecognitionModel model =
DeliveryNoteRecognitionUtils.Get(url);
                txtAddress.Text = model.CustomerAddress;
                txtDate.Text = model.DeliveryDate;
                txtAmount.Text = model.TotalAmount;
                txtBillNo.Text = model.DeliveryNoteNumber;
                txtReceiveCompany.Text = model.Consignee;
            }
        }
```

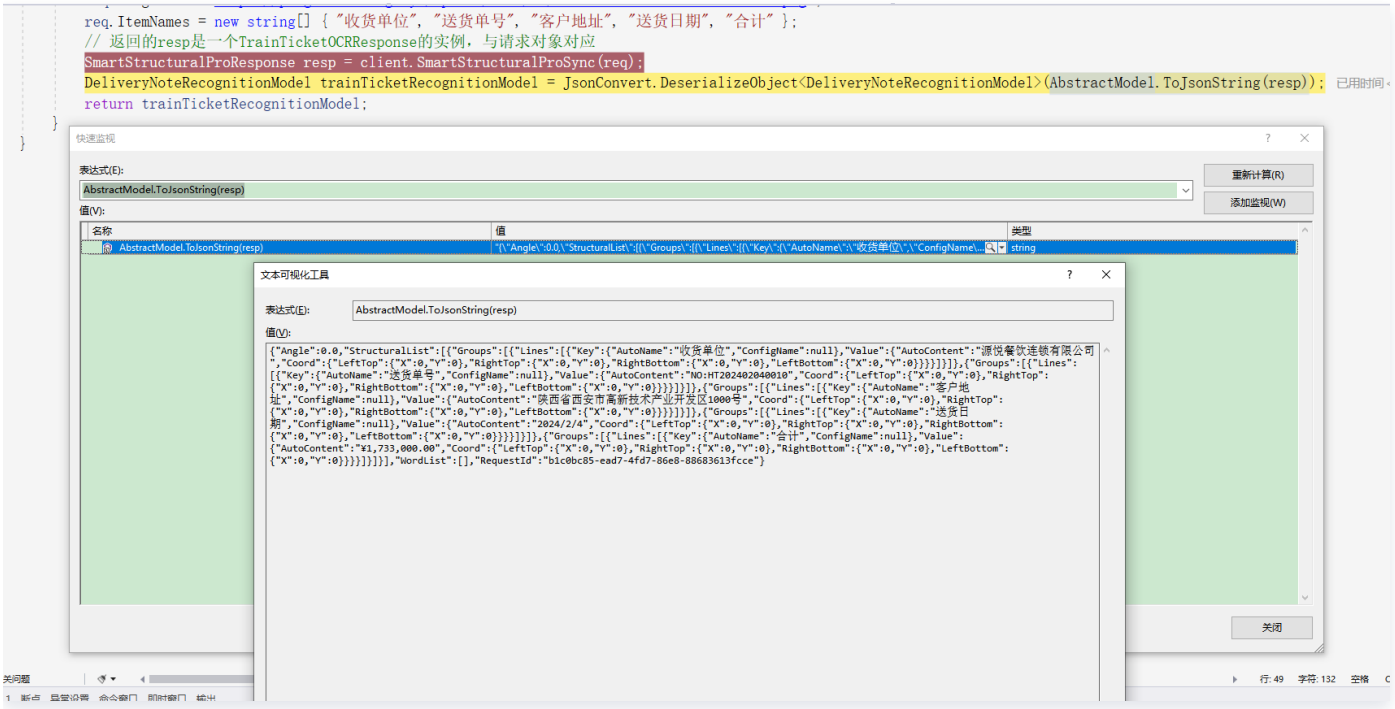
界面效果如下：



调试获取请求的数据如下图：



json 格式的数据：



返回的json数据


```
{
    "Angle": 0.0,
    "StructuralList": [{
        "Groups": [{
            "Lines": [{
                "Key": {
                    "AutoName": "收货单位",
                    "ConfigName": null
```

```
                },
                "Value": {
                    "AutoContent": "源悦餐饮连锁有限公司",
                    "Coord": {
                        "LeftTop": {
                        "X": 0,
                        "Y": 0
                        },
                        "RightTop": {
                            "X": 0,
                            "Y": 0
                        },
                        "RightBottom": {
                            "X": 0,
                            "Y": 0
                        },
                        "LeftBottom": {
                            "X": 0,
                            "Y": 0
                        }
                    }
                }
            }]
        }]
}, {
    "Groups": [{
        "Lines": [{
            "Key": {
                "AutoName": "送货单号",
                "ConfigName": null
                },
                "Value": {
                    "AutoContent": "NO:HT202402040010",
                    "Coord": {
                        "LeftTop": {
                            "X": 0,
                            "Y": 0
                        },
                        "RightTop": {
                            "X": 0,
                            "Y": 0
                        },
                        "RightBottom": {
```

```
                        "X": 0,
                        "Y": 0
                    },
                    "LeftBottom": {
                      "X": 0,
                      "Y": 0
                    }
                }
            }
        }]
    }]
}, {
    "Groups": [{
        "Lines": [{
            "Key": {
                "AutoName": "客户地址",
                    "ConfigName": null
            },
            "Value": {
                "AutoContent": "陕西省西安市高新技术产业开发区1000号",
                "Coord": {
                    "LeftTop": {
                        "X": 0,
                        "Y": 0
                    },
                    "RightTop": {
                        "X": 0,
                        "Y": 0
                    },
                    "RightBottom": {
                        "X": 0,
                        "Y": 0
                    },
                    "LeftBottom": {
                        "X": 0,
                        "Y": 0
                    }
                }
            }
        }]
    }]
}, {
    "Groups": [{
```

```
            "Lines": [{
                "Key": {
                    "AutoName": "送货日期",
                    "ConfigName": null
                },
                "Value": {
                    "AutoContent": "2024/2/4",
                    "Coord": {
                        "LeftTop": {
                            "X": 0,
                            "Y": 0
                        },
                        "RightTop": {
                            "X": 0,
                            "Y": 0
                        },
                        "RightBottom": {
                            "X": 0,
                            "Y": 0
                        },
                        "LeftBottom": {
                            "X": 0,
                            "Y": 0
                        }
                    }
                }
            }]
        }]
    }, {
        "Groups": [{
            "Lines": [{
                "Key": {
                    "AutoName": "合计",
                    "ConfigName": null
                },
                "Value": {
                    "AutoContent": "¥1,733,000.00",
                    "Coord": {
                        "LeftTop": {
                            "X": 0,
                            "Y": 0
                        },
                        "RightTop": {
```

```
                            "X": 0,
                            "Y": 0
                        },
                        "RightBottom": {
                            "X": 0,
                            "Y": 0
                        },
                        "LeftBottom": {
                            "X": 0,
                            "Y": 0
                        }
                    }
                }
            }]
        }]
    }],
    "WordList": [],
    "RequestId": "b1c0bc85-ead7-4fd7-86e8-88683613fcce"
}
```


最终界面结果如下：

三、总结

通过以上案例可以借助腾讯云文档智能服务轻松提取送货单的信息，大家如果使用的时候只需要把该服务器封装为 API 接口就可以轻松和其他系统进行对接集成，非常实用。当然也可以通过该案例去集成更多的场景，例如出货、发货标签的识别、质量检验单的识别、出入库单据的识别、客户发票、应付单等等各种单据的识别都可以做到。通过腾讯云文档智能服务可以大大节省人工录入的准确率和效率。还是非常实用的，感兴趣的朋友可以自己体验一下，大家如果有问题欢迎评论区沟通交流！

基于腾讯云文档智能 OCR 能力的最佳技术实践

最近更新时间：2025-05-27 17:59:31

引言

在信息爆炸的时代，企业面临着海量数据的挑战。如何高效地获取、处理和利用这些数据，成为提升竞争力的关键。光学字符识别（OCR）技术作为数据采集的重要工具，已在多个行业得到广泛应用。腾讯云文档智能OCR 能力凭借其高效、准确、智能的特性，成为众多企业数字化转型的理想选择。本博客将深入探讨基于腾讯云文档智能 OCR 能力的最佳技术实践，涵盖应用背景、解决的问题、接入指引、技术优势，以及实际应用后的效果与收益。通过详细的项目描述和丰富的代码示例，帮助读者全面理解和应用这一技术。
产品官网/文档：文档智能 OCR 定制模板 OCR 自定义文字识别
产品 demo 体验：OCR Demo
OCR 专项特惠：文字识别特惠活动文字识别购买文字识别选购

应用背景

数据处理需求的增长

随着企业业务的不断扩展，数据处理需求呈指数级增长。无论是金融、医疗、物流，还是教育、零售，各行业都面临着海量文档和数据的管理与分析需求。传统的手工数据录入方式不仅效率低下，而且容易出错，难以满足现代企业对数据处理速度和准确性的要求。

非结构化数据的困扰

企业日常运营中产生的大量数据往往以非结构化或半结构化的形式存在，如扫描的纸质文档、照片、PDF 文件等。这些数据难以直接用于后续的分析和决策，制约了数据价值的发挥。如何将这些非结构化数据高效地转化为结构化数据，成为企业亟待解决的问题。

智能 OCR 技术的崛起

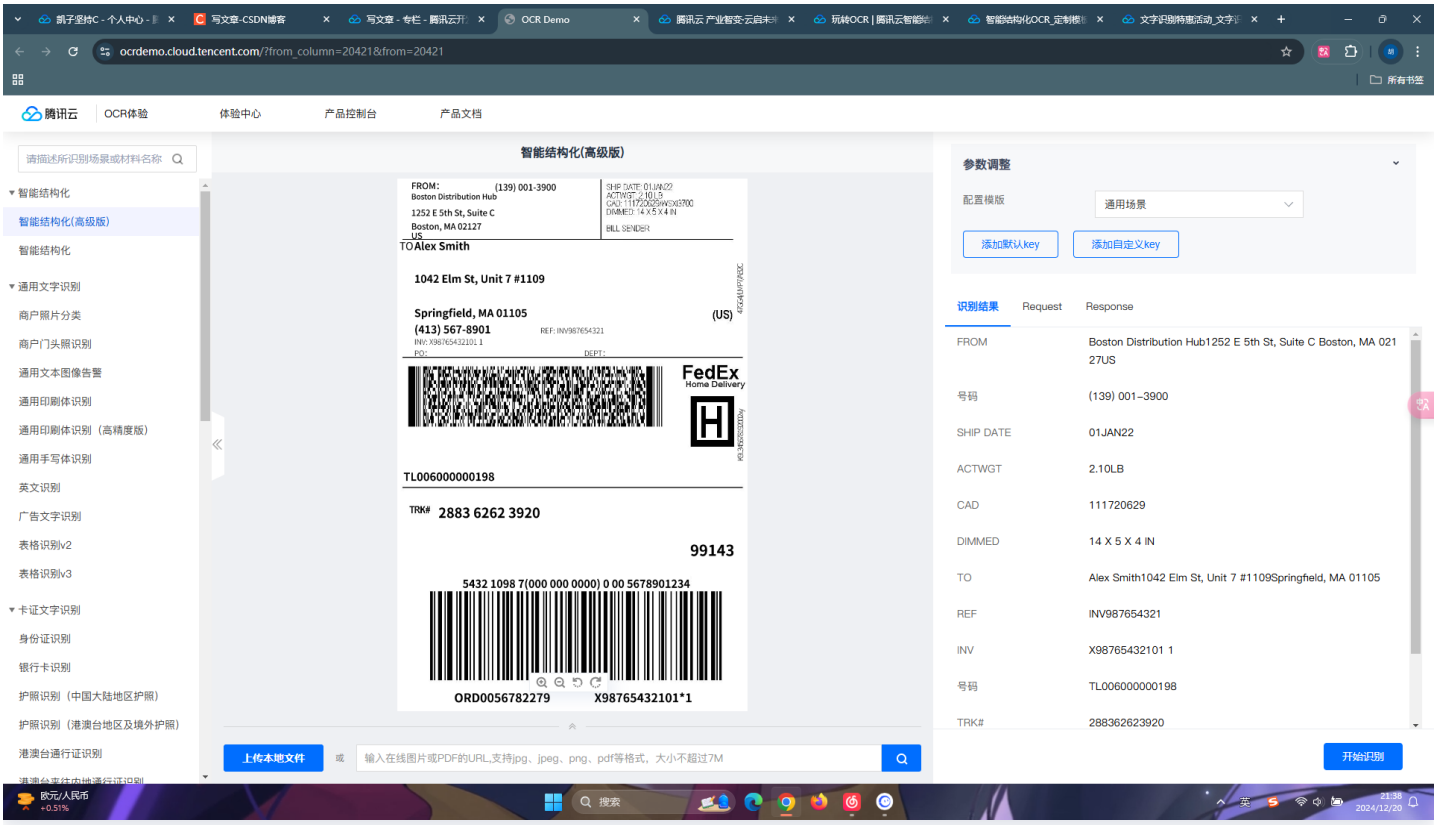
随着人工智能和机器学习技术的快速发展，智能 OCR 技术取得了显著进展。腾讯云文档智能 OCR 能力，基于深度学习和大规模数据训练，提供高准确率、快速响应、多语言支持等优势，成为企业数据采集和处理的重要工具。

解决的问题

引入腾讯云文档智能 OCR 能力，企业能够有效解决以下关键问题：

1. **效率低下**：传统手工数据录入需要大量人力和时间，难以应对快速增长的数据量。智能 OCR 实现自动化处理，大幅提升数据处理速度。
2. **数据准确性差**：手工录入容易出错，影响数据的可靠性和决策的准确性。高准确率的 OCR 技术确保数据的可靠性，减少错误率。
3. **数据处理不便**：非结构化数据难以进行后续的数据分析和利用，限制了数据价值的发挥。文档智能 OCR 能力将非结构化数据转化为可分析的数据，提升数据利用率。

4. **成本高昂**：大量的人力投入导致企业运营成本增加，影响盈利能力。自动化 OCR 解决方案减少对人力资源的依赖，降低运营成本。
5. **扩展性差**：传统数据处理方式难以快速扩展，无法适应业务的快速增长和变化。云端 OCR 服务具备高扩展性，能够灵活应对业务需求变化。



接入指引

接入腾讯云文档智能 OCR 能力，企业可以按照以下步骤进行操作。本文将以 Python 为例，详细介绍 API 集成和 SDK 使用的方法。

注册与配置

注册腾讯云账号

1. 访问 腾讯云官网 进行账号注册。
2. 完成注册后，登录腾讯云控制台。

开通 OCR 服务

1. 在控制台中，导航至“产品与服务” > “人工智能” > “文字智能 OCR”。
2. 根据业务需求选择合适的服务套餐，点击“开通”。
3. 进入“访问管理”页面，创建 API 密钥（SecretId 和 SecretKey），用于后续 API 调用的身份验证。

API 集成

通过 API 接口，开发者可以直接调用腾讯云的 OCR 服务。以下是使用 Python 调用 OCR API 的详细步骤和代码示例。

安装必要的库

首先，确保已安装 `requests` 库，用于发送 HTTP 请求：

```
pip install requests
```

构建请求参数和签名

腾讯云 API 请求需要进行签名验证。以下代码示例展示了如何构建请求参数并生成签名：

```
import json
import base64
import hashlib
import hmacimport time
```

```
import requests

# 配置API密钥
SECRET_ID = 'YOUR_SECRET_ID'
SECRET_KEY = 'YOUR_SECRET_KEY'
ENDPOINT = 'ocr.tencentcloudapi.com'
REGION = 'ap-guangzhou'
ACTION = 'GeneralBasicOCR'
VERSION = '2022-03-23'

ef generate_signature(secret_key, params):
      sorted_params = sorted(params.items())
      canonical_query_string = '&'.join(['{}={}'.format(k, v) for k, v
in sorted_params])
      string_to_sign = 'GETocr.tencentcloudapi.com/?' +
canonical_query_string
      signature = hmac.new(secret_key.encode('utf-8'),
string_to_sign.encode('utf-8'), hashlib.sha1).digest()
      signature = base64.b64encode(signature).decode()
      return signature

def call_ocr_api(image_path):
      with open(image_path, 'rb') as f:
          image_data = base64.b64encode(f.read()).decode('utf-8')
      params = {
          'Action': ACTION,
          'Version': VERSION,
          'Region': REGION,
          'Timestamp': int(time.time()),
          'Nonce': int(time.time()),
          'SecretId': SECRET_ID,
          'ImageBase64': image_data,
          'LanguageType': 'CHN_ENG'
      }

      signature = generate_signature(SECRET_KEY, params)
      params['Signature'] = signature

      response = requests.get(f'https://{ENDPOINT}/', params=params)
      return response.json()

  # 调用OCR API
  result = call_ocr_api('path_to_image.jpg')
```

```
    print(json.dumps(result, indent=2, ensure_ascii=False))
```

解析 API 响应

API 响应通常包含识别结果和相关元数据。以下是解析响应的示例代码：

```
def parse_ocr_response(response):
      if 'Response' in response and 'Error' not in response['Response']:
          items = response['Response']['TextDetections']
          for item in items:
              print(f"Detected Text: {item['DetectedText']}")
              print(f"Confidence: {item['Confidence']}")
              print(f"Bounding Box: {item['Polygon']}")
              print('-' * 30)
          else:
              print("Error:", response['Response']['Error']['Message'])

# 解析并打印识别结果
parse_ocr_response(result)
```

SDK使用

腾讯云提供了多种编程语言的 SDK，简化了 API 调用的复杂性。以下以 Python SDK 为例，展示如何调用 OCR 服务。

安装腾讯云 SDK

首先，安装腾讯云 Python SDK：

```
pip install tencentcloud-sdk-python
```

使用 SDK 调用 OCR

以下是使用腾讯云 Python SDK 调用 OCR 服务的完整示例：

```
from tencentcloud.common import credential
from tencentcloud.ocr.v20181119 import ocr_client, models
import base64
import json

# 配置API密钥
cred = credential.Credential("YOUR_SECRET_ID", "YOUR_SECRET_KEY")
client = ocr_client.OcrClient(cred, "ap-guangzhou")
```

```
# 读取并编码图像with open("path_to_image.jpg", "rb") as image_file:
image_data = base64.b64encode(image_file.read()).decode('utf-8')# 创建请求
对象req = models.GeneralBasicOCRRequest()params = {    "ImageBase64":
image_data,    "LanguageType":
"CHN_ENG"}req.from_json_string(json.dumps(params))# 调用OCR接口resp =
client.GeneralBasicOCR(req)print(resp.to_json_string())
```

解析 SDK 响应

SDK 响应格式与 API 相似，以下是解析响应的示例：

```
def parse_sdk_response(response):
      for item in response.TextDetections:
          print(f"Detected Text: {item.DetectedText}")
          print(f"Confidence: {item.Confidence}")
          print(f"Bounding Box: {item.Polygon}")
          print('-' * 30)
# 解析并打印识别结果
parse_sdk_response(resp)
```

实施步骤

为了确保 OCR 能力的顺利接入和应用，企业应按照以下步骤进行实施：

1. 需求分析
   - 明确数据处理需求，确定需要识别的文档类型和关键信息。
   - 评估现有数据处理流程，识别痛点和改进空间。
2. 环境准备
   - 配置开发环境，安装必要的 SDK 和依赖库。
   - 确保网络环境允许与腾讯云 API 进行通信。
3. API 集成
   - 按照 API 文档进行接口调用和数据处理。
   - 编写测试代码，验证 OCR 功能的正确性和稳定性。
4. 测试与优化
   - 对接入的 OCR 功能进行全面测试，涵盖不同类型和质量的文档。
   - 根据测试结果优化识别参数，提高识别准确率和处理效率。
5. 部署与上线
   - 将集成好的 OCR 功能部署到生产环境。
   - 监控运行状态，确保系统稳定性和可靠性。

6. 持续改进
    - 根据实际使用情况，不断优化识别模型和处理流程。
    - 关注腾讯云 OCR 服务的更新和新功能，及时集成和应用。

技术优势

腾讯云文档智能 OCR 能力具备多项技术优势，使其在市场中具有显著竞争力：

高准确率

腾讯云 OCR 基于深度学习和大规模数据训练，具备出色的文本识别和布局分析能力。无论是印刷体还是手写体，复杂布局还是多语言文本，腾讯云 OCR 都能保持高准确率。

多语言支持

支持中英文、日文、韩文等多种语言的文本识别，满足全球化企业的需求。同时，支持多种字符集和编码格式，确保在不同语言环境下的识别效果。

丰富的文档类型

能够处理多种类型的文档，包括扫描件、照片、PDF 等。无论是票据、合同、病历，还是身份证、驾驶证等证件，腾讯云 OCR 都能高效识别和提取关键信息。

快速响应

高效的处理速度，支持实时数据处理和大规模并发调用。适用于高频次的数据采集场景，如在线表单提交、实时监控数据采集等。

易于集成

提供多种编程语言的 SDK 和详细的 API 文档，降低了技术集成门槛。无论是前端应用、后端服务，还是移动端应用，都能轻松集成 OCR 能力。

数据安全

严格的数据保护措施，确保用户数据的安全和隐私。符合多项国际和国内数据保护法规，适用于对数据安全性要求高的行业，如金融、医疗、政府等。

智能结构化

不仅进行文本识别，还能自动提取和结构化关键信息。通过自定义模板和规则，支持对特定格式文档的智能解析，方便后续的数据分析和利用。

成本效益

灵活的计费模式，根据实际使用量收费，帮助企业控制成本。无需大规模前期投资，按需使用，适合不同规模和需求的企业。

实际应用效果与收益

采用腾讯云文档智能 OCR 能力，企业在实际应用中能够获得显著的效果和收益。以下通过具体案例，详细阐述应用后的实际效果。

案例一：金融行业的贷款审批流程优化

背景

某大型银行的贷款审批流程需要处理大量的申请表和相关文件。传统的手工录入方式不仅耗时长，而且容易出错，影响客户的审批体验和银行的运营效率。

解决方案

银行决定引入腾讯云文档智能 OCR 能力，自动识别和提取贷款申请表中的关键信息，如姓名、身份证号码、收入情况等。通过与内部审批系统集成，实现数据的自动化流转和审批。

实施过程

1. 需求分析
    - 确定需要识别的文档类型（贷款申请表、身份证扫描件等）。
    - 确定需要提取的关键信息字段（姓名、身份证号、收入、贷款金额等）。
2. 环境准备
    - 配置开发环境，安装腾讯云 Python SDK。
    - 获取 API 密钥，配置访问权限。
3. API 集成
    - 编写 Python 脚本，调用腾讯云 OCR API，上传贷款申请表图片并获取识别结果。
    - 解析 OCR 响应，提取关键信息。
4. 系统集成
    - 将 OCR 识别结果与银行内部审批系统对接，实现数据的自动化流转。
    - 开发前端接口，支持在线提交和实时审批。
5. 测试与优化
    - 使用多种格式和质量的申请表进行测试，优化识别参数和处理流程。
    - 根据测试结果调整识别模型和信息提取规则，提高准确率。
6. 部署与上线
    - 将集成好的 OCR 功能部署到生产环境。
    - 监控系统运行状态，确保稳定性和可靠性。

效果

- 审批速度提升：贷款申请处理时间从平均3天缩短至数小时，显著提高了业务响应速度。
- 错误率降低：数据录入错误率降低了90%，提升了数据的准确性和可靠性。

- 客户满意度提高：快速的审批流程显著提升了客户的满意度和信任度，促进了业务增长。
- 运营成本降低：减少了对人力资源的依赖，降低了运营成本，提高了银行的盈利能力。

案例二：医疗行业的病历管理

背景

某大型医院的病历管理需要处理大量的手写或扫描的病历文档。手工录入不仅费时费力，而且难以保证数据的完整性和准确性，影响了医疗服务的质量和效率。

解决方案

医院采用腾讯云文档智能 OCR 能力，将病历文档中的文本信息自动识别和提取，生成结构化的数据记录。通过与医院的信息管理系统集成，实现病历数据的快速存储和检索。

实施过程

1. 需求分析
    - 确定需要识别的病历文档类型（手写病历、扫描病历等）。
    - 确定需要提取的关键信息字段（患者姓名、诊断结果、治疗方案等）。
2. 环境准备
    - 配置开发环境，安装腾讯云 Python SDK。
    - 获取 API 密钥，配置访问权限。
3. API 集成
    - 编写 Python 脚本，调用腾讯云 OCR API，上传病历文档图片并获取识别结果。
    - 解析 OCR 响应，提取关键信息。
4. 系统集成
    - 将 OCR 识别结果与医院信息管理系统对接，实现数据的自动化流转和存储。
    - 开发前端接口，支持医务人员在线查看和管理病历数据。
5. 测试与优化
    - 使用多种格式和质量的病历文档进行测试，优化识别参数和处理流程。
    - 根据测试结果调整识别模型和信息提取规则，提高准确率。
6. 部署与上线
    - 将集成好的 OCR 功能部署到生产环境。
    - 监控系统运行状态，确保稳定性和可靠性。

效果

- 数据处理效率提升：病历录入时间减少了80%，大幅提升了医务人员的工作效率。
- 数据检索便捷：结构化数据便于快速检索和分析，支持临床决策和研究，提升医疗服务质量。

- 医疗服务质量提升：准确的数据记录有助于提升诊断和治疗的准确性，改善患者的医疗体验。
- 运营成本降低：减少了对人力资源的依赖，降低了运营成本，提高了医院的盈利能力。

案例三：物流行业的运输单据处理

背景

某物流公司每天需要处理大量的运输单据，包括发货单、收货单、运单等。传统的手工录入方式不仅耗时长，而且容易出错，影响了物流运营的效率和准确性。

解决方案

物流公司引入腾讯云文档智能 OCR 能力，自动识别和提取运输单据中的关键信息，如发货人、收货人、货物名称、数量、运输日期等。通过与物流管理系统集成，实现数据的自动化流转和管理。

实施过程

1. 需求分析
    - 确定需要识别的运输单据类型（发货单、收货单、运单等）。
    - 确定需要提取的关键信息字段（发货人、收货人、货物名称、数量、运输日期等）。
2. 环境准备
    - 配置开发环境，安装腾讯云 Python SDK。
    - 获取API密钥，配置访问权限。
3. API 集成
    - 编写 Python 脚本，调用腾讯云 OCR API，上传运输单据图片并获取识别结果。
    - 解析 OCR 响应，提取关键信息。
4. 系统集成
    - 将 OCR 识别结果与物流管理系统对接，实现数据的自动化流转和管理。
    - 开发前端接口，支持物流人员在线查看和管理运输单据数据。
5. 测试与优化
    - 使用多种格式和质量的运输单据进行测试，优化识别参数和处理流程。
    - 根据测试结果调整识别模型和信息提取规则，提高准确率。
6. 部署与上线
    - 将集成好的 OCR 功能部署到生产环境。
    - 监控系统运行状态，确保稳定性和可靠性。

效果

- 运营效率提升：运输单据处理时间减少了70%，显著提高了物流运营的效率。
- 错误率降低：数据录入错误率降低了85%，提升了数据的准确性和可靠性。

- 客户满意度提高：快速准确的数据处理提升了客户的满意度和信任度，促进了业务增长。
- 运营成本降低：减少了对人力资源的依赖，降低了运营成本，提高了物流公司的盈利能力。

技术细节与代码实现

为了更好地理解和应用腾讯云文档智能 OCR 能力，以下将详细介绍具体的代码实现和技术细节。

图像预处理

在进行 OCR 识别前，图像的预处理是提升识别准确率的重要环节。以下是常见的图像预处理步骤：

1. 灰度化：将彩色图像转换为灰度图像，减少颜色信息对识别的干扰。
2. 二值化：将灰度图像转换为二值图像，增强文本的对比度。
3. 去噪：去除图像中的噪点和杂色，提高文本的清晰度。
4. 旋转校正：检测并校正图像的倾斜角度，确保文本的水平排列。

以下是使用 OpenCV 进行图像预处理的示例代码：

```
import cv2
import numpy as np

def preprocess_image(image_path):
    # 读取图像
    image = cv2.imread(image_path)

    # 灰度化
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # 二值化
    _, binary = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY_INV)

    # 去噪
    kernel = np.ones((1,1), np.uint8)
    denoised = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel)

    # 旋转校正
    coords = np.column_stack(np.where(denoised > 0))
    angle = cv2.minAreaRect(coords)[-1]
    if angle < -45:
        angle = -(90 + angle)
    else:
        angle = -angle
    (h, w) = denoised.shape
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
```

```
        rotated = cv2.warpAffine(denoised, M, (w, h),
flags=cv2.INTER_CUBIC, borderMode=cv2.BORDER_REPLICATE)
        return rotated

# 示例
preprocessed_image = preprocess_image('path_to_image.jpg')
cv2.imwrite('preprocessed_image.jpg', preprocessed_image)
```

自定义模板与规则

对于特定格式的文档，利用自定义模板和规则，可以进一步提升信息提取的准确率。以下是一个示例，展示如何针对贷款申请表自定义模板提取关键信息。

```
def extract_fields(ocr_result):
        fields = {
            'Name': '',
            'ID Number': '',
            'Income': '',
            'Loan Amount': '',
            'Date': ''
        }

        for item in ocr_result['Response']['TextDetections']:         text
= item['DetectedText']          if '姓名' in text:
fields['Name'] = text.split('姓名')[-1].strip()          elif '身份证号' in
text:              fields['ID Number'] = text.split('身份证号')[-1].strip()
elif '收入' in text:              fields['Income'] = text.split('收入')
[-1].strip()          elif '贷款金额' in text:              fields['Loan
Amount'] = text.split('贷款金额')[-1].strip()          elif '日期' in text:
fields['Date'] = text.split('日期')[-1].strip()          return fields# 示例
fields = extract_fields(result)print(fields)
```

异常处理与错误管理

在实际应用中，可能会遇到各种异常情况，如图像质量差、文字模糊、字段缺失等。为了提高系统的鲁棒性，需实现有效的异常处理和错误管理机制。

```
def safe_call_ocr_api(image_path):
        try:
            response = call_ocr_api(image_path)
            if 'Response' in response and 'Error' not in
response['Response']:
```

```
                return response
            else:
                error_message = response['Response']['Error']['Message']
                print(f"OCR API Error: {error_message}")
                return None

        except Exception as e:
            print(f"Exception occurred: {e}")
            return None

# 示例
response = safe_call_ocr_api('path_to_image.jpg')
if response:
    fields = extract_fields(response)
    print(fields)
else:
    print("OCR processing failed.")
```

批量处理与并发优化

对于大规模数据处理，批量处理和并发调用是提升效率的关键。以下示例展示如何使用多线程进行并发OCR调用。

```
import threading

def process_batch(image_paths):
    results = []
    threads = []

    def worker(image_path):
        response = safe_call_ocr_api(image_path)
        if response:
            fields = extract_fields(response)
            results.append(fields)
    for path in image_paths:
        t = threading.Thread(target=worker, args=(path,))
        threads.append(t)
        t.start()
    for t in threads:
        t.join()
    return results

# 示例
image_paths = ['image1.jpg', 'image2.jpg', 'image3.jpg']
```

```
batch_results = process_batch(image_paths)print(batch_results)
```

日志记录与监控

为了确保系统的稳定性和可维护性，需实现详细的日志记录和监控机制，记录每次OCR调用的结果和异常情况。

```
import logging

# 配置日志
logging.basicConfig(filename='ocr_processing.log', level=logging.INFO,
                    format='%(asctime)s %(levelname)s:%(message)s')
def safe_call_ocr_api_with_logging(image_path):
      try:
          response = call_ocr_api(image_path)
          if 'Response' in response and 'Error' not in
response['Response']:
              logging.info(f"OCR Success: {image_path}")
              return response
          else:
              error_message = response['Response']['Error']['Message']
              logging.error(f"OCR API Error for {image_path}:
{error_message}")
              return None

      except Exception as e:
          logging.error(f"Exception for {image_path}: {e}")
          return
          None

# 示例
response = safe_call_ocr_api_with_logging('path_to_image.jpg')
if response:
      fields = extract_fields(response)
      print(fields)
else:
      print("OCR processing failed.")
```

潜在应用思考

腾讯云文档智能 OCR 能力不仅在金融、医疗和物流行业有广泛的应用前景，在其他领域同样具有巨大的潜力。以下是一些潜在的应用场景和思考：

教育行业

应用场景：自动化处理学生成绩单、入学申请表、考试试卷等文档。
潜在应用：

- 自动录入学生信息，减少手工输入错误。
- 识别考试试卷中的答案，辅助评分和分析。
- 提取课程资料，支持数字化教学和学习资源管理。

零售行业

应用场景：自动化处理发票、收据、商品标签等财务和库存文档。
潜在应用：

- 识别和记录销售数据，优化库存管理和财务分析。
- 自动化处理供应商发票，提升采购和支付效率。
- 通过商品标签识别，实现智能库存盘点和管理。

法律行业

应用场景：自动化处理法律文书、合同、案件材料等文档。
潜在应用：

- 自动识别和提取合同中的关键信息，支持合同管理和审查。
- 数字化案件材料，提升案件管理和查询效率。
- 支持法律文书的智能检索和分析，辅助法律研究和决策。

政府部门

应用场景：优化政务服务，自动化处理各类行政文件和表单。
潜在应用：

- 自动录入和处理居民申请表，提高政务服务效率。
- 数字化存档各类行政文件，提升文件管理和检索能力。
- 支持智能分析和统计，辅助政策制定和决策。

电子商务

应用场景：自动化处理订单、发货单、用户评价等文档和数据。
潜在应用：

- 自动识别和记录订单信息，优化订单管理和物流调度。
- 通过用户评价识别产品反馈，支持产品改进和服务优化。
- 数字化处理发货单，提升仓储和配送效率。

未来发展

随着人工智能技术的不断进步，腾讯云文档智能 OCR 能力将持续优化和扩展，未来的发展方向主要包括以下几个方面：

更高的识别准确率

通过持续的模型训练和优化，进一步提升文本识别和信息提取的准确性，特别是在复杂背景、低对比度和多语言环境下的表现。

多模态识别

结合图像识别、语音识别等多种技术，实现更全面的数据采集和处理。例如，通过结合语音识别，实现语音转文字和图像文字的联合识别，提升数据处理的多样性和灵活性。

个性化定制

根据不同行业和企业的需求，提供定制化的 OCR 解决方案，支持特定格式文档的智能解析和信息提取，满足多样化的业务场景需求。

智能分析与决策支持

在 OCR 的基础上，集成数据分析和决策支持功能，帮助企业从数据中获取更多的商业价值。例如，通过智能分析识别结果，生成数据报表和可视化图表，辅助业务决策和战略规划。

跨平台支持

扩展对更多操作系统和平台的支持，提升服务的普适性和便捷性。支持移动端、桌面端和云端的多平台应用，满足不同用户的使用需求。

增强的安全性与隐私保护

加强数据加密和访问控制，提升数据安全性和隐私保护能力，确保用户数据在传输和存储过程中的安全，符合更多的行业和地区数据保护法规。

自动化流程集成

支持更多的自动化流程和工作流集成，实现OCR与其他企业系统的无缝连接。例如，与企业资源规划（ERP）、客户关系管理（CRM）系统的集成，提升整体业务流程的自动化和智能化水平。

结论

腾讯云文档智能 OCR 能力为企业提供了一种高效、准确、智能的数据采集和处理解决方案。在业务团队项目中，通过合理的技术实践和集成，企业能够显著提升运营效率，降低成本，提升数据准确性和用户体验。本文通过详细的项目案例和丰富的代码示例，展示了 OCR 能力在金融、医疗、物流等多个行业的应用效果和收益。随着技术的不断进步和应用场景的拓展，智能 OCR 将在更多行业发挥重要作用，助力企业实现数字化转型和持续发展。
通过本文的技术指导和实践经验，企业可以更好地理解和应用腾讯云文档智能 OCR 能力，充分发挥其优势，实现业务流程的自动化和智能化，提升企业的核心竞争力。

附录：完整代码示例

为了帮助大家更好地理解和应用本文中的技术内容，以下提供一个完整的 Python 代码示例，涵盖图像预处理、OCR 调用、信息提取和异常处理等功能。

```
import json
import base64
import hashlib
import hmac
import time
import requests
import cv2import numpy as np
import threading
import logging
from tencentcloud.common import credential
from tencentcloud.ocr.v20181119 import ocr_client, models

# 配置日志
logging.basicConfig(filename='ocr_processing.log', level=logging.INFO,
                    format='%(asctime)s %(levelname)s:%(message)s')

# 配置API密钥
SECRET_ID = 'YOUR_SECRET_ID'
SECRET_KEY ='YOUR_SECRET_KEY'
ENDPOINT = 'ocr.tencentcloudapi.com'
REGION = 'ap-guangzhou'
ACTION = 'GeneralBasicOCR'
VERSION = '2022-03-23'

def preprocess_image(image_path):
    # 读取图像
    image = cv2.imread(image_path)

    # 灰度化
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # 二值化
    _, binary = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY_INV)

    # 去噪
    kernel = np.ones((1,1), np.uint8)
    denoised = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel)

    # 旋转校正
    coords = np.column_stack(np.where(denoised > 0))
```

```
    angle = cv2.minAreaRect(coords)[-1]
    if angle < -45:
        angle = -(90 + angle)    else:
        angle = -angle
    (h, w) = denoised.shape
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    rotated = cv2.warpAffine(denoised, M, (w, h), flags=cv2.INTER_CUBIC,
borderMode=cv2.BORDER_REPLICATE)
    # 保存预处理后的图像
    preprocessed_path = 'preprocessed_' + image_path
    cv2.imwrite(preprocessed_path, rotated)
    return preprocessed_path

def generate_signature(secret_key, params):
    sorted_params = sorted(params.items())
    canonical_query_string = '&'.join(['{}={}'.format(k, v) for k, v in
sorted_params])
    string_to_sign = 'GETocr.tencentcloudapi.com/?' +
canonical_query_string
    signature = hmac.new(secret_key.encode('utf-8'),
string_to_sign.encode('utf-8'), hashlib.sha1).digest()
    signature = base64.b64encode(signature).decode()
    return signature

def call_ocr_api(image_path):
    preprocessed_image = preprocess_image(image_path)
    with open(preprocessed_image, 'rb') as f:
        image_data = base64.b64encode(f.read()).decode('utf-8')

    params = {
        'Action': ACTION,
        'Version': VERSION,
        'Region': REGION,
        'Timestamp': int(time.time()),
        'Nonce': int(time.time()),
        'SecretId': SECRET_ID,
        'ImageBase64': image_data,
        'LanguageType': 'CHN_ENG'
    }

    signature = generate_signature(SECRET_KEY, params)
    params['Signature'] = signature
```

```
    try:
        response = requests.get(f'https://{ENDPOINT}/', params=params,
timeout=10)
        response.raise_for_status()
        return response.json()
    except requests.exceptions.RequestException as e:
        logging.error(f"HTTP Request failed: {e}")
        return None

def extract_fields(ocr_result):
    fields = {
        'Name': '',
        'ID Number': '',
        'Income': '',
        'Loan Amount': '',
        'Date': ''
    }

    for item in ocr_result.get('Response', {}).get('TextDetections',
[]):
        text = item.get('DetectedText', '')
        if '姓名' in text:
            fields['Name'] = text.split('姓名')[-1].strip()
        elif '身份证号' in text:
            fields['ID Number'] = text.split('身份证号')[-1].strip()
        elif '收入' in text:
            fields['Income'] = text.split('收入')[-1].strip()
        elif '贷款金额' in text:
            fields['Loan Amount'] = text.split('贷款金额')[-1].strip()
        elif '日期' in text:
            fields['Date'] = text.split('日期')[-1].strip()
    return fields

def safe_call_ocr_api_with_logging(image_path):
    response = call_ocr_api(image_path)
    if response and 'Response' in response and 'Error' not in
response['Response']:
        logging.info(f"OCR Success: {image_path}")
        fields = extract_fields(response)
        return fields
    else:
```

```
        error_message = response.get('Response', {}).get('Error',
{}).get('Message', 'Unknown Error')
        logging.error(f"OCR API Error for {image_path}:
{error_message}")
        return None

def process_image(image_path):
    fields = safe_call_ocr_api_with_logging(image_path)
    if fields:
        print(f"Processed {image_path}: {fields}")
    else:
        print(f"Failed to process {image_path}")

def main():
      image_paths = ['loan_application1.jpg', 'loan_application2.jpg',
'loan_application3.jpg']
      threads = []

      for path in image_paths:
          t = threading.Thread(target=process_image, args=(path,))
          threads.append(t)
          t.start()

      for t in threads:
      t.join()

if __name__ == "__main__":
      main()
```

代码说明

1. 图像预处理：使用 OpenCV 进行图像的灰度化、二值化、去噪和旋转校正，提升图像质量，确保 OCR 识别的准确性。
2. OCR 调用：通过 API 调用腾讯云 OCR 服务，上传预处理后的图像并获取识别结果。
3. 信息提取：根据自定义规则，从 OCR 响应中提取关键信息字段，如姓名、身份证号、收入等。
4. 异常处理与日志记录：实现对 OCR 调用的异常处理和日志记录，确保系统的稳定性和可维护性。
5. 并发处理：使用多线程对多张图像进行并发处理，提升处理效率。
6. 主函数：定义需要处理的图像路径列表，启动多线程进行并发处理。

通过这些代码示例，大家可以全面了解和应用腾讯云文档智能 OCR 能力，实现高效、准确的数据采集和处理。

结束语

腾讯云文档智能 OCR 能力为企业提供了一种高效、准确、智能的数据采集和处理解决方案。通过本文的详细介绍和丰富的代码示例，企业可以更好地理解和应用这一技术，提升数据处理效率，降低运营成本，提升业务的准确性和可靠性。随着技术的不断发展，智能 OCR 将在更多领域发挥重要作用，助力企业实现数字化转型和持续发展。
如果大家在应用过程中遇到任何问题或有进一步的需求，欢迎联系腾讯云技术支持团队，获取专业的技术指导和服务支持。

结尾

感谢您阅读本文，希望对您在实际项目中应用腾讯云文档智能 OCR 能力有所帮助。如有任何疑问或建议，欢迎在评论区留言交流。

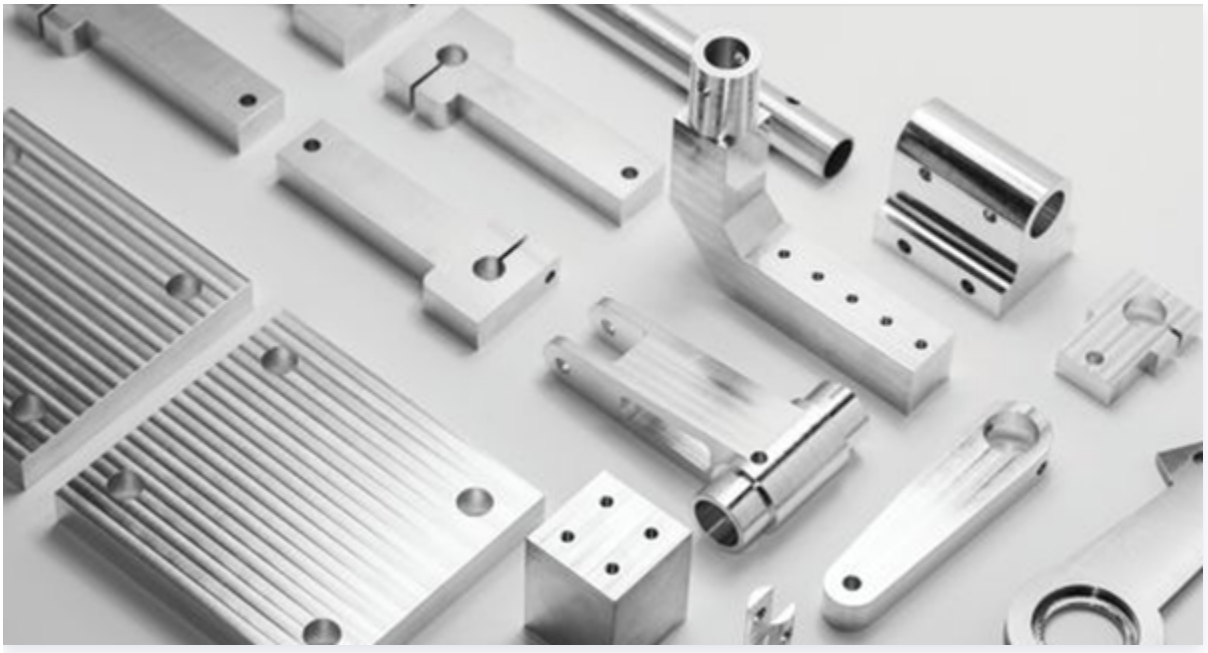
使用 OCR 实现自动识别与分类 CNC 加工铝件产品

最近更新时间：2025-05-27 17:59:31

厂家制造业正逐步向智能化转型。在这一背景下，如何提高生产效率、减少人工错误成为我们企业关注的重点之一。目前我们公司大力推广智能化，自动化产品及系统。我主要负责如何利用 OCR 技术来解决一家专注于使用CNC机床加工铝件产品的公司所面临的问题，并评估其潜在收益。

问题背景

- 业务场景：公司拥有大量的 CNC 机床用于加工铝件产品。每个产品上都会被刻有标识符，包括但不限于 SN（序列号，包含了产品型号、生产时间及机台编号等）。完成加工后，所有产品都需要经过清洗工序去除表面残留物，随后通过 AXI 检测确保尺寸符合标准。
- 挑战：当前流程中，对于成品的分拣主要依赖于人工操作，这不仅耗时费力，还容易出现误判或遗漏的情况。为了提高工作效率和准确性，计划开发一套自动化系统能够自动读取并解析这些标记信息，进而对不同批次的产品进行准确分类统计。



方案评估

根据现有技术寻找不同的厂家，我们 OCR 技术进行一个全面的比较。
综合评估 OCR 技术各有特点，适合不同的需求和应用场景。发现腾讯 OCR 比较适合我们公司应用场景。
腾讯 OCR 的优势其特点：

1. 数据安全与隐私保护：腾讯云 OCR 支持私有云部署，这意味着用户可以全权管理所有文件，从而更好地保护数据安全和隐私。这对于那些对数据安全有严格要求的企业来说，是一个非常重要的优势。

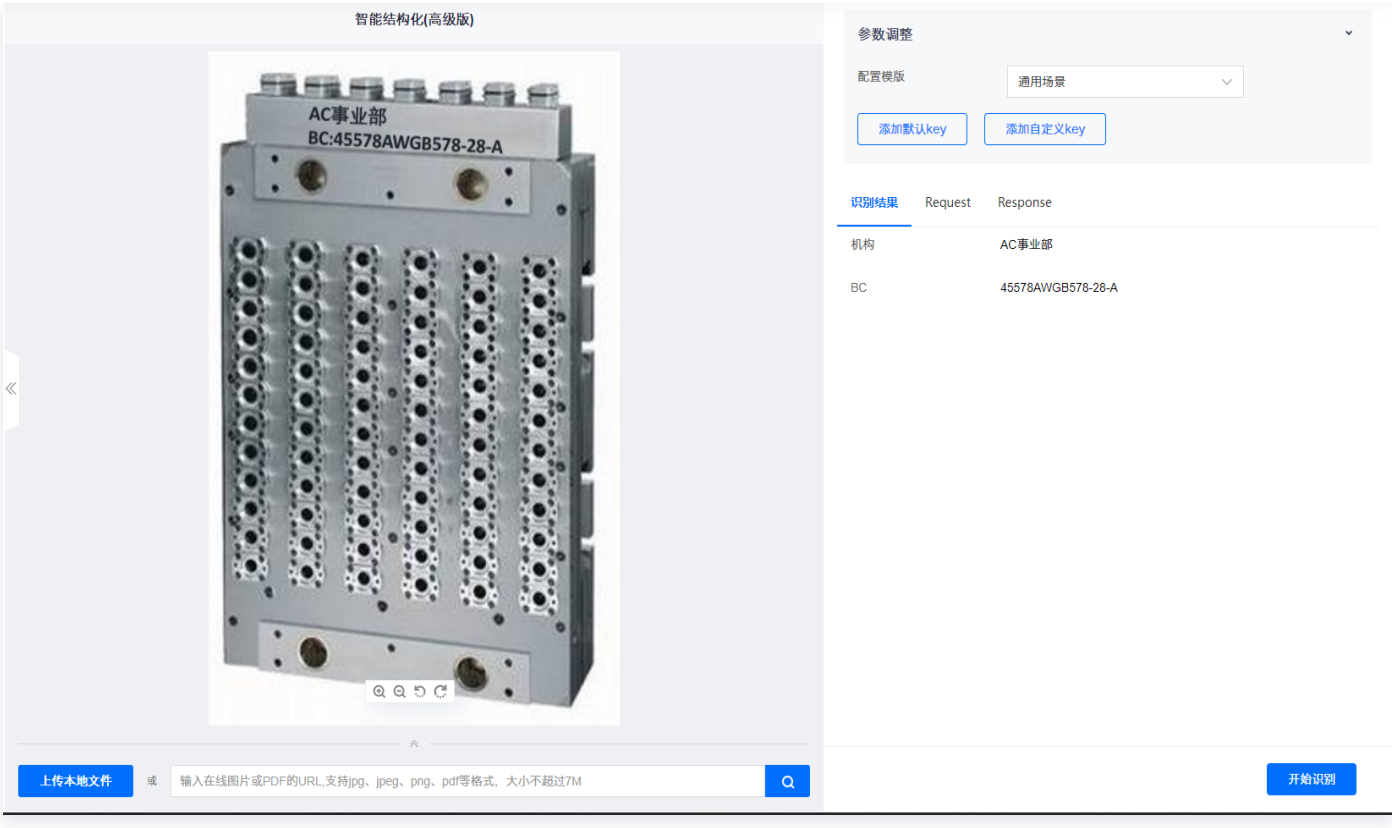
2. 高准确性：腾讯云 OCR 在印刷体和手写体的文字识别上表现出色，平均准确率分别可达90%以上和85%以上。这种高准确性的识别能力，确保了即使在复杂的文本环境下，也能保持较高的识别效率。
3. 服务稳定性：腾讯云 OCR 的服务稳定性得到了用户的广泛认可。这意味着即使在处理大量数据时，腾讯 OCR 也能保持高效和稳定的服务，这对于需要连续、可靠服务的企业至关重要。
4. 广泛的技术支持：腾讯云 OCR 支持中文简繁体、英文、数字、标点共10000+标签，覆盖上百种字体，甚至生僻字版本更支持2w+标签。这种广泛的技术支持，使得腾讯 OCR 能够适应多种不同的识别需求，具有很高的灵活性。
5. 性能提升：腾讯云 OCR 在性能方面也做出了显著的努力，通过优化，其通用 OCR 的平均耗时从1815ms减少到824ms，提升了2.2倍，显示出腾讯在持续优化其服务性能方面的决心和能力。

综上所述，腾讯 OCR 在数据安全、准确性、稳定性、技术支持范围以及性能优化方面都有显著的优势，这使得它成为我们 OCR解决方案首选，特别适合对数据安全和识别准确性有高要求。目前还处于验证阶段。

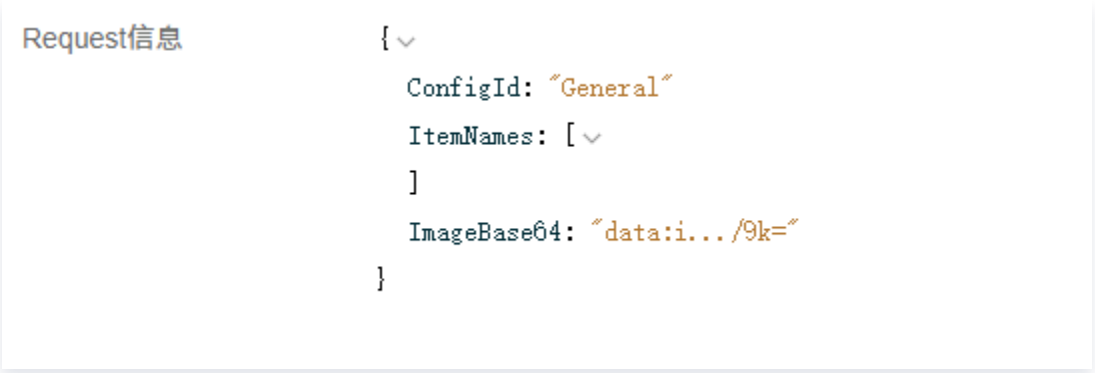
开发逻辑

1. 图像采集

2. 预处理
   - 图像增强：应用图像增强技术，如对比度调整、锐化等，以提高文字部分的清晰度。
   - 去噪处理：去除图像中的噪声，使 OCR 识别更加准确。
   - 校正倾斜：自动检测并校正图像中的倾斜角度，确保文字水平。
3. OCR 识别



4. 信息解析与分类



5. 统计报告生成
   - 数据汇总：定期汇总已分类的数据，生成详细的报表。
   - 数据分析：对数据进行分析，提供生产效率、质量控制等方面的见解。
   - 可视化展示：使用图表和仪表盘等方式，直观展示统计数据，便于管理层参考决策。

系统设计细节

1. 硬件选型：选择具有高分辨率和快速响应能力的摄像头，确保图像质量和采集速度。同时，需要配备足够强大的处理器和存储设备，以支持 OCR 识别和数据处理的高效运行。
2. 软件开发：开发 OCR 识别和数据处理模块。这些模块需要集成文档抽取（多模态版）API，并根据业务需求进行定制开发。需要注意的是，系统应具备良好的扩展性和兼容性，以便未来可能的功能升级和技术更新。

3. 数据库设计：设计合理的数据库结构，以存储识别和处理后的 SN 信息。数据库需要具备高效的查询和存储能力，以支持实时数据分析和报表生成。
4. 用户界面：开发友好的用户界面，以便于操作人员进行数据查询、报表生成和系统管理。用户界面应具备直观的图形化界面，简化操作流程，提高使用效率。

潜在收益

实施上述方案后，预计可带来以下几个方面的好处：

- 显著提升生产线整体运作效率：自动化系统可以大幅减少人工操作时间，提高生产效率。效率预计提升80%。
- 减少因人为因素造成的错误率：通过自动化识别和分类，减少了人工操作带来的误差。 误查可由原来5%减少为1%
- 为后续质量控制环节提供更多可靠依据：准确的数据记录和分类为质量控制提供了更多可靠的依据。
- 增强企业的市场竞争力：高效的生产和质量控制能力提升了企业的市场竞争力。
- 节省人力资源：自动化系统可以替代大量的人工工作，节省人力资源成本。 节省分拣人力，精简4人，效益为4 * 6w = 24W/年。
- 提高客户满意度：高质量的产品和快速的交付时间将提高客户的满意度。

总之，通过引入先进的 OCR 技术，不仅可以帮助我们公司解决现有难题，还能为其未来发展打下坚实基础。

腾讯云 OCR 在制造业的应用：内存模组产品识别实战指南

最近更新时间：2025-05-27 17:59:31

一、背景

制造业在产品识别环节经历着前所未有的挑战。传统的依赖人工进行产品识别的方法效率低下，难以满足现代化生产线高速运转的需求，导致生产周期延长和交货延迟。高昂的人工成本进一步加剧了这个问题，尤其是在需要大量人力进行细致检查和分类的场景下。此外，人工识别容易出错，即使经验丰富的工人也可能出现视觉疲劳和判断失误。因此，提高产品识别效率、降低人工成本和错误率，已成为许多制造企业亟待解决的关键问题。
腾讯云 OCR 是腾讯云提供的一项强大的图像识别服务，能够快速、准确地将图片或文档中的文字信息提取出来。它基于腾讯云领先的深度学习技术，具备卓越的图像识别能力，能够处理多种语言、多种字体、不同清晰度和复杂的背景图像。

腾讯云 OCR 的优势在于其高精确性和高效率。 处理速度快，能够快速识别大量的图像，有效减少人工处理的时间，大幅提升工作效率。此外，腾讯云 OCR 还提供灵活的 API 接口和 SDK，方便用户集成到各种应用系统中，降低了开发和部署成本。
本文通过内存模组产品识别案例，详细讲解腾讯云 OCR 在制造业的应用，并提供可操作的实战指南。

二、腾讯云 OCR 技术概述

运用顶尖的深度学习、图像检测和 OCR 大模型技术，腾讯云 OCR 实现了对任意版式文档的结构化信息提取。从标准化证件到复杂的物流单据，都能精准识别。系统预先学习并建立了键值对映射关系，并支持客户自定义模板，显著提升数据录入效率，广泛应用于政务、票据审核、行业表单和国际物流等领域。
核心功能：

- 灵活键值定义：用户可自行定义键值对（key-value），系统将自动匹配图片文字与自定义键值，从而解析任意版式图片的结构化信息。

- 智能模板匹配：系统能智能识别上传图片，并将其自动匹配到已有的模板，无需人工分类，快速实现结构化信息提取。
- 自定义字段类型：支持创建多种字段类型，针对不同内容（如金额、日期、数字）进行精准优化，提升识别准确率。更进一步，用户可自定义字段的可能取值范围，实现智能校正和规范化输出。

核心优势：

- 精准识别：支持各种版式的证件和票据识别，其字段识别精度处于行业领先地位，文本识别准确率超过98%，结构化识别准确率超过85%。
- 全面覆盖：涵盖多种常见证件和票据的结构化信息提取，例如警官证、教师资格证、道路运输证等，广泛适用于各行各业。
- 便捷易用：只需简单的几步操作即可创建个性化模板，快速提取结构化数据，大幅提升数据录入效率。
- 多语言支持，API 接口易于集成。

适用场景：

- 政务及身份认证：适用于政务、教育、金融等行业，可高效处理各类标准化证件（如警官证、英语等级证书、教师资格证、临时身份证等）的结构化信息提取，简化身份认证流程，提升业务办理效率。
- 行业表单自动化：针对医疗、物流、金融、制造等行业，支持定制专属模板，自动识别和录入财务票据、体检报告、物流单据等各类表单信息，实现业务流程自动化，提升行业信息化水平。

腾讯云 文档智能 OCR 提供两种方案，满足不同需求：

基础版：就像一个通用的文字识别工具，能轻松搞定各种常见文档，例如身份证、银行卡、发票等等，只要文字比较规范，它都能准确识别。

高级版：如果您需要更强大的功能，高级版就是您的选择。它可以根据您的具体情况定制识别模板，即使是格式复杂、设计独特的文档，它也能轻松应对，特别适合一些对识别精度要求很高的特殊行业。 您可以把它想象成一个私人定制版文字识别专家，能满足您所有个性化需求。

三、内存模组产品识别需求

简单来说就是：用电脑“看”懂内存条上的信息！您想想，内存条上那些型号、容量、频率、生厂商等等，密密麻麻的小字，人工一个一个看太费劲了，对吧?

所以需要有一个系统，能快速、准确地从内存条的图片或者视频里，直接把这些关键信息都识别出来。
不过这活儿还真有点难！内存条上的标签往往很小，字体也五花八门，有的字还模糊不清；拍照的时候，光线不好或者角度不对，也会影响识别效果。
但我们希望最后能做到：系统自动把这些信息提取出来，整理成规规矩矩的表格，这样就能方便地用来分析和处理数据了，不用再人工录入，省时省力！
例如这样的图片中提取出标签的所有信息：

想体验腾讯云文档智能 OCR 的强大功能?

- **了解产品详情及文档：** 文档智能 OCR 点击这里，更多的产品信息和使用说明。
- **立即体验 Demo：** OCR Demo — 亲自动手试试，感受OCR的便捷！
- **限时优惠活动：** 文字识别特惠活动文字识别购买文字识别选购 。

可以点击上面的“了解产品详情及文档”链接，然后在页面找到“立即体验”按钮，开始 OCR 之旅！

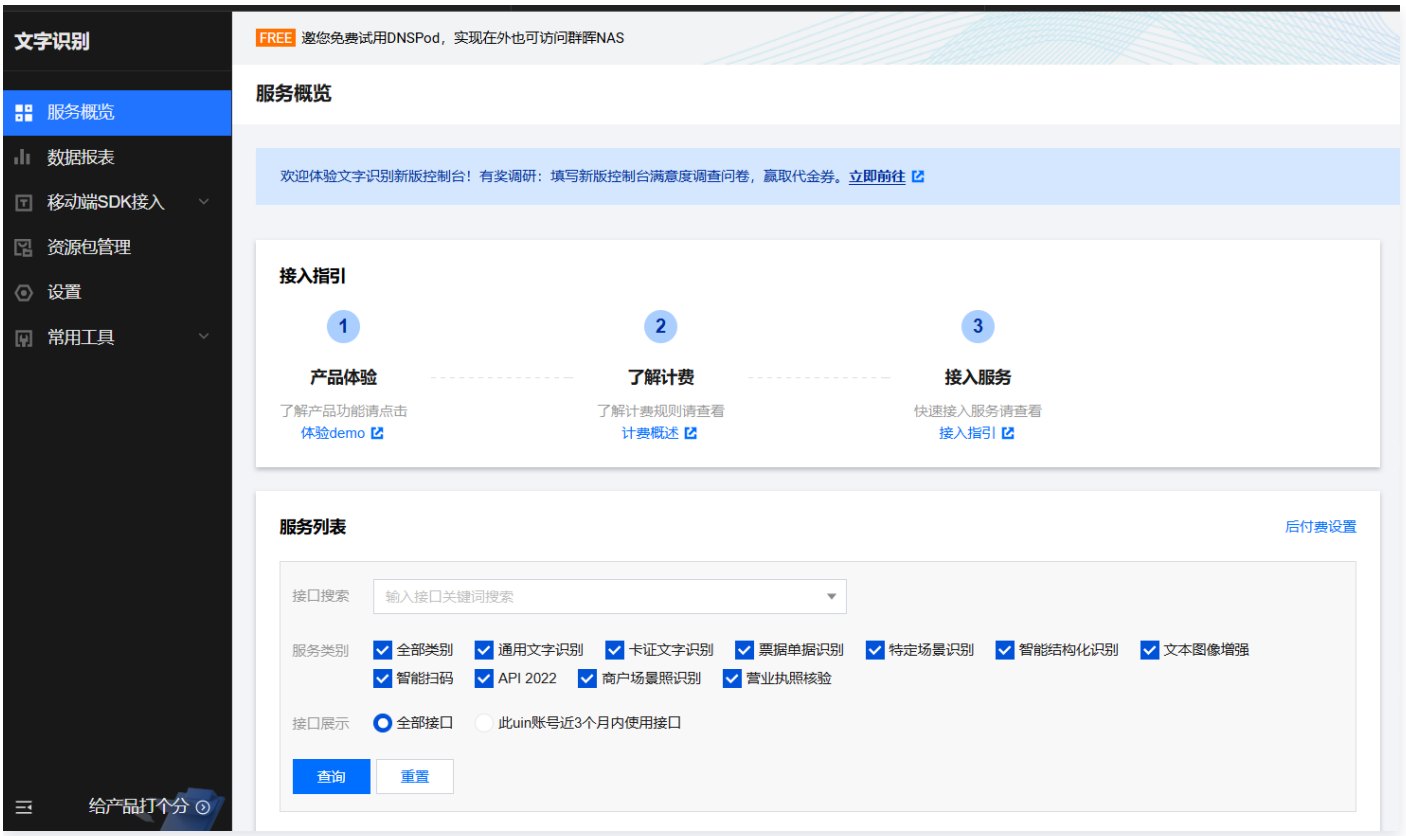
四、基于腾讯云 OCR 的内存模组产品识别

玩转腾讯云智能 OCR!

准备工作

第一步：先登录腾讯云控制台。还没账号？官方提供了账号注册教程 快速完成注册和实名认证。

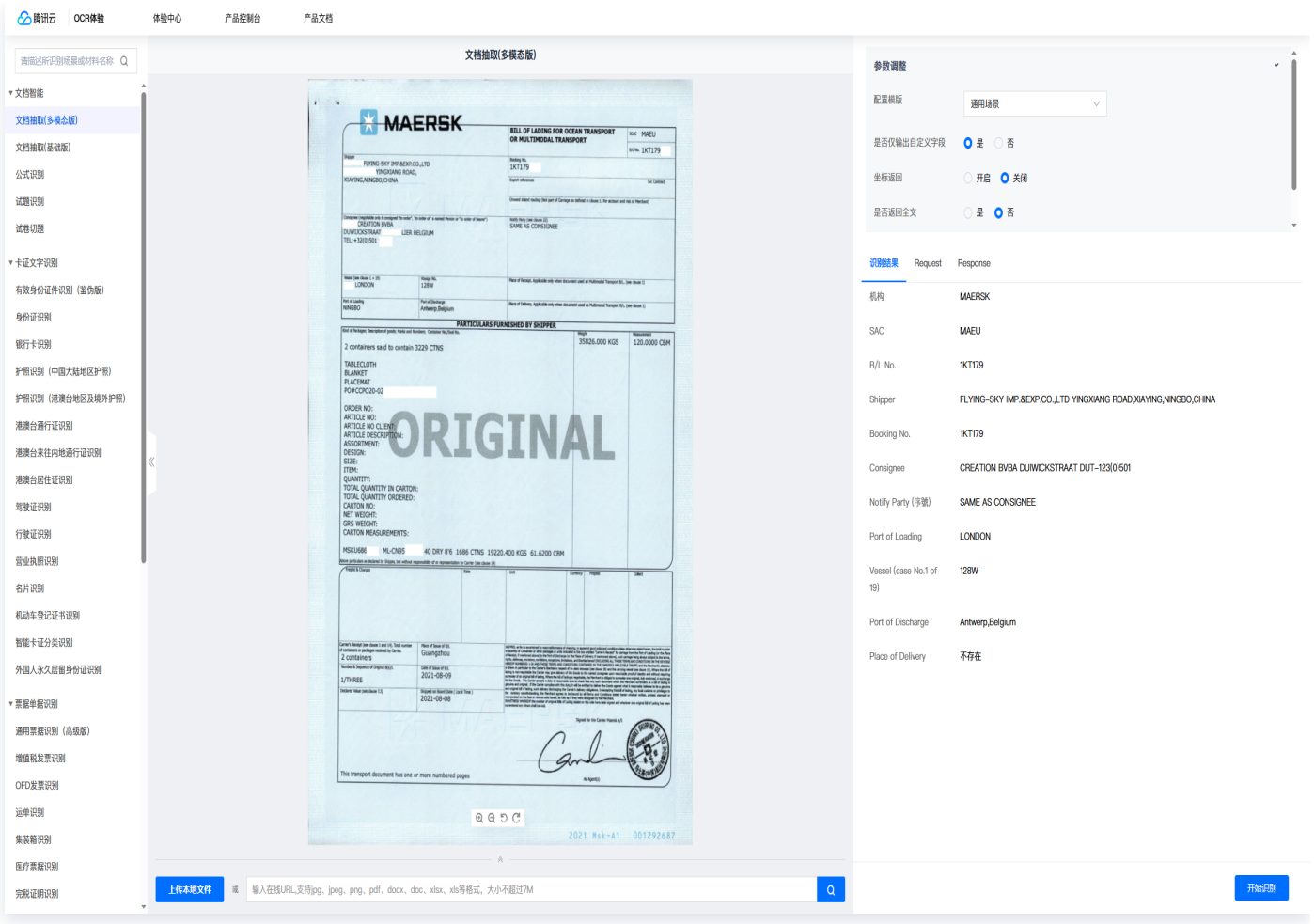
第二步：开通服务 点击进入 文字识别控制台 ，轻松开通服务。开通后如下：



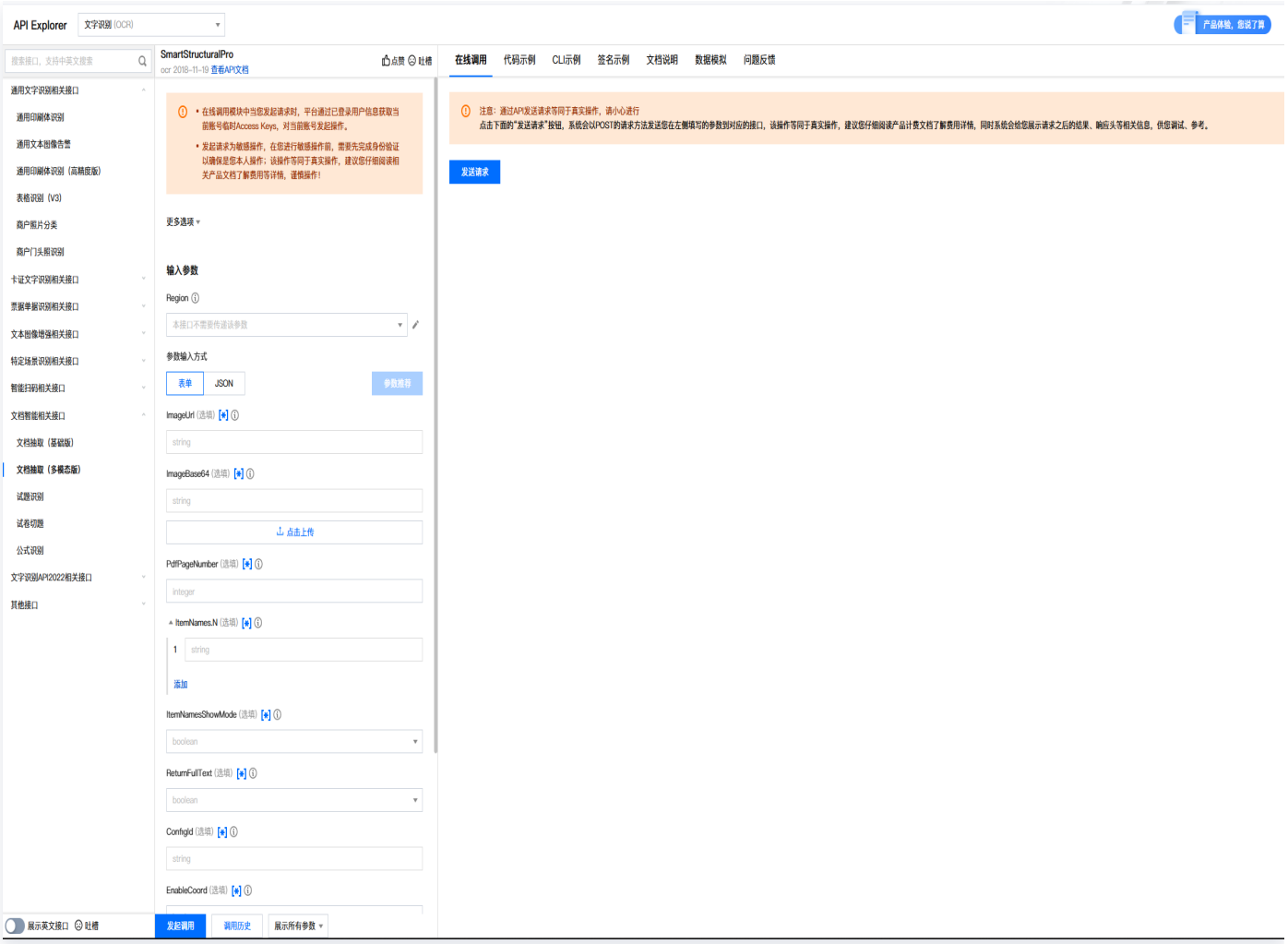
第三步：根据需求，可以选择以下四种方式之一：

1. **零代码体验：想快速体验？直接使用 文字识别体验 Demo ，选择“ 文档智能 > 文档抽取(多模态版)”即可。**

   注意：此方式仅供体验，每次只能识别一张图片，不适合实际开发。



2. 在线 API 调用(初学者友好)：如果您懂 HTTP 请求和 API 调用，可以使用 API 3.0 Explorer 在线测试 API，体验在线调用、签名验证、代码生成等功能。



3. 服务端 API 开发(开发者)：腾讯云 OCR 提供了多种语言 SDK(Python, Java, C++, PHP, Go, NodeJS, .Net等)，快速集成文字识别功能。服务端 API 接入指南 将帮助快速上手。
4. 客户端 SDK 集成(移动端开发者)：针对 Android 和 iOS 平台，腾讯云 OCR 提供了客户端 SDK，轻松将文字识别功能集成到 App 中。客户端 SDK 下载 和 客户端Demo 将帮助快速完成集成。

腾讯云 API 3.0 全新升级的开发者工具套件(SDK 3.0)现已推出，作为C++后端开发工程师，本文首选C++编程语言高效接入腾讯云服务。

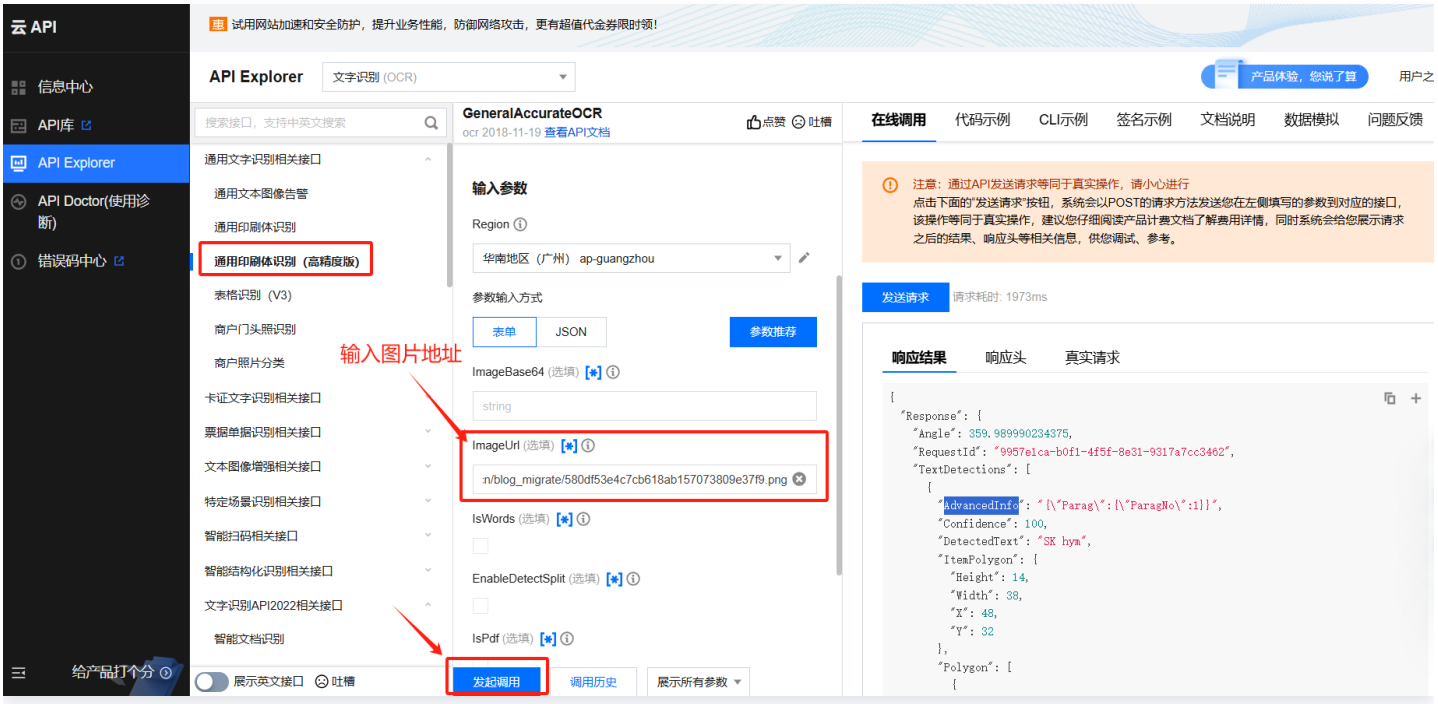
API 调用与代码实现

关于 C++ SDK 的安装教程可以参考 腾讯云 SDK 中心 的教程，这里不一一赘述了。

```
git clone https://github.com/TencentCloud/tencentcloud-sdk-cpp.git
cd tencentcloud-sdk-cpp
mkdir sdk_build
cd sdk_build
# centos 下使用 cmake3 ..
# 指定产品编译，分号;分隔
cmake -DBUILD_MODULES="cvm;cbs" ..
make
```

```
sudo make install
```

下面我们通过 API Explorer 中在线这个功能，以通用印刷体识别（高精度版）为例。



调用后会得到类似如下的 JSON 结果：

```
{
  "Response": {
    "Angle": 359.989990234375,
    "RequestId": "9957e1ca-b0f1-4f5f-8e31-9317a7cc3462",
    "TextDetections": [
      {
        "AdvancedInfo": "{\"Parag\":{\"ParagNo\":1}}",
        "Confidence": 100,
        "DetectedText": "SK hym",
        "ItemPolygon": {
          "Height": 14,
          "Width": 38,
          "X": 48,
          "Y": 32
        },
        "Polygon": [
          {
            "X": 49,
            "Y": 32
          },
          {
```

```
            "X": 85,
            "Y": 34
          },
          {
            "X": 84,
            "Y": 45
          },
          {
            "X": 48,
            "Y": 43
          }
        ],
        "WordCoordPoint": [],
        "Words": []
      },
      {
        "AdvancedInfo": "{\"Parag\":{\"ParagNo\":4}}",
        "Confidence": 100,
        "DetectedText": "豆",
        "ItemPolygon": {
          "Height": 20,
          "Width": 13,
          "X": 576,
          "Y": 117
        },
        "Polygon": [
          {
            "X": 576,
            "Y": 117
          },
          {
            "X": 588,
            "Y": 117
          },
          {
            "X": 588,
            "Y": 136
          },
          {
            "X": 576,
            "Y": 136
          }
        ],
```

```
      "WordCoordPoint": [],
      "Words": []
    },
    {
      "AdvancedInfo": "{\"Parag\":{\"ParagNo\":2}}",
      "Confidence": 100,
      "DetectedText": "SKhynix",
      "ItemPolygon": {
        "Height": 17,
        "Width": 54,
        "X": 149,
        "Y": 145
      },
      "Polygon": [
        {
          "X": 149,
          "Y": 145
        },
        {
          "X": 202,
          "Y": 145
        },
        {
          "X": 202,
          "Y": 161
        },
        {
          "X": 149,
          "Y": 161
        }
      ],
      "WordCoordPoint": [],
      "Words": []
    },
    {
      "AdvancedInfo": "{\"Parag\":{\"ParagNo\":2}}",
      "Confidence": 100,
      "DetectedText": "16GB 2Rx8 PC4-2933Y-RE2-12",
      "ItemPolygon": {
        "Height": 18,
        "Width": 214,
        "X": 206,
        "Y": 145
```

```
      },
      "Polygon": [
        {
          "X": 206,
          "Y": 145
        },
        {
          "X": 419,
          "Y": 145
        },
        {
          "X": 419,
          "Y": 162
        },
        {
          "X": 206,
          "Y": 162
        }
      ],
      "WordCoordPoint": [],
      "Words": []
    },
    {
      "AdvancedInfo": "{\"Parag\":{\"ParagNo\":3}}",
      "Confidence": 100,
      "DetectedText": "ECO",
      "ItemPolygon": {
        "Height": 13,
        "Width": 25,
        "X": 536,
        "Y": 153
      },
      "Polygon": [
        {
          "X": 536,
          "Y": 153
        },
        {
          "X": 560,
          "Y": 153
        },
        {
          "X": 560,
```

```
            "Y": 165
          },
          {
            "X": 536,
            "Y": 165
          }
        ],
        "WordCoordPoint": [],
        "Words": []
      },
      {
        "AdvancedInfo": "{\"Parag\":{\"ParagNo\":2}}",
        "Confidence": 100,
        "DetectedText": "KOREA",
        "ItemPolygon": {
          "Height": 19,
          "Width": 52,
          "X": 147,
          "Y": 160
        },
        "Polygon": [
          {
            "X": 148,
            "Y": 160
          },
          {
            "X": 198,
            "Y": 162
          },
          {
            "X": 197,
            "Y": 178
          },
          {
            "X": 147,
            "Y": 175
          }
        ],
        "WordCoordPoint": [],
        "Words": []
      },
      {
        "AdvancedInfo": "{\"Parag\":{\"ParagNo\":2}}",
```

```
        "Confidence": 100,
        "DetectedText": "HMA82GR7CJR8N-WM TG AC 2008",
        "ItemPolygon": {
          "Height": 21,
          "Width": 228,
          "X": 204,
          "Y": 165
        },
        "Polygon": [
          {
            "X": 205,
            "Y": 165
          },
          {
            "X": 431,
            "Y": 167
          },
          {
            "X": 430,
            "Y": 185
          },
          {
            "X": 204,
            "Y": 184
          }
        ],
        "WordCoordPoint": [],
        "Words": []
      }
    ]
  }
}
```

点击代码示例得到如下内容：

```
#include <tencentcloud/core/Credential.h>
#include <tencentcloud/core/profile/ClientProfile.h>
#include <tencentcloud/core/profile/HttpProfile.h>
#include <tencentcloud/ocr/v20181119/OcrClient.h>
#include <tencentcloud/ocr/v20181119/model/GeneralAccurateOCRRequest.h>
#include <tencentcloud/ocr/v20181119/model/GeneralAccurateOCRResponse.h>
#include <iostream>
#include <string>
```

```
#include <vector>

using namespace TencentCloud;
using namespace TencentCloud::Ocr::V20181119;
using namespace TencentCloud::Ocr::V20181119::Model;
using namespace std;

int main() {
    // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需
注意密钥对的保密
        // 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的
安全性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见:
https://cloud.tencent.com/document/product/1278/85305
        // 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi
进行获取
    Credential cred = Credential("SecretId", "SecretKey");
    // 实例化一个http选项，可选的，没有特殊需求可以跳过
    HttpProfile httpProfile = HttpProfile();
    httpProfile.SetEndpoint("ocr.tencentcloudapi.com");

    // 实例化一个client选项，可选的，没有特殊需求可以跳过
    ClientProfile clientProfile = ClientProfile();
    clientProfile.SetHttpProfile(httpProfile);
    // 实例化要请求产品的client对象,clientProfile是可选的
    OcrClient client = OcrClient(cred, "ap-guangzhou", clientProfile);

    // 实例化一个请求对象,每个接口都会对应一个request对象
    GeneralAccurateOCRRequest req = GeneralAccurateOCRRequest();

    req.SetImageUrl("https://i-
blog.csdnimg.cn/blog_migrate/580df53e4c7cb618ab157073809e37f9.png");

    // 返回的resp是一个GeneralAccurateOCRResponse的实例，与请求对象对应
    auto outcome = client.GeneralAccurateOCR(req);
    if (!outcome.IsSuccess())
    {
        cout << outcome.GetError().PrintAll() << endl;
        return -1;
    }
    GeneralAccurateOCRResponse resp = outcome.GetResult();
    // 输出json格式的字符串回包
    cout << resp.ToJsonString() << endl;
```

```
    return 0;
}
```

其中 resp.ToJsonString() 就是上面的 JSON 结果，我们只要对 JSON 字符串结果进行解析即可得到想要的结果。

五、代码示例

解析 JSON 结果的代码：

```
#include <tencentcloud/core/Credential.h>
#include <tencentcloud/core/profile/ClientProfile.h>
#include <tencentcloud/core/profile/HttpProfile.h>
#include <tencentcloud/ocr/v20181119/OcrClient.h>
#include <tencentcloud/ocr/v20181119/model/GeneralAccurateOCRRequest.h>
#include <tencentcloud/ocr/v20181119/model/GeneralAccurateOCRResponse.h>
#include <iostream>
#include <string>
#include <vector>
#include <json/json.h> // 需要包含jsoncpp库

using namespace TencentCloud;
using namespace TencentCloud::Ocr::V20181119;
using namespace TencentCloud::Ocr::V20181119::Model;
using namespace std;

int main() {
    // ... (Credential and Client initialization remains the same) ...

    auto outcome = client.GeneralAccurateOCR(req);
    if (!outcome.IsSuccess())
    {
        cout << outcome.GetError().PrintAll() << endl;
        return -1;
    }
    GeneralAccurateOCRResponse resp = outcome.GetResult();
    string jsonString = resp.ToJsonString();

    Json::Reader reader;
    Json::Value root;
    if (reader.parse(jsonString, root)) {
        const Json::Value& textDetections = root["Response"]
["TextDetections"];
```

```
        for (const auto& detection : textDetections) {
            if (!detection["AdvancedInfo"].isNull()) {
                cout << "AdvancedInfo: " <<
detection["AdvancedInfo"].asString() << endl;
            }
            if (!detection["DetectedText"].isNull()) {
                cout << "DetectedText: " <<
detection["DetectedText"].asString() << endl;
            }
        }
    } else {
        cerr << "Failed to parse JSON: " <<
reader.getFormattedErrorMessages() << endl;
        return -1;
    }

    return 0;
}
```

为了方便，这里我使用了 jsoncpp 库， 源码地址是 https://github.com/open-source-parsers/jsoncpp 。
如果使用 CMake 构建项目，步骤会更简单：

```
add_subdirectory(path/to/jsoncpp) # Replace with the path to your
jsoncpp source
target_link_libraries(your_target PRIVATE jsoncpp::jsoncpp)
include_directories(path/to/jsoncpp/include)
```

记得将 `"SecretId"` 和 `"SecretKey"` 替换成您的实际腾讯云 SecretId 和 SecretKey。
腾讯云 OCR 就像一个超级快的、不会疲倦的、又很准确的抄写员，它能帮您快速、廉价地把照片上的文字转换成电脑能识别的文字，省时省力又省钱！ 尤其是当您需要处理很多照片的时候，这个优势就更加明显了。
腾讯云 OCR 通过其高效率、高准确率和低成本的特点，为企业和个人提供了比人工识别更优越的解决方案。

六、应用场景

腾讯云 OCR 在制造业的应用远不止于简单的文字识别，它还可以赋能诸多场景，例如零件识别和质量检测，显著提升效率和降低成本：
零件识别:

- 场景: 在生产线上快速识别各种零件，包括型号、批次、生产日期等信息，用于库存管理、生产追踪和质量追溯。 许多零件上印有难以人工识别的微小字符或二维码。
- 应用: 腾讯云 OCR 结合图像识别技术，可以对零件上的标识进行快速、准确的识别，即使是模糊、污损或角度倾斜的标识也能有效识别。 这可以避免人工逐一检查的低效和出错率，实现自动化识别和数据录入。 识别结果

可以实时上传到生产管理系统，方便后续的流程管理。

- 优势：提高生产效率，降低人工成本，减少错误率，实现生产过程的数字化和可追溯性。

质量检测：

- 场景：检测产品外观缺陷，例如划痕、裂纹、污点等；识别产品包装上的瑕疵；验证产品上的标识是否符合标准。
- 应用：腾讯云 OCR 可以结合图像处理技术，对产品图片进行分析，自动识别出产品缺陷，并进行分类和统计。例如，检测电路板上的元器件是否缺失或焊接不良；识别纺织品上的污点和瑕疵；检查药品包装上的标识是否完整清晰。

其他应用场景：

- 工装管理：识别工装工具上的标识，方便管理和维护。
- 设备维护：识别设备铭牌上的信息，方便维护人员快速了解设备状态。
- 安全管理：识别安全标识，确保生产环境安全。
- 文档管理：数字化生产文件，如图纸、说明书等，方便检索和管理。

腾讯云 OCR 在制造业场景的优势：

- 高准确率：即使在光线不佳或标识模糊的情况下也能保持较高的识别准确率。
- 高效率：可以大幅度提高识别速度，降低人工成本。
- 可扩展性：可以轻松集成到现有的生产管理系统中。
- 降低成本：减少人工成本、降低次品率、提高生产效率，整体降低生产成本。
- 数据可视化：将识别结果转化为数据，方便进行分析和决策。

腾讯云 OCR 未来的发展方向将是朝着更加智能化、自动化、个性化和普适化的方向发展，最终目标是让 OCR 技术成为一种简单易用、高效可靠的工具，广泛应用于各个行业和领域。

七、总结

腾讯云 OCR 在内存模组产品识别中高效识别芯片型号、容量等关键信息，显著提升了生产效率和数据准确性，减少人工错误。这凸显了其在制造业中的重要意义：实现自动化质检、精细化管理，最终提升产品质量和竞争力。

参考：

- 腾讯云文字识别产品官网
- 腾讯云 OCR 技术文档
- 腾讯云 OCR API 接口文档
- 腾讯云 OCR API 在线调用页面
- 腾讯云 OCR 体验 Demo
- 腾讯云文档智能 OCR 产品页

玩转腾讯云文档智能：OCR 推动文档处理与数据提取进入新时代

最近更新时间：2025-05-27 17:59:31

引言

在数字化时代，文档处理与数据提取已成为企业和个人日常工作中不可或缺的一部分。随着技术的不断进步，OCR（光学字符识别）技术已逐渐成为这一领域的关键支撑。OCR 技术通过将图像中的文字转换为可编辑的文本，极大地提高了文档处理的效率和准确性。腾讯云 文档智能 OCR 作为这一领域的佼佼者，凭借其强大的功能和广泛的应用场景，正引领着文档处理与数据提取的新时代。

从传统的证件识别、票据处理，到如今复杂文档的结构化提取，OCR 技术的发展日新月异。它不仅极大地简化了人工处理文档的繁琐流程，还通过自动化的方式显著减少了错误，提高了数据准确性和工作效率。在政务、金融、零售等多个行业中，OCR 技术的应用已经深入人心，成为推动业务流程优化和决策效率提升的关键因素。

一、腾讯云文档智能 OCR 技术原理与优势

技术原理

腾讯云文档智能 OCR 技术是一种基于深度学习、图像检测及 OCR 大模型等多种先进技术相融合的综合性解决方案。其工作流程可以大致分为以下几步：

1. 图像采集与预处理

   通过高精度图像采集设备获取文档图像。然后，对图像进行预处理，包括去噪、二值化、倾斜校正等操作，以提高后续识别的准确性。

2. 图像特征提取

   利用深度学习算法对预处理后的图像进行特征提取，识别出图像中的文字、表格等关键信息。

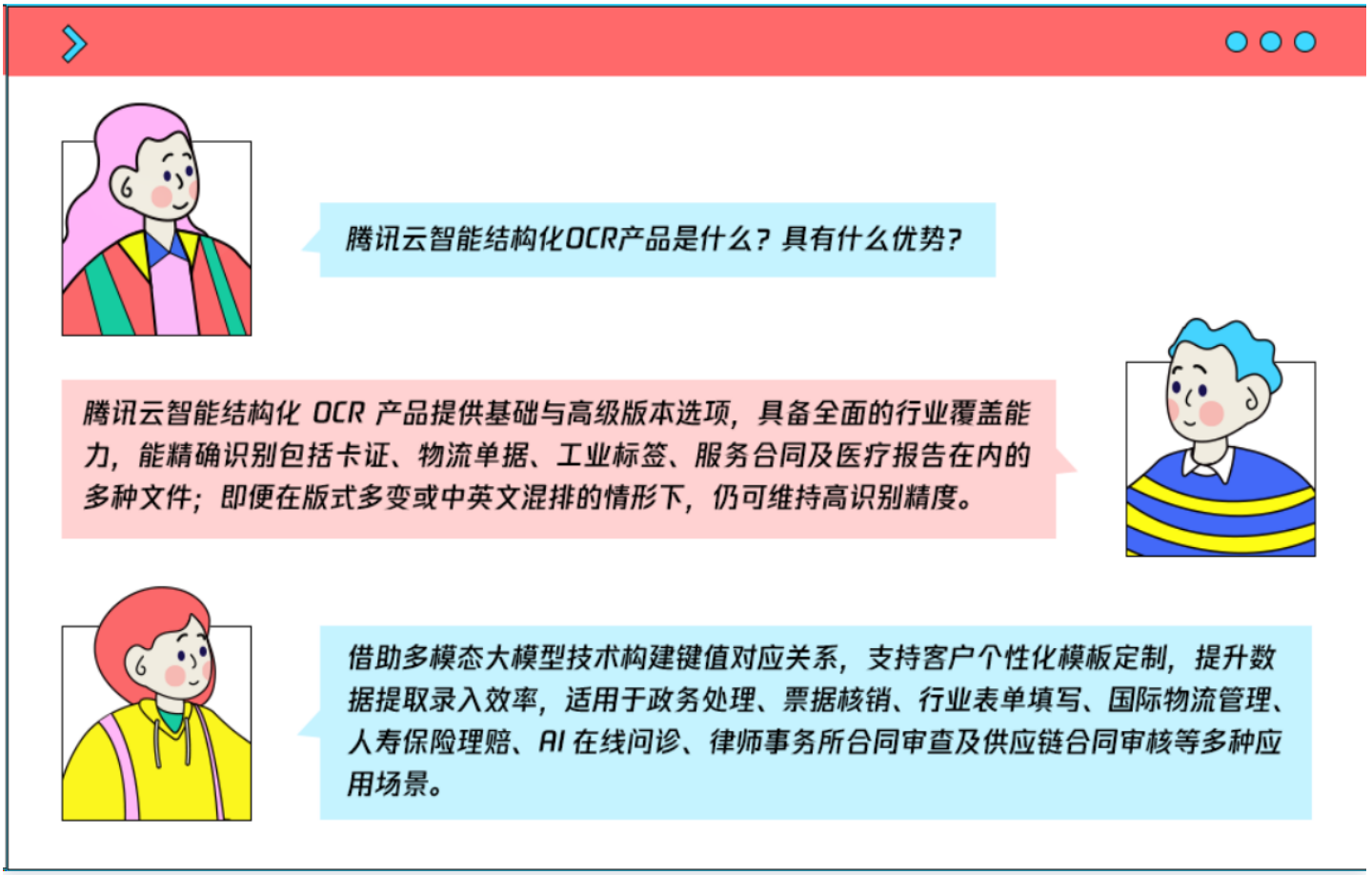
3. OCR 识别

基于 OCR 大模型，对提取出的特征进行文字识别，将图像中的文字转换为可编辑的文本。

4. 结构化输出

根据用户需求，将识别结果进行结构化处理，输出为 Excel、XML 等格式，方便后续的数据处理和分析。

技术优势



1. 高识别准确率

基于深度学习和 OCR 大模型的融合应用，腾讯云文档智能 OCR 技术能够实现对复杂文档的高效、精准识别，识别准确率高达98%以上。

2. 自定义模板

用户可以根据业务需求自定义识别模板，实现特定文档的快速、准确识别。这一功能大大提升了产品的灵活性和适用性。

3. 易于集成

腾讯云文档智能 OCR 提供了友好的 API 接口和丰富的 SDK，可以轻松集成到各类业务系统中，无需复杂的开发工作。

4. 多场景适用

无论是证件识别、票据处理，还是文档扫描与转换，腾讯云文档智能 OCR 都能轻松应对，满足不同场景下的需求。

二、腾讯云文档智能 OCR 使用教程

注册与登录

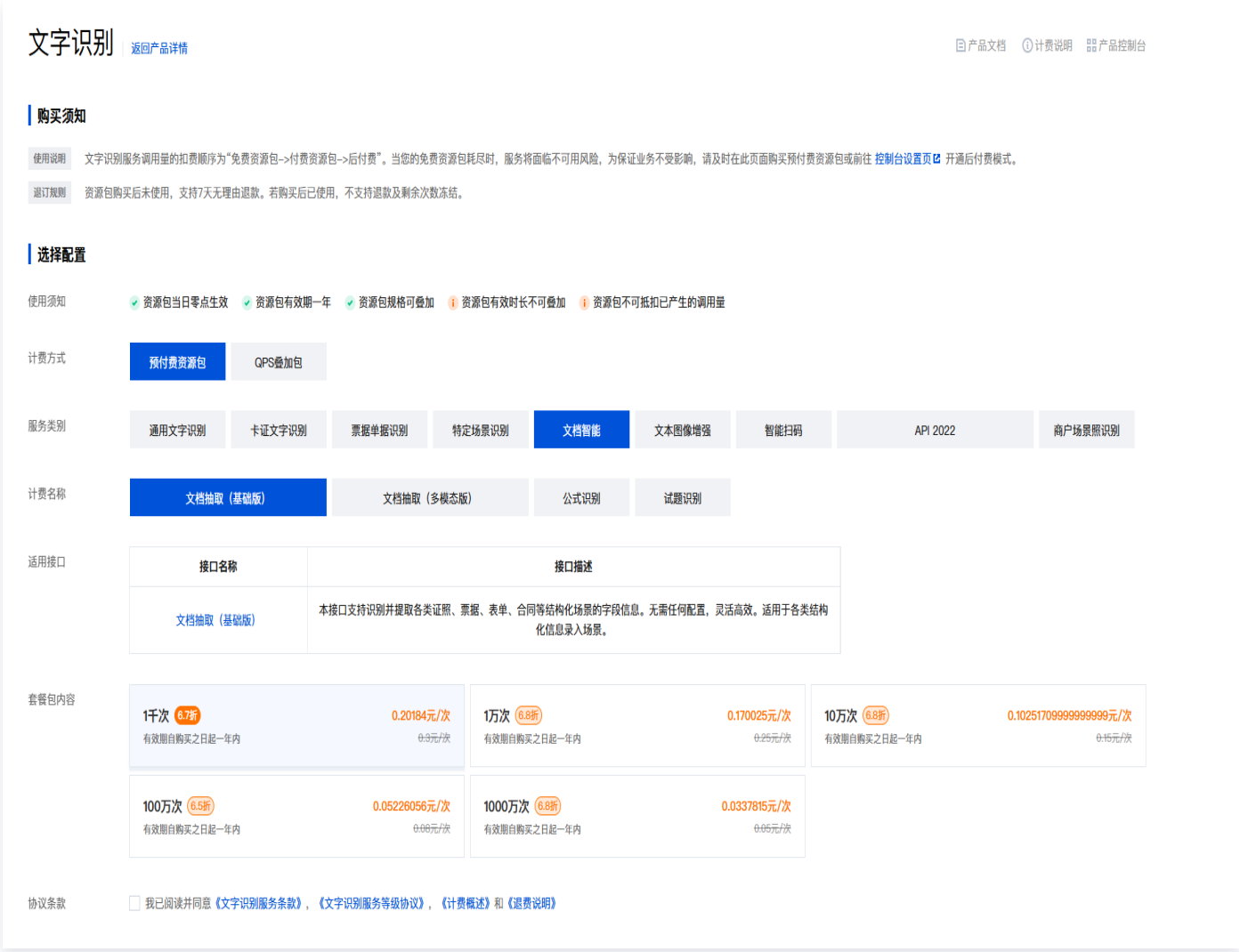
1. 访问腾讯云官方网站，注册并登录您的腾讯云账号。

2. 在控制台中，找到并点击“文档智能 OCR”服务。

开通服务

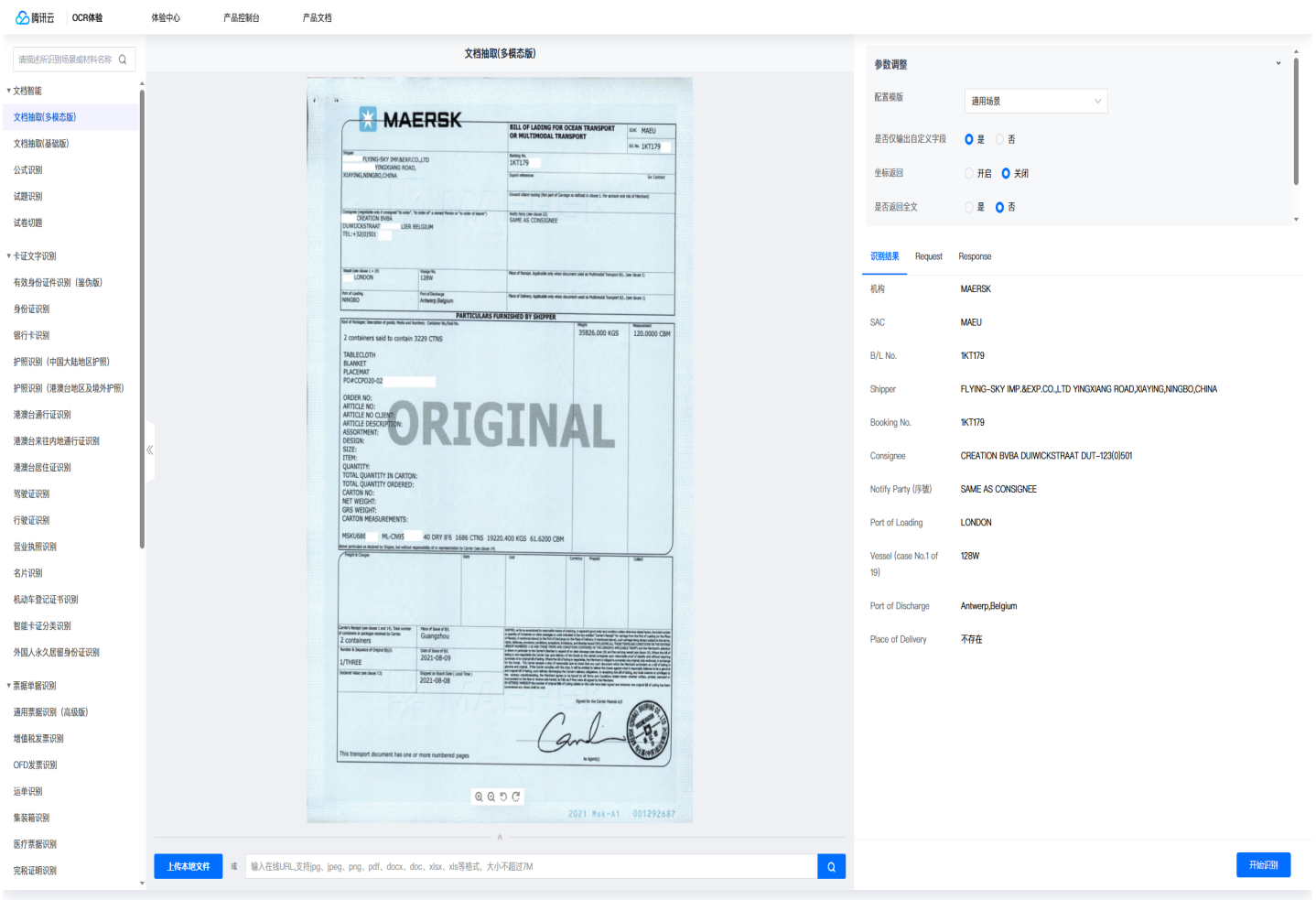
1. 进入控制台，可对相关功能进行操作。

2. 根据使用需求购买相应的资源包



体验 demo

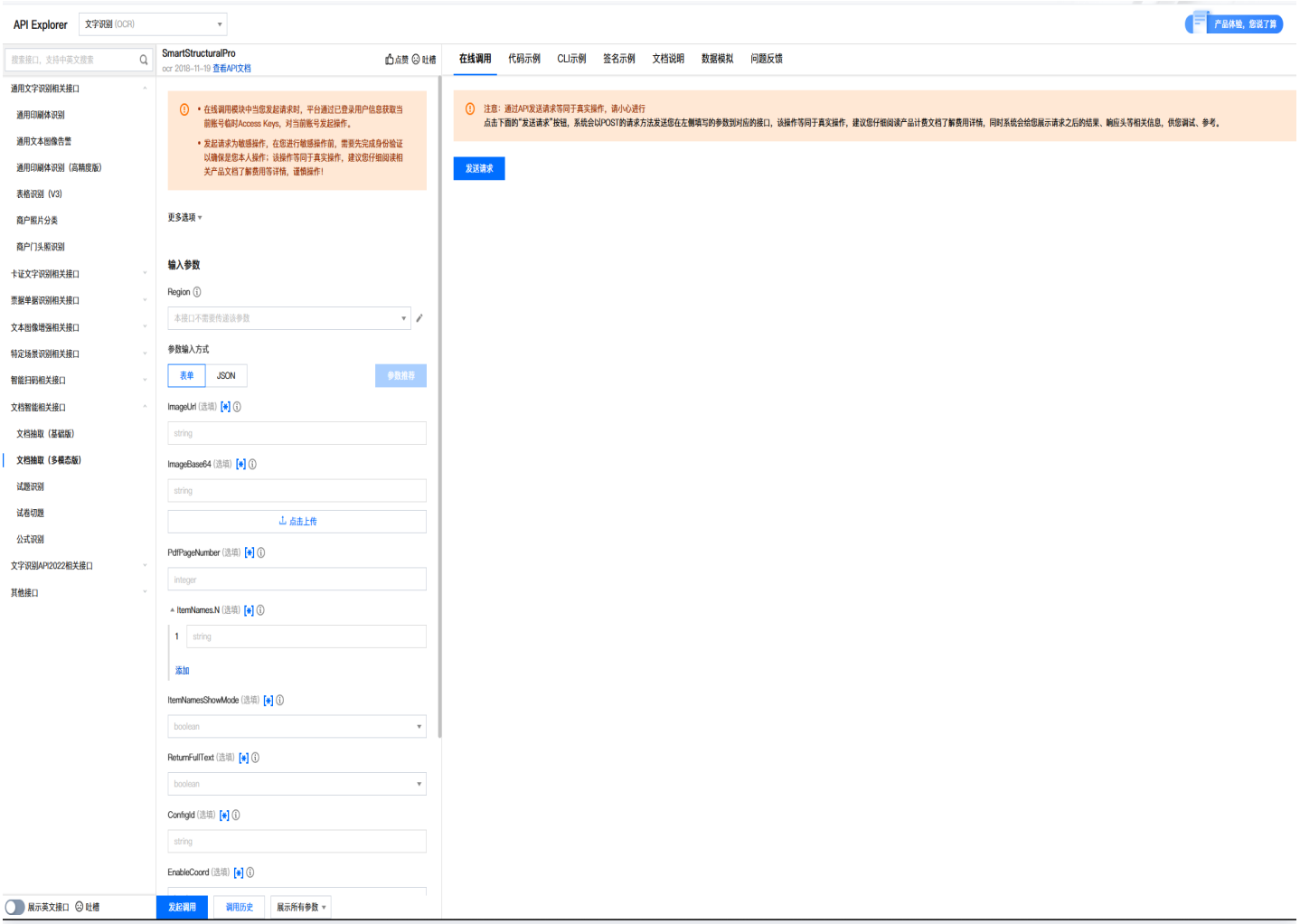
官方文档提供了 demo 体验，可以供大家体验学习。



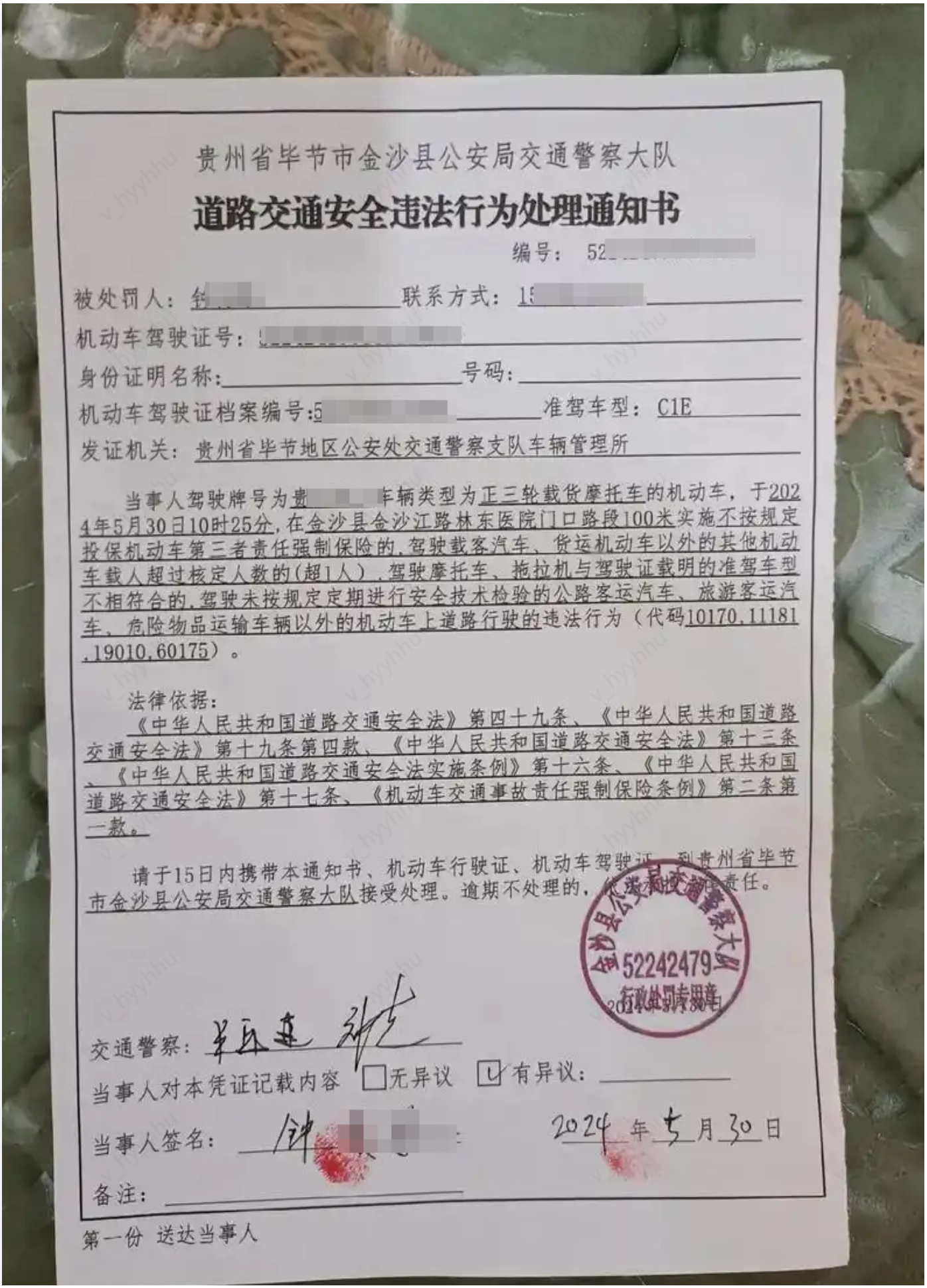
通过 API 3.0 Explorer 进行在线调用

如果您是开发初学者，有代码编写基础，对 HTTP 请求和 API 调用有一定的了解，您可以通过此方式使用文字识别服务。

该方式能够实现在线调用、签名验证、SDK 代码生成和快速检索接口等能力。



1. 选取一张需要识别的票据图片。



贵州省毕节市金沙县公安局交通警察大队

道路交通安全违法行为处理通知书

编号：52[redacted]

被处罚人：钟[redacted] 联系方式：15[redacted]

机动车驾驶证号：[redacted]

身份证明名称：[redacted] 号码：[redacted]

机动车驾驶证档案编号：5[redacted] 准驾车型：C1E

发证机关：贵州省毕节地区公安处交通警察支队车辆管理所

当事人驾驶牌号为贵[redacted]车辆类型为正三轮载货摩托车的机动车，于2024年5月30日10时25分，在金沙县金沙江路林东医院门口路段100米实施不按规定投保机动车第三者责任强制保险的，驾驶载客汽车、货运机动车以外的其他机动车载人超过核定人数的（超1人），驾驶摩托车、拖拉机与驾驶证载明的准驾车型不相符合的，驾驶未按规定定期进行安全技术检验的公路客运汽车、旅游客运汽车、危险物品运输车辆以外的机动车上道路行驶的违法行为（代码10170,11181,19010,60175）。

法律依据：

《中华人民共和国道路交通安全法》第四十九条、《中华人民共和国道路交通安全法》第十九条第四款、《中华人民共和国道路交通安全法》第十三条、《中华人民共和国道路交通安全法实施条例》第十六条、《中华人民共和国道路交通安全法》第十七条、《机动车交通事故责任强制保险条例》第二条第一款。

请于15日内携带本通知书、机动车行驶证、机动车驾驶证到贵州省毕节市金沙县公安局交通警察大队接受处理。逾期不处理的，依法承担责任。

贵州省金沙县公安局交通警察大队 52242479 行政处罚专用章 2024年5月30日

交通警察：[redacted]

当事人对本凭证记载内容 □无异议 ☑有异议：[redacted]

当事人签名：钟[redacted] 2024年5月30日

备注：[redacted]

第一份 送达当事人

2. 上传选取的票据至在线调用区域。

3. 发送调用请求，并获取导数据。

响应结果　响应头　真实请求


```
{
  "Response": {
    "Angle": 0,
    "RequestId": "c38ec4a2-d1f6-4243-91a9-dc88ddfab7d8",
    "StructuralList": [
      {
        "Groups": [
          {
            "Lines": [
              {
                "Key": {
                  "AutoName": "标题",
                  "ConfigName": null
                },
                "Value": {
                  "AutoContent": "贵州省毕节市金沙县公安局交通警察大队道路交通安全违法行为处理通知书",
                  "Coord": {
                    "LeftBottom": {
                      "X": 0,
                      "Y": 0
                    },
                    "LeftTop": {
                      "X": 0,
                      "Y": 0
                    },
                    "RightBottom": {
                      "X": 0,
                      "Y": 0
                    },
                    "RightTop": {
                      "X": 0,
                      "Y": 0
                    }
```


通过代码调用服务 API

如果您是开发工程师，熟悉代码编写，您可以通过腾讯云已编写好的开发工具集（SDK）来调用文字识别服务 API 。SDK 已支持多种语言，包括 Python、Java、PHP、Go、NodeJS、.Net 等。

下面我将以 Node.js 配合HTML，CSS，JavaScript开发一款小工具：

1. 初始化 node 服务项目

   搭建过程可以参考我之前的文章：从零开始：Node.js服务端搭建教程 。

2. 安装相关依赖文件

```
npm install express body-parser tencentcloud-sdk-nodejs-ocr
```

3. 创建服务器文件

修改 index.js 的文件，并添加以下代码：

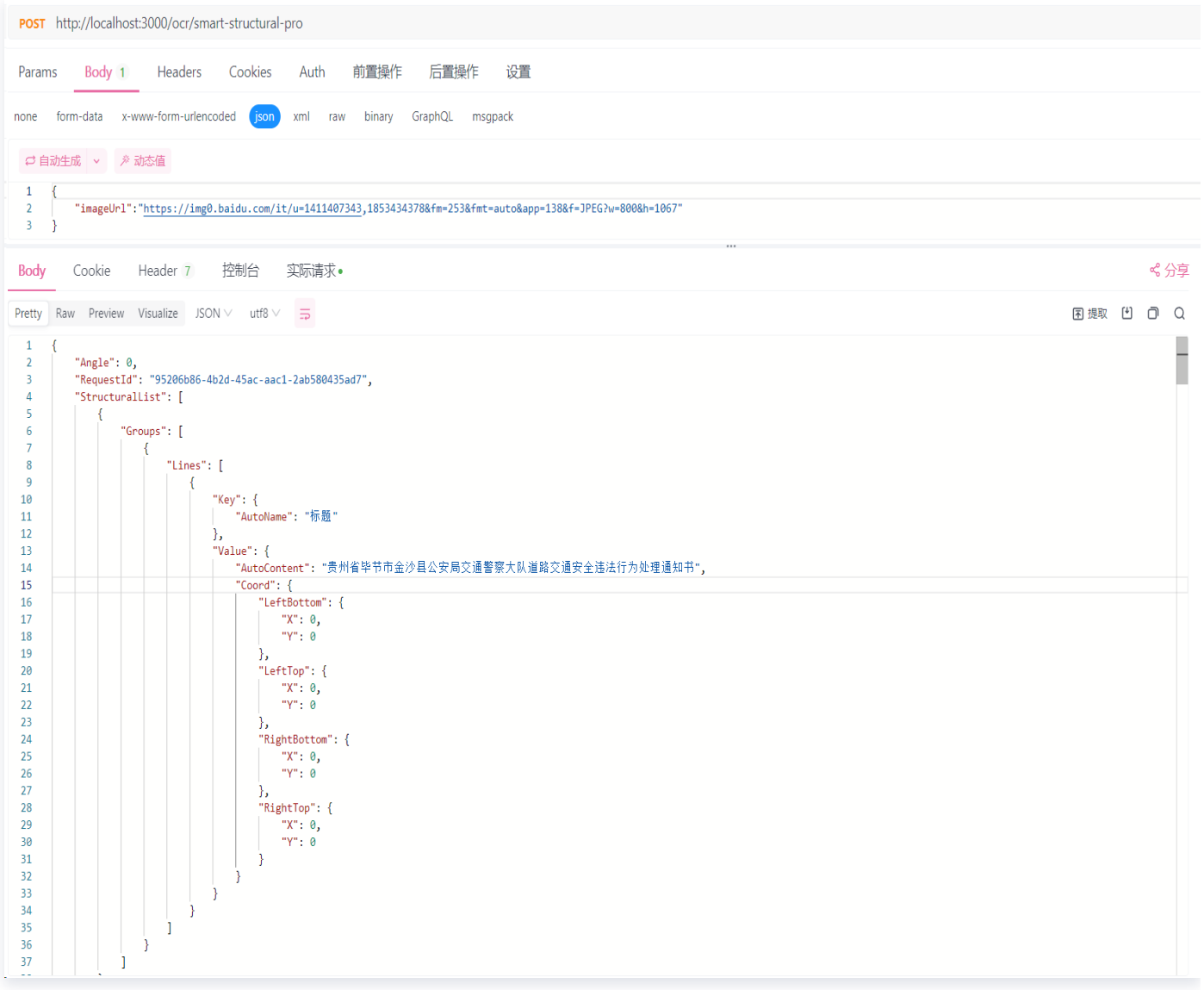
```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const port = 3000;

const ocrRoutes = require('./server.js');


// 使用 body-parser 中间件解析 JSON 请求体
app.use(bodyParser.json());

// 使用 OCR 路由
app.use('/ocr', ocrRoutes);
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

创建一个名为 server.js 的文件，并添加以下代码：

```
// server.js
// src/ocr.js

const express = require('express');
```

```
const router = express.Router();
const tencentcloud = require("tencentcloud-sdk-nodejs-ocr");

const OcrClient = tencentcloud.ocr.v20181119.Client;

const clientConfig = {
  credential: {
    secretId: process.env.SECRET_ID,
    secretKey: process.env.SECRET_KEY,
  },
  region: process.env.REGION || "ap-guangzhou",
  profile: {
    httpProfile: {
      endpoint: "ocr.tencentcloudapi.com",
    },
  },
};

const client = new OcrClient(clientConfig);

router.post('/smart-structural-pro', async (req, res) => {
  try {
    const { imageUrl } = req.body;

    if (!imageUrl) {
      return res.status(400).json({ error: '缺少 imageUrl 参数' });
    }

    const params = {
      ImageUrl: imageUrl,
    };

    const data = await client.SmartStructuralPro(params);
    res.json(data);
  } catch (error) {
    console.error("OCR 请求错误:", error);
    res.status(500).json({ error: 'OCR 请求失败', details:
error.message });
  }
});
module.exports = router;
```

4. 通过 Apifox 工具进行测试。



5. 创建前端界面

为了更方便快捷的使用腾讯 OCR，开发配套的前端界面示例，供大家学习交流，创建界面，未来可以集成在我们的应用软件中。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>OCR Upload</title>
    <style>
    </style>
</head>
<body>
    <div class="container">
```

```
        <h1>Upload Image for OCR</h1>
        <input type="file" id="imageInput" accept="image/*">
        <button onclick="uploadImage()">Upload</button>
        <div id="result"></div>
    </div>
</body>
</html>
```

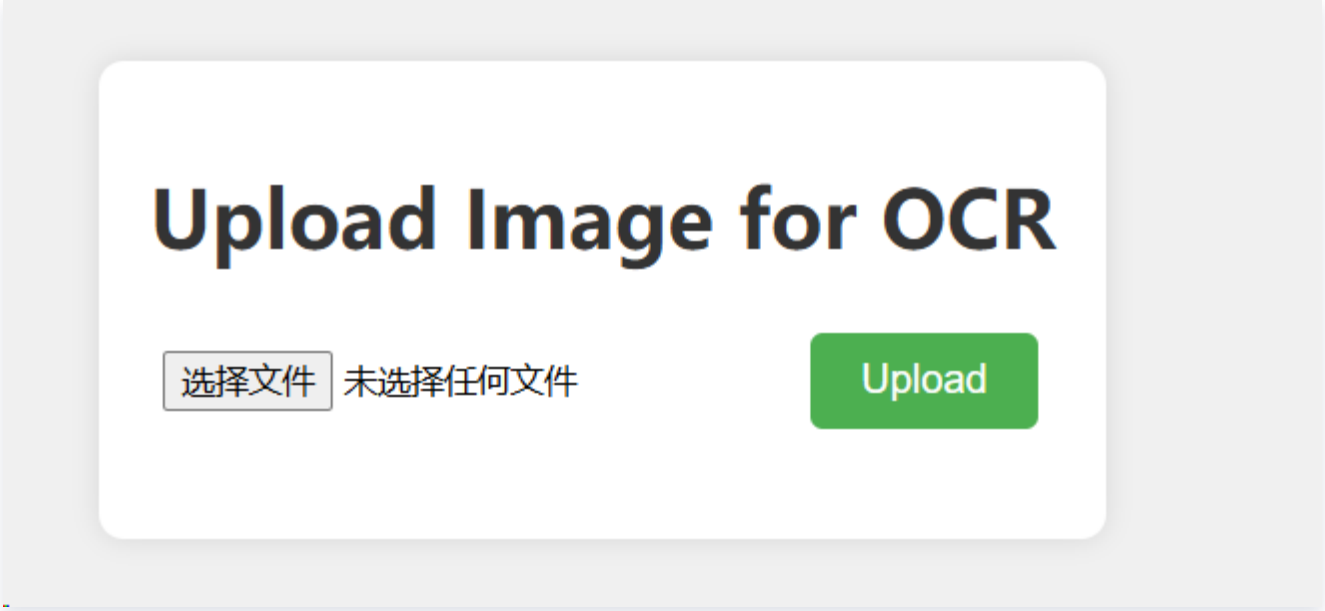
```
body {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f0f0f0;
    margin: 0;
}
.container {
    background-color: #fff;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    text-align: center;
}
h1 {
    color: #333;
}
#result {
    margin-top: 20px;
    text-align: left;
}
.item {
    margin-bottom: 10px;
}
button {
    background-color: #4CAF50;
    color: white;
    border: none;
    padding: 10px 20px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
```

```
    margin: 4px 2px;
    cursor: pointer;
    border-radius: 5px;
}
button:hover {
    background-color: #45a049;
}
input[type="file"] {
    margin-bottom: 10px;
}
```

```
function uploadImage() {
    const input = document.getElementById('imageInput');
    if (input.files.length === 0) {
        alert('Please select an image file.');
        return;
    }
    const formData = new FormData();
    formData.append('image', input.files[0]);
    fetch('http://localhost:3000/ocr/smart-structural-pro', {
        method: 'POST',
        body: formData,
    })
    .then(response => {
        if (!response.ok) {
            throw new Error(`HTTP error! status: ${response.status}`);
        }
        return response.json();
    })
    .then(data => {
        displayResult(data.StructuralList);
    })
    .catch(error => {
        console.error('Error:', error);
    });
    console.log(2);

}
function displayResult(structuralList) {
    const resultDiv = document.getElementById('result');
    resultDiv.innerHTML = ''
    structuralList.forEach(group => {
```

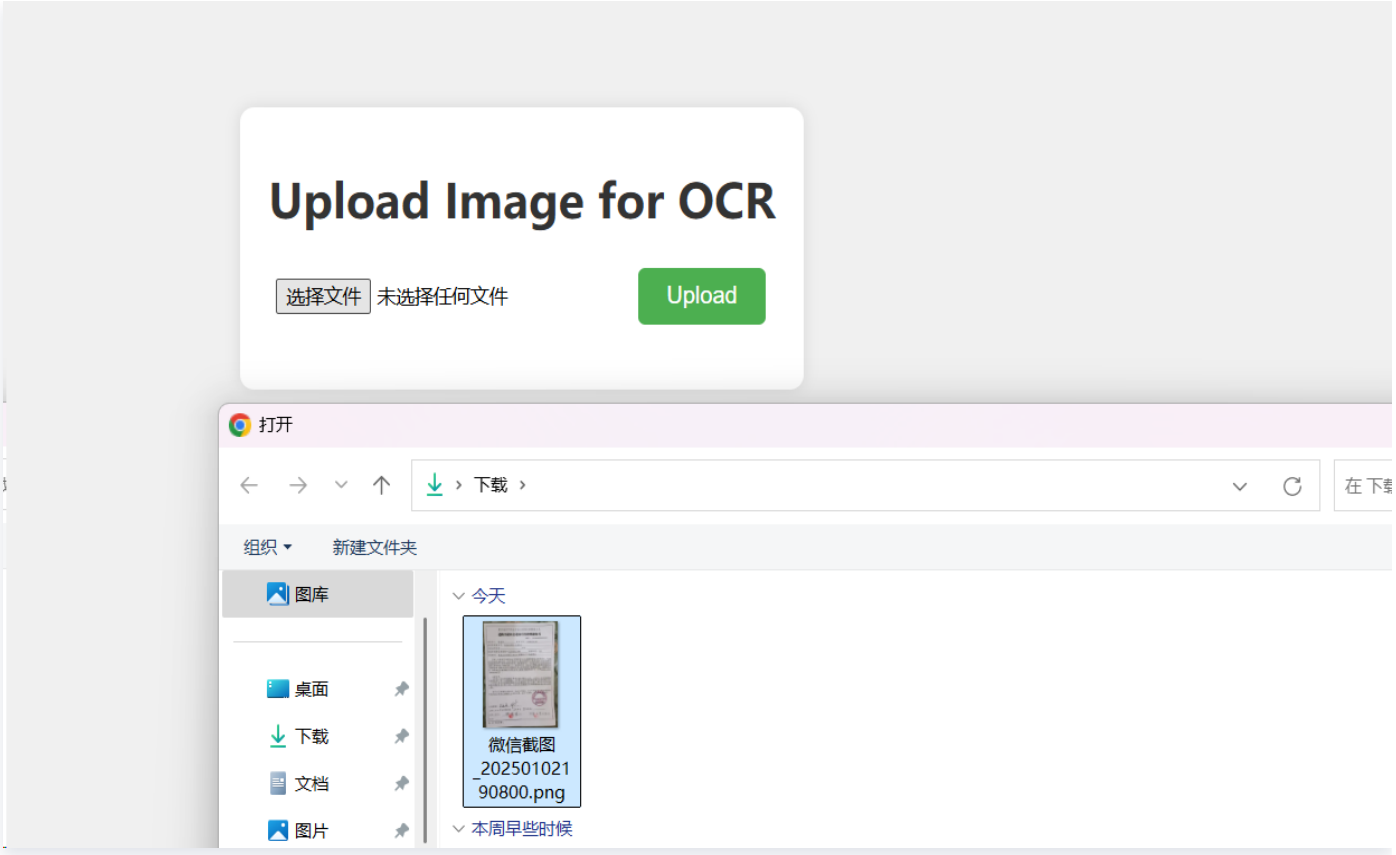
```
        group.Groups.forEach(subGroup => {
            subGroup.Lines.forEach(line => {
                const itemDiv = document.createElement('div');
                itemDiv.className = 'item';
                itemDiv.innerHTML = `<strong>${line.Key.AutoName}:
</strong> ${line.Value.AutoContent}`;
                resultDiv.appendChild(itemDiv);
            });
        });
    });
}
```

界面效果：



6. 上传图片及效果预览

首先选择一张需要识别的图片（以交通领域中道路交通安全违法行为处理通知书为例）；



点击上传等待执行完成，可得到执行结果；

通过测试，识别的效果以及精确程度符合预期，自此，我们完成了腾讯云 OCR 的所有流程，希望对大家有所启发。

三、腾讯云文档智能 OCR 的应用场景

物流行业

在物流行业中，货物的运输和仓储管理需要处理大量的单据和标签。腾讯云文档智能 OCR 技术可以快速准确地识别这些单据和标签上的信息，如发货人、收货人、货物名称、数量、重量等，实现自动化处理，提高物流效率。
例如，某大型物流公司利用腾讯云文档智能 OCR 技术，实现了对运单的自动识别和信息提取。通过 OCR 技术，系统能够快速识别运单上的信息，并将其自动录入到物流管理系统中，大大减少了人工录入的工作量，提高了信息处理的准确性和效率。

XXX物流有限公司托运单

XXX ←→XXXX 专线往返 货物查询电话：XXXXXXX
中转：XXXX-XXXX-XXXX-XXXX-XXXX-XXXX

单号： 202402240001

发站： 徐州　　到站： 深圳　　开票员： 1234　　开票日期： 2024-02-24

发货人	张三	电话	55		身份证号	320555			
收货人	李四	电话	66		地址	深圳某某区			
货物名称		件数	包装	重量kg	体积m3	运费	付款方式	交接方式	转送费
纸箱		4	纸箱	20	50	115	月结	自提	
代收货款大写： 玖佰捌拾元整				¥ 980		'备注	备注信息		
合计金额大写： 壹仟零玖拾伍元整				¥ 1095					
货物丢失赔付金额					温馨提示：				
发货人签字		收货人签字			收货人身份证号				

徐州地址：徐州市XXXXXXX　　电话：XXXXXXXXXX
深圳地址：XXXXXXXXXXXX　　电话：XXXXXXXXXX
广州地址：XXXXXXXXXXXXX　　电话：XXXXXXXXXXX

识别效果：

选择文件 2dfb279c-358…ec55b49b.jpg Upload

货物名称: 纸箱
件数: 4
包装: 纸箱
重量kg: 20
体积m3: 50
运费: 115
付款方式: 月结
交接方式: 自提
标题: XXX物流有限公司托运单
中转: XXXX-XXXX-XXXX-XXXX-XXXX-XXXX
专线往返: XXXXXXX
单号: 202402240001
发站: 徐州
到站: 深圳
开票员: 1234
开票日期: 2024-02-24
发货人: 张三
电话: 55
身份证号: 320555
收货人: 李四
电话: 66
地址: 深圳某某区
代收货款大写: 玖佰捌拾元整
金额: ¥980
合计金额大写: 壹仟零玖拾伍元整
金额: ¥1095
徐州地址: 徐州市XXXXXXX
深圳地址: XXXXXXXXXXXX
电话: XXXXXXXXXXXX

金融行业

在金融行业中，贷款申请、信用卡申请、财务报表等业务需要处理大量的证件和表格。腾讯云文档智能 OCR 技术可以快速准确地识别这些证件和表格上的信息，如申请人信息、财务状况等，实现自动化处理，提高金融效率。

例如，某银行利用腾讯云文档智能 OCR 技术，实现了对贷款申请人的证件信息自动识别。通过 OCR 技术，系统能够快速识别身份证、户口本等证件上的信息，并将其自动录入到银行的贷款申请系统中，大大减少了人工审核的工作量，提高了审批效率和客户满意度。

k 公司资产负债表

20**年 12 月 31 日

资产	2010 年	2009 年	负债及所有者权益	2010 年	2009 年
流动资产：			流动负债：		
货币资金	1 510 000	1 450 000	短期借款	805 000	1 810 000
交易性金融资产	250 000	330 000	交易性金融负债		
应收票据	80 000	90 000	应付票据	230 000	160 000
应收账款	995 000	796 000	应付账款	1 280 000	950 000
预付账款	188 000	168 000	预收账款	140 000	124 000
其他应收款	46 000	30 000	应付职工薪酬	180 000	210 000
存货	2 900 000	2 400 000	应交税费	200 000	180 000
一年内到期的非流动资产			应付股利	100 000	60 000
其他流动资产	75 600	70 000	其他应付款	98 000	93 000
流动资产合计	6 044 600	5 334 000	流动负债合计	3 033 000	3 587 000
可供出售金融资产			长期借款	1 880 000	1 500 000
持有至到期投资			应付债券	1 300 000	1 200 000
长期股权投资	1 000 000	2 000 000	长期应付款		
投资性房地产			长期负债合计	3 180 000	2 700 000
固定资产	10 450 000	9 500 000	负债合计	6 213 000	6 287 000
在建工程			实收资本	9 400 000	8 600 000
工程物资			资本公积	800 000	800 000
固定资产清理			盈余公积	1 600 000	1 500 000
无形资产	450 000	300 000	未分配利润	11 600	47 000
长期待摊费用	80 000	100 000	所有者权益合计	11 811 600	10 947 000

其他非流动资产					
资产总计	18 024 600	17 234 000	负债和所有者权益总计	18 024 600	17 234 000

识别效果（截取部分）：

负债及所有者权益: 应交税费
2010年: 200 000
2009年: 180000
资产: 一年内到期的非运动资产
负债及所有者权益: 应付股利
2010年: 100 000
2009年: 60 000
资产: 其他流动资产
2010年: 75600
2009年: 70 000
负债及所有者权益: 其他应付款
2010年: 98000
2009年: 93000
资产: 流动资产合计
2010年: 6044600
2009年: 5334000
负债及所有者权益: 流动负债合计
2010年: 3033000
2009年: 3 587000
资产: 可供出售金融资产
负债及所有者权益: 长期借款
2010年: 1880000
2009年: 1500000
资产: 持有至到期投资

零售行业

在零售行业中，商品管理、库存管理、销售分析等业务需要处理大量的商品信息和销售数据。腾讯云文档智能 OCR 技术可以快速准确地识别商品标签、销售发票等信息，实现自动化处理，提高零售效率。
例如，某大型超市利用腾讯云文档智能 OCR 技术，实现了对商品标签信息的自动识别和录入。通过 OCR 技术，系统能够快速识别商品上的条形码、名称、价格等信息，并将其自动录入到超市的商品管理系统中，大大减少了人工录入的工作量，提高了商品管理的效率和准确性。

库存管理查询表（自动核算）

查询品名：	打印纸	原库存	3500	入库数量	1500	出库数量	580	现库存	4420

序号	名称	编号	单位	规格	原库存	入库数量	出库数量	现库存	入库时间
1	僧纸				5000	2000	500	6500	
2	笔记本				1000	2000	800	2200	
3	打印纸				3500	1500	580	4420	
4								0	
5								0	
6								0	

识别效果：

序号: 1
名称: 倡纸
原库存: 5000
入库数量: 2000
出库数量: 500
现库存: 6500
序号: 2
名称: 笔记本
原库存: 1000
入库数量: 2000
出库数量: 800
现库存: 2200
序号: 3
名称: 打印纸
原库存: 3500
入库数量: 1500
出库数量: 580
现库存: 4420
序号: 4
现库存: 0
序号: 5

此外，腾讯云文档智能 OCR 还广泛应用于教育、医疗、法律等行业，为各行业提供高效、准确的文档处理和数据提取解决方案。

四、腾讯云文档智能 OCR 的定制化与灵活性

腾讯云文档智能 OCR 产品不仅提供了强大的标准功能，还具备高度的定制化和灵活性，以满足不同行业和客户的个性化需求。

预学习机制

通过预学习机制，腾讯云文档智能 OCR 产品能够根据用户提供的少量样本数据，自动调整和优化识别模型，以适应特定场景和数据类型的需求。这种机制大大降低了定制化开发的成本和时间，使用户能够快速获得符合自身需求的 OCR 解决方案。

自定义模板

用户可以根据实际业务需求，自定义识别模板。通过简单的配置和标注，用户可以实现对特定文档的快速、准确识别。自定义模板功能使得 OCR 产品能够适应各种复杂场景和格式，满足用户的个性化需求。

多语言支持

腾讯云文档智能 OCR 产品支持多种语言的识别，包括中文、英文、日文、韩文等。这使得产品能够广泛应用于全球范围内的文档处理和数据提取场景，满足不同国家和地区用户的需求。

接口与集成

腾讯云文档智能 OCR 产品提供了丰富的 API 接口和 SDK，可以轻松集成到各类业务系统中。用户可以根据自身系统的架构和需求，选择合适的接口和集成方式，实现 OCR 功能与现有系统的无缝对接。

五、腾讯云文档智能 OCR 的安全性与稳定性

在处理敏感和重要的文档数据时，安全性和稳定性是用户最为关心的问题。腾讯云文档智能 OCR 产品在这方面同样表现出色。

数据加密与安全传输

腾讯云文档智能 OCR 产品采用业界领先的数据加密技术，确保在数据传输和存储过程中的安全性。同时，产品还支持 HTTPS 等安全传输协议，进一步保障用户数据的安全。

严格的权限管理

腾讯云文档智能 OCR 产品提供了严格的权限管理功能，用户可以根据自身需求设置不同的访问权限和操作权限，确保数据的安全性和访问的可控性。

合规性与审计

腾讯云文档智能 OCR 产品符合国家相关法规和行业标准的要求，同时提供了丰富的审计日志和监控功能，方便用户对系统的运行状态和安全情况进行监控和审计。

六、OCR技术的创新发展趋势

智能化识别

随着 人工智能 技术的不断发展，OCR 识别技术将更加智能化。未来的 OCR 系统将能够自动识别文档的类型、格式和布局，并根据这些信息进行智能化的处理和分析。此外，OCR 系统还将能够识别不同语言和字体，提高识别的准确性和效率。

深度学习与大数据的结合

深度学习和大数据技术的结合将为 OCR 识别技术带来新的突破。通过深度学习算法对大量的文档数据进行训练和学习，OCR 系统将能够不断优化识别模型，提高识别的准确性和效率。同时，大数据技术将为 OCR 系统提供更丰富的数据资源，支持更复杂的文档处理和数据提取任务。

云化与边缘计算的结合

随着云计算和 边缘计算 技术的发展，未来的 OCR 系统将更加灵活和高效。云化技术将使 OCR 系统能够处理更大规模的文档数据，提供更强大的计算能力和存储能力。而边缘计算技术将使 OCR 系统能够在本地设备上进行部分识

别和处理，提高系统的响应速度和隐私保护能力。

跨领域融合与应用

OCR 技术将与其他领域进行更多的融合和应用。例如，在医疗领域，OCR 技术可以用于病历数据的提取和分析；在教育领域，OCR 技术可以用于试卷的自动批改和成绩录入；在交通领域，OCR 技术可以用于车辆牌照的自动识别和交通管理。

结语

腾讯云文档智能 OCR 技术作为文档处理和数据提取领域的佼佼者，正以其强大的功能、出色的性能和广泛的应用场景引领着新时代的发展。未来，随着技术的不断进步和应用需求的不断扩展，OCR 技术将在更多领域发挥更大的作用，推动数字化转型的进程。

文档智能助力在大规模突发事件背景下社交媒体图片中时间、地点等关键信息的有效提取

最近更新时间：2025-05-27 17:59:32

文档智能的社会效益

在大规模社会性突发事件背景下，由于传播渠道有限和实时性要求，处于困境中的个人往往会在公开社交媒体上发布求助信息；
这种信息的格式通常是：何时何地何人需要何种帮助，
尤其是时间、地点这两个关键信息非常重要，时间可以用来记录存档和判断信息是否过期，假如没有地点信息，就好比消防队员不知道去哪救火、红十字会不知道去哪发放物资...
每逢这种背景，通常有这样的监控系统出现，实时采集各大平台中的相关话题的求助信息，智能解析信息后给予当事人及时的救助。
例如从一段微博文本中，提取时间、地点、人物、联系电话等关键信息，
但是信息不都是纯文本，特别是在这样一个信息过载的时代，越来越多的人选择发图片文字，这样一种更加直观易读的方式。
所以在从文本中提取结构性信息前，还需要增加一步 OCR 处理，一种将输入、手写或印刷体文本从图片转换为机器编码文本的基础技术。
这两个标准步骤，不仅在社会救助实践上大有可为，而且在不少应急管理的科研课题上提供技术支撑。

几年前我如何做内容文档抽取

我几年前的时候，就给老师做过这样一个科研项目，有关 2021 河南暴雨事件，采集一个河南暴雨互助超话下的所有微博，并下载所有图片，然后利用 OCR 提取图片中的数据，交给下游任务处理。
当时好像用的是 Tesseract OCR 框架处理图片转文本，再使用从改编的算法从文本中提取信息，不仅流程长，依赖多，精确率有待提高。

当今腾讯云的智能文档抽取

但是现在是2025年了，刚好看到腾讯云出了文档智能产品，官网是这样宣传的：
具备全面的行业覆盖能力，能精确识别包括卡证、物流单据、工业标签、服务合同及医疗报告在内的多种文件。
听这名字，不仅 OCR，而且文档抽取，敢情是把上面两个步骤一步到位了？
说干就干，于是我参考在腾讯云 OCR 文档，折腾了下，果真如此，特此记录分享出来。

现在的我们可以怎么做智能文档抽取

获取 SecretId 和 SecretKey

为了保护隐私，我将使用下面这个文本图片（demo.jpg）作为演示，提取图片中的时间、地点等结构信息，
文本本身不具有真实性，仅供测试。

2025.1.5

由于魂殿慕骨等人捣乱

星域即将崩溃

主角萧炎被困星域

急需斗尊以上强者相助

圣丹市丹塔县星域镇未名村

事成后将有八品丹药等丰厚报酬

以及一名斗帝强者好友

第一步，需要先在腾讯云开通文档智能服务，开通控制台和文档都在下面这个链接，现在开通还有免费额度赠送，
`https://cloud.tencent.com/product/doc-ai`
按照文档中的最佳实践操作，需要注意的是，为了资源安全，最好开通一个子账号，
因为腾讯云所有的产品默认都可以由主账号的一个 `SecretId` 和 `SecretKey` 控制，这很危险，不小心被别人拿到开通各种付费服务就麻烦了。

所有推荐创建一个子账号，然后只授权这个子账号 `QcloudOCRReadOnlyaccess` 这一个文字识别只读访问权限，只允许访问文字识别所有读接口，包含调用所有 OCR 接口、允许查看所有账号用量等；
在下面这个页面可以开通子账号：
`https://console.cloud.tencent.com/cam`
授权后在这个子账号下新建一个 API 密钥，注意在新建时保存 `SecretId` 和 `SecretKey`，`SecretKey` 只在创建时可见可复制。
拿到 `SecretId` 和 `SecretKey` 后，第一步就算完成了。

使用 Python 请求服务

首先安装腾讯云最新版本的 Python SDK：

```
pip install --upgrade tencentcloud-sdk-python
```

然后使用下面这份代码，坑我踩了，0 报错 0 警告，拿去用即可。

```
// step 1 初始化 SDK Auth Client
def initClient():
    # 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注意密钥对的保密
    # 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全性。
    cred = credential.Credential("替换成您自己的 SecretId", "替换成您自己的 SecretKey")
    # 实例化一个http选项，可选的，没有特殊需求可以跳过
    http_profile = HttpProfile()
    http_profile.endpoint = "ocr.tencentcloudapi.com"

    # 实例化一个client选项，可选的，没有特殊需求可以跳过
    client_profile = ClientProfile()
    client_profile.httpProfile = http_profile
    # 实例化要请求产品的client对象,clientProfile是可选的
    client = ocr_client.OcrClient(cred, "", client_profile)
    return client
```

由于调用服务需要把二进制图片转成 Base64 字符串，下面是两个工具函数。

```
def getFileContent(file_path):
    with open(file_path, 'rb') as fp:
        return fp.read()
```

```
def imageToBase64(image_path):
    # 读取图片文件的二进制数据

    binary_data = getFileContent(image_path)
    # 使用 base64 编码二进制数据
    base64_encoded_data = base64.b64encode(binary_data)
    # 将 base64 编码的二进制数据转换为字符串
    base64_message = base64_encoded_data.decode('utf-8')

    return base64_message
```

构建文档抽取（多模态版）请求，并且获得响应：

```
def buildTencentSmartStructuralOCRV2(image_file_path):
    try:
        client = initClient()
        # 实例化一个请求对象,每个接口都会对应一个request对象
        req = models.SmartStructuralOCRV2Request()
        params = {
            "ImageBase64": imageToBase64(image_file_path),
            "ItemNames": ["日期", "地点"],
            "ReturnFullText": False
        }
        req.from_json_string(json.dumps(params))

        # 返回的resp是一个SmartStructuralOCRV2Response的实例，与请求对象对应
        resp = client.SmartStructuralOCRV2(req)
        # 输出json格式的字符串回包
        print(resp.to_json_string())
        return json.loads(resp.to_json_string())

    except TencentCloudSDKException as err:
        print(err)
        return None
```

参数 `ItemNames` 指明了我们需要解析的文本字段列表，您也可以尝试更多的一些字段，
SDK 给我们的响应输出是一个字符串，在此我把它转成了 JSON 格式，方便后续的响应解析，获得我们目标数据。
传入上文的测试文本图片 demo.jpg，输出内容如下：

```
{
    "Angle": -0.02881504036486149,
```

```
    "StructuralList": [
        {
            "Groups": [
                {
                    "Lines": [
                        {
                            "Key": {
                                "AutoName": "日期",
                                "ConfigName": null
                            },
                            "Value": {
                                "AutoContent": "2025年1月5日",
                                "Coord": {
                                    "LeftTop": {
                                        "X": 457,
                                        "Y": 86
                                    },
                                    "RightTop": {
                                        "X": 774,
                                        "Y": 85
                                    },
                                    "RightBottom": {
                                        "X": 774,
                                        "Y": 157
                                    },
                                    "LeftBottom": {
                                        "X": 457,
                                        "Y": 158
                                    }
                                }
                            }
                        }
                    ]
                }
            ]
        },
        {
            "Groups": [
                {
                    "Lines": [
                        {
                            "Key": {
                                "AutoName": "地点",
```

```
                            "ConfigName": null
                        },
                        "Value": {
                            "AutoContent": "圣丹市丹塔县星域镇未名村",
                            "Coord": {
                                "LeftTop": {
                                    "X": 249,
                                    "Y": 956
                                },
                                "RightTop": {
                                    "X": 1181,
                                    "Y": 955
                                },
                                "RightBottom": {
                                    "X": 1181,
                                    "Y": 1035
                                },
                                "LeftBottom": {
                                    "X": 249,
                                    "Y": 1036
                                }
                            }
                        }
                    }
                ]
            }
        ]
    }
    ],
    "WordList": [],
    "RequestId": "月小水长 buyixiao 的 RID"
}
```


Angle 是图片相对于水平的旋转角度，文本的水平方向为 0，顺时针为正，逆时针为负；
AutoName 是我们传入的待解析字段，对应的AutoContent 就是解析的字段值；
X 和 Y 就是文本在图片中的位置边框，其四个点的坐标。
所以下面的我们的目标就是解析所有的 AutoContent了。

解析响应获取目标结果

看起来这个 json一点也不规则，不能直接清晰地通过下标或者键名获取想要的结果，
当然可以选择诸如 `json["StructuralList"][0]["Groups"][0]["Lines"]` 之类的 ugly 写法，但是这样看起来不易读不好维护，而且不一定通用，

如果您读过我在云社区写的这篇文章：工程实践善用简单算法，事半功倍。
就能清楚的知道，本文解析任务也类似，都可以通过设计递归算法简化之，
递归解析引擎我写好了，测试通过，拿去用即可。

```
def parseEntity(resp_json):
    entity_collected = {}

    def recurse_seek(obj):
        if isinstance(obj, dict):
            if "Key" in obj.keys() and "AutoName" in obj["Key"] and \
                    "Value" in obj.keys() and "AutoContent" in
obj["Value"]:
                entity_collected[obj["Key"]["AutoName"]] = obj["Value"]
["AutoContent"]
            else:
                for key, value in obj.items():
                    if isinstance(value, list):
                        for vv in value:
                            recurse_seek(vv)
        elif isinstance(obj, list):
            for item in obj:
                recurse_seek(item)

    if resp_json:
        recurse_seek(resp_json)

    return entity_collected
```

最终输出的结果如下：

```
{'日期': '2025年1月5日', '地点': '圣丹市丹塔县星域镇未名村'}
```

运行代码的注意事项
新建一个 py 文件，
1. 需要导入的包如下，复制到文件最前面。

```
import base64
import json
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
```

```
from tencentcloud.common.exception.tencent_cloud_sdk_exception import
TencentCloudSDKException
from tencentcloud.ocr.v20181119 import ocr_client, models
```

2. 接着复制上面所有的工具函数、请求函数、解析函数（json 输出不要复制了）。
3. 最后复制调用入口，demo.jpg 放到和 py 文件一个目录下，就能完整运行起来了。

```
if __name__ == '__main__':
    response_json =
buildTencentSmartStructuralOCRV2(image_file_path='./demo.jpg')
    entity_dict = parseEntity(response_json)
    print(entity_dict)
```

总结

腾讯云文档智能给 OCR 提取和结构信息提取提供了新的选择，而且产品和文档都做的比较好，有需要的朋友们可以动手试试，我仅小小抛砖引玉了下，其实这个产品还有其他更多强大功能可以探索，在大模型能力的加持下，想象空间很大，我在官方文档找到了如下表述：
即便在版式多变或中英文混排的情形下，仍可维持高识别精度。
借助多模态大模型技术构建键值对应关系，支持客户个性化模板定制，提升数据提取录入效率，
适用于政务处理、票据核销、行业表单填写、国际物流管理、人寿保险理赔、AI 在线问诊、律师事务所合同审查及供应链合同审核等多种应用场景。

所以，海阔天空，不妨一试~

文档智能|OCR 助力 OA，如何实现报销两天到账

最近更新时间：2025-05-27 17:59:32

前言

第一次接触 OCR，还是在18年刚毕业的时候。那时候热衷于爬虫，在爬取数据的过程中总会遇到形形色色的验证码问题。虽然有很多打码平台可以解决这个问题，但是我还是趁着这个机会去学习了 OCR 的知识。

OCR 应用

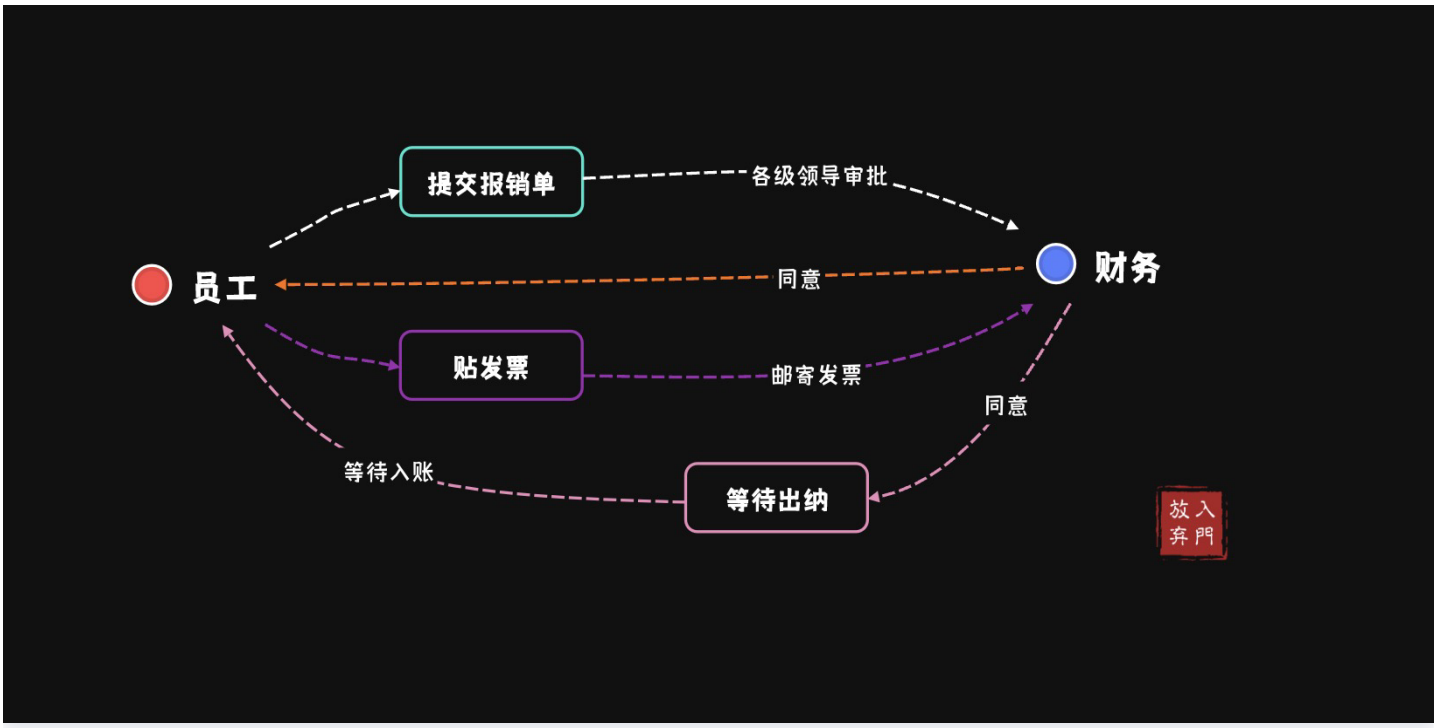
时隔多年，再来回顾 OCR ，当时是使用 PIL 和 pytesseract 来简单的实现图片文字的识别。

```
from PIL import Image
import pytesseract

# 如果 Tesseract 未配置在系统路径中，指定其路径
# pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-
OCR \tesseract.exe'
# 打开图片
image = Image.open("example_image.png")
# 使用 Tesseract  OCR  进行文字识别
text = pytesseract.image_to_string(image, lang="eng")

print("识别结果:", text)
```

以上代码仅仅是对于文字的识别，而对于发票这种结构化的识别，仅仅这样是完成不了的。所以后来在OA系统的财务报销模块改造升级的时候，果断使用了第三方的接口来实现发票的 OCR 。

OA 报销流程

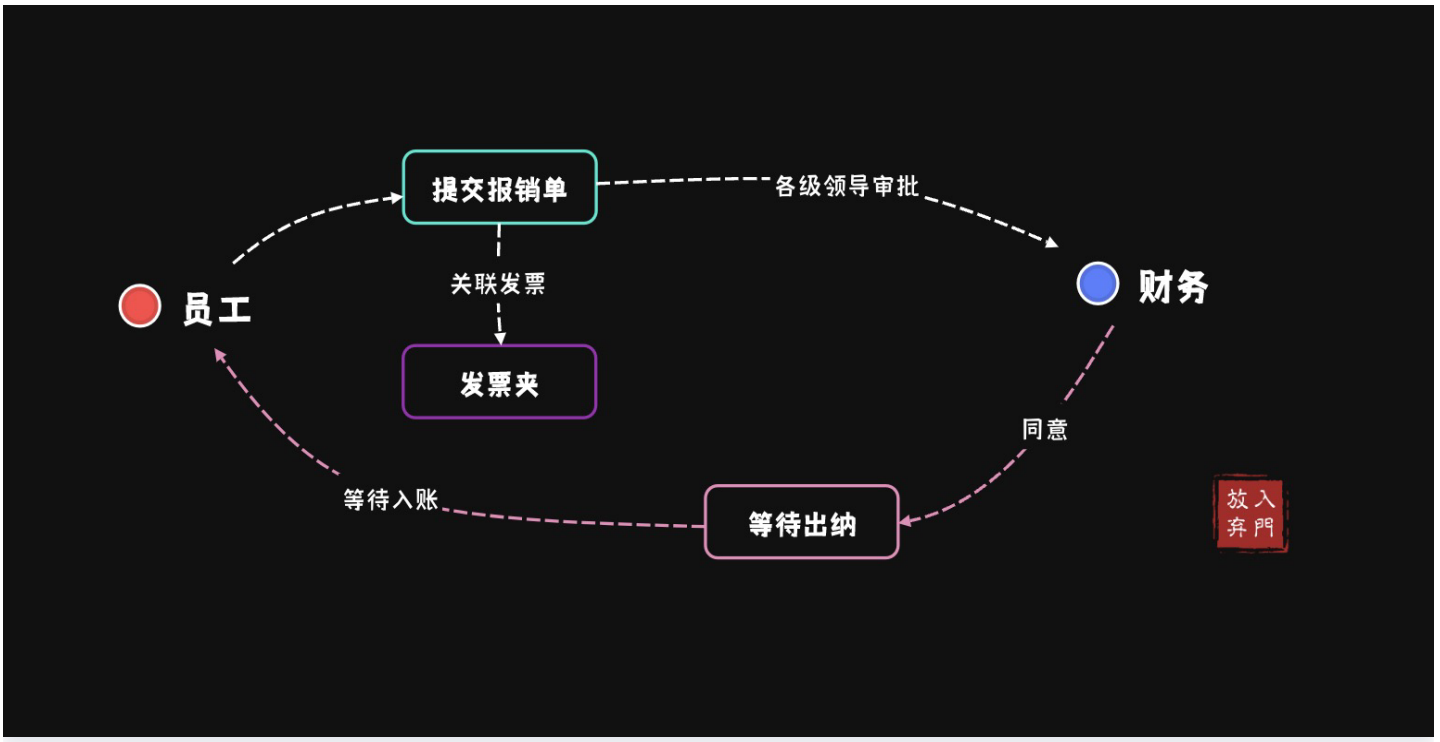
最初公司在 OA 上的报销流程甚是复杂。首先我们需要从 OA 上提交报销工单，除了需要填写项目编号之外，就是填写报销金额，这个金额是需要根据手里需要报销的发票（电子发票或者纸质发票）自行计算。然后经过各级领导审批之后，财务在最终确认报销单，然后进入到贴发票环节。
将财务确认的报销单打印签字之后，需要将一张张发票用胶水粘到一张A4纸上，电子发票需要打印在纸上再粘，然后与报销单一起交给前台，等到到固定日期统一邮寄到总公司的财务处。然后财务收到您的报销单和发票之后进行核对校验，审核通过之后再等待财务出纳日，报销款才能到我的工资卡。
如果发票贴错了，或者报销金额与发票金额不对等怎么办？那么恭喜您，财务不仅会把发票寄还给您，您还要重新贴好发票或者修改报销单之后，等待前台邮寄、财务审核...



我当时经常加班打车，出租车票经常一攒一大堆，然后就用计算器挨张累加金额，反复计算好几次，生怕填错了，电子发票也经常搞得混乱不堪，所以每次报销对于我来说都是很累的事情。

OCR 在 OA 中的应用

后来，公司推行各种智能化，包括商旅、打车等，OA 接入订票平台和打车平台等，这样出差和打车就不用员工自己垫付。然后顺便也把报销平台进行了改造。OA 新增了发票夹功能。
将电子发票或者纸质发票的照片上传到发票夹，然后调用 OCR 接口，识别出来发票金额等信息之后，我们在提交报销单的时候，就能直接关联发票，同时发票金额也是自动计算填入报销单，再也不用我们自己再去挨个计算。



同时，报销平台除了接入 OCR，同时也接入了发票核验功能，报销单和电子发票也不用打印出来邮寄，全程实现了无纸化，报销周期也从之前的两个星期缩减到了现在的2天，极大提高了报销效率。
在之前 Tessera 的 OCR 代码中，只能对纯文本进行识别。如果想要识别发票这种结构化的表格信息，需要结合区域检测或专用 OCR 工具，所以本篇文章主要是对腾讯云的 OCR 产品 文档智能 的发票 OCR 能力来做一个深入探究。使用腾讯云文档智能来实现OA中的发票夹，并验证识别准确性是否满足 OA 发票识别的需求。

文档智能

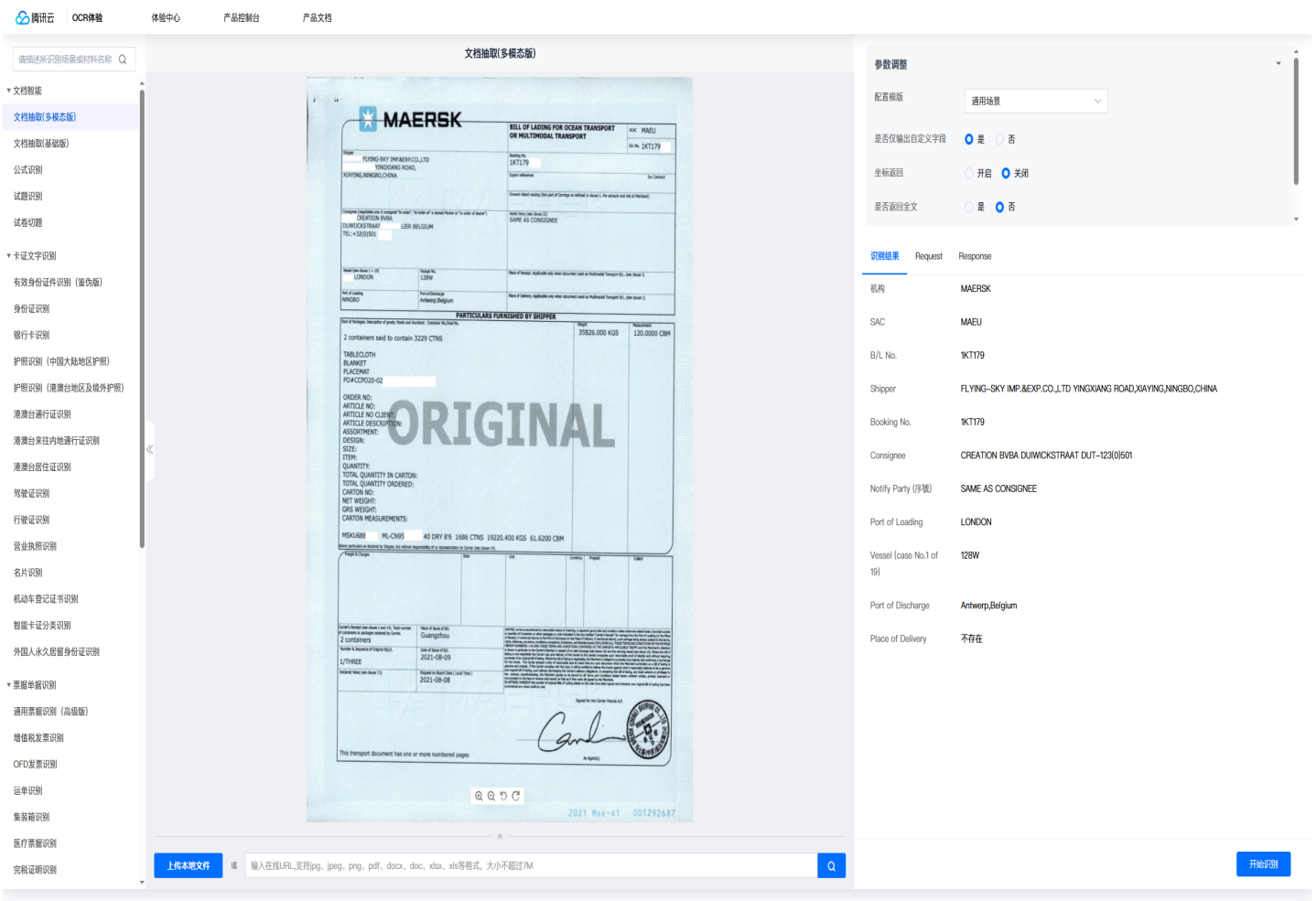
文档智能（Document AI）融合了业界领先的深度学习技术、图像检测技术以及 OCR 大模型能力，能够实现不限版式的结构化信息抽取。支持智能提取各类证照、票据、表单、合同等结构化场景的 key:value 字段信息，并支持提取表格信息的 key:value 组的结构化。
在产品官网/文档： 文档智能 中，我们可以通过点击页面的开通服务按钮，开通文档智能的服务。

开通之后，进入 OCR 控制台，在资源包管理菜单下，我们可以看到腾讯云赠送的文档抽取（基础版）和文档抽取（多模态版）各1000次的调用次数。

除了每个月会免费赠送文档抽取的资源包之外，还会赠送银行卡识别、英文识别的资源包等。

产品体验

腾讯云提供了文档抽取（多模态版）的产品 demo 体验，点击 OCR Demo，在产品 demo 中识别的是国际物流的单据，可以看到精准识别了物流单据中的结构化信息。



我们也可以上传一些本地的单据，点击开始识别就可以调用 OCR 能力，有兴趣的小伙伴可以试用一下。

版本对比

文档智能一共提供了基础版和多模态版的识别接口调用。

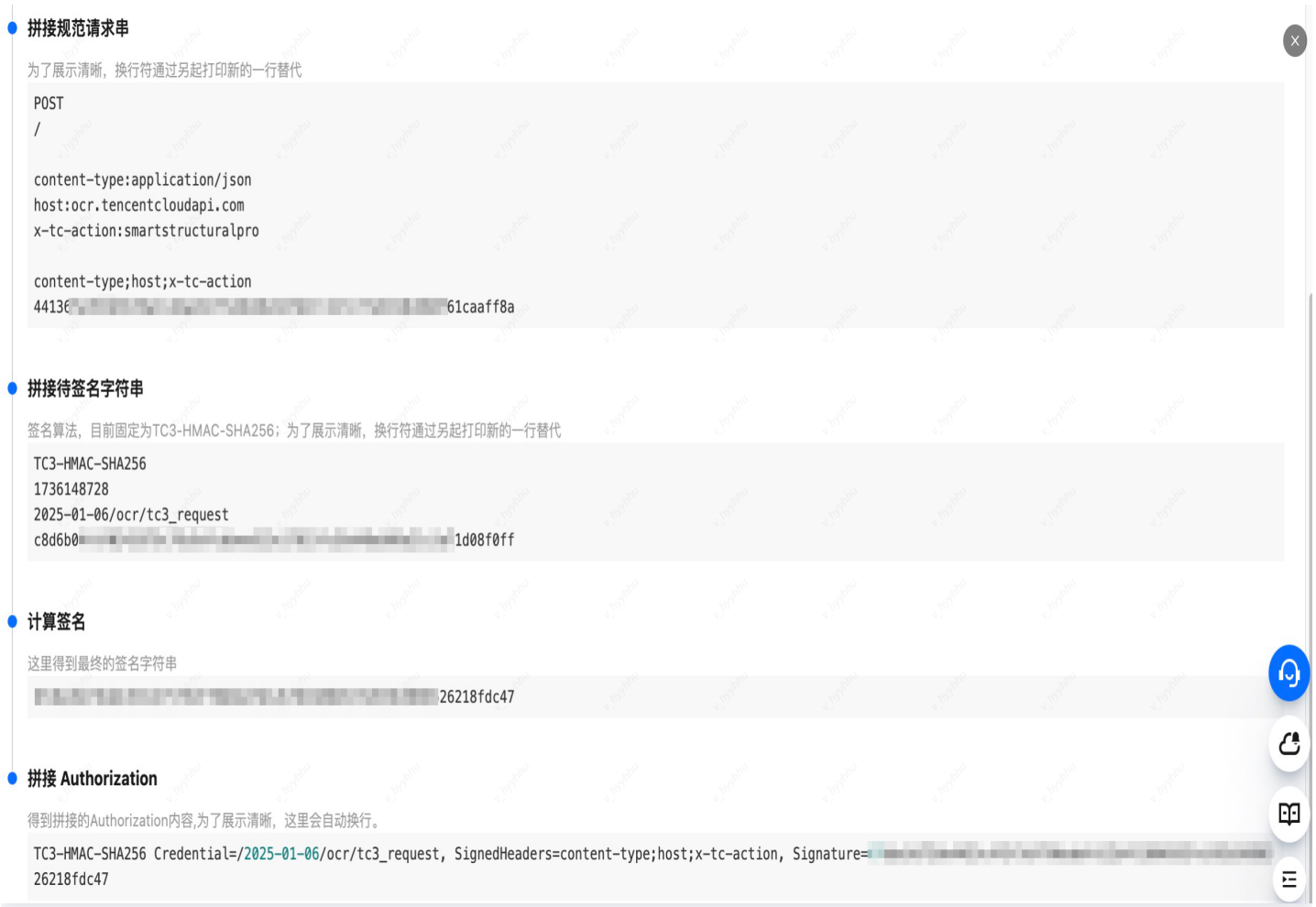
文档抽取（多模态版）不限定版式，支持海外票据、提单、运单、进出口报关单、装箱单、过磅单、采购单等物流单据，识别率高且有更大的模型参数算力支撑。
而基础版只支持固定版式的识别，例如网约车行程单、驾驶证、运输证、房产证、不动产证、毕业证等各类证件单据，这种固定版式识别相对于简单，这也意味着相对有高级版来说识别速度更快。
我在这里选择了文档抽取（多模态版），用来开发发票夹，在开通和体验完文档抽取之后，就开始发票夹应用的开发环节。

后端开发

在开发之前首先要确定开发架构，考虑到想要做一个多端的适配，且需要对用户的发票和请求做一个管理，所以这里我选择使用 uni-app 为开发底座，vue3+springBoot 前后端的架构来实现这个发票夹。

接口调用

腾讯云产品提供了两种接口调用的方式。一是自己是实现接口请求，但是需要使用 签名方法v3 生成请求凭证。
第二种方式是使用腾讯云提供的 tencentcloudapi SDK，这样就不要自己封装签名，通过 SDK 就可以直接调用文档抽取（多模态版）的接口。




```
<dependency>
    <groupId>com.tencentcloudapi</groupId>
    <artifactId>tencentcloud-sdk-java</artifactId>
    <version>3.1.322</version>
</dependency>
```

Service

初始化客户端

实现 OcrService，初始化 tencentcloudapi 的请求客户端 CommonClient，代码如下：

```
public class OcrService {
    private final CommonClient client;

    public OcrService(OcrConfig ocrConfig) {
        // 实例化一个认证对象
        Credential cred = new Credential(ocrConfig.getSecretId(),
ocrConfig.getSecretKey());
```

```
        // 实例化一个 HTTP 选项
        HttpProfile httpProfile = new HttpProfile();
        httpProfile.setEndpoint(ocrConfig.getEndpoint());

        // 实例化一个客户端配置对象
        ClientProfile clientProfile = new ClientProfile();
        clientProfile.setHttpProfile(httpProfile);

        // 实例化要请求产品的client对象
        this.client = new CommonClient("ocr", "2018-11-19", cred,
ocrConfig.getRegion(), clientProfile);
    }
}
```

在初始化请求的时候，除了要填写必要的请求参数，还要设置访问密钥：SecretId和SecretKey。在控制台 就可以生成密钥。

构造 OCR 请求

调用文档抽取（多模态版）接口时，可以将图片上传到公网上，然后使用 ImageUrl 数调用接口。我使用的是第二种方法，现将图片转换成 Base64 格式，然后通过 ImageBase64 将图片发送到接口。

```
public String recognizeInvoice(String imageBase64) throws
TencentCloudSDKException {
    try {
        // 构造请求参数
        String params = String.format("{\"ImageBase64\":\"%s\"}",
imageBase64);
        // 发起请求
        log.info("开始调用OCR识别服务...");
        String response = client.call("SmartStructuralPro", params);
        log.info("OCR识别完成：{}", response);

        return response;

    } catch (TencentCloudSDKException e) {
        log.error("OCR识别失败：{}", e.getMessage(), e);
        throw e;
    }
}
```

核心代码就是使用 client.call 调用文档抽取（多模态版）接口，然后将接口的响应原样返回。

Controller

实现 Controller 对外暴露 /api/ocr/recognize 接口，通过调用service的方法，实现前端页面与高级版 OCR 接口的连通。

```
@Slf4j
@RestController
@RequestMapping("/api/ocr")
@CrossOrigin
public class OcrController {
    private final OcrService ocrService;

    public OcrController(OcrService ocrService) {
        this.ocrService = ocrService;
    }

    @PostMapping("/recognize")
    public OcrResponse<String> recognizeInvoice(@RequestBody OcrRequest
request) {
        try {
            log.info("收到OCR识别请求，图片大小：{} bytes",
request.getImageBase64().length());
            String response =
ocrService.recognizeInvoice(request.getImageBase64());
            return OcrResponse.success(response);
        } catch (TencentCloudSDKException e) {
            log.error("OCR识别失败：{}", e.getMessage(), e);
            return OcrResponse.error(e.getMessage());
        }
    }
}
```

至此，后端开发接口服务已经开发完成。接下来就是开发前端页面，与将后端接口集成到前端中实现发票的 OCR 调用。

前端开发

前端使用 uni-app 和 vue3 完成开发，这里以 OA 系统中发票夹为中心进行开发，实现用户上传发票、识别发票和保存发票的功能，详细功能点如下：

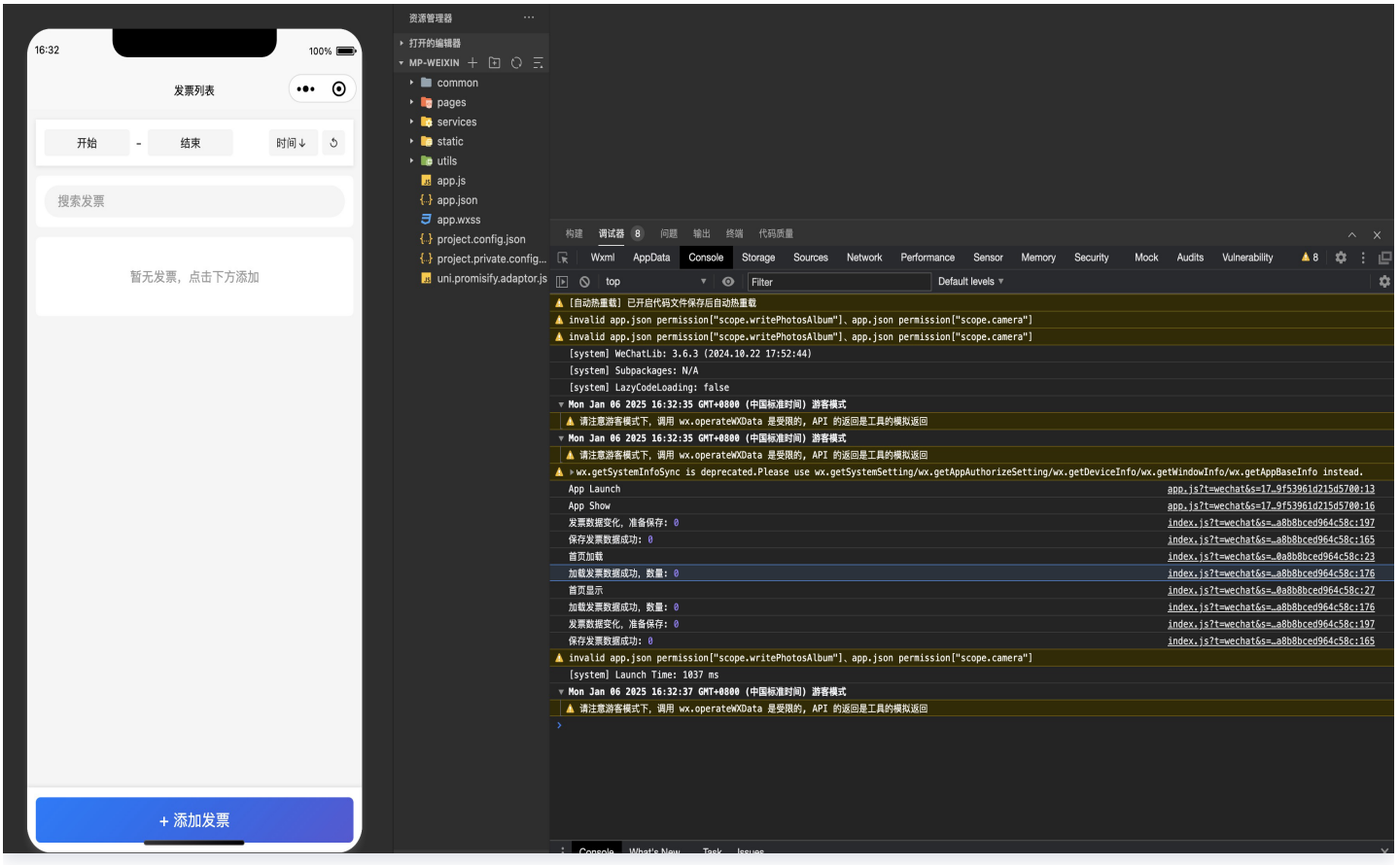
1. 发票 OCR 识别；
2. 发票导入重复检查；
3. 发票列表展示、详情展示；

4. 发票排序；

5. 发票删除；

6. 发票搜索。

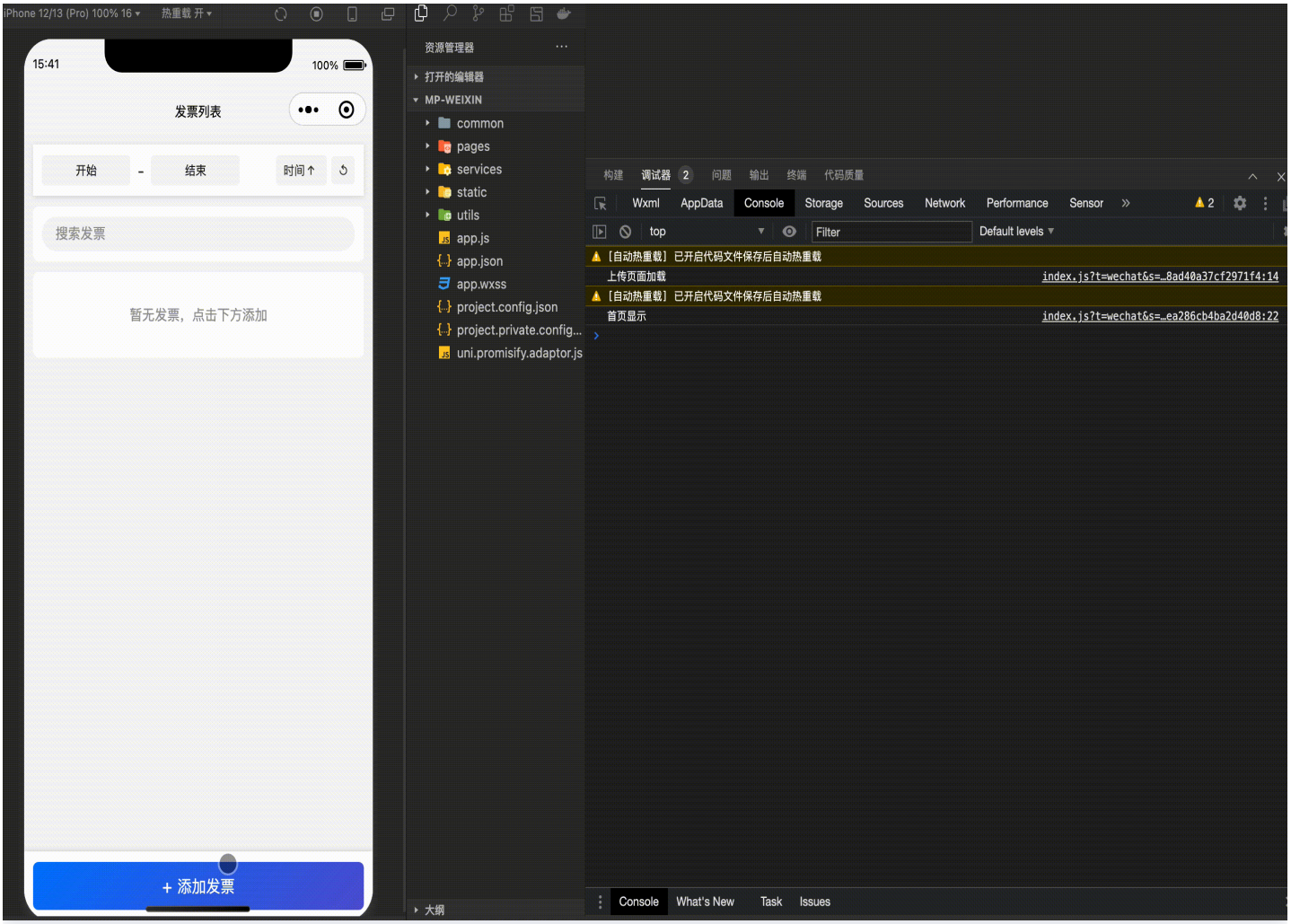
在发票夹的首页，实现了发票列表展示、筛选、搜索、删除的功能，同时也是用户添加发票的入口。



员工需要先添加发票，才能实现筛选、搜索、删除等功能，所以这里我们先看看添加、实现发票的功能是如何实现的，然后再探索其他的功能模块。

添加发票

点击添加发票按钮，会弹出拍照/相册选择的提示框，这里点击相册，就会跳转到发票添加页面。



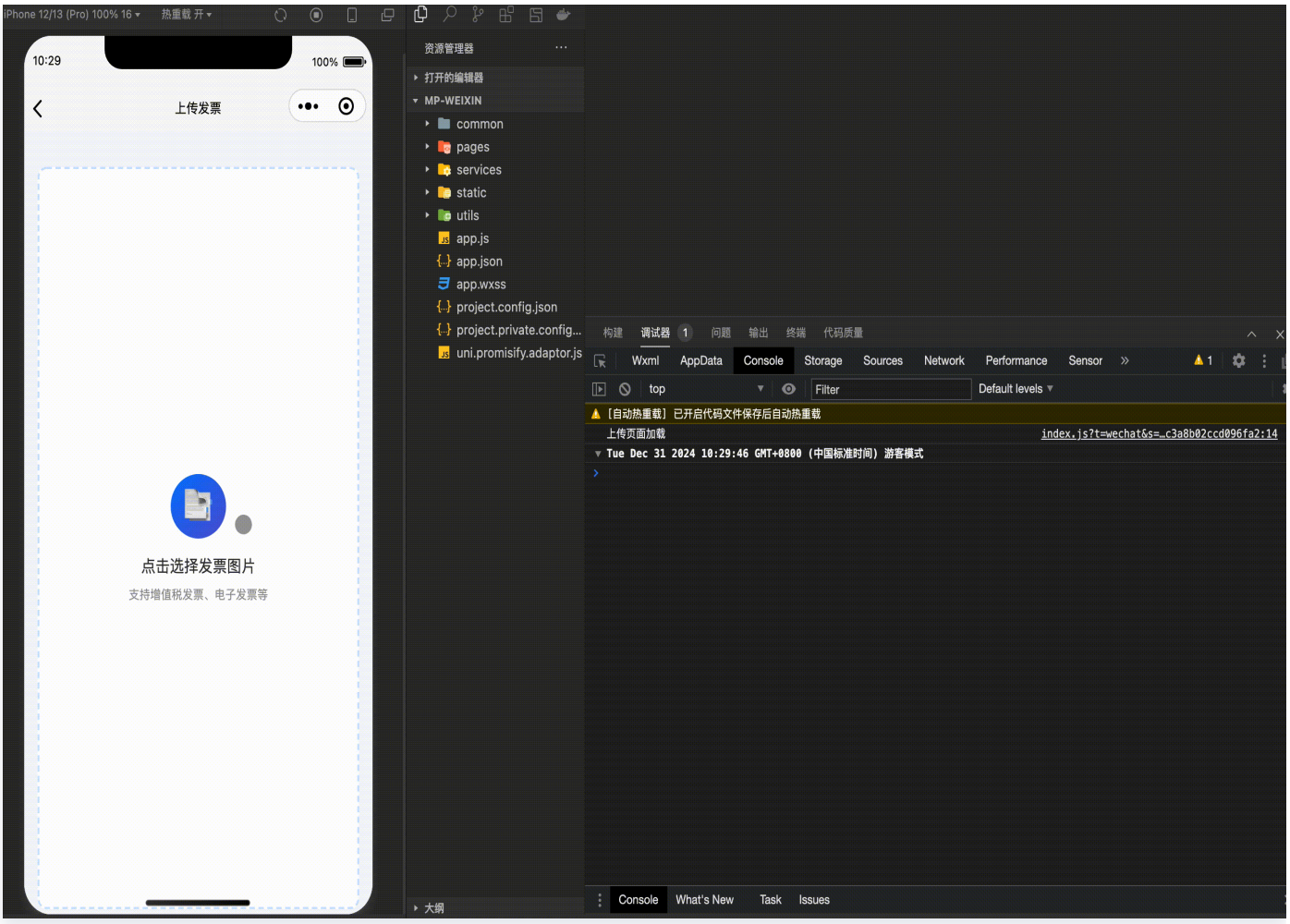
发票上传页面的实现就是简单的布局：

```
<template>
  <view class="container">
    <!-- 选择图片前的提示界面 -->
    <view class="upload-area" v-if="!tempImage" @tap="chooseImage">
      <view class="upload-icon">
        <text class="iconfont">📄</text>
      </view>
      <text class="upload-text">点击选择发票图片</text>
      <text class="upload-tip">支持增值税发票、电子发票等</text>
    </view>

    <!-- 图片预览和确认界面 -->
    <view class="preview-area" v-else>
      <image
        class="preview-image"
        :src="tempImage"
        mode="aspectFit"
      />
      <view class="action-buttons">
        <button
          class="action-btn cancel"
          @tap="cancelUpload"
        >重新选择</button>
        <button
          class="action-btn confirm"
          @tap="confirmUpload"
          :loading="loading"
        >{{ loading ? '识别中...' : '确认上传' }}</button>
      </view>
    </view>
  </view>
</template>
```

而页面除了需要实现 OCR 发票识别功能，还需要实现发票图片上传、发票查重、预览等功能。

1. 发票上传

点击页面，进入到相册选择发票。



选择发票之后，点击确认上传，就会调用 OCR 接口上传。

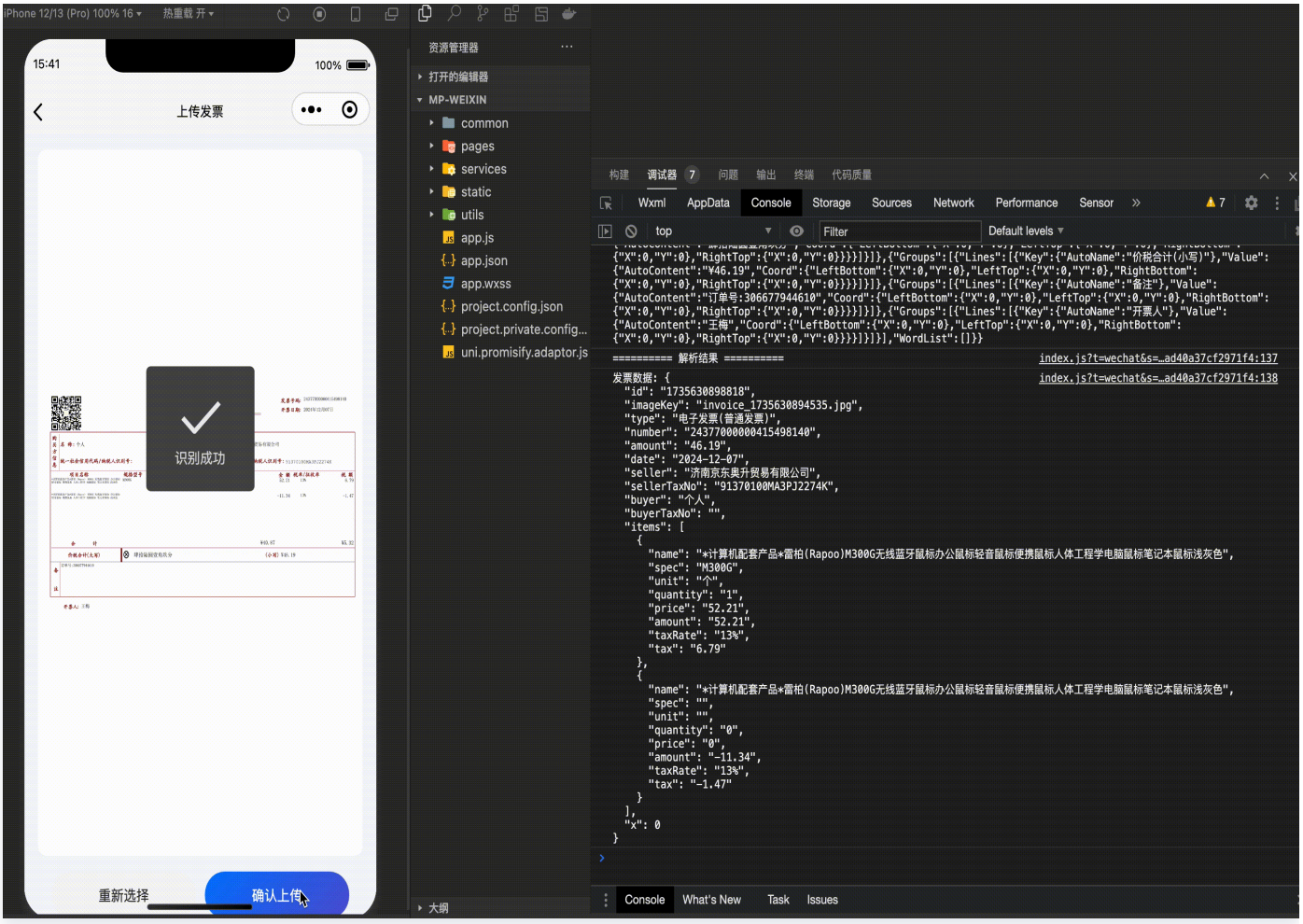
2. 电子发票识别

在点击确认上传之后，开始触发 OCR 识别，这时候就会调用后端服务中 /api/ocr/recognize 接口对发票进行识别。这里封装了 call OCR API 方法来实现接口的调用。

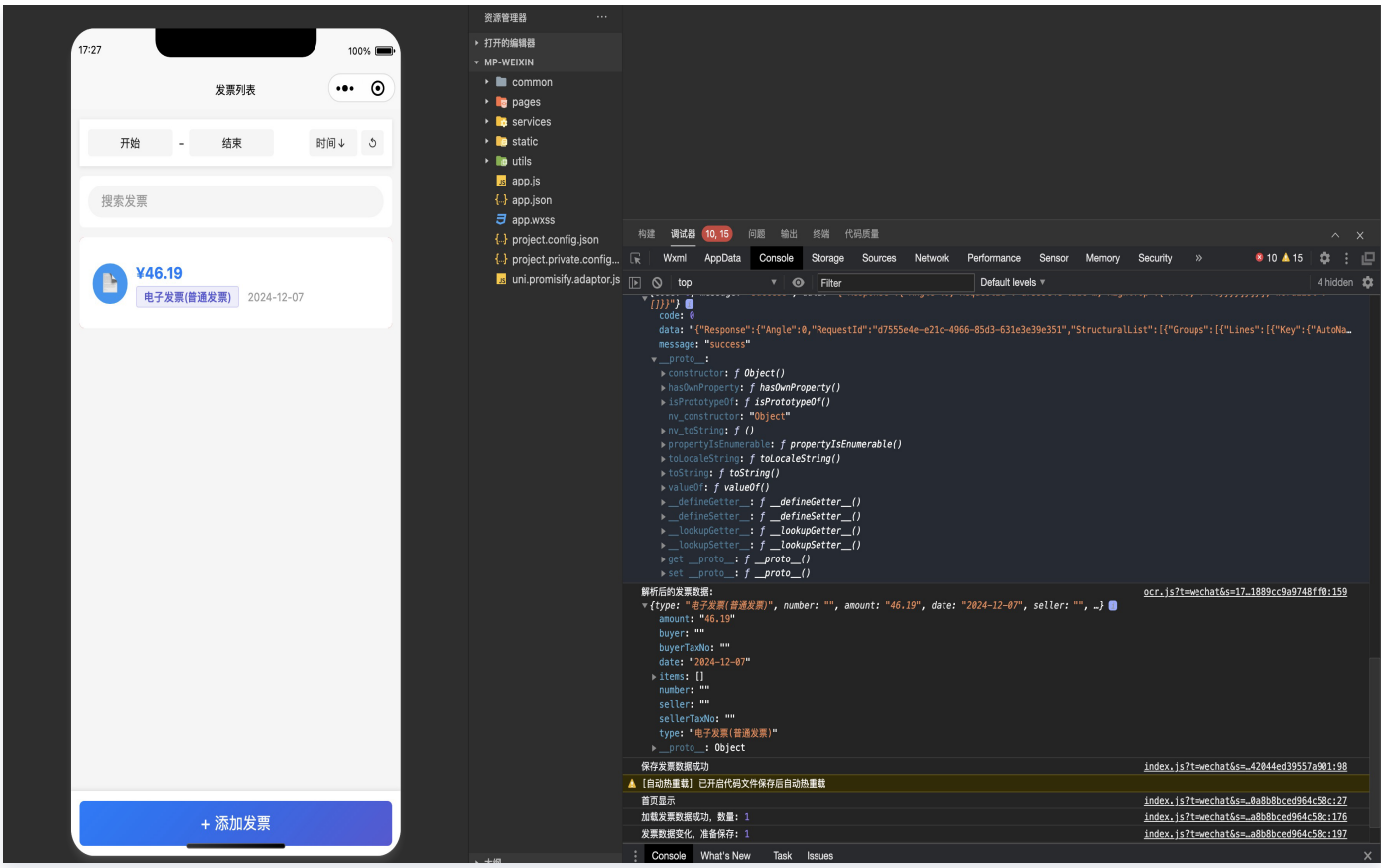
```
const API_CONFIG = {
  baseUrl: 'http://localhost:8080/api/ocr'  // Java 后端服务地址
};

// 调用 OCR API
async function callOCRAPI(imagePath) {
  try {
    // 将本地图片转为 base64
    const base64 = await new Promise((resolve, reject) => {
      uni.getFileSystemManager().readFile({
        filePath: imagePath,
        encoding: 'base64',
        success: res => resolve(res.data),
        fail: err => reject(err)
      });
    });

    // 发起请求
    return await new Promise((resolve, reject) => {
      uni.request({
        url: `${API_CONFIG.baseUrl}/recognize`,
        method: 'POST',
        header: {
          'Content-Type': 'application/json'
        },
        data: {
          imageBase64: base64
        },
        success: (res) => {
          if (res.statusCode === 200 && res.data?.code === 0) {
            resolve(res.data);
          } else {
            reject(new Error(res.data?.message || '识别失败'));
          }
        },
        fail: (err) => {
          reject(err);
        }
      });
    });
  } catch (error) {
    throw error;
  }
```

```
}
```

在上面的代码中主要实现了两个功能，首先实现图片转base64，然后发起后端服务的请求。



发票识别成功之后，发票就会添加到到首页。同时在控制台可以看到后台请求返回的数据，以及解析到的数据。

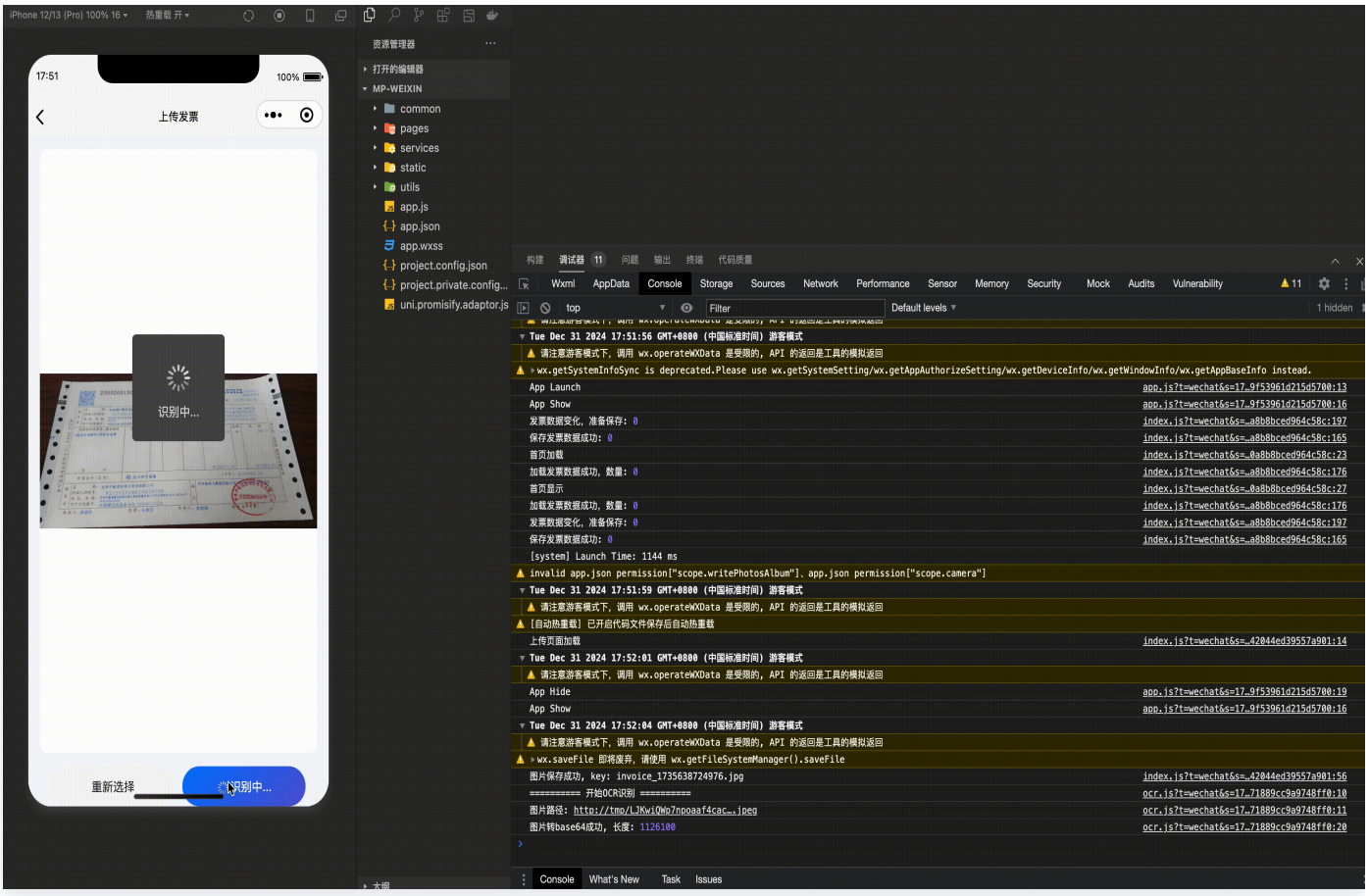


从后台也可以看到 OCR 请求的调用和返回日志。

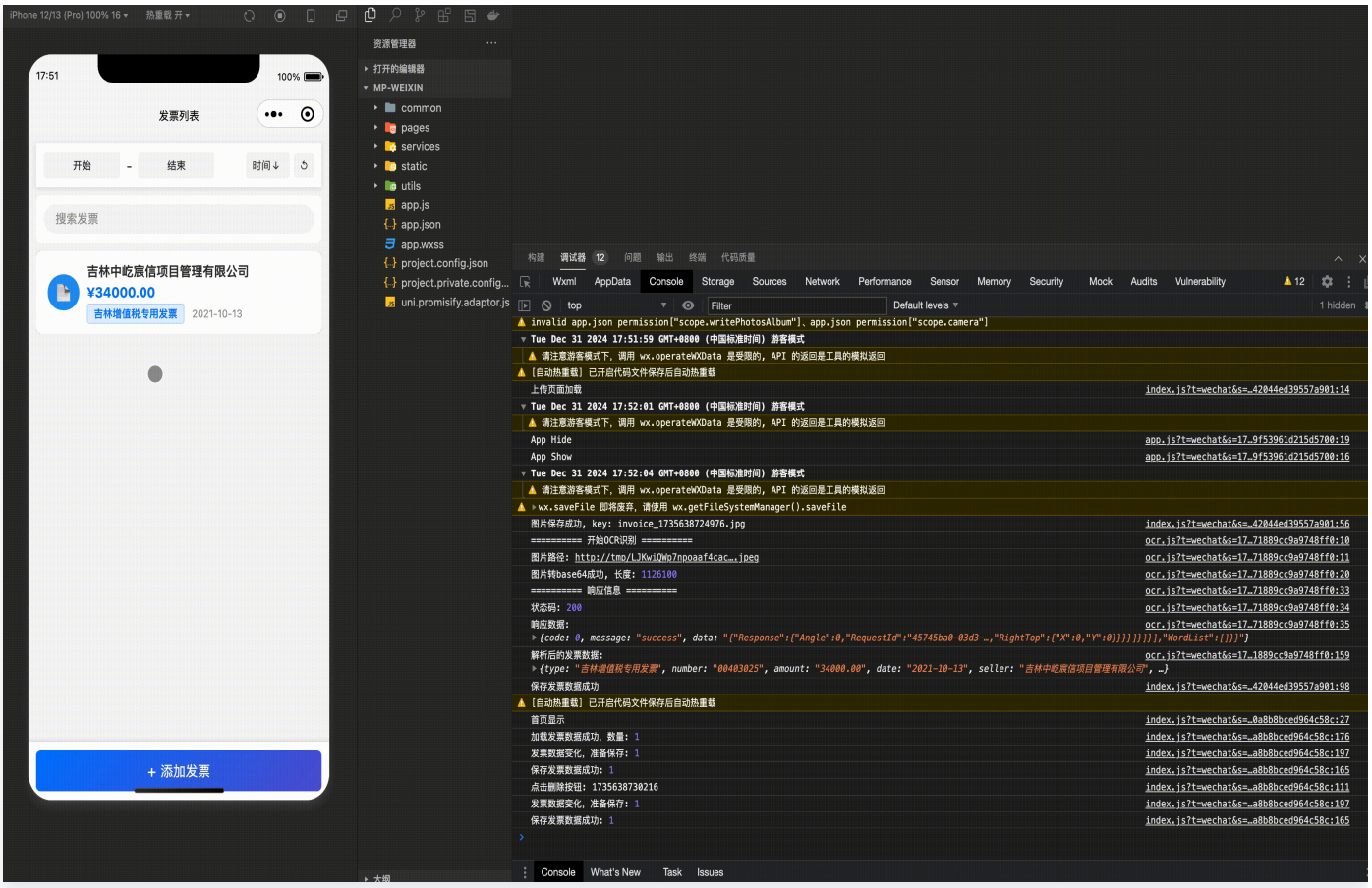
3. 纸质发票识别

这里同样需要实现纸质发票的识别，这里也相当于测试了相机拍照识别发票的功能，我们先选择一张纸质发票。

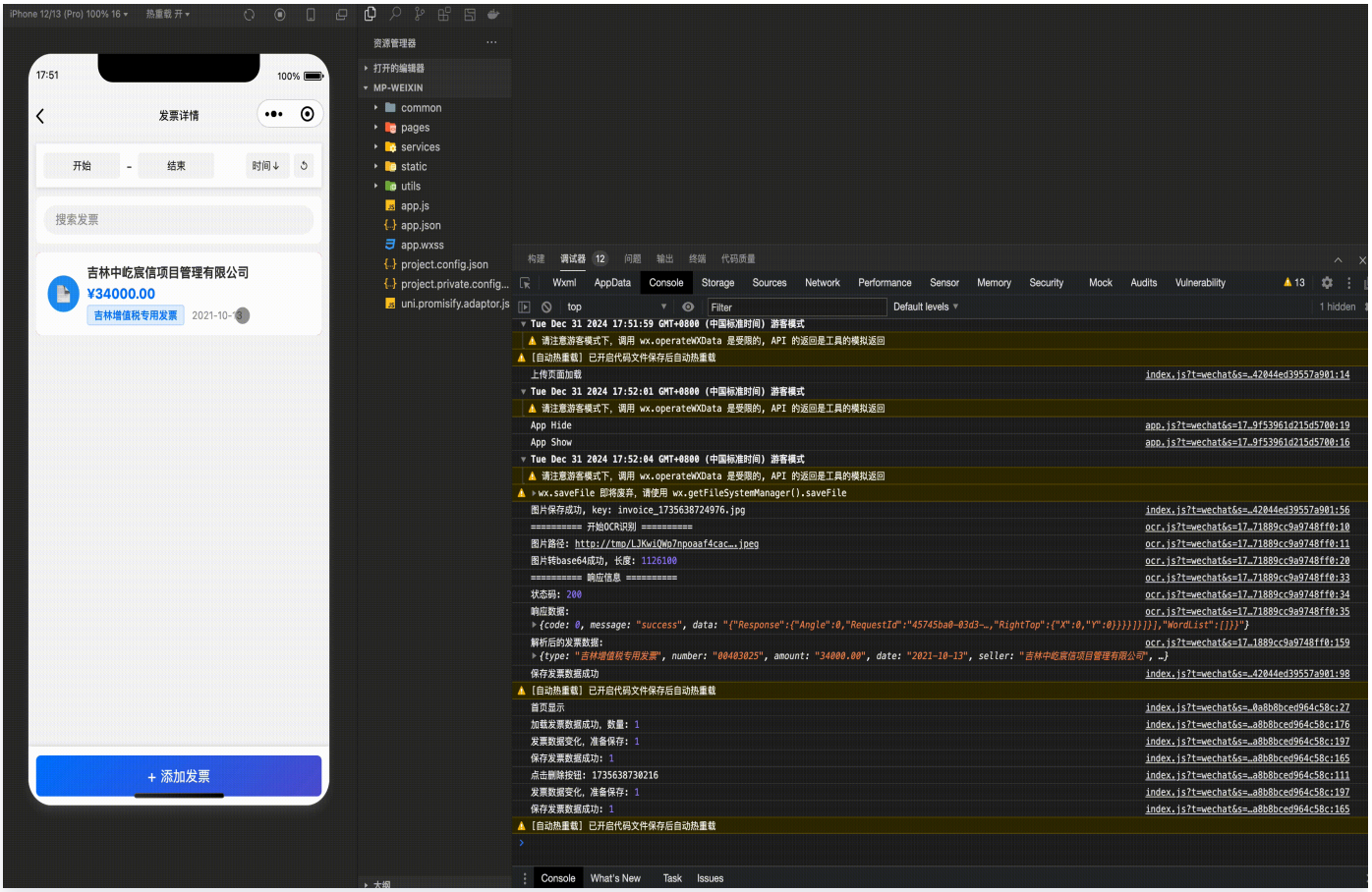
可以看到纸质发票比电子发票的结构更为复杂，而且照片中也有其他的干扰因素，所以识别起来应该更困难一些。



确认上传，可以看到纸质发票被识别成功。长按可以看到详细的发票信息：



进入详情页也可以预览发票图片：

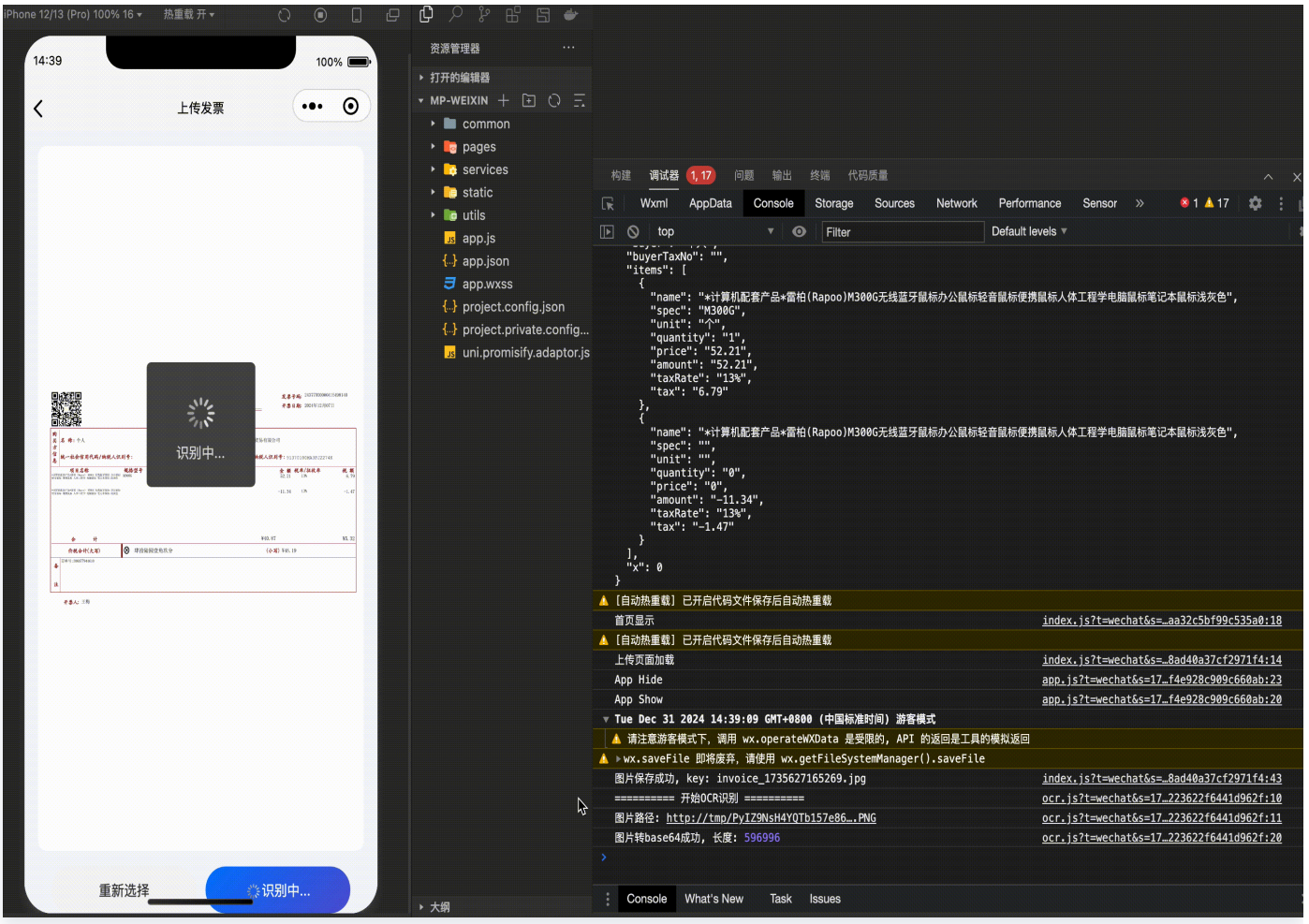


4. 发票去重

在识别发票的模块中，为了避免一张发票多次添加到发票夹中，这里对发票做了一个排重的处理。在生产中需要在后台实现识别重复的逻辑，使用发票编号作为查找条件，去数据库中查找该用户是否已经存此张发票。
这里我在前端页面简单实现了去重的逻辑：

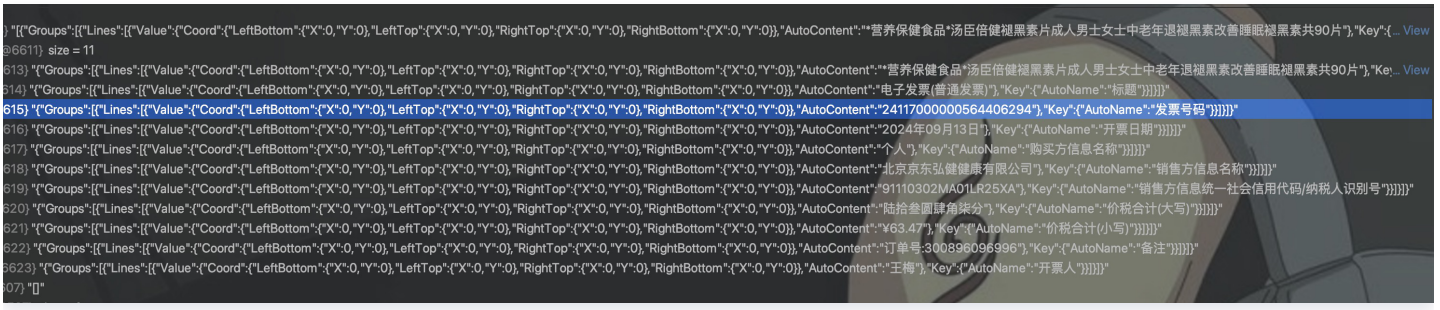
```
// 发票查重实现
checkInvoiceExists(number) {
  try {
    const savedInvoices = uni.getStorageSync('invoices');
    if (savedInvoices) {
      const invoices = JSON.parse(savedInvoices);
      return invoices.some(invoice => invoice.number === number);
    }
    return false;
  } catch (error) {
    console.error('检查发票失败:', error);
    return false;
  }
}

// 查重处理
if (this.checkInvoiceExists(response.number)) {
  uni.hideLoading();
  uni.showModal({
    title: '提示',
    content: '该发票已存在，请勿重复添加',
    showCancel: false,
    success: () => {
      this.loading = false;
    }
  });
  return;
}
```

使用 number 发票号码作为去重条件，最终展现效果：



数据解析

发票详细信息的展示，主要是对文档抽取（多模态版）接口返回信息的解析，结构将识别每条发票信息都会以相同的结构返回：



如图，响应数据中 key 的 AutoName 字段是发票中每一项的 key，value 中的 AutoContent 字段是发票中每一项的值。

1. 结构定义

   这里我定义了字段结构，解析响应数据，将数据存储在 invoiceData 变量中：

```
{
  id: String,           // 发票唯一标识
  imageKey: String,     // 发票图片的存储key
```

```
  number: String,       // 发票号码
  type: String,         // 发票类型（增值税专用发票/普通发票/电子发票）
  amount: String,       // 发票金额
  date: String,         // 开票日期
  seller: String,       // 销售方名称
  sellerTaxNo: String,  // 销售方纳税人识别号
  buyer: String,        // 购买方名称
  buyerTaxNo: String,   // 购买方纳税人识别号
  items: [              // 发票商品明细
    {
      name: String,     // 商品名称
      spec: String,     // 规格型号
      unit: String,     // 单位
      quantity: String, // 数量
      price: String,    // 单价
      amount: String,   // 金额
      taxRate: String,  // 税率
      tax: String       // 税额
    }
  ]
}
```

2. 结构分析

在调用文档抽取（多模态版）接口识别时，因为是是对 key:value 进行结构化识别，但是不同的发票之间相同的 value，key 是不一样的，例如下面两张电子发票的纳税人识别号字段的key就不一样：

发票1：

电子发票(普通发票)

发票号码: 24377000000415498140
开票日期: 2024年12月07日

购买方信息	名 称: 个人 统一社会信用代码/纳税人识别号:	销售方信息	名 称: 济南京东奥升贸易有限公司 统一社会信用代码/纳税人识别号: 91370100MA3PJ2274K

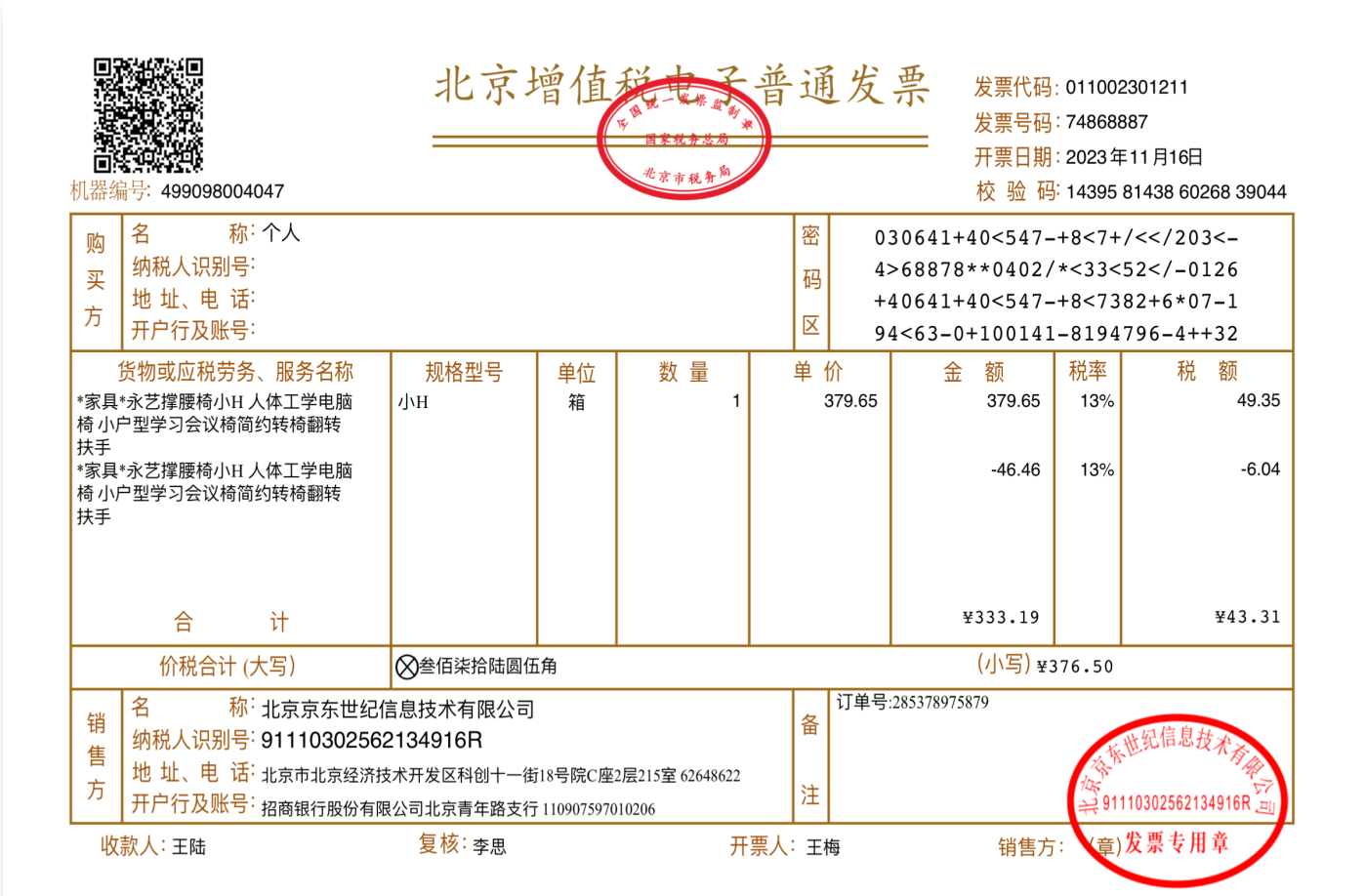
项目名称	规格型号	单位	数量	单价	金额	税率/征收率	税额
*计算机配套产品*雷柏（Rapoo） M300G 无线蓝牙鼠标 办公鼠标 轻音鼠标 便携鼠标 人体工程学 电脑鼠标 笔记本鼠标 浅灰色	M300G	个	1	52.21	52.21	13%	6.79
*计算机配套产品*雷柏（Rapoo） M300G 无线蓝牙鼠标 办公鼠标 轻音鼠标 便携鼠标 人体工程学 电脑鼠标 笔记本鼠标 浅灰色					-11.34	13%	-1.47
合 计					¥40.87		¥5.32

价税合计(大写) ⊗ 肆拾陆圆壹角玖分 (小写) ¥46.19

备注 订单号:306677944610

开票人: 王梅

发票2:



北京增值税电子普通发票

发票代码：011002301211
发票号码：74868887
开票日期：2023年11月16日
校 验 码：14395 81438 60268 39044

机器编号：499098004047

购买方	名　　称：个人 纳税人识别号： 地 址、电 话： 开户行及账号：	密码区	030641+40<547-+8<7+/<</203<- 4>68878**0402/*<33<52</-0126 +40641+40<547-+8<7382+6*07-1 94<63-0+100141-8194796-4++32

货物或应税劳务、服务名称	规格型号	单位	数量	单价	金额	税率	税额
*家具*永艺撑腰椅小H 人体工学电脑椅 小户型学习会议椅简约转椅翻转扶手	小H	箱	1	379.65	379.65	13%	49.35
*家具*永艺撑腰椅小H 人体工学电脑椅 小户型学习会议椅简约转椅翻转扶手					-46.46	13%	-6.04
合　　计					¥333.19		¥43.31
价税合计（大写）	⊗叁佰柒拾陆圆伍角				（小写）¥376.50		

销售方	名　　称：北京京东世纪信息技术有限公司 纳税人识别号：91110302562134916R 地 址、电 话：北京市北京经济技术开发区科创十一街18号院C座2层215室 62648622 开户行及账号：招商银行股份有限公司北京青年路支行 110907597010206	备注	订单号:285378975879

收款人：王陆　　复核：李思　　开票人：王梅　　销售方：（章）发票专用章

发票1返回的纳税人识别号结构：

```
{"AutoName":"购买方信息名称"},"Value":{"AutoContent":"个人","Coord":{"LeftBottom":{"X":0,"Y":0},"LeftTop":{"X":0,"Y":0},"
{"AutoName":"销售方信息名称"},"Value":{"AutoContent":"济南京东奥升贸易有限公司","Coord":{"LeftBottom":{"X":0,"Y":0},"LeftTop"
[{"Lines":[{"Key":{"AutoName":"销售方信息统一社会信用代码/纳税人识别号"},"Value":{"AutoContent":"91370100MA3PJ2274K","Coord":
op":{"X":0,"Y":0}}}}]}]},{"Groups":[{"Lines":[{"Key":{"AutoName":"价税合计(大写)"},"Value":{"AutoContent":"肆拾陆圆壹角玖分
:0,"Y":0},"RightTop":{"X":0,"Y":0}}}}]}]},{"Groups":[{"Lines":[{"Key":{"AutoName":"价税合计(小写)"},"Value":{"AutoConten
m":{"X":0,"Y":0},"RightTop":{"X":0,"Y":0}}}}]}]},{"Groups":[{"Lines":[{"Key":{"AutoName":"备注"},"Value":{"AutoContent
```

发票2返回的纳税人识别号结构：

```
'北京京东世纪信息技术有限公司","Coord":{"LeftBottom":{"X":0,"Y":0},"LeftTop":{"X":0,"Y":0},"Right
':{"AutoName":"备注"},"Value":{"AutoContent":"订单号:285378975879","Coord":{"LeftBottom":{"X"
'Lines":[{"Key":{"AutoName":"销售方纳税人识别号"},"Value":{"AutoContent":"91110302562134916R","
}]}]},{"Groups":[{"Lines":[{"Key":{"AutoName":"销售方地址、电话"},"Value":{"AutoContent":"北京市
ightBottom":{"X":0,"Y":0},"RightTop":{"X":0,"Y":0}}}}]}]},{"Groups":[{"Lines":[{"Key":{"Aut
商银行股份有限公司北京青年路支行110907597010206","Coord":{"LeftBottom":{"X":0,"Y":0},"LeftTop":{"X
':{"AutoName":"收款人"},"Value":{"AutoContent":"王陆","Coord":{"LeftBottom":{"X":0,"Y":0},"Le
':{"AutoName":"复核"},"Value":{"AutoContent":"李思","Coord":{"LeftBottom":{"X":0,"Y":0},"Left
```

对于纳税人的 key，一个是销售方信息统一社会信用代码/纳税人识别号，一个是销售方纳税人识别号。当时我是按照发票1的结构开发的解析程序，所以就导致发票2的许多字段没有解析出来。

3. 代码开发

所以，这里需要通过对不同的发票数据结构进行判断提取：

```
switch (AutoName) {
      case '标题':
        invoiceData.type = AutoContent || '电子发票';
        break;
      case '发票号码':
      case '号码':
        invoiceData.number = AutoContent || '';
        break;
      case '开票日期':
        invoiceData.date = AutoContent ? AutoContent.replace('年', '-
').replace('月', '-').replace('日', '') : '';
        break;
      case '销售方名称':
      case '销售方信息名称':
        invoiceData.seller = AutoContent || '';
        break;
      case '销售方纳税人识别号':
      case '销售方信息统一社会信用代码/纳税人识别号':
        invoiceData.sellerTaxNo = AutoContent || '';
        break;
      case '购买方名称':
```

```
        case '购买方信息名称':
          invoiceData.buyer = AutoContent || '';
          break;
        case '价税合计(小写)':
          invoiceData.amount = AutoContent ? AutoContent.replace('¥',
'').replace('￥', '') : '0';
          break;
      }
// ...省略部分代码
```

这样，不同格式的发票也成功识别了。

同样也要将增值税专用发票、纸质发票结构的key考虑进去。

发票展示

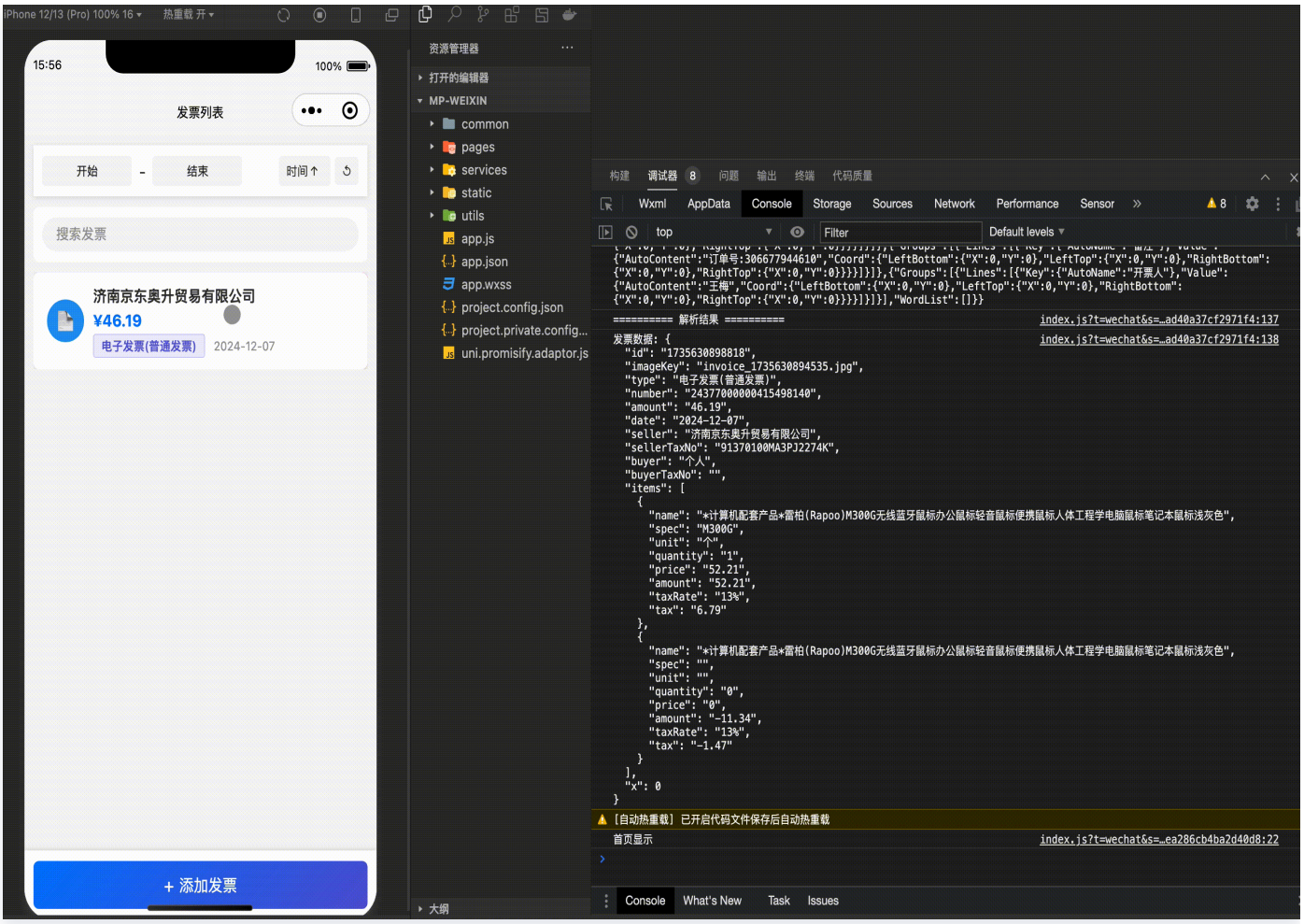
发票识别成功之后，就会展示在首页。

1. 发票详情弹出层

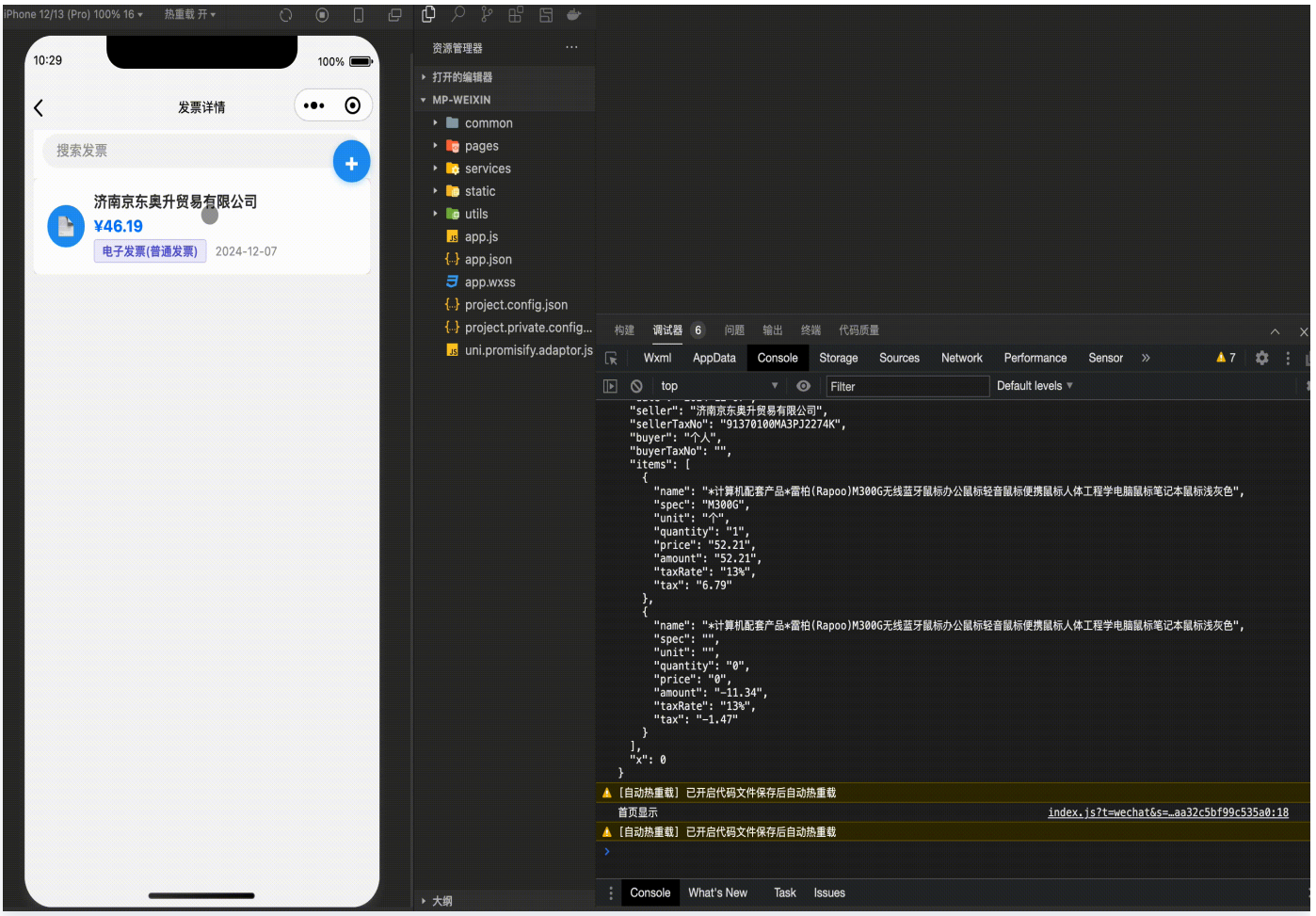
   这里主要实现了两个发票的详情展示。第一个是长按列表中的发票，就会弹出发票的信息，包括发票号码等。绑定长按事件：

```
<view
  class="invoice-item"
  @tap="goToDetail(invoice)"
  @longpress="showInvoicePopup(invoice)"
>
```

   showInvoicePopup会修改showPopup变量，用来控制弹出层显示。

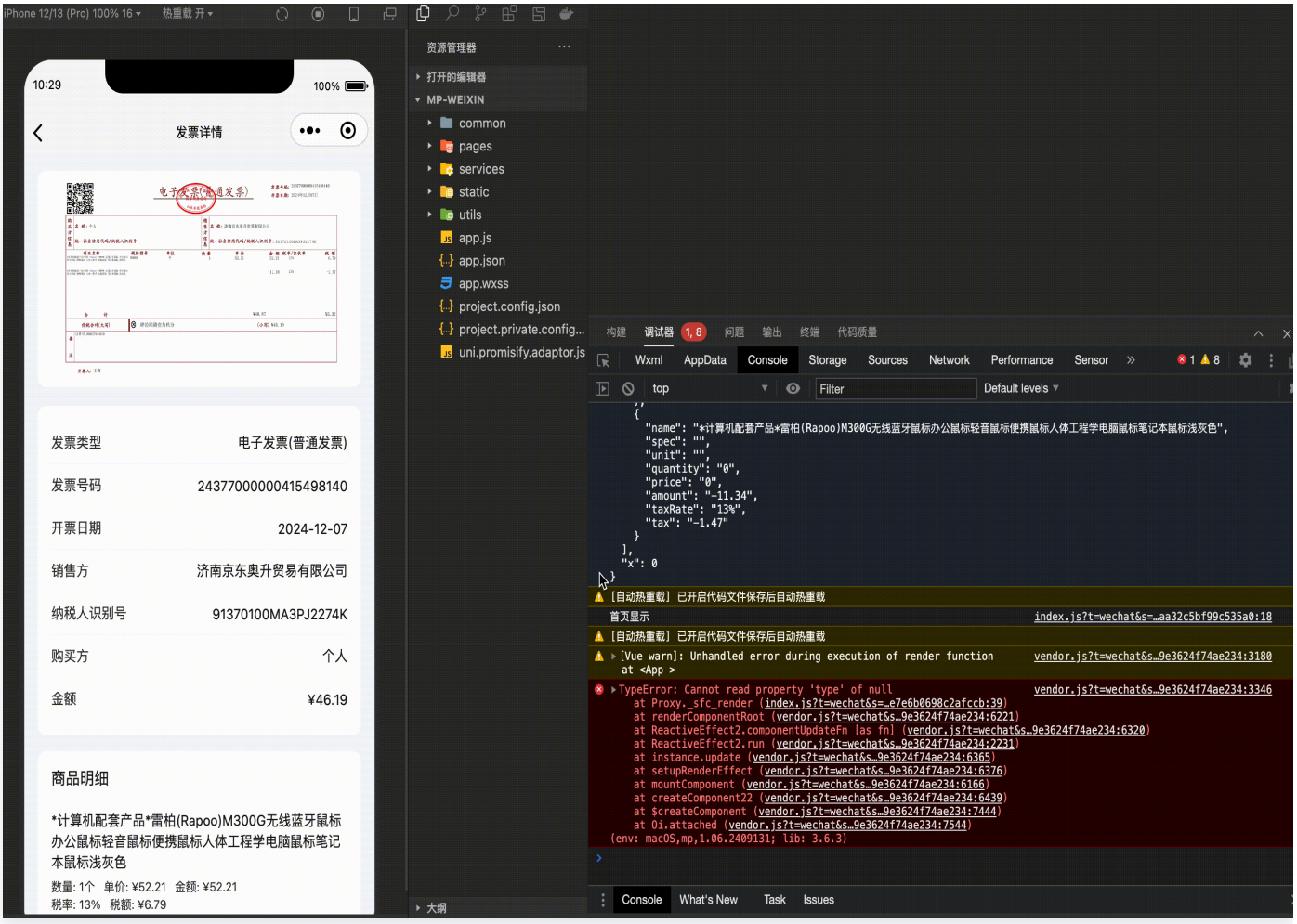

iPhone 12/13 (Pro) 100% 16
热重载 开
15:56
100%
发票列表
开始
结束
时间↑
搜索发票
济南京东奥升贸易有限公司
¥46.19
电子发票(普通发票)
2024-12-07
+ 添加发票
资源管理器
打开的编辑器
MP-WEIXIN
common
pages
services
static
utils
app.js
app.json
app.wxss
project.config.json
project.private.config...
uni.promisify.adaptor.js
大纲
构建
调试器 8
问题
输出
终端
代码质量
Wxml
AppData
Console
Storage
Sources
Network
Performance
Sensor
top
Filter
Default levels
{"AutoContent":"订单号:306677944610","Coord":{"LeftBottom":{"X":0,"Y":0},"LeftTop":{"X":0,"Y":0},"RightBottom":{"X":0,"Y":0},"RightTop":{"X":0,"Y":0}}}]}],{"Groups":[{"Lines":[{"Key":{"AutoName":"开票人"},"Value":{"AutoContent":"王梅","Coord":{"LeftBottom":{"X":0,"Y":0},"LeftTop":{"X":0,"Y":0},"RightBottom":{"X":0,"Y":0},"RightTop":{"X":0,"Y":0}}}}]}],"WordList":[]}}
========== 解析结果 ==========
index.js?t=wechat&s=…ad40a37cf2971f4:137
发票数据: {
index.js?t=wechat&s=…ad40a37cf2971f4:138
"id": "1735630898818",
"imageKey": "invoice_1735630894535.jpg",
"type": "电子发票(普通发票)",
"number": "24377000000415498140",
"amount": "46.19",
"date": "2024-12-07",
"seller": "济南京东奥升贸易有限公司",
"sellerTaxNo": "91370100MA3PJ2274K",
"buyer": "个人",
"buyerTaxNo": "",
"items": [
{
"name": "*计算机配套产品*雷柏(Rapoo)M300G无线蓝牙鼠标办公鼠标轻音鼠标便携鼠标人体工程学电脑鼠标笔记本鼠标浅灰色",
"spec": "M300G",
"unit": "个",
"quantity": "1",
"price": "52.21",
"amount": "52.21",
"taxRate": "13%",
"tax": "6.79"
},
{
"name": "*计算机配套产品*雷柏(Rapoo)M300G无线蓝牙鼠标办公鼠标轻音鼠标便携鼠标人体工程学电脑鼠标笔记本鼠标浅灰色",
"spec": "",
"unit": "",
"quantity": "0",
"price": "0",
"amount": "-11.34",
"taxRate": "13%",
"tax": "-1.47"
}
],
"x": 0
}
【自动热重载】已开启代码文件保存后自动热重载
首页显示
index.js?t=wechat&s=…ea286cb4ba2d40d8:22
Console
What's New
Task
Issues

最终效果如下：

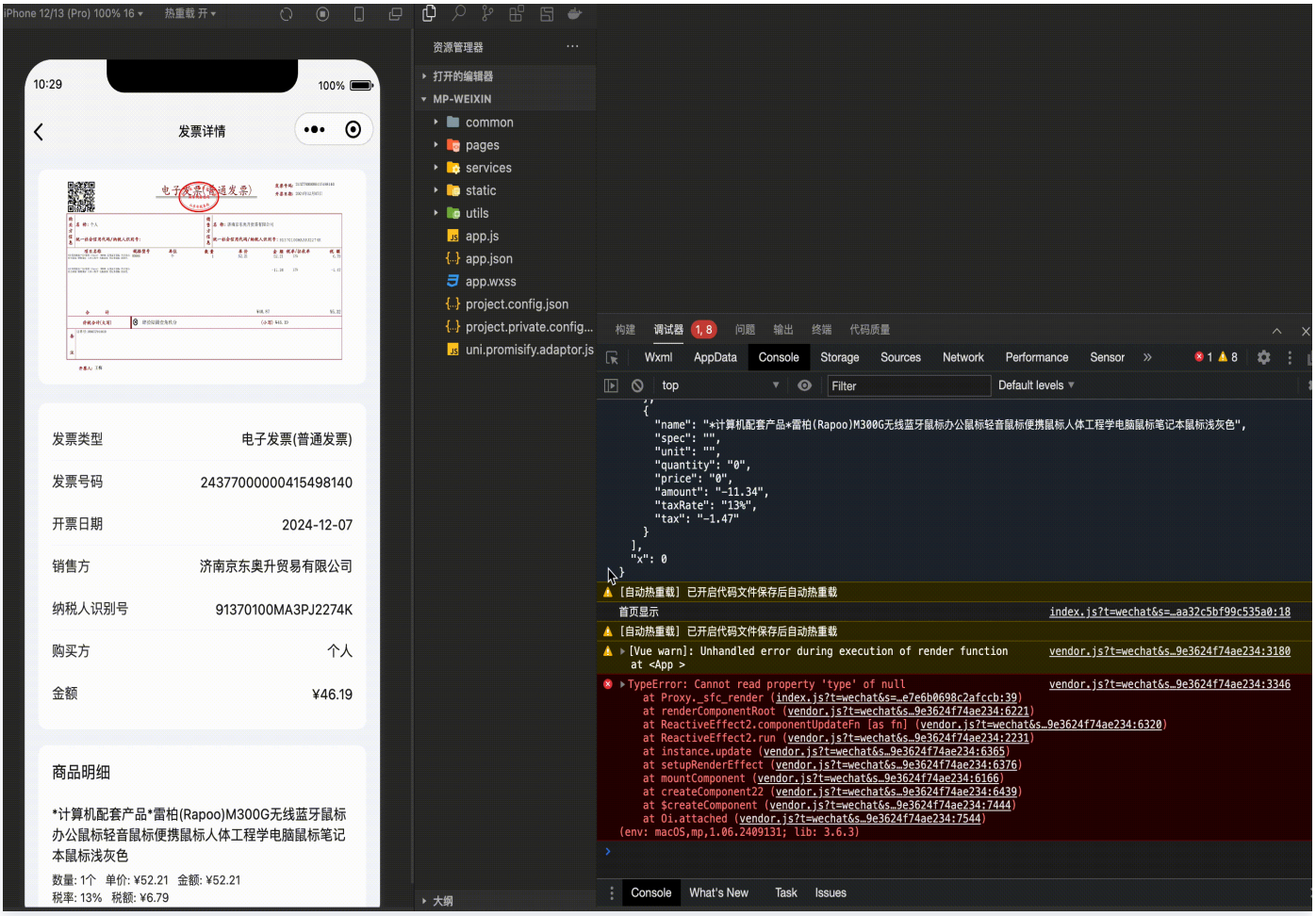


2. 发票详情页面

第二个就是点击发票，跳转发票详情页面：



在这个页面可以预览发票图片：



整体就是一个页面展示，主要是定义图片的预览事件 previewImage。

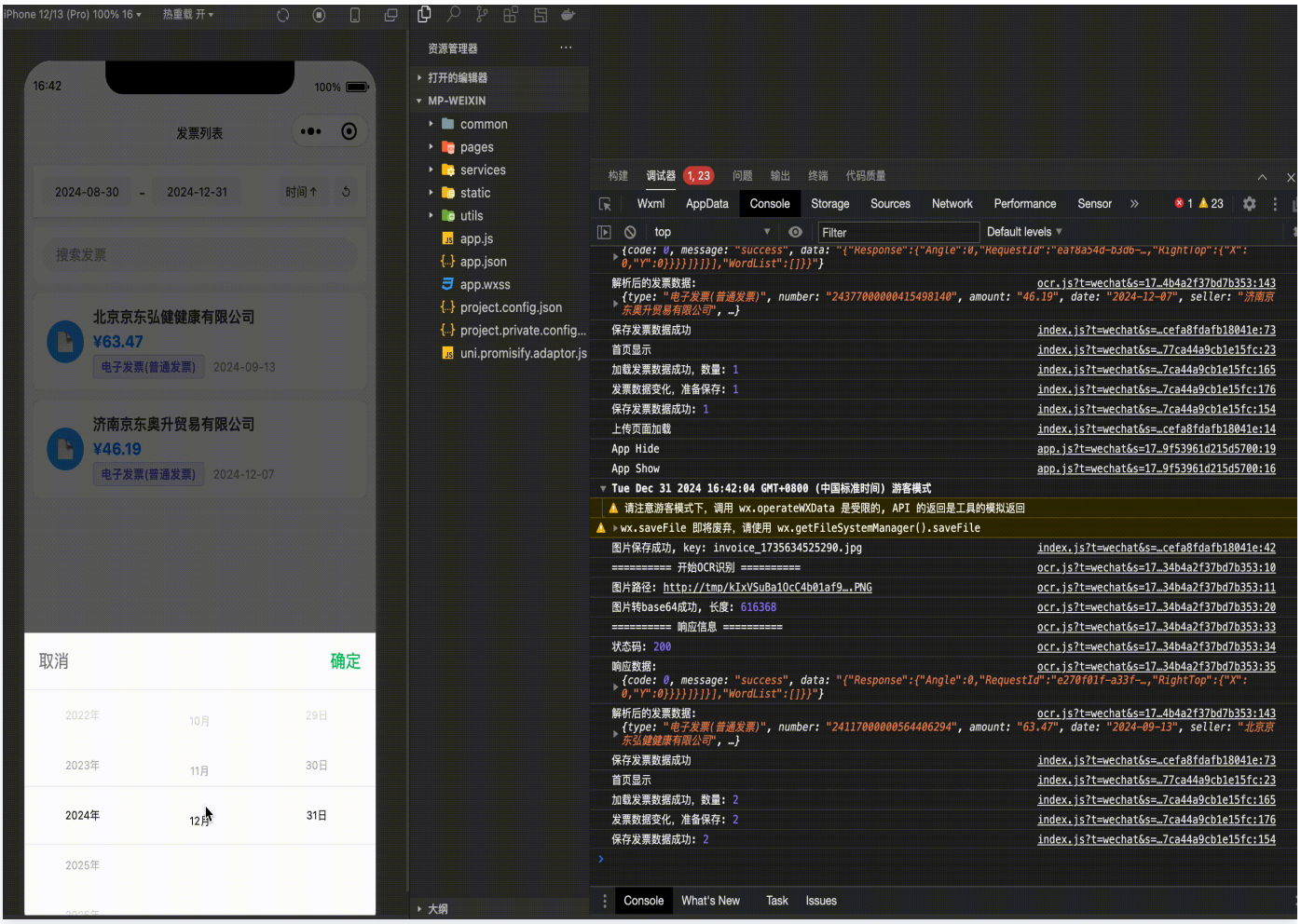
```
<image
  class="invoice-image"
  :src="imageUrl"
  mode="aspectFit"
  @tap="previewImage"
/>
```

当触发点击事件（tap）时，通过uni.previewImage就可以实现图片预览。

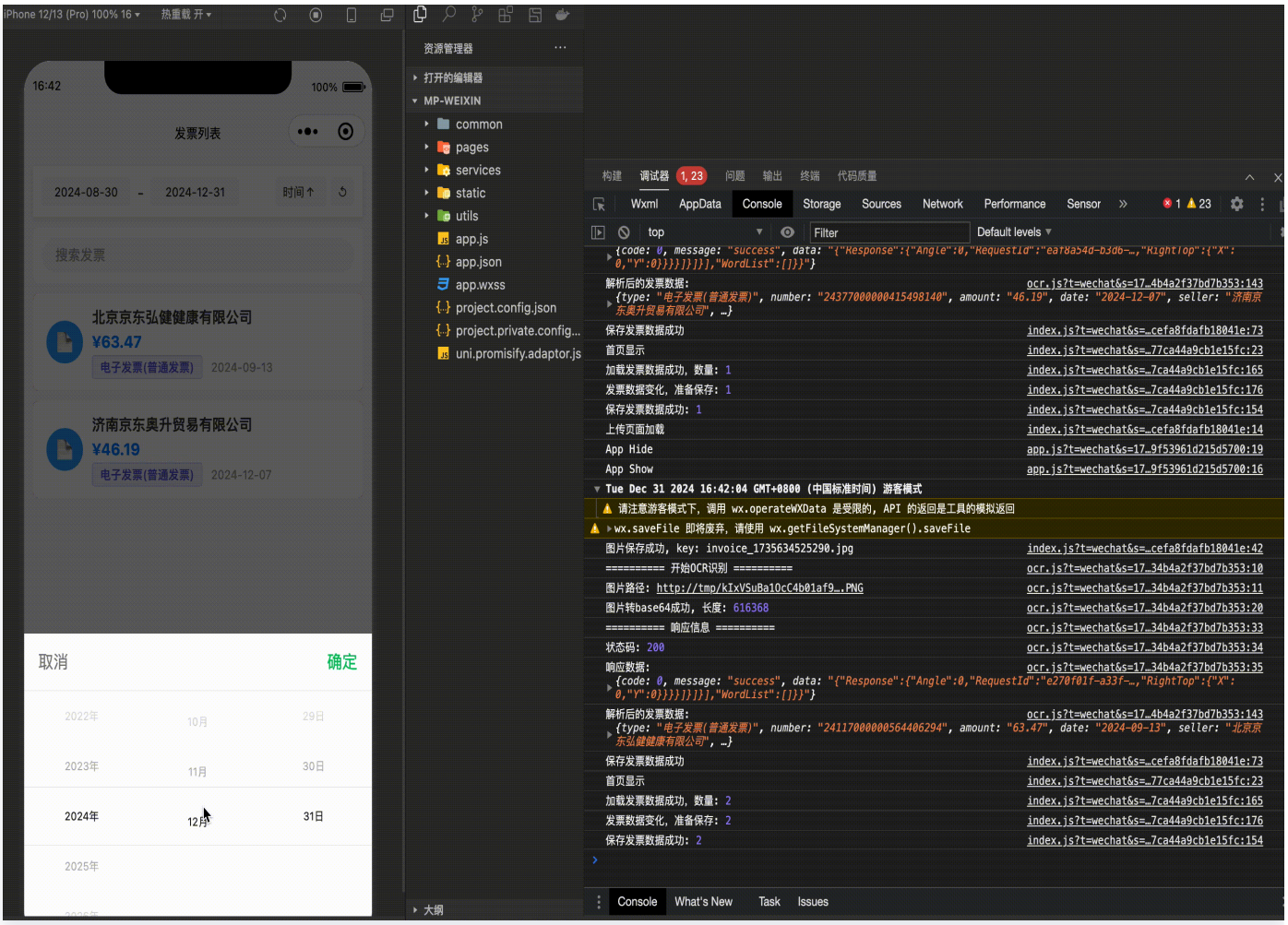
```
  previewImage() {
  if (this.imageUrl) {
    uni.previewImage({
      urls: [this.imageUrl]
    });
  }
}
```

3. 发票筛选

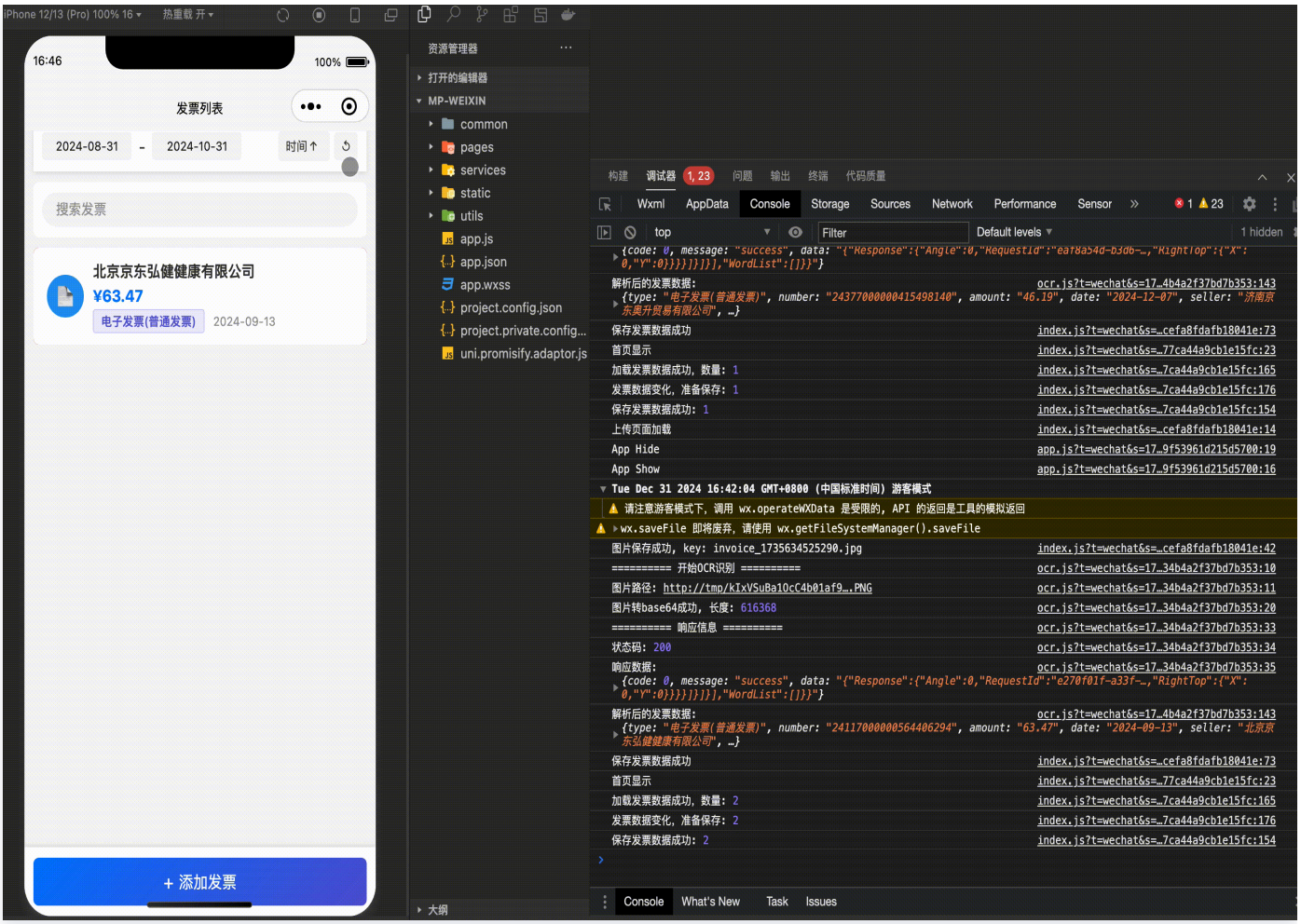
发票列表支持按日期范围筛选。



支持按时间正序/倒序排序。



提供快速重置筛选条件功能。



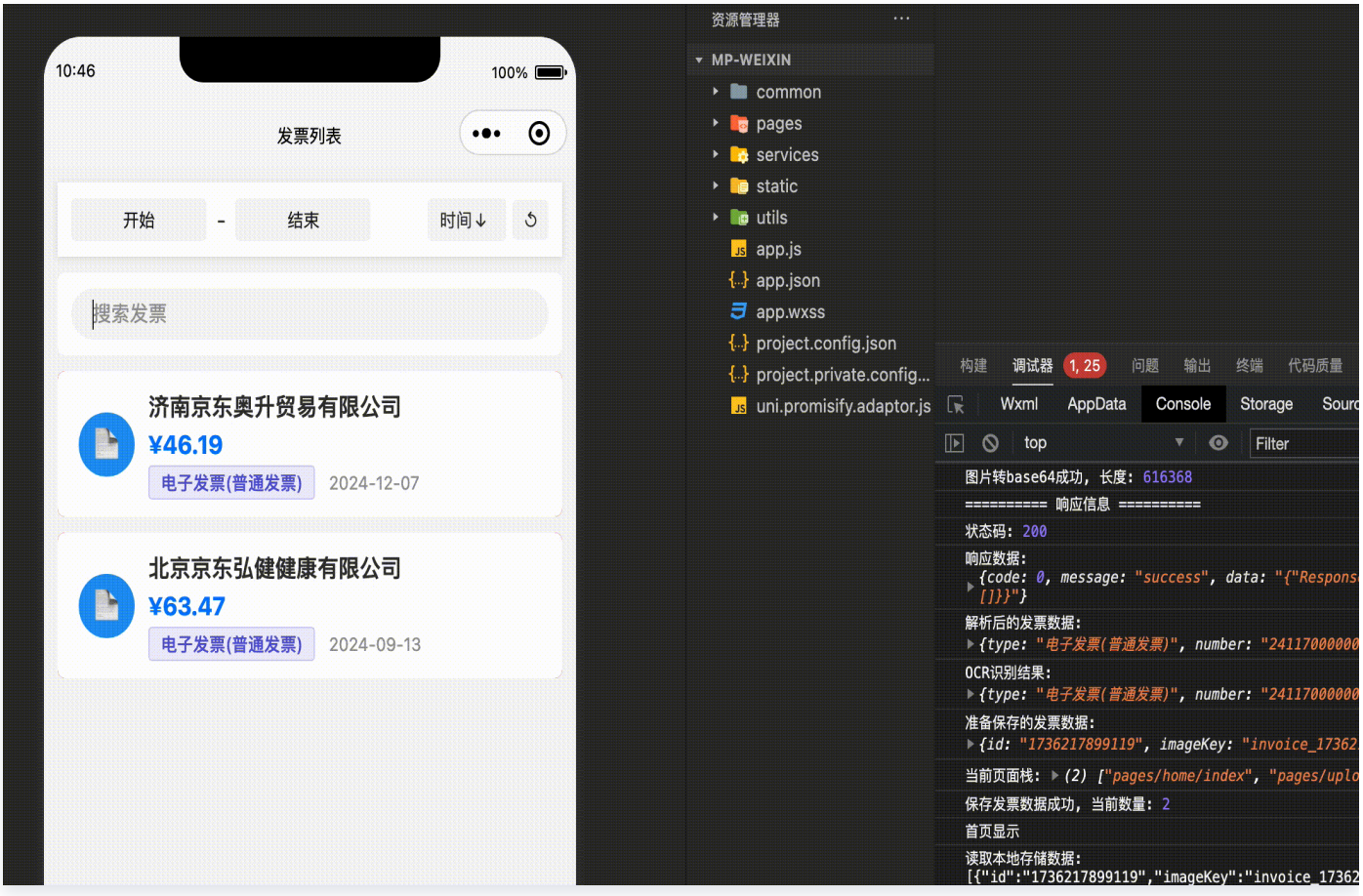
这个功能逻辑比较简单，主要是对发票时间进行一个排序和筛选，代码如下：

```
filteredInvoices() {
  let result = [...this.invoices];

  // 时间区间筛选
  if (this.startDate || this.endDate) {
    result = result.filter(invoice => {
      const invoiceDate = new Date(invoice.date);
      let isValid = true;

      if (this.startDate) {
        const start = new Date(this.startDate);
        isValid = isValid && invoiceDate >= start;
      }

      if (this.endDate) {
        const end = new Date(this.endDate);
        isValid = isValid && invoiceDate <= end;
      }
```

```
      return isValid;
    });
  }

  // 时间排序
  result.sort((a, b) => {
    const dateA = new Date(a.date);
    const dateB = new Date(b.date);
    return this.sortDesc ? dateB - dateA : dateA - dateB;
  });
  return result;
}
```

4. 发票搜索

支持实时搜索，可按销售方名称、发票号码、金额搜索。



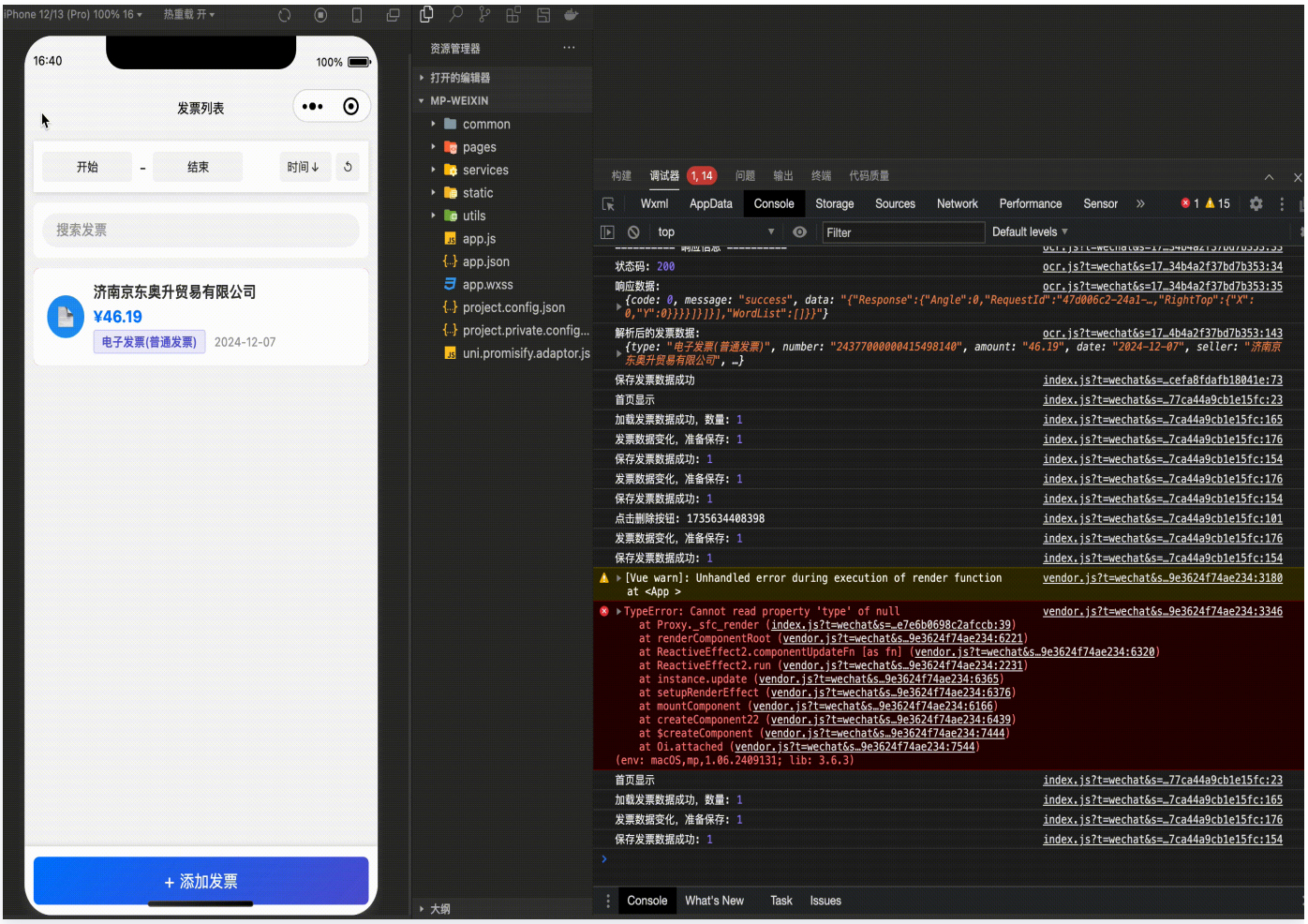
并使用防抖处理优化性能。

```
  debounce(fn, delay = this.debounceDelay) {
  return (...args) => {
    if (this.searchDebounceTimer) {
```

```
      clearTimeout(this.searchDebounceTimer);
    }
    this.searchDebounceTimer = setTimeout(() => {
      fn.apply(this, args);
    }, delay);
  };
}
```

5. 发票删除

实现了左滑显示删除按钮、删除前二次确认，当用户点击确认之后，发票才会被删除。



实现代码如下：

```
onDeleteTap(event, invoice) {
  if (event) {
    event.stopPropagation();
  }

  uni.showModal({
    title: '确认删除',
```

```
      content: '是否确认删除该发票？',
      confirmColor: '#ff4444',
      showCancel: true,
      success: (res) => {
        if (res.confirm) {
          this.invoices = this.invoices.filter(item => item.id !==
invoice.id);
          uni.showToast({
            title: '删除成功',
            icon: 'success'
          });
          this.saveInvoices();
        }
        this.$nextTick(() => {
          invoice.x = 0;
        });
      }
    });
}
```

结语

本篇文章主要使用腾讯云的文档抽取（多模态版）接口，实现了 OA 系统中的发票识别模块。在整体的开发过程中，文档抽取（多模态版）使用起来比较方便，通过 SDK 几行代码就能接入到系统中。
通过对各种版式的电子发票，以及纸质增值税专用发票的识别结果来看，文档抽取（多模态版）有着精准的识别能力，适合于各种版式的单据识别。

探索腾讯云文档智能：技术解析与实践指南

最近更新时间：2025-05-27 17:59:32

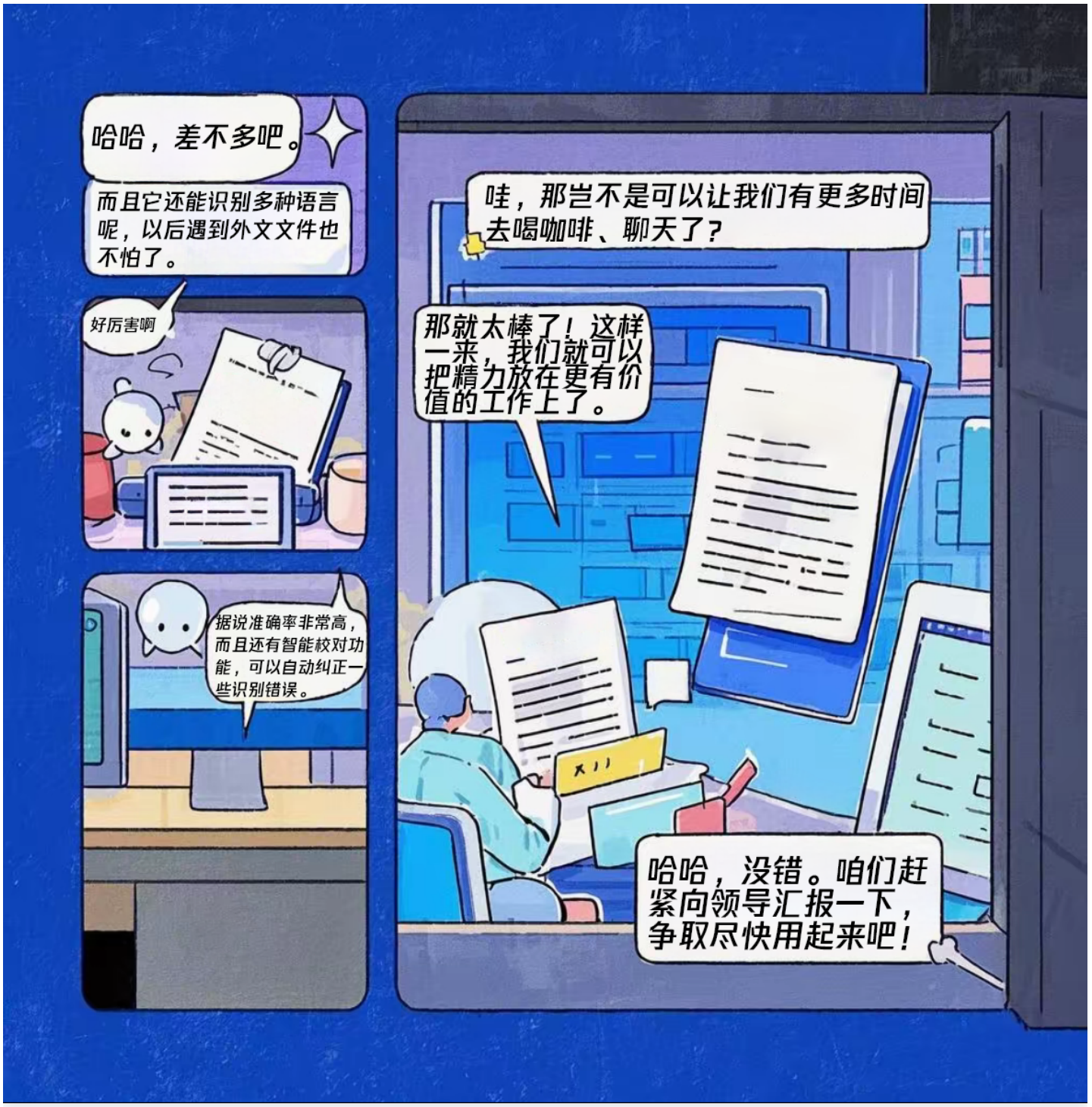
概述

本文详细介绍了腾讯云文档智能 OCR 技术，突出其高精度、灵活性和易用性。通过多模态大模型技术，该 OCR 服务支持个性化模板定制，显著提升数据抓取效率，适用于多种文档识别场景。文章还提供了获取 SecretId 和 SecretKey 的指南，以及如何使用 OCR 服务的步骤，包括前端界面设计和后端 Flask 应用实现。最后，作者鼓励读者亲自尝试 OCR 服务，并对其功能和界面表示满意。

引言


哎，小明，你今天怎么这么早到办公室啊？
我昨天工作太多了，加完班工作还没做完想着早点来做完。
最近咱们部门的文件处理工作量越来越大了，你有没有什么好办法？
是啊,我也在想这个问题呢
昨天我偶然看到了一个叫腾讯云智能结构化OCR的产品，感觉挺有意思的。

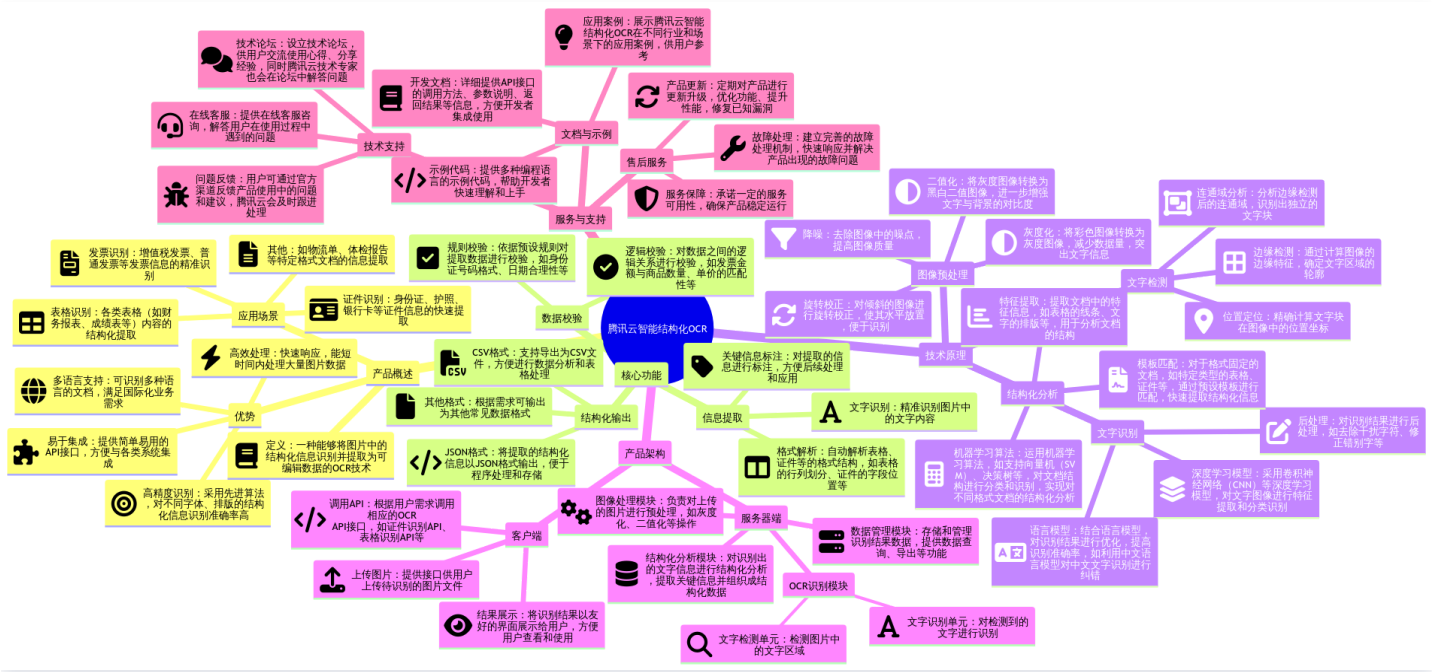

它可以自动识别和提取各种结构化数据，比如表格、发票、名片什么的。
哇，听起来像是个“懒人神器”啊！
对啊，而且有时候还会出错。
是的，用这个OCR产品就可以把合同上的信息自动识别出来，直接转换成电子格式，省去了人工录入的麻烦。



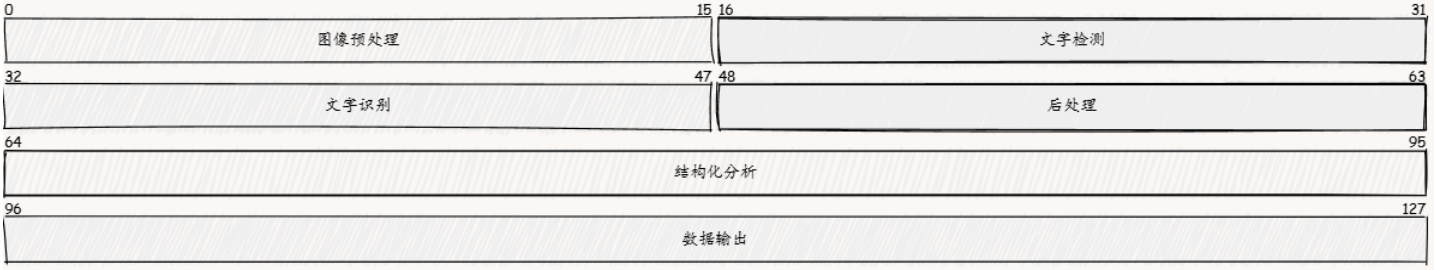
在上述情境中，我们仅对腾讯云文档智能 OCR 这一技术进行了简要提及。凭借其卓越的高精度、出色的灵活性和极佳的易用性，腾讯云文档智能 OCR 正开启跨行业高效且精准的文档处理及数据提取的新篇章。腾讯云文档智能 OCR 以多模态大模型技术为核心，融合了深度学习、图像检测技术及 OCR 大模型能力，通过智能建立键值对应关系，支持客户根据自身需求定制个性化模板，显著提高数据抓取与录入的效率。其基础版本为企业搭建了通用型文本识别的稳固框架，可精准识别常见的印刷体文字、数字及基本符号，满足一般性的数据提取需求。而高级版本则进一步融入了语义理解、上下文关联分析等高级功能模块，能够应对诸如法律合同、医学报告等专业性强、语义复杂且版式多变的文档识别任务，实现关键信息的深度挖掘与结构化输出。例如，在处理法律合同文档时，腾讯云文档智能 OCR 能够快速提取合同双方、金额信息、时间节点等关键信息，大大提高了合同处理的效率和准确性。

其实，撰写这篇文章我内心纠结良久，迟迟未能动笔。究其根本，是因我对这款产品知之甚少，无从下笔。在日常生活中，我们或许接触过类似 OCR 功能的场景，例如部分手机自带的图片文字自动识别功能，又或是微信、QQ中通过截图进行的文字识别，它们的准确率着实令人称赞。眼看活动即将落幕，我终于挤出些许时间，匆匆写下这篇文章，现在就让我们直奔主题。

或许您和我一样，面对产品文档或产品界面时，感到一头雾水，不知该如何着手。为此，我精心整理了一份思维导图，您可以借助这张图，迅速把握产品的核心要点。



借助这份思维导图，我们能够对产品形成初步的宏观认知。那么，您是否对这款产品有了新的认识呢？又或者，还有哪些疑问未解呢？接下来，让我们一同探究 OCR 识别的原理，这将助我们深入理解产品背后的技术支撑，进而更高效地运用它来满足自身需求。虽然下面这张图较为简单，但它却是最直观地解释文字识别原理的方式。将两张图对照来看，既能让您对产品有更深入的看法，也能增进您对产品使用的了解。下面，就让我们来了解一下如何使用这款产品。目前而言，这款产品几乎没有任何使用门槛，希望大家都能多去尝试，说不定您会喜欢去使用这款产品。



OCR 识别原理图

文档智能的注册

首先，我们需要访问腾讯云文档智能的官方网站。点击此处即可直达官网。进入官网后，您会看到如下图所示的界面。接下来，只需点击图中的“开通服务”按钮，并按照提示逐步操作，便能顺利开通此项服务。

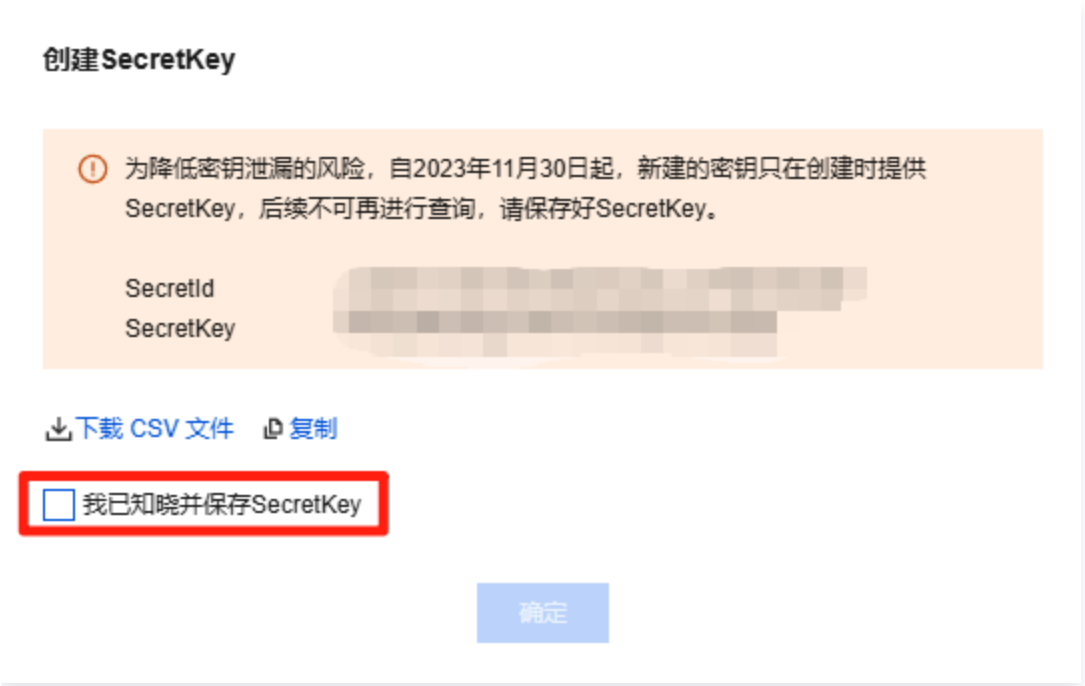
完成首次注册后，系统将贴心地赠送一份免费资源包，有效期为一个月。这份资源包对于普通用户而言，可谓相当充足。其详细资源列表涵盖了从几十次到几千次不等的使用额度，足以满足我们在短期内的使用需求，无需急于再次购买。激活资源后请确保 SecretId 和 SecretKey 的唯一性。

2025年1月免费资源包(37个)	赠送	32250	0 (0%)	32250 (100%)	2025-01-01 00:00:00	2025-01-31 23:59:59	退款
智能结构化高级版-1千次免费资源包	赠送	1000	0 (0%)	1000 (100%)	2025-01-01 00:00:00	2025-01-31 23:59:59	退款
英文识别-1000次免费套餐包	赠送	1000	0 (0%)	1000 (100%)	2025-01-01 00:00:00	2025-01-31 23:59:59	退款
银行卡识别-1k次免费套餐包	赠送	1000	0 (0%)	1000 (100%)	2025-01-01 00:00:00	2025-01-31 23:59:59	退款
智能结构化-1k次免费资源包	赠送	1000	0 (0%)	1000 (100%)	2025-01-01 00:00:00	2025-01-31 23:59:59	退款
通用印刷体识别（高速版）-1k次免费套餐包	赠送	1000	0 (0%)	1000 (100%)	2025-01-01 00:00:00	2025-01-31 23:59:59	退款

获取 SecretId 和 SecretKey

可能有些小伙伴不清楚 SecretId 和 SecretKey 的获取位置，接下来就让我来为大家详细讲解。首先，点击此处进入腾讯云的访问管理页面。在页面中找到“用户”选项，再点击“用户列表”。接着，点击“新建用户”按钮。当然，您也可以直接使用主账号，但出于安全和资源管理的考虑，我更推荐大家创建一个子账号，这里勾选 QcloudOCRReadOnlyaccess 账号权限。这样既能有效避免他人误用资源导致不必要的消费，又能更好地管理权限和资源使用情况。

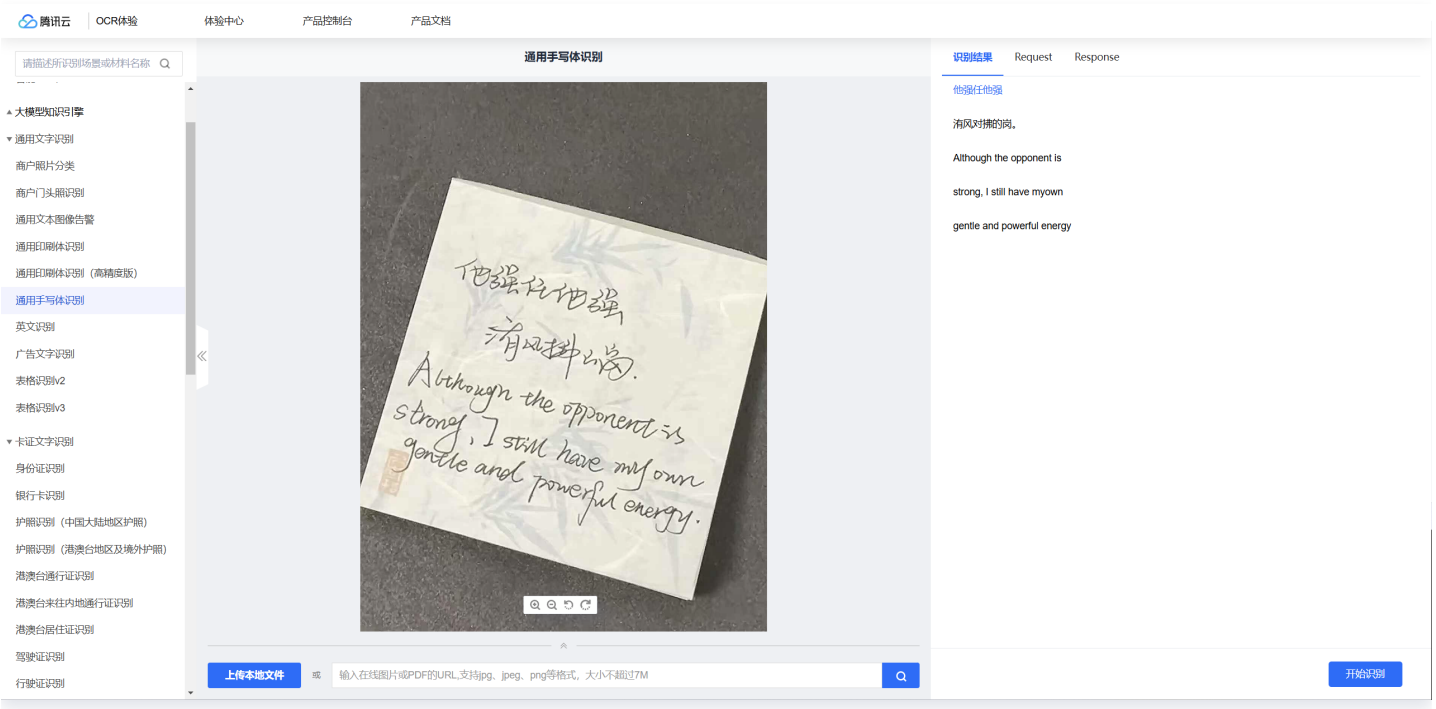
进入子用户面板后，点击“API密钥”选项。再次点击“新建密钥”，此时系统会立即弹出显示 SecretId 和 SecretKey 的界面。需要特别注意的是，这个密钥信息目前仅展示一次，因此务必要及时对密钥进行保存，或者下载 CSV 文件以便安全存储。同时，别忘了勾选“我已知晓并保存 SecretKey”这一选项。完成以上步骤，您的 SecretId 和 SecretKey 就已成功获取完毕啦。



文档智能 OCR 的使用

OCR demo

点击此处，便可直接进入 OCR 的 demo 界面。在这个界面中，您可以充分测试其接口的各项能力，建议对所有功能都进行测试，以全面了解其性能。接下来，我将使用自己准备的几张图片，在 demo 中进行实际测试。
我打算用自己手写的字去测试一下它的识别效果。我的字可能有点潦草，个人感觉要是识别出现些许错误也是情有可原的，毕竟手写字体千差万别，尤其是我这种不太规整的书写风格，对识别系统来说是个不小的挑战。只要不是错误太多，基本上我还是能接受的。那我们现在就一起打开网页，看看它到底能识别成什么样吧！

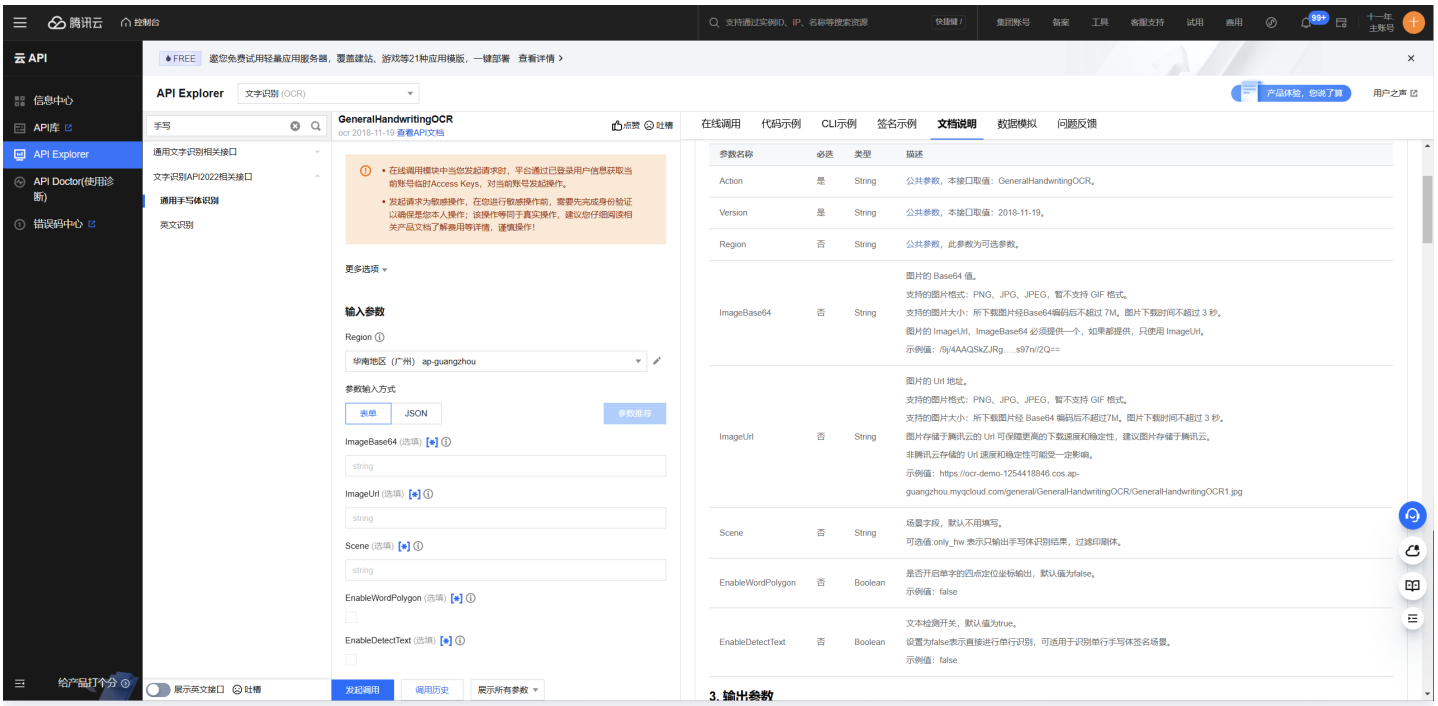


我们仔细观察后发现，虽然每行字的划分很清晰，但文字识别确实存在一些错误。不过这也情有可原，毕竟我写的字实在太像另一个字了。原话应该是“他强任他强，清风拂山岗”。
接下来，我将通过一本书的内页来进行识别测试，以检验其识别率的精准度。这本书是熊培云老师送给我的第一本书，它非常值得一读，我极力推荐大家去阅读。

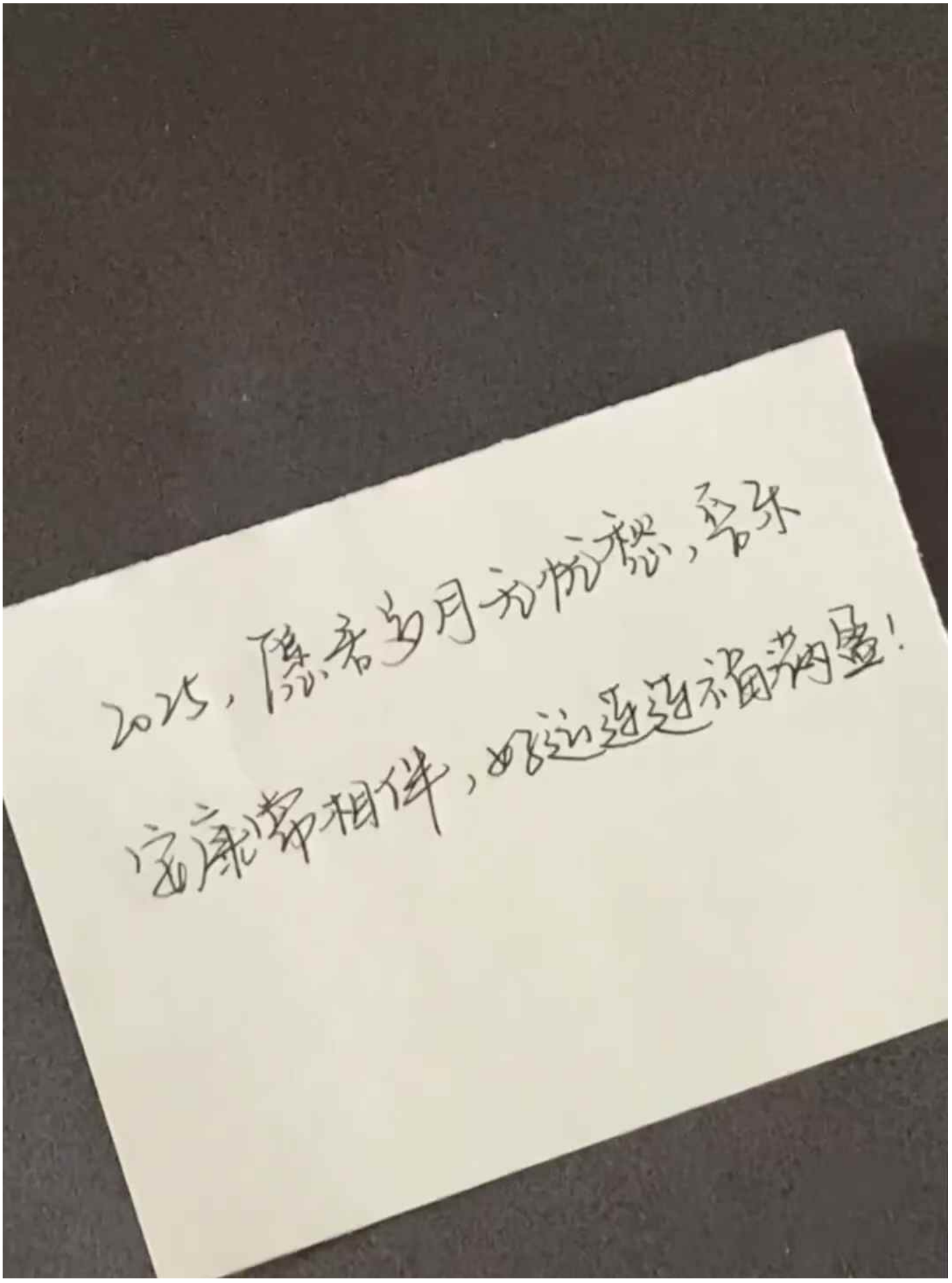
我们发现，此次识别的准确率相当高，几乎未出现任何错误。这充分证明了对于纯印刷体文字，其识别效果是非常出色的。

经过与源文档的细致比对，输出结果毫无瑕疵，图片等内容也均能精准识别，这足以证明其文档解析能力依旧卓越。我仅通过这三个演示，便足以略窥一二。那么，此次演示便暂且告一段落。不过，我也发现了一些待改进之处。例如，在复制公式时，并未保留其原有的格式。许多文档编辑软件都具备插入公式的功能，因此，我在此向产品团队提出一点小建议，希望能尽快实现对公式格式的复制支持。或者，当我们将鼠标悬停在公式上时，能自动弹出选项，让我们能够将其以公式格式复制，以便更便捷地导入腾讯文档等办公软件中，进一步提升使用体验。
既然我们已经通过 Demo 亲眼见证了其出色的实际效果，那么接下来是不是该轮到我们亲自上手部署一番了呢？也就是在自己的本地环境或者服务器上请求调用 OCR 的 SDK，从而真正实现文字识别的强大功能。值得一提的是，官方贴心地为多种编程语言都提供了详细的示例，这无疑为我们接下来的部署工作提供了极大的便利。

简单调用服务接口



当我们访问这个页面（点击此处直达），您会发现我选择了手写识别作为演示功能。在页面的右侧，有一份文档说明，建议您详细阅读以获取更多信息。实际上，在使用过程中，您并不需要填写太多信息。最关键的部分是提供图片的 ImageBase64 值。请注意，目前仅支持 PNG、JPG、JPEG 这三种图片格式，其他格式的图片将不被接受。这意味着您上传的图片必须是上述格式之一。接下来，我将为您演示如何操作。

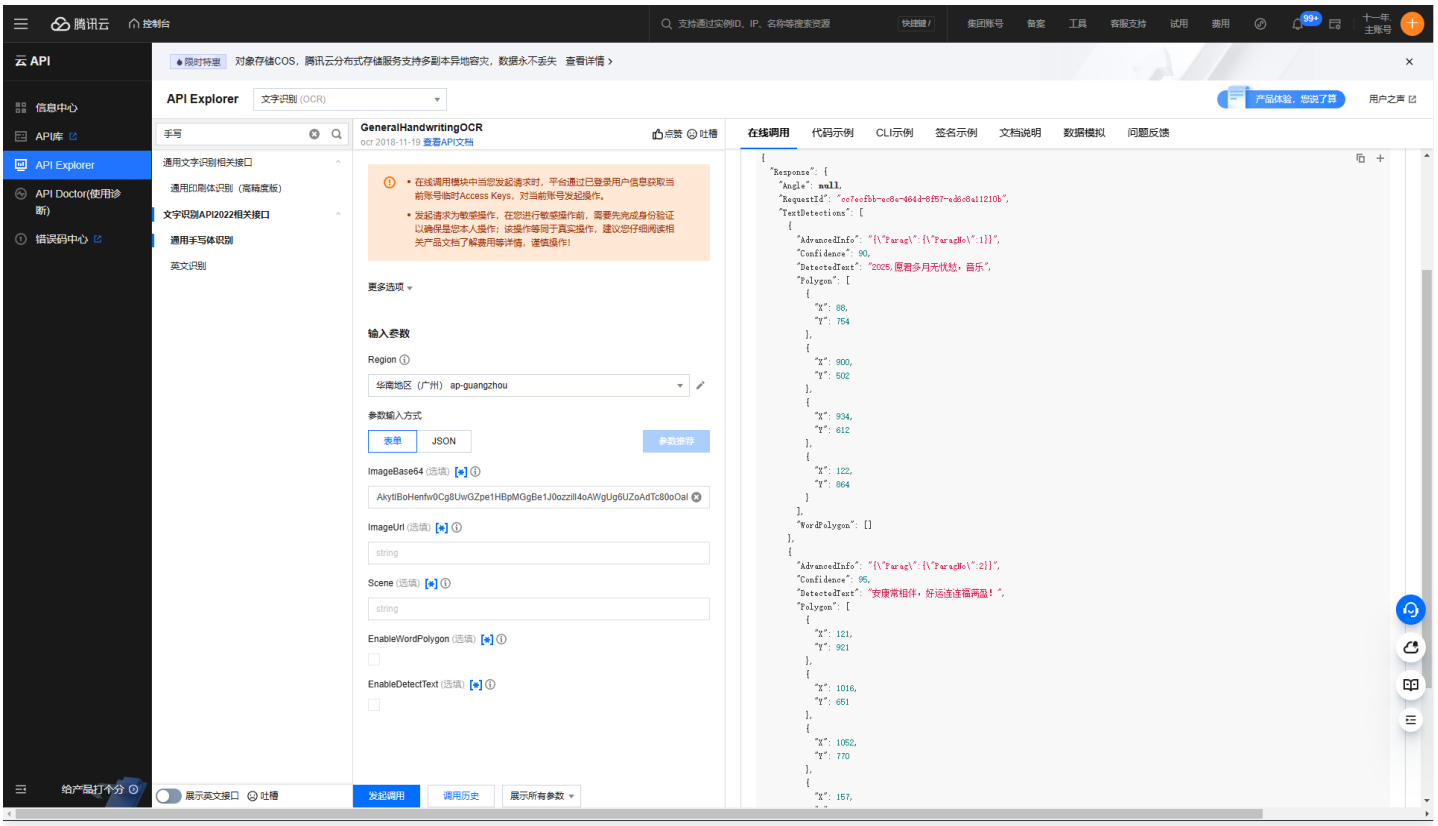


您已经选择了要上传的图片，接下来的关键步骤是获取这张图片的 ImageBase64 值。您可以轻松地通过在线工具或编程语言中的库来完成这一转换过程。为了简化操作，这里我将直接展示如何使用在线工具来转换图片为 ImageBase64 值。

转换过程非常简单，您只需上传您的图片到在线转换工具，它将自动为您生成 ImageBase64 编码。下面，我将展示这张图片转换后的 ImageBase64 值，您可以将其用于上传。同时，我也鼓励您尝试使用自己的图片进行转换，体验这一过程。

图片 ImageBase64 编码.docx

由于 ImageBase64 编码的字符数量过于庞大，导致无法在编辑器内完整展示，因此已将其以文件形式附上。您可自行下载该文件获取完整的编码内容。至此，我们已成功获取到 ImageBase64 编码，接下来的操作便十分简便。您只需将编码粘贴至相应位置，切换到“在线调用”选项卡，然后单击“发送请求”按钮，即可开启实际效果测试，直观地查看调用结果。



当您完成上述操作后，输出结果便呈现在眼前。不过，正如您所言，由于您的字迹较为潦草，导致这部分内容的识别效果并不理想，而其他文字的识别则毫无问题，准确无误。

此时，您可以留意到选项卡右侧有一个“代码示例”的入口。点击进入后，您会发现这里提供了多种编程语言的示例代码，涵盖了Golang、Python、Java、C++、node.js、PHP、.Net等。这些丰富的示例代码无疑为我们的开发工作提供了极大的便利，让我们能够更加便捷地将相关功能集成到自己的项目中，大大提高了开发效率。

确认本地依赖环境满足以下条件：

编程环境	SDK 集成要求
Node.js	需要7.10.1版本及以上
Python	需要2.7至3.6版本
Java	需要 JDK 7版本及以上
Go	需要 Go 1.9版本及以上
.Net	需要 .NET Framework 4.5+ 和 .NET Core 2.1
PHP	需要5.6.33版本及以上
C++	需要 GCC 4.8版本及以上的 C++编辑器和 cmake 3.0版本及以上的编译工具，暂时仅支持 Linux 环境，不支持 Windows 环境

在进行后续操作之前，务必要留意本地依赖环境的版本号。请确保安装与本地依赖环境相匹配的腾讯云文字识别SDK版本，这是保障服务能够稳定、高效运行的关键前提。只有版本适配，才能避免因环境差异导致的兼容性问题，从而确保文字识别功能在本地环境中顺畅地发挥其应有的作用。

1. 安装依赖

```
pip install Flask tencentcloud-sdk-python
```

2. 创建 Flask 应用 (app.py)

```
from flask import Flask, request, render_template, jsonify
import json
import base64
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from tencentcloud.ocr.v20181119 import ocr_client, models

app = Flask(__name__)

def initClient():
    cred = credential.Credential("您的SecretId", "您的SecretKey")
    httpProfile = HttpProfile()
    httpProfile.endpoint = "ocr.tencentcloudapi.com"
    clientProfile = ClientProfile()
    clientProfile.httpProfile = httpProfile
    return ocr_client.OcrClient(cred, "ap-guangzhou", clientProfile)

@app.route('/', methods=['GET', 'POST'])
```

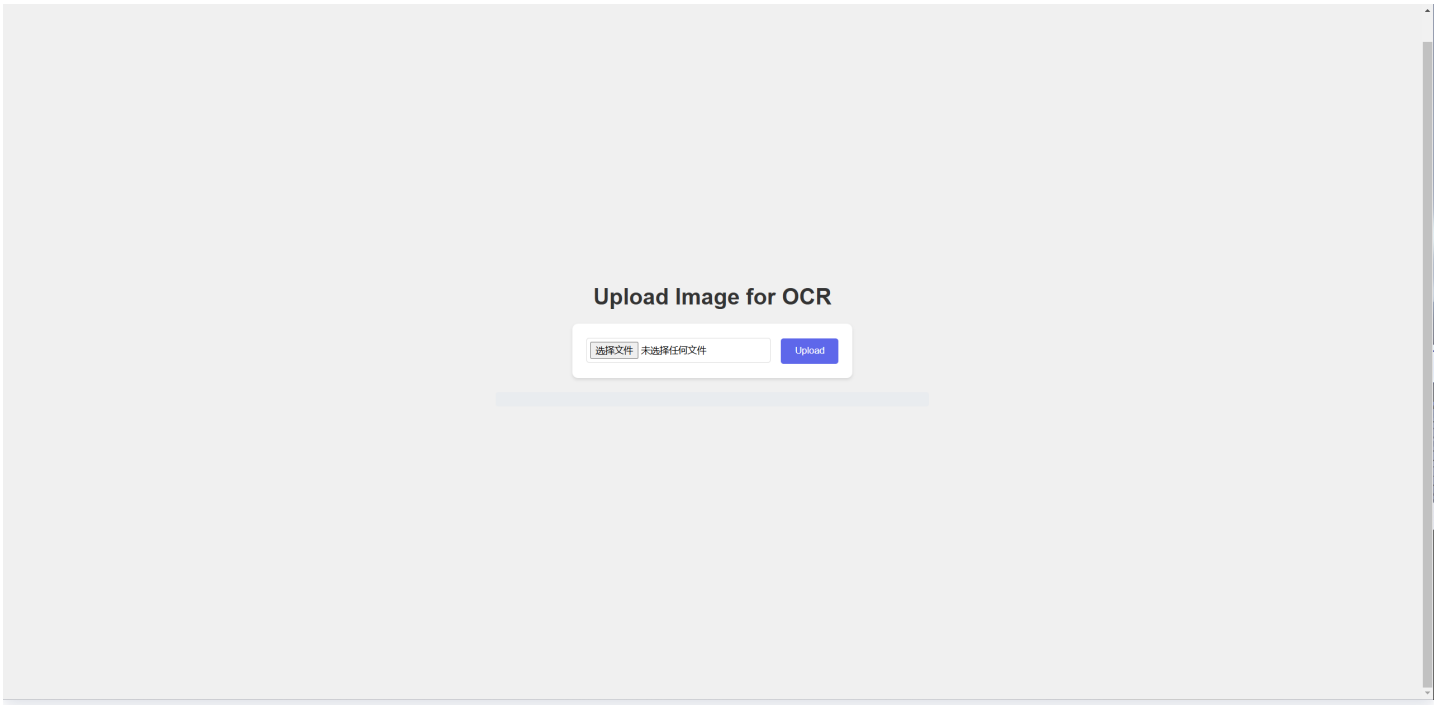
```
def upload_file():
    if request.method == 'POST':
        f = request.files['file']
        if f:
            try:
                client = initClient()
                req = models.GeneralHandwriting OCR Request()
                params = {
                    "ImageBase64": imageToBase64(f)
                }
                req.from_json_string(json.dumps(params))
                resp = client.GeneralHandwriting OCR (req)
                return jsonify(json.loads(resp.to_json_string()))
            except Exception as err:
                return jsonify({"error": str(err)})
    return render_template('upload.html')

def imageToBase64(file_object):
    binary_data = file_object.read()
    base64_encoded_data = base64.b64encode(binary_data)
    base64_message = base64_encoded_data.decode('utf-8')
    return base64_message

if __name__ == '__main__':
    app.run(debug=True)
```

3. 创建 HTML 模板 (templates/upload.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Upload Image for  OCR </title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background-color: #f0f0f0;
            margin: 0;
            padding: 20px;
            display: flex;
            flex-direction: column;
```

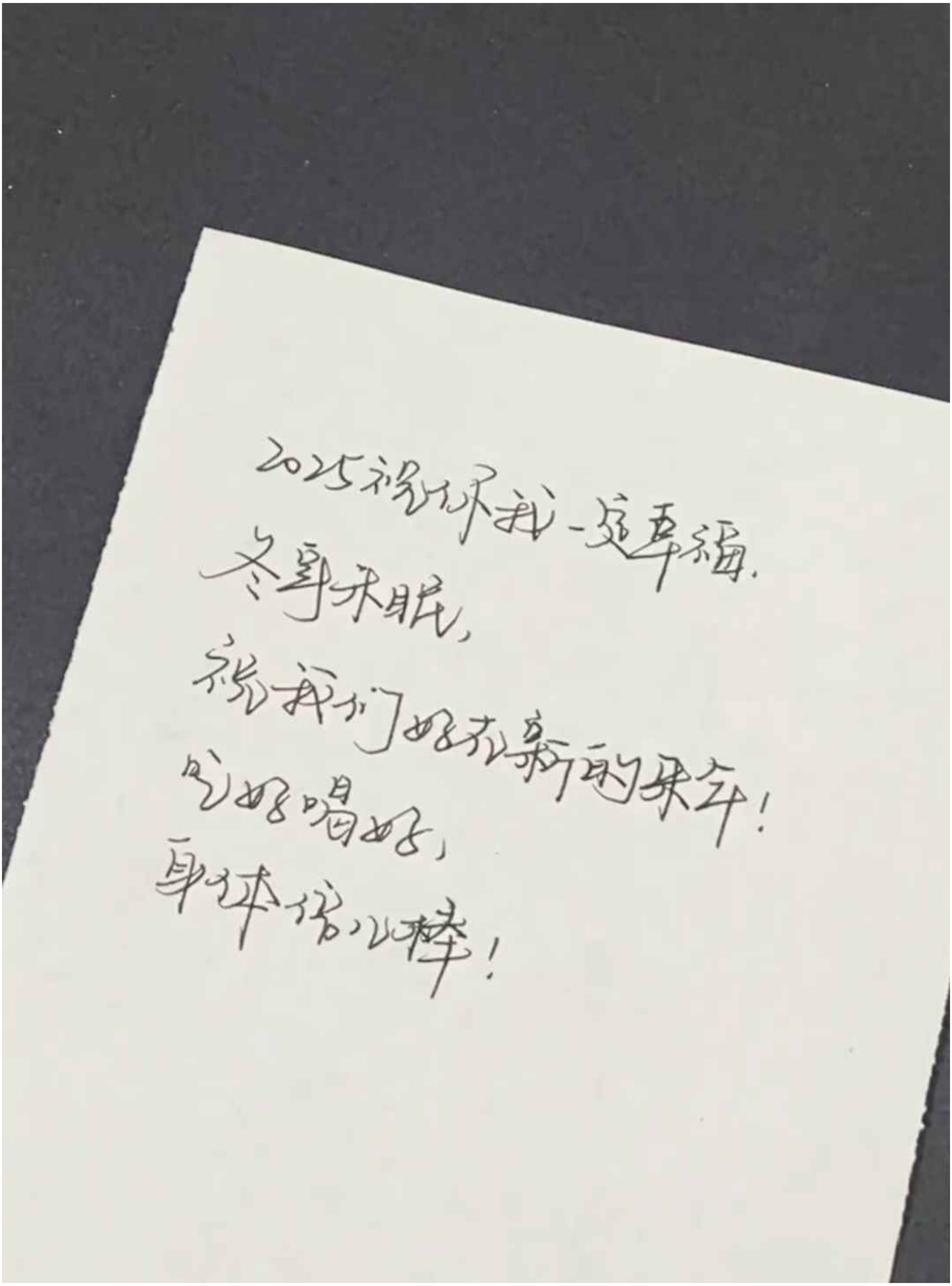
```
            align-items: center;
            justify-content: center;
            height: 100vh;
        }
        h1 {
            color: #333;
            margin-bottom: 20px;
        }
        form {
            background: white;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
        }
        input[type="file"] {
            margin-right: 10px;
            border: 1px solid #ddd;
            border-radius: 4px;
            padding: 5px;
        }
        input[type="submit"] {
            background-color: #5c67f2;
            color: white;
            border: none;
            padding: 10px 20px;
            border-radius: 4px;
            cursor: pointer;
            transition: background-color 0.3s;
        }
        input[type="submit"]:hover {
            background-color: #4a54e1;
        }
        #result {
            margin-top: 20px;
            width: 100%;
            max-width: 600px;
            padding: 10px;
            background-color: #e9ecef;
            border-radius: 4px;
            overflow-x: auto;
            white-space: pre-wrap;
            word-wrap: break-word;
        }
```

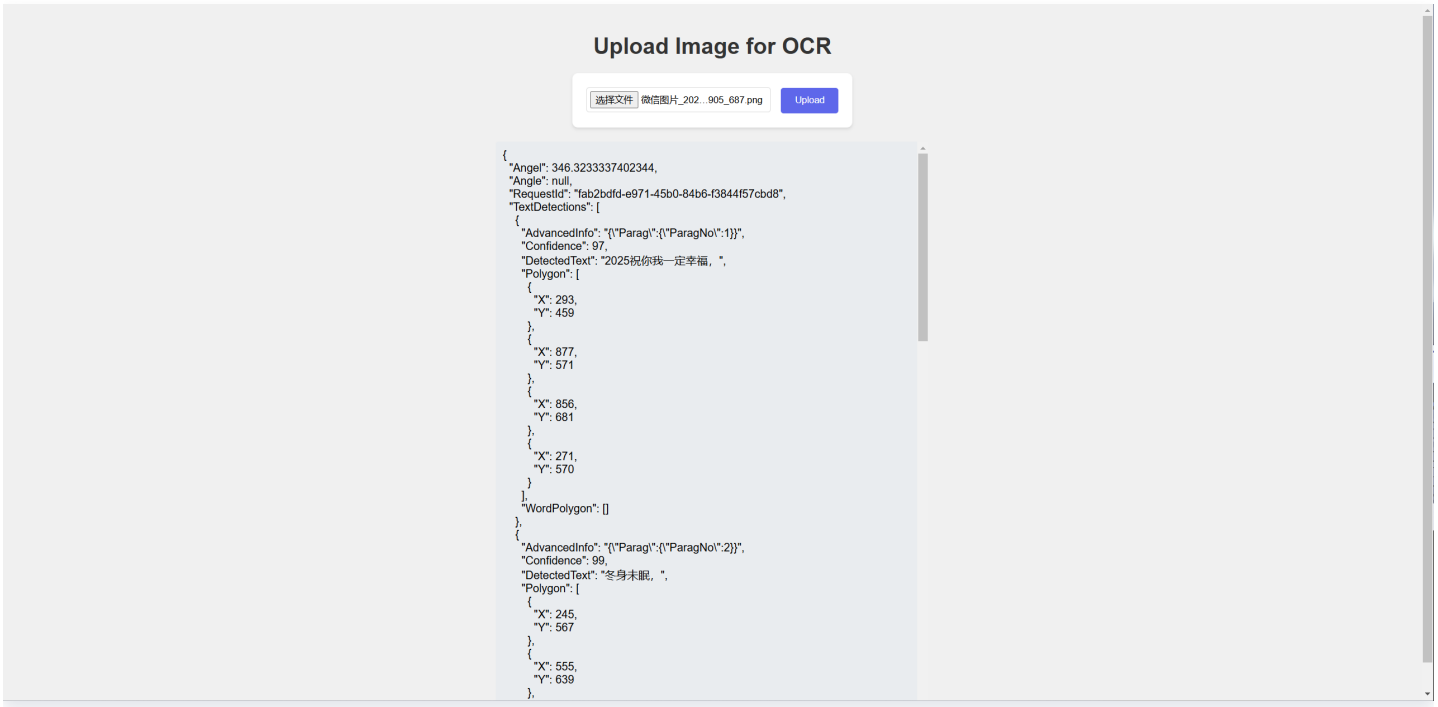
```
    </style>
</head>
<body>
    <h1>Upload Image for  OCR </h1>
    <form action="" method="post" enctype="multipart/form-data">
        <input type="file" name="file" required>
        <input type="submit" value="Upload">
    </form>
    <div id="result"></div>
    <script>
        document.querySelector('form').onsubmit = function(event) {
            event.preventDefault();
            const formData = new FormData(this);
            fetch('/', {
                method: 'POST',
                body: formData
            }).then(response => response.json())
            .then(data => {
                document.getElementById('result').innerText =
JSON.stringify(data, null, 2);
            })
            .catch(error => {
                document.getElementById('result').innerText = 'Error: '
+ error.message;
            });
        };
    </script>
</body>
</html>
```



我打造了一个简洁的前端界面，使其不再显得空旷。虽说是随意构思，但仅用几行 CSS 和 JS 代码便实现了效果。经过严谨的实际测试，确认所有代码精准无误，在使用过程中未出现任何重大问题，大家可以放心直接使用。当然，如果觉得这个前端界面过于简单，也可自行重新设计一个。接下来，让我们检验一下它上传图片时是否能够准确识别吧。

这次我选取了之前另一张手写体图片来进行识别测试。在识别过程中，我同样对效果充满好奇，不知其准确度如何。现在，就让我们一同揭晓答案。图片也已附在下方，大家可以仔细查看，核对识别结果是否精准无误。


2025祝你我一定幸福.
冬季不眠,
祝我们好在新的来年!
吃好喝好,
身体倍儿棒!



由于篇幅有限，这里仅展示这张图片。前端输出的样式，我个人认为相当出色，毕竟对于实用工具来说，简洁的界面往往是最佳选择。大家可以对比图片仔细查看，我经过核对，发现识别结果并无差错。因篇幅限制，只能展示这张图，但各位完全可以亲自动手尝试一番。 不得不提，OCR 的手写体识别功能确实表现出色，值得大力夸赞。这段话也是我送给各位的 2025 年祝福，愿大家在新的一年里，都能顺利达成自己心中的目标，收获满满。

实际上，我并未深入讲解接口调用的细节，原因在于我认为凭借参考代码示例与文档说明，大家已足以清晰地理解其原理与操作。因此，本文直接附上代码，未对代码进行逐行解释。但我认为，在学习与实践过程中，大家应多投入自己的思考，深入探究代码背后的逻辑与原理。

结语

在本文的探索之旅中，我们一同领略了腾讯云文档智能 OCR 的强大功能和应用潜力。从基础的文本识别到复杂的文档处理，这款产品无疑为我们的工作和生活带来了前所未有的便利。通过精心设计的前端界面和稳健的后端支持，我们不仅提升了交互体验，也确保了功能的高效与准确。

随着技术的不断进步，OCR 技术的应用场景将越来越广泛，其重要性也日益凸显。我们期待腾讯云文档智能 OCR 在未来能够持续优化，加入更多创新功能，如公式格式的保留、更丰富的API支持等，以满足用户日益增长的需求。

在此，我想对每一位读者表达最诚挚的感谢。感谢您们的关注与支持，也感谢您们在探索过程中的耐心与热情。希望本文能为您提供有价值的信息和启发，也希望您能在实践中发现更多 OCR 技术的魅力。

玩转 OCR|智能 Excel 数据分析助手

最近更新时间：2025-02-14 15:04:12

项目背景

项目地址：Excel智能分析助手 。
本项目旨在构建一个智能化的 Excel 数据分析助手，通过结合 OCR 技术和自然语言处理，实现从图片到数据分析的端到端解决方案。通过腾讯云的 OCR 技术，将图片中的数据转化为可分析的数字资产。再加上混元大模型的自然语言的解析能力与 DuckDB 的高性能查询能力相结合，实现了自动化字段解析、数据类型推断与高效数据入库等功能。无论是对复杂数据的快速处理，还是多源数据的灵活支持，满足用户对实时性和准确性的需求。
在当今数字化时代，大量的业务数据仍以 Excel 表格和纸质文档的形式存在。如何高效地将这些数据转化为可分析的数字资产，是很多企业和个人面临的挑战。

解决方式

本项目旨在构建一个智能化的 Excel 数据分析助手，通过结合 OCR 技术和自然语言处理，实现从图片到数据分析的端到端解决方案。
通过腾讯云的 OCR 技术，将图片中的数据转化为可分析的数字资产。在加上混元大模型的自然语言的解析能力与 DuckDB 的高性能查询能力相结合，实现了自动化字段解析、数据类型推断与高效数据入库等功能。无论是对复杂数据的快速处理，还是多源数据的灵活支持，系统均能高效响应，满足用户对实时性和准确性的需求。
同时，项目提供了轻量化的分析引擎，在处理数据的同时保持资源占用的低门槛，让用户无需掌握复杂的技术知识即可完成高质量的数据分析。

主要功能

1. 数据输入模块
    - Excel 文件直接读取
    - 图片 OCR 表格识别
    - 数据预处理和清洗
2. 分析引擎模块
    - 自动生成分析报告
    - 智能对话分析
    - 统计分析功能
    - 可视化图表生成
3. 交互界面模块
    - 文件上传和预览
    - 智能对话交互
    - 图表定制生成

- 分析报告导出

4. 可视化实现

支持11种专业图表类型：

- 基础图表
  - 折线图：展示趋势变化
  - 柱状图：数值比较
  - 散点图：相关性分析
  - 饼图：占比展示
- 统计图表
  - 箱线图：分布特征
  - 直方图：频率分布
  - 热力图：相关性矩阵
- 高级图表
  - 3D散点图：三维关系
  - 气泡图：多维数据
  - 堆叠图：层次关系
  - 面积图：累积趋势

Excel 智能分析助手

上传Excel文件或图片，开始智能分析对话

数据分析　数据可视化

上传数据文件

微信图片_20250111120136.jpg　413.5 KB

数据概览

基本信息

- 总行数：39
- 总列数：8
- 列名：序号, 学号, 姓名, 专业班级, 加权平均成绩, 学年操评, 学年综合成绩, 是否符

数据类型

```
序号          int64
学号          int64
姓名         object
专业班级       object
加权平均成绩    float64
学年操评       float64
学年综合成绩    float64
是否符        float64
```

完整数据

序号	学号	姓名	专业班级	加权平均成绩	学年操评	学年综合成绩	是否符

智能对话

Chatbot

前3个人的学年综合成绩的平均数

前3个人的学年综合成绩分别为89.530769、89.438462和87.153846。计算平均数为：

$$\text{平均数} = \frac{89.530769 + 89.438462 + 87.153846}{3} = \frac{266.123077}{3} = 88.707692$$

所以，前3个人的学年综合成绩的平均数是88.71。

分析哪些孩子是比较突出的

数据洞察

1 关键特征和模式：

Textbox

请输入你的问题...

发送

生成分析报告 导出报告

1 数据分析报告

1 基本信息

- 总行数：39
- 总列数：8
- 数据维度：(39, 8)

2 数据类型分析

```
序号          int64
学号          int64
姓名         object
专业班级       object
加权平均成绩    float64
学年操评       float64
学年综合成绩    float64
是否符        float64
```

3 基本统计分析

	序号	学号	加权平均成绩	学年操评	学年综合成绩	是否符
count	39	39	39	39	39	0
mean	20	2.02304e+09	84.8333	2.91154	82.7782	nan
std	11.4018	185872	2.26935	1.69576	2.65559	nan
min	1	2.02225e+09	80.7981	0.2	79.4615	nan
25%	10.5	2.02308e+09	82.9856	1.9	80.7038	nan

Excel 智能分析助手

上传Excel文件或图片，开始智能分析对话

数据分析 数据可视化

选择图表类型

折线图

X轴数据

姓名

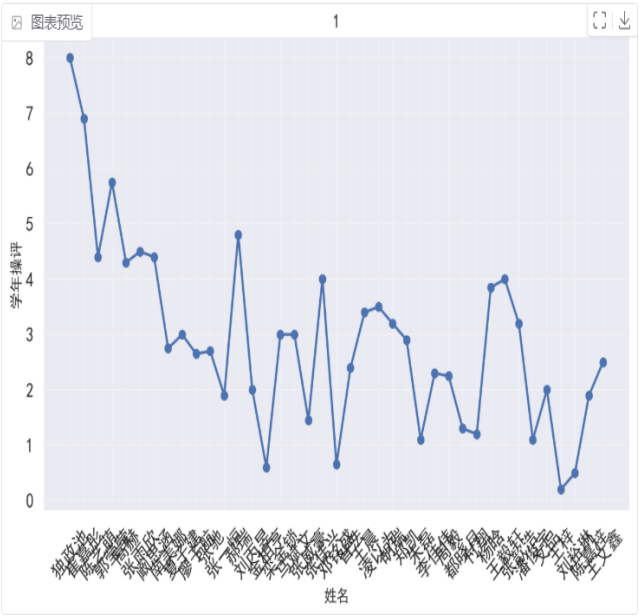
Y轴数据

学年操评

图表标题

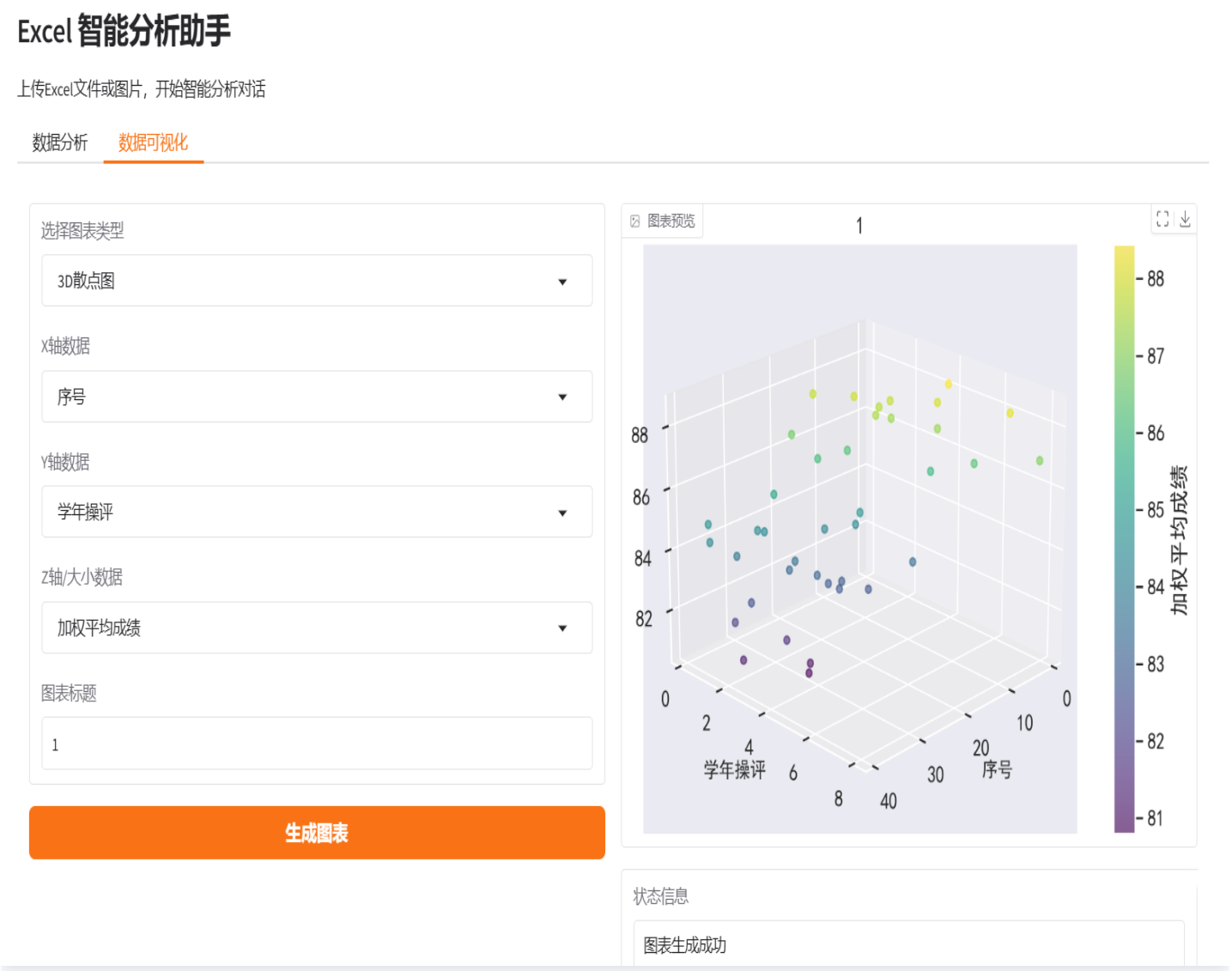
1

生成图表



状态信息

图表生成成功



主要技术实现

如何使用腾讯云 OCR

官方文档：API Explorer – 云 API – 控制台

OCR 识别表格

代码

```
class OcrTableAccurate:
    def __init__(self, secret_id, secret_key):
        self.secret_id = secret_id
        self.secret_key = secret_key

    def recognize_table_accurate(self, image_path):
        # 读取图片文件
        with open(image_path, "rb") as img_file:
```

```
            img_data = img_file.read()

        # 将图片转换为Base64编码
        image_base64 = b64encode(img_data).decode()

        # 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey
        cred = credential.Credential(self.secret_id, self.secret_key)

        # 实例化一个http选项，可选的，没有特殊需求可以跳过
        httpProfile = HttpProfile()
        httpProfile.endpoint = "ocr.tencentcloudapi.com"

        # 实例化一个client选项，可选的，没有特殊需求可以跳过
        clientProfile = ClientProfile()
        clientProfile.httpProfile = httpProfile
        # 实例化要请求产品的client对象,clientProfile是可选的
        client = ocr_client.OcrClient(cred, "ap-guangzhou",
clientProfile)

        # 实例化一个请求对象,每个接口都会对应一个request对象
        req = models.RecognizeTableAccurateOCRRequest()
        params = {
            "ImageBase64": image_base64
        }
        req.from_json_string(json.dumps(params))

        # 返回的resp是一个RecognizeTableAccurateOCRResponse的实例，与请求对象
对应
        resp = client.RecognizeTableAccurateOCR(req)
        # 输出json格式的字符串回包
        return json.loads(resp.to_json_string())
```

图片转 Excel（解析识别的参数）

```
table_data = result['TableDetections'][0]['Cells']  # 获取单元格数据
```

参数

```
result = {
    'TableDetections': [  # 检测到的所有表格列表
        {
            'Cells': [      # 单个表格中的所有单元格
```

```
                {
                    # 单元格位置参数
                    'RowTl': 0,   # 行的起始位置（Top-Left，左上角），0行
                    'ColTl': 0,   # 列的起始位置（Top-Left，左上角），0列。
                    'RowBr': 1,   # 行的结束位置（Bottom-Right，右下角）。
                    'ColBr': 1,   # 列的结束位置（Bottom-Right，右下角）。
                    'Text': '单元格文本内容'   # 识别出的文本内容
                    'Type': 单元格的类型。
                    'Confidence': 置信度或识别准确度。
                    'Polygon': 定义单元格的多边形边界。此处定义表格中的4个点的位
置。顺时针

                },
                # ... 更多单元格
            ]
        }
    ]
}
```

将识别的参数转化为 DataFrame

```
# 找出表格的最大行列数
                max_row = max(cell['RowBr'] for cell in table_data)
                max_col = max(cell['ColBr'] for cell in table_data)

                # 创建空的二维数组
                rows = [['' for _ in range(max_col)] for _ in
range(max_row)]

                # 填充数据
                for cell in table_data:
                    row_start = cell['RowTl']
                    col_start = cell['ColTl']
                    text = cell['Text']
                    # 处理合并单元格
                    row_end = cell['RowBr']
                    col_end = cell['ColBr']

                    # 填充所有涉及的单元格
                    for r in range(row_start, row_end):
                        for c in range(col_start, col_end):
                            rows[r][c] = text
```

解析数据类

ExcelToDuckDB 是一个 Python 类，用于将 Excel 数据文件快速解析并导入到 DuckDB 数据库中。

通过混元大模型构建智能体

如何使用混元大模型的 API：腾讯混元大模型 简介-API 中心-腾讯云

```
    def chat_completions(self, input_text, template_name=None):
        """
        发送聊天请求
        :param input_text: 输入文本或SQL查询结果
        :param template_name: 模板名称
        :return: API响应或错误信息
        """
        try:
            # 实例化认证对象
            cred = credential.Credential(self.secret_id,
self.secret_key)
            httpProfile = HttpProfile()
            httpProfile.endpoint = "hunyuan.tencentcloudapi.com"
            clientProfile = ClientProfile()
            clientProfile.httpProfile = httpProfile

            # 实例化客户端
            client = hunyuan_client.HunyuanClient(cred, "ap-guangzhou",
clientProfile)
            req = models.ChatCompletionsRequest()

            # 构建系统提示词
            system_prompt = self.get_prompt_template(template_name)

            # 构建请求参数
            params = {
                "Messages": [
                    {
                        "Role": "system",
                        "Content": system_prompt
                    },
                    {
                        "Role": "user",
                        "Content": input_text
                    }
                ],
```

```
        "Model": "hunyuan-turbo-20241120"
    }

    req.from_json_string(json.dumps(params))
    resp = client.ChatCompletions(req)
```

Prompt 模板

构建的提示词：会根据输入问题的分析是否需要深入回答。

简单回答


```
def _build_data_context(self, message):
        """构建数据上下文"""
        basic_stats = {
            'total_rows': self.current_df.shape[0],
            'total_cols': self.current_df.shape[1],
            'null_count': self.current_df.isnull().sum().sum(),
            'numeric_cols': self.current_df.select_dtypes(include=
['int64', 'float64']).columns.tolist(),
            'categorical_cols': self.current_df.select_dtypes(include=
['object']).columns.tolist()
        }

        # 使用模型判断问题类型
        question_analysis_prompt = f"""
        请分析以下问题的类型，判断是否需要深度分析。

        问题：{message}

        请只回答"简单"或"深入"。
        - 简单：如计算平均值、求和、查找最大最小值等基础操作。
```

```
            - 深入：如分析趋势、寻找关系、提供建议、评估情况等需要综合分析的问题
            """

            # 调用模型判断
            analysis_type =
self.analyzer.chat.chat_completions(question_analysis_prompt)
            needs_deep_analysis = "深入" in
self._format_response(analysis_type)

            # 基础数据信息
            data_context = f"""
            数据基本信息：
            - 数据规模：{basic_stats['total_rows']}行 ×
{basic_stats['total_cols']}列
            - 缺失值：{basic_stats['null_count']}
            - 数值列：{', '.join(basic_stats['numeric_cols'])}
            - 分类列：{', '.join(basic_stats['categorical_cols'])}

            数据内容：
            {self.current_df.to_string()}
            """

            # 根据模型判断结果添加分析要求
            if needs_deep_analysis:
                data_context += """
                分析要求：
                1. 数据洞察
                - 识别关键特征和模式
                - 发现数据中的趋势
                - 分析异常和特殊情况
                2. 统计分析
                - 提供相关统计指标
                - 解释数据分布特征
                - 分析相关性（如适用）
                3. 业务建议
                - 提供数据驱动的见解
                - 给出可行的改进建议
                - 指出潜在的机会
                4. 可视化建议
                - 推荐合适的图表类型
                - 建议关注的关键维度
                5. 改进建议
                - 指出数据质量问题
```

```
            - 建议额外需要的数据
            - 提供优化分析的方向
            """
        else:
            data_context += """
            回答要求：
            - 直接回答问题
            - 确保计算准确
            - 使用简洁的语言
            """

        data_context += f"\n用户问题：{message}"

        return data_context
```

应用场景

1. 企业数据分析
   - 销售数据分析
   - 财务报表分析
   - 运营数据分析
   - 市场调研分析
2. 个人数据处理
   - 成绩单分析
   - 消费记录分析
   - 个人财务分析
   - 数据整理归档

结语

Excel 智能分析助手通过结合 OCR 技术和自然语言处理，为用户提供了一个简单易用的数据分析工具。无论是数据的导入转换，还是分析可视化，都能以智能化的方式完成，大大提高了数据分析的效率。