

## 太保OCR项目经验总结

太保OCR项目是我用通用的定位模型和识别模型完成OCR任务的首次尝试，在这里记录一下做这个项目过程中积累的一些经验和trick。

总体思路如下

1. 先用通用定位+识别的API获得初步的定位和识别结果；
2. 利用要提取字段的特征完成字段的提取；
3. 根据提取字段的候选字符对识别的结果做修正。

步骤1直接调用API接口，不再赘述；步骤3是对识别出的概率矩阵加mask，将候选字符以外的概率值全部置0，然后再进行greedy decode，此外还可以根据固定位数、常见字串等先验信息进行修正。后面主要描述字段提取部分。

### 批单、确认书字段提取方案：

**保险合同退保业务批单（暨领款通知书）**

批单号: 901906089027271      受理日期: 2019年05月08日

尊敬的杜超辉：  
感谢您选择了我公司的产品及服务。您申请办理的保全业务，经我们审核后回复如下：

【申请项目】退保  
保单号：SHJ 061EL0267930

主/附	险种名称	被保险人	退保份数	退费净额	退税金
主险	太平盛世·长安安康终身寿险(B)	杜超辉	10.00	2,006.00	-

【实付退保金合计】人民币贰仟零陆元整    ¥2006.00  
其中【退税金合计】人民币零元整    ¥0.00

您选择的领款方式：银行集中转账  
户名：杜超辉    证件号码：130225197401140020  
开户行：中国建设银行    银行账号：6227000181610278005

1、退保会给您造成一定的损失，请您慎重决策。如为退保，以上保险合同于申请之日起，保险责任即行终止；如为减保，我司按减保后合同约定的保险金额或份额承担保险责任。  
2、申请人承诺以上保险合同并无任何转让、抵押之事实；申请人未有破产和涉及与以上保险合同的诉讼事项。

经办人: 刘迎迎      打印日期: 2019年05月08日

批单和确认书需要识别的字段类似，分别是批单号/受理单号和受理日期。

批单号/受理单号提取逻辑：

1. box从上到下依次排序，取前k个box的结果；

2. 取box的左上点p1位于图片左侧的box，过滤掉  $p1 > \text{ration} * \text{image\_width}$  的box；
3. 在剩余的box中，选取数字字符最长的box。

受理日期字段的提取方法类似，该方法比先定位key再取value的方案更加简洁有效，主要利用了提取字段与其他无关字段在内容上（数字位数很多）有较大的区分度。

## 保单字段提取方案



除上述描述的情况以外还存在一些难点：

1. key字段与value字段有时在同一个box，有时在两个box；
2. key字段因为模板不同导致内容是不一样的，如险种字段的内容：“投保险种名称”、“险种名称及款式”；
3. key字段在有些情况除了真正需要提取的内容外，还在其他信息，如身份证号字段有时在某些模板会和性别、出身日期在同一个box内；
4. 被保险人身份证号、投保人身份证号在有些模板是不存在的；

具体方法如下：

1. 将定位输出的boxes按位置从上到下排序，确定字段出现的大致范围[min\_index, max\_index];
2. 计算box内识别的内容的前k个字符与key字段的编辑距离，如：key字段为“保险单号”，某个box识别的内容为“保险早号：123456789”，因key字段有4个字符，即k=4，则编辑距离为1；
3. 对所有[min\_index, max\_index]内的所有box识别内容计算一遍编辑距离，取编辑距离最小的box；
4. 判断value与key是否在同一box；
5. 若key与value不在同一box内，查找value所在的box（图片有角度，不能直接往右找）：
  1. 获取所有box的方向，用聚类算法过滤掉方向差异较大的box（可能是水印）；
  2. 计算出text angle，根据key box的上下边界，确定text line region；
  3. 计算[min\_index, max\_index]内的box与text line region交的面积，比去该box自身的面积，比值大于一定阈值加入候选box；
  4. 后候选box内去距离key box最近的box。

实施的过程中还用上了其他一些trick，没什么通用性，不赘述。

## 分类方案：

分类的任务是将所有表单分为保单、批单、确认书3类。同样是因为样本量的原因不宜用分类模型，保单模板不固定，也不宜用模板匹配。观察了大量样本后确定的方案如下：

1. 对于每一类单据，挑选出k个区分度较大的文字片段作为text anchor，如保单的anchor为{"保险保险单", "保险单号", "出生日期", "个人"}，批单的anchor为{"保险合同", "批单号", "受理日期", "通知书"}，确认书的anchor为{"保险合同", "确认书", "受理单号", "受理日期"}；
2. 对每个测试样本，对box按从上到下排序，取前n个定位识别结果；
3. 对每个测试样本的n个识别结果，判断anchor存在情况，如某个样本的保单、批单、确认书的anchor判别结果为[1, 1, 1, 0], [0, 0, 0, 1], [0, 0, 1, 0]，1表示存在，0表示不存在；
4. 对每类anchor的判别结果求和，如上述的和为（3, 1, 1），则将该样本判定为保单。

PS：

1. 对于每一类的anchor可适当多取几个；
2. 对每类anchor可同时设置一个权重序列，使得区分度越大的文字片段有较大的权重，如保单的anchor中，可以将“保险保险单”的权重设大，因为其区分度很高。

其他：

## pdf 转图片

转换过程主要考虑的问题是是否会有信息损失，有两方面：

1. 分辨率不够；
2. 字体发生变化；

对于第一点，转换时指定较大的分辨率即可。在本次项目中使用的是一个Python库，GitHub地址为：<https://github.com/cnych/pdf2images>，该Python库进一步依赖wand、pypdf2等库，同时需要系统库方面的支持，主要有ImageMagick、GhostScript等，安装好这些依赖即可正常使用。

对于第二点，问题比较复杂。正常情况下，生成PDF文件时，一般会将原文件中使用的字体或字体子集一起嵌入到PDF文件中（可以在Adobe reader里的文件->属性->字体中查看），但由于字体的版权问题和一些设置上的原因，有可能使原文件中使用的字体没有嵌入到PDF，这时系统如果也没有相应字体，在转换过程中，GhostScript就会去找近似的字体去代替。但是GhostScript似乎不太智能，即使我已经把缺失的字体放在了系统字体路径里（/usr/share/fonts/），GhostScript转换PDF时依然没用上该字体。后来查找资料发现可以指定字体名到字体路径的映射。配置文件的路径为/usr/share/ghostscript/8.70/Resource/Init/cidmap.GS，如缺失的字体名为SimSun，配置语句可类似这样写：

```
/SimSun << /FileType /TrueType /Path (/usr/share/fonts/simsun.ttf)
/SubfontID 0 /CSI [(GB1) 4] >>;
```

具体语法可参考<http://www.voidcn.com/article/p-fdqt dali-rx.html>，这里摘录重要的几句：

cidmap 文件中对格式有详悉的解释，很简单。我这里只想说一下最后方括号中的两个值，第一个称为ordering，对简体中文永远是(GB1)，繁体是(CNS1)。第二个称为 supplement，相当于版本号，简体中文有 0-4 五个值。大体上 0和1对应GB2312，2和3 对应GBK，4对应GB18030。具体用那一个取决字体包含多少字符，如文鼎的gbsn00lp.ttf 应该用0，simsun.ttc应该用2，SimSun18030应该用4。