

第四届“泰迪杯” 全国数据挖掘挑战赛

优
秀
作
品

作品名称：基于深度学习和语言模型的印刷文字 OCR 系统

荣获奖项：特等并获企业冠名奖

作品单位：华南师范大学

作品成员：苏剑林 曾玉婷

基于深度学习和语言模型的印刷文字 OCR 系统

苏剑林 华南师范大学数学科学学院

曾玉婷 华南师范大学数学科学学院

2016 年 5 月 15 日

中文摘要

我们设计了一系列的算法,完成了文字特征提取、文字定位等工作,并基于卷积神经网络 (CNN) 建立了字符识别模型,最后结合统计语言模型来提升效果,成功构建了一个完整的 OCR(光学字符识别)系统。

在特征提取方面,我们抛弃了传统的“边缘检测 + 腐蚀膨胀”的方法,基于一些基本假设,通过灰度聚类、图层分解、去噪等步骤,得到了良好的文字特征。这部分文字特征既可以用于第二步做文字定位,又可以直接输入到第三步的模型中进行识别,而不用做额外的特征提取工作。

在文字定位方面,我们通过邻近搜索的方法先整合特征碎片,得到了单行的文字特征,然后通过前后统计的方法将单行的文字切割为单个字符。测试表明,这种切割思路能够很好地应对中英文混排的文字切割。

在光学识别方面,我们基于 CNN 的深度学习模型建立了单字识别模型,自行生成了 140 万的样本进行训练,最终得到了一个良好的单字识别模型,训练正确率为 99.7%,测试正确率为 92.1%,即便增大图片噪音到 15%,也能有 90% 左右的正确率。

最后,为了在前面的工作的基础上再次提升效果,我们结合了语言模型,通过微信的数十万文本计算了常见汉字的转移概率矩阵,由 Viterbi 算法动态规划,得到最优的识别组合。

将以上四部分工作结合起来,就是一个完整的 OCR 系统。经过测试,我们的系统对印刷文字的识别有着不错的效果,可以作为电商、微信等平台的图片文字识别工具。

关键词: 光学字符识别,特征提取,文本定位,卷积神经网络,深度学习,语言模型

Abstract

In this article, we design a series of algorithm to extract features and position text. Next we use convolutional neural network to train a character recognition system. And then we use language model to improve recognition effect. Based to the above steps, we achieve a complete OCR (Optical Character Recognition) system.

For feature extraction, we discover a new approach better than traditional way which is based on boundary detection and dilation-erosion. According to some fundamental assumptions, we gain excellent text features via grey clustering, layer decomposition, noise reduction, and so on. The features we gain can not only be use for text poistioning at step II , but also text recognition at step III.

For text positioning, we integrate the feature patches via neighbor searching, and gain the features of single line texts. Then we use a statistic way to cut the single line into single character. Our result show that this way can work well even if Chinese and English mixed in the one line.

And for optical recognition, we use convolutional neural network to build up our model for single character and train it with 1.4 milion samples produced by ourselves. Fortunately, we gain a good model which has a 99.7% train accurary, 92.7% test accurary, even a 90% accurary for the samples who has 15% noise.

Finally, for the better result, we use language model to improve our work. We calculate the probability transition matrix from hundreds of thousands wechat articles, and use Viterbi algorithm to dynamicly produce the optimal result.

Combined the above works, we gain a complete OCR system. And the result show that our system work well for the printed text recognition.

Keywords : *OCR, feature extraction, text positioning, CNN, deep learning, language model*

目录

1 研究背景 1

2 建模说明 1

2.1 研究假设 1

2.2 分析流程 1

2.3 实验平台 2

3 特征提取 2

3.1 图像预处理 3

3.2 灰度聚类 3

3.2.1 核密度估计 4

3.2.2 极大极小值分割 5

3.3 逐层识别 5

3.3.1 连通性 6

3.3.2 抗腐蚀能力 6

3.3.3 池化操作 8

3.3.4 密度排除 8

3.3.5 孤立区排除 9

4 文字定位 9

4.1 邻近搜索 10

4.1.1 目的 10

4.1.2 距离 10

4.1.3 结果 11

4.2 文本切割 11

4.2.1 均匀切割 12

4.2.2 统计切割 12

4.2.3 前后比较 12

5 光学识别 12

5.1 模型选择 12

5.2 训练数据 13

5.3 模型结构 13

5.4 模型实现 14

5.5 模型检验 15

5.5.1 训练集检验 15

5.5.2 测试集检验 16

6 语言模型	16
6.1 转移概率	16
6.2 动态规划	17
6.2.1 转移概率矩阵	18
6.2.2 Viterbi 算法	18
6.3 提升效果	19
7 综合评估	19
7.1 数据验证	19
7.2 模型综述	19
7.3 结果反思	19
参考文献	21

1 研究背景

关于光学字符识别 (Optical Character Recognition, 下面都简称 OCR), 是指将图像上的文字转化为计算机可编辑的文字内容, 众多的研究人员对相关的技术研究已久, 也有不少成熟的 OCR 技术和产品产生, 比如汉王 OCR、ABBYY FineReader、Tesseract OCR 等. 值得一提的是, ABBYY FineReader 不仅正确率高 (包括对中文的识别), 而且还能保留大部分的排版效果, 是一个非常强大的 OCR 商业软件.

然而, 在诸多的 OCR 成品中, 除了 Tesseract OCR 外, 其他的都是闭源的、甚至是商业的软件, 我们既无法将它们嵌入到我们自己的程序中, 也无法对其进行改进. 开源的唯一选择是 Google 的 Tesseract OCR, 但它的识别效果不算很好, 而且中文识别正确率偏低, 有待进一步改进.

综上所述, 不管是为了学术研究还是实际应用, 都有必要对 OCR 技术进行探究和改进. 我们队伍将完整的 OCR 系统分为“特征提取”、“文字定位”、“光学识别”、“语言模型”四个方面, 逐步进行解决, 最终完成了一个可用的、完整的、用于印刷文字的 OCR 系统. 该系统可以初步用于电商、微信等平台的图片文字识别, 以判断上面信息的真伪.

2 建模说明

2.1 研究假设

在本文中, 我们假设图像的文字部分有以下的特征:

1. **印刷字体** 假设我们要识别的图像字体都是比较规范的印刷字体, 如宋体、黑体、楷体、行书等;
2. **对比度** 文字与背景应该有比较明显的对比度;
3. **横向排版** 在设计模型的时候, 我们假设了图片文本是横向排版的;
4. **厚度** 文字的笔画应该有一定的宽度, 不可以太细;
5. **渐变性** 同一个文字的色彩应该最多是渐变的;
6. **复杂性** 一般文字是通过比较密集的笔画成字的, 并且很多时候都具有一定的连通性.

可以看到, 这些特征都是常见的电商宣传海报等的常见特点, 因此这些假设都是比较合理的.

2.2 分析流程

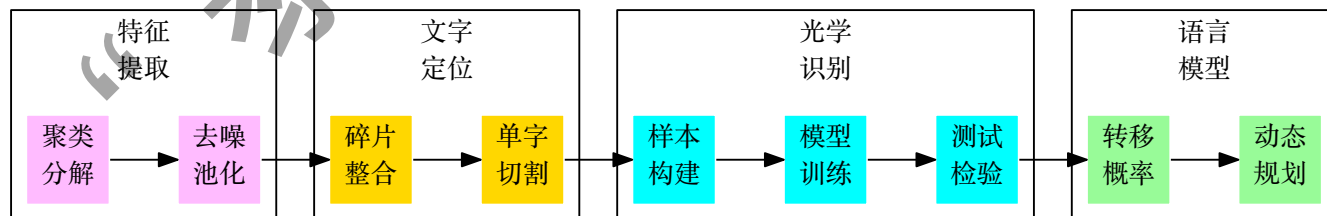


图 1: 我们的实验流程图

2.3 实验平台

本文的实验在 CentOS 7 + Python 2.7 的环境下完成. 其中, 图像处理部分用到了下列拓展库 : Numpy、SciPy、Pandas、Pillow ; 卷积神经网络模型用到了下述拓展库 : Keras、Theano. 具体的实验配置可以到 5.4 节浏览.

3 特征提取

作为 OCR 系统的第一步, 特征提取是希望找出图像中候选的文字区域特征, 以便我们在第二步进行文字定位和第三步进行识别. 在这部分内容中, 我们集中精力模仿肉眼对图像与汉字的处理过程, 在图像的处理和汉字的定位方面走了一条创新的道路. 这部分工作是整个 OCR 系统最核心的部分, 也是我们队伍工作中最核心的部分.

传统的文本分割思路大多数是“边缘检测 + 腐蚀膨胀 + 联通区域检测”, 如论文 [1]. 然而, 在复杂背景的图像下进行边缘检测会导致背景部分的边缘过多 (即噪音增加), 同时文字部分的边缘信息则容易被忽略, 从而导致效果变差. 如果在此时进行腐蚀或膨胀, 那么将会使得背景区域跟文字区域粘合, 效果进一步恶化.

因此, 在本文中, 我们放弃了边缘检测和腐蚀膨胀, 通过聚类、分割、去噪、池化等步骤, 得到了比较好的文字部分的特征, 整个流程大致如图 2, 这些特征甚至可以直接输入到文字识别模型中进行识别, 而不用做额外的处理. 由于我们每一部分结果都有相应的理论基础作为支撑, 因此能够模型的可靠性得到保证.

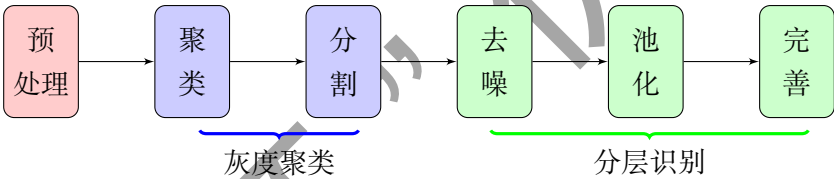


图 2: 特征提取大概流程

在这部分的实验中, 我们以图 3 来演示我们的效果. 这个图像的特点是尺寸中等, 背景较炫, 色彩较为丰富, 并且文字跟图片混合排版, 排版格式不固定, 是比较典型的电商类宣传图片. 可以看到, 处理这张图片的要点就是如何识别图片区域和文字区域, 识别并剔除右端的电饭锅, 只保留文字区域.



图 3: 小米电饭锅介绍图

3.1 图像预处理

首先, 我们将原始图片以灰度图的形式读入, 得到一个 $m \times n$ 的灰度矩阵 M , 其中 m, n 是图像的长、宽. 这样读入比直接读入 RGB 彩色图像维度更低, 同时没有明显损失文字信息. 转换为灰度图事实上就是将原来的 RGB 图像的三个通道以下面的公式整合为一个通道:

$$Y = 0.299R + 0.587G + 0.114B \quad (1)$$

图 3 的灰度图如图 4(a).

图像本身的尺寸不大, 如果直接处理, 则会导致文字笔画过小, 容易被当成噪音处理掉, 因此为了保证文字的笔画有一定的厚度, 可以先将图片进行放大. 在我们的实验中, 一般将图像放大为原来的两倍就有比较好的效果了.

不过, 图像放大之后, 文字与背景之间的区分度降低了. 这是因为图片放大时会使用插值算法来填补空缺部分的像素. 这时候需要相应地增大区分度. 经过测试, 在大多数图片中, 使用次数为 2 的“幂次变换”效果较好. 幂次变换为

$$x \mapsto x^r \quad (2)$$

其中 x 代表矩阵 M 中的元素, r 为次数, 在这里我们选取为 2. 然后将结果映射到 $[0, 255]$ 区间:

$$x \mapsto \frac{x - M_{\min}}{M_{\max} - M_{\min}} \times 255 \quad (3)$$

其中 M_{\max}, M_{\min} 是矩阵 M 的最大值和最小值. 经过这样处理后, 图像如图 4(b).



图 4: 图像的预处理

3.2 灰度聚类

接着我们就对图像的色彩进行聚类. 聚类的有两个事实依据:

1. **灰度分辨率** 肉眼的灰度分辨率大概为 40, 因此对于像素值 254 和 255, 在我们肉眼看来都只是白色;
2. **设计原则** 根据我们一般的审美原则, 在考虑海报设计、服装搭配等搭配的时候, 一般要求在服装、海报等颜色搭配不超过三种颜色.

更通俗地说, 虽然灰度图片色阶范围是 $[0, 255]$, 但我们能感觉到的整体的色调一般不多, 因此, 可以将相近的色阶归为一类, 从而减少颜色分布, 有效地降低噪音.

事实上，聚类是根据图像的特点自适应地进行多值化的过程，避免了传统的简单二值化所带来的信息损失。由于我们需要自动地确定聚类数目，因此传统的 KMeans 等聚类方法被我们抛弃了，而且经过我们测试，诸如 MeanShift 等可行的聚类方法又存在速度较慢等缺陷。因此，我们自行设计了聚类方法，使用的是“核概率密度估计”的思路，通过求颜色密度极值的方式来聚类。

3.2.1 核密度估计

经过预处理的图像，我们可以对每个色阶的出现次数进行统计，得到如图 5 的频率分布直方图：可以看

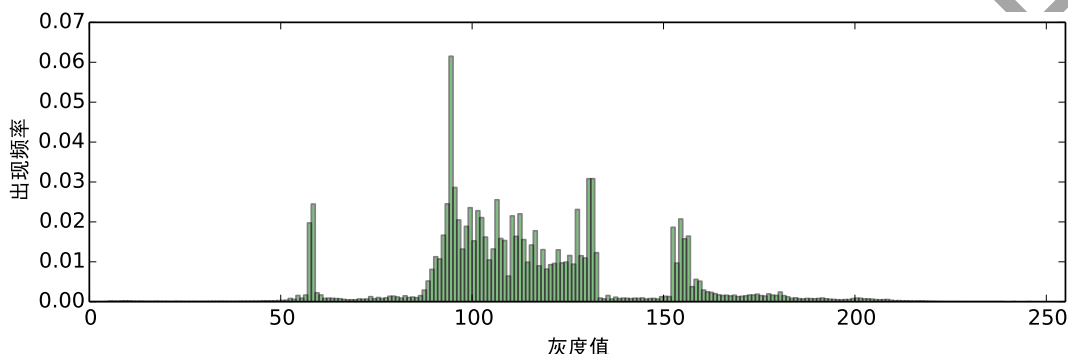


图 5: 对预处理后的图像进行灰色阶统计

到，色阶的分布形成了几个比较突出的峰，换言之，存在一定的聚类趋势。然而，直方图的统计结果是不连续的，一个平滑的结果更便于我们分析研究，结果也更有说服力。将统计结果平滑化的方法，就是核密度估计 (kernel density estimation)。

核密度估计方法是一种非参数估计方法，由 Rosenblatt 和 Parzen 提出，在统计学理论和应用领域均受到高度的重视 [2]。当然，也可以简单地将它看成一种函数平滑方式。我们根据大量的数据来估计某个值出现的概率（密度）时，事实上做的是如下估算：

$$\hat{p}(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right) \quad (4)$$

其中 $K(x)$ 称为核函数，当 h 取为 1，且 $K(x)$ 取

$$K(x) = \begin{cases} 1, & x = 0 \\ 0, & x \neq 0 \end{cases} \quad (5)$$

时，就是我们上述的直方图估计。 $K(x)$ 这一项的含义很简单，它就是告诉我们在范围 h 内的 x_i 都算入到 x 中去，至于怎么算，由 $K\left(\frac{x-x_i}{h}\right)$ 给出。可见， h 的选择对结果的影响很大， h 我们称之为带宽 (bandwidth)，它主要影响结果的平滑性。

如果 $K(x)$ 是离散的，得到的结果还是离散的，但如果 $K(x)$ 是光滑的，得到的结果也是比较光滑的。一个常用的光滑函数核是高斯核：

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \quad (6)$$

所得到的估计也叫高斯核密度估计。在这里，我们使用 scott 规则自适应地选取 h ，但需要手动指定一个平滑因子，在本文中，我们选取为 0.2。对于示例图片，我们得到如图 6 的红色曲线的结果。

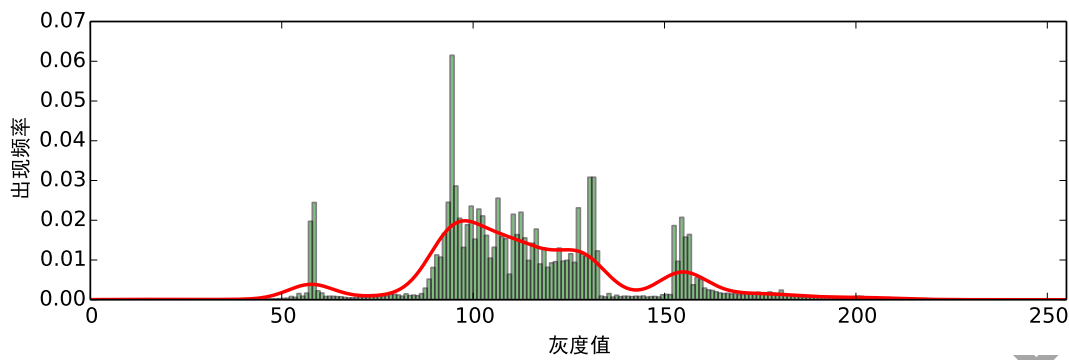


图 6: 频率分布的高斯核密度估计

3.2.2 极大极小值分割

从图 6 中我们进一步可以看出，图像确实存在着聚类趋势. 这表现为它有几个明显的极大值和极小值点，这里的极大值点位于 $x = 10, 57, 97, 123, 154$ ，极小值点位于 $25, 71, 121, 142$.

因此，一个很自然的聚类方法是：有多少个极大值点，就聚为多少类，并且以极小值点作为类别之间的边界. 也就是说，对于图 3，可以将图像分层 5 层，逐层处理. 分层之后，每一层的形状如图 7，其中白色是 1，黑色是 0.



图 7: 通过聚类将图像分为 5 个图层 (可放大看)

可见，由于 2.1 节的“对比度”和“渐变性”假设，通过聚类确实可以将文字图层通过核密度估计的聚类方法分离开来. 而且，通过聚类分层的思路，无需对文字颜色作任何假定，即便时文字颜色跟背景颜色一致时，也可以获得有效检测。

3.3 逐层识别

当图像有效地进行分层后，我们就可以根据 2.1 节的假设，进一步设计相应的模型，通过逐层处理的方式找出图像中的文字区域.

3.3.1 连通性

可以看到,每一层的图像是由若干连通区域组成的,文字本身是由笔画较为密集组成的,因此往往文字也能够组成一个连通区域.这里的连通定义为8邻接,即某个像素周围的8个像素都定义为邻接像素,邻接的像素则被定义为同一个连通区域.

定义了连通区域后,每个图层被分割为若干个连通区域,也就是说,我们逐步地将原始图像进行分解,如图9.

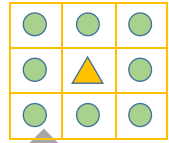


图8: 在本文中我们使用的是8邻接

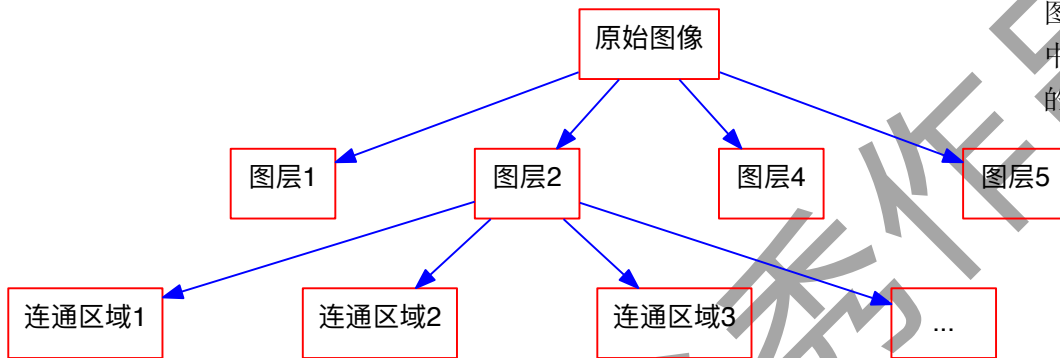


图9: 图像分解结构图

3.3.2 抗腐蚀能力

将图像分解至连通区域这一粒度后,我们就不再细分了,下一步开始识别哪些区域是可能的文字区域.这里我们要求文字具有一定的抗腐蚀能力.因此我们先来定义腐蚀.

腐蚀是一种图像上的形态学变换,一般针对于二值图像,对于二值图像中的非零像素(即取值为1的像素),如果它邻接的像素都为1,则保持不变,否则变为0,这里我们同样采用的是8邻接的定义.可以看到,如果连通区域的边界线越长,那么腐蚀运算对它的“伤害”就越大,反之,如果连通区域的边界线越短,那么腐蚀运算对它的“伤害”就越小.

根据以上腐蚀的定义,我们可以给出一个对文字区域的要求:

抗腐蚀要求 文字所在的连通区域应当具有一定的抗腐蚀能力.

这里的“一定”是指在一个连续的范围,不能太大,也不能太小.比如,一个面积较大的方形区域,它的抗腐蚀能力是很强的,因为它边界线很短,但这些区域明显不是文字区域,图7(e)的电饭锅便是属于这一类型;此外,抗腐蚀能力太弱也不可以,比如细长的线条,腐蚀之后可能就消失了,这些也不作为候选的文字区域,图7(d)的文字边界线就属于这一类型.

这里可以定义一个抗腐蚀能力的指标:

$$\text{连通区域的抗腐蚀能力} = \frac{\text{该区域被腐蚀后的总面积}}{\text{该区域被腐蚀前的总面积}} \quad (7)$$

经过测试,文字区域的抗腐蚀能力大概在[0.1, 0.9]这个区间中.

经过抗腐蚀能力筛选分解的5个图层,得到如图10的特征层.

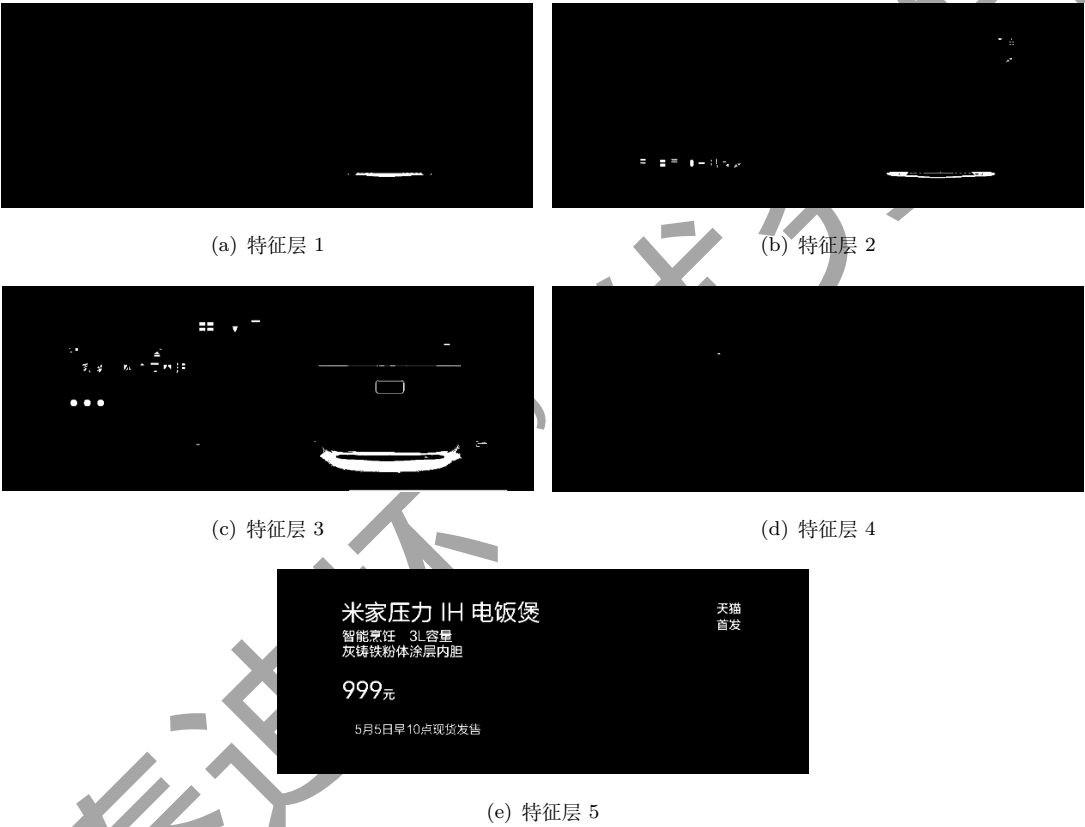


图 10: 只保留抗腐蚀能力在 $[0.1, 0.9]$ 这个区间中的连通区域

3.3.3 池化操作

到现在为止，我们得到了 5 个特征层，虽然肉眼可以看到，文字主要集中在第 5 个特征层。但是，对于一般的图片，文字可能分布在多个特征层，因此需要对特征层进行整合。我们这里进行特征整合的方法，类似于卷积神经网络中的“池化”，因此我们也借用了这个名称。

首先，我们将 5 个特征层进行叠加，得到一幅整体的图像特征（称为叠加特征）。这样的图像特征可以当作最后的特征输出，但并不是最好的方法。我们认为，某个区域内的主要文字特征应该已经集中分布在某个特征层中，而不是分散在所有的特征层。因此，得到叠加特征后，使用类似“最大值池化”的方式整合特征，步骤如下：

1. 直接叠加特征，然后对叠加特征划分连通区域；
2. 检测每个连通区域的主要贡献是哪个特征层，该连通区域就只保留这个特征层的来源。

经过这样的池化操作后，得到的最终特征结果如图 11。



图 11: 池化后的特征

对于我们演示的这幅图像，经过上述操作后，得到的特征图 11 已经不用再做什么处理了。然而，对于一般的图片，还有可能出现一些没处理好的区域，这时候需要在前述结果的基础上进一步排除。排除过程，主要有两个步骤，一个是低/高密度区排除，另外则是孤立区排除。

3.3.4 密度排除

一种明显不是文字区域的连通区域是低密度区，一个典型的例子就是由表格线组成的连通区域，这样的区域范围较大，但点很少，也就是密度很低，这种低密度区可以排除。首先我们来定义连通区域密度和低密度区：

连通区域密度 从一个连通区域出发，可以找到该连通区域的水平外切矩形，该区域的密度定义为

$$\text{连通区域密度} = \frac{\text{连通区域的面积}}{\text{外切矩形的面积}} \times \frac{\text{原图像总面积}}{\text{外切矩形的面积}} \quad (8)$$

低密度区 如果一个连通区域的密度小于 16，那么这个连通区域定义为低密度区。

直觉上的定义应该是 $\frac{\text{连通区域的面积}}{\text{外切矩形的面积}}$ ，但这里多了一个因子 $\frac{\text{原图像总面积}}{\text{外切矩形的面积}}$ ，目的是把面积大小这个影响因素加进去，因为文字一般有明显的边界，容易被分割开来，所以一般来说面积越大的区域越不可能是文本区域。这里的参数 16 是经验值。

低密度区排除是排除表格等线条较多的非文字区域的有效方法。类似地，范围较大的高密度区也是一类需要排除的区域。有了低密度区之后，就很容易定义高密度区了：

高密度区定义 * 如果一个连通区域以水平外切矩形反转后的区域是一个低密度区，那个这个连通区域定义为高密度区。

这个定义是很自然的，但是却有一定的不合理性。比如“一”字，是一个水平的矩形，于是翻转后的密度为 0，于是这个“一”字就被排除了，这是不合理的。解决这个问题一个方案是：

高密度区定义 当且仅当下面条件满足时才被定义为高密度区：

$$\frac{1 + \text{外切矩形的面积} - \text{连通区域的面积}}{\text{外切矩形的面积}} \times \frac{\text{原图像总面积}}{\text{外切矩形的面积}} \leq 16 \quad (9)$$

这是在原来定义的基础上加上了 1，防止了翻转后密度为 0 的情况。

还有另外一种失效的情况，就是假如输入图片是单字图片，那么只有一个连通区域，且 $\frac{\text{原图像总面积}}{\text{外切矩形的面积}}$ 接近于 1，因此它就被判为低密度区，这样就排除了单字。这种情形确实比较难兼顾。一个可行的解决办法是通过人工指定是单字模式、单行模型还是整体图片模式，Google 的 Tesseract OCR 也提供了这样的选项。

3.3.5 孤立区排除

孤立区排除的出发点是：文字之间、笔画之间应该会比较紧凑的，如果一个区域明显地孤立于其他区域，那么这个区域很可能不是文字区域。也就是说，可以把孤立区给排除掉。首先我们定义孤立区的概念：

孤立区 从一个连通区域出发，可以找到该连通区域的水平外切矩形，将这个矩形中心对称地向外扩张为原来的 9 倍（长、宽变为原来的 3 倍，如图 12），扩展后的区域如果没有包含其他的连通区域，那么原来的连通区域称为孤立区。



图 12: 孤立区演示

在大多数情况，孤立区排除是一种非常简单的去噪方法，因为很多噪音点都是孤立区。但是孤立区排除是存在一定风险的。如果一幅图像只有一个文字，构成了唯一一个连通区域，那么这个连通区域就是孤立的，于是这个文字就被排除了。因此，要对孤立区加上更多的限制，一个可选的额外限制是：被排除的孤立区的占比 $\frac{\text{连通区域的面积}}{\text{外切矩形的面积}}$ 要大于 0.75（这个值源于圆与外切正方形的面积之比 $\pi/4$ ）。

4 文字定位

经过第一部分，我们已经较好地提取了图像的文本特征，下面进行文字定位。主要过程分两步：1、邻近搜索，目的是圈出单行文字；2、文本切割，目的是将单行文本切割为单字。

4.1 邻近搜索

我们可以对提取的特征图进行连通区域搜索，得到的每个连通区域视为一个汉字. 这对于大多数汉字来说是适用，但是对于一些比较简单的汉字却不适用，比如“小”、“旦”、“八”、“元”这些字，由于不具有连通性，所以就被分拆开了，如图 13. 因此，我们需要通过邻近搜索算法，来整合可能成字的区域，得到单行的文本区域.

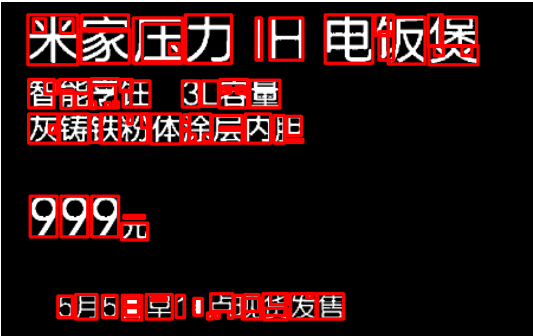


图 13: 直接搜索连通区域，会把诸如“元”之类的字分拆开

4.1.1 目的

邻近搜索的目的是进行膨胀，以把可能成字的区域“粘合”起来. 如果不进行搜索就膨胀，那么膨胀是各个方向同时进行的，这样有可能把上下行都粘合起来了. 因此，我们只允许区域向单一的一个方向膨胀. 我们正是要通过搜索邻近区域来确定膨胀方向 (上、下、左、右)：

邻近搜索 * 从一个连通区域出发，可以找到该连通区域的水平外切矩形，将连通区域扩展到整个矩形. 当该区域与最邻近区域的距离小于一定范围时，考虑这个矩形的膨胀，膨胀的方向是最邻近区域的所在方向.

既然涉及到了邻近，那么就需要有距离的概念. 下面给出一个比较合理的距离的定义.

4.1.2 距离

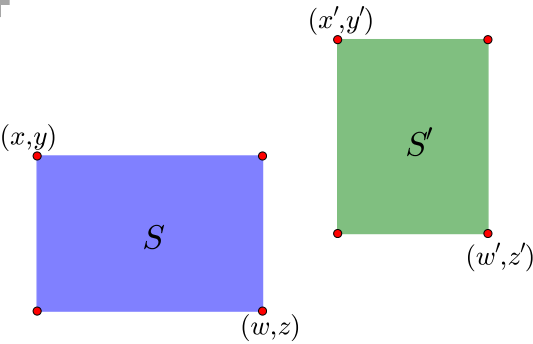


图 14: 两个示例区域

如图 14, 通过左上角坐标 (x, y) 和右下角坐标 (z, w) 就可以确定一个矩形区域, 这里的坐标是以左上角为原点来算的. 这个区域的中心是 $(\frac{x+z}{2}, \frac{y+w}{2})$. 对于图中的两个区域 S 和 S' , 可以计算它们的中心向量差

$$(x_c, y_c) = \left(\frac{x' + z'}{2} - \frac{x + z}{2}, \frac{y' + w'}{2} - \frac{y + w}{2} \right) \quad (10)$$

如果直接使用 $\sqrt{x_c^2 + y_c^2}$ 作为距离是不合理的, 因为这里的邻近应该是按边界来算, 而不是中心点. 因此, 需要减去区域的长度:

$$(x'_c, y'_c) = \left(x_c - \frac{w - x}{2} - \frac{w' - x'}{2}, y_c - \frac{z - y}{2} - \frac{z' - y'}{2} \right) \quad (11)$$

距离定义为

$$d(S, S') = \sqrt{[\max(x'_c, 0)]^2 + [\max(y'_c, 0)]^2} \quad (12)$$

至于方向, 由 (x_c, y_c) 的幅角进行判断即可.

然而, 按照前面的“邻近搜索*”方法, 容易把上下两行文字粘合起来, 因此, 基于我们的横向排版假设, 更好的方法是只允许横向膨胀:

邻近搜索 从一个连通区域出发, 可以找到该连通区域的水平外切矩形, 将连通区域扩展到整个矩形. 当该区域与最邻近区域的距离小于一定范围时, 考虑这个矩形的膨胀, 膨胀的方向是最邻近区域的所在方向, 当且仅当所在方向是水平的, 才执行膨胀操作.

4.1.3 结果

有了距离之后, 我们就可以计算每两个连通区域之间的距离, 然后找出最邻近的区域. 我们将每个区域向它最邻近的区域所在的方向扩大 4 分之一, 这样邻近的区域就有可能融合为一个新的区域, 从而把碎片整合. 实验表明, 邻近搜索的思路能够有效地整合文字碎片, 结果如图 15

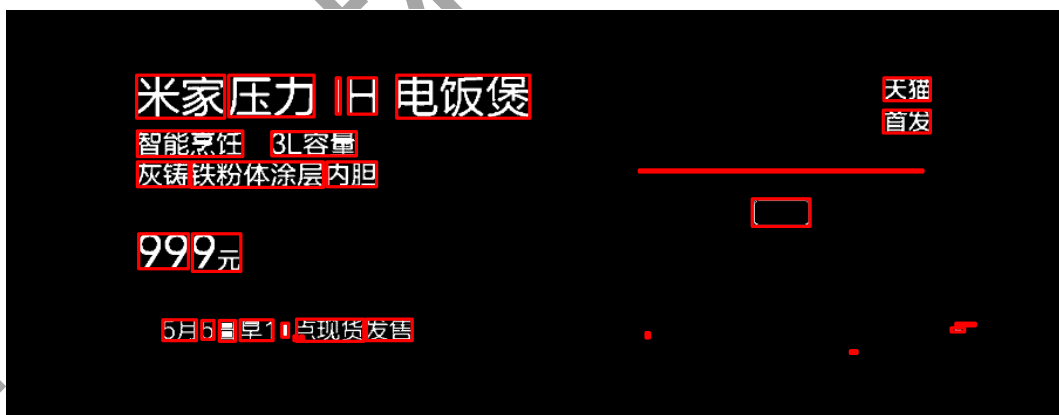


图 15: 通过邻近搜索后, 圈出的文字区域

4.2 文本切割

经过上一步, 得到单行的文本区域之后, 我们就可以想办法将单行的文本切割为单个的字符了. 因为第三步的模型师针对单个的字符建立的, 因此这一步也是必须的.

4.2.1 均匀切割

基于方块汉字的假设,事实上最简单的切割方法是均匀切割,也就是说不加任何判断,直接按照高度来将单行文本切割为一个一个的正方形图片.这种思路可以应对大部分的单行文本,如图 16(a).



图 16: 均匀切割成单字

当然,均匀切割的弊端也是很明显的.大多数汉字都是方块汉字,但多数英语和数字都不是,因此如果出现中英文混排的时候,均匀切割就失效了,如图 16(b).

4.2.2 统计切割

从图 15 中可以看出,经过前面的操作,字与字都被很好地分离开了.因此,另外一种比较简单的思路是对单行的文字图片进行垂直方向的求和,和为 0 的所在的列就是被切割的列.

用这种统计的思路可以很好地解决中英文混排的单行文字图片分割的问题,但是它也存在一定的弊端.最明显的就是诸如“小”、“的”等字就被切割开为两部分了.

4.2.3 前后比较

一个更好的思路是结合前面两部分结果,通过比较前后两部分区域是否组成方形来确定是否切割.具体步骤是:

1. 通过统计求和的思路,得出候选的切割线;
2. 如果该候选切割线到左右两条候选切割线的距离之和超过宽长度的 1.2 倍,那么该候选切割线确定为切割线;
3. 如果得到的区域是一个明显的长条矩形,并且没办法按照上面两个步骤切割,那个就均匀切割.

这三个步骤比较简单,基于两个假设:1、数字、英文字符的底与高之比大于 60%;2、汉字的底与高之比低于 1.2. 经过测试,该算法可以很好地用于前面步骤所提取的图片文本特征的分割.详细的评测结果请阅读 7.1 节.

5 光学识别

经过第一、二步,我们已经能够找出图像中单个文字的区域,接下来可以建立相应的模型对单字进行识别.

5.1 模型选择

在模型方面,我们选择了深度学习中的卷积神经网络模型,通过多层卷积神经网络,构建了单字的识别模型.

卷积神经网络是人工神经网络的一种，已成为当前图像识别领域的主流模型。它通过局部感知野和权值共享方法，降低了网络模型的复杂度，减少了权值的数量，在网络结构上更类似于生物神经网络，这也预示着它必然具有更优秀的效果。事实上，我们选择卷积神经网络的主要原因有：

1. **对原始图像自动提取特征** 卷积神经网络模型可以直接将原始图像进行输入，免除了传统模型的人工提取特征这一比较困难的核心部分；
2. **比传统模型更高的精度** 比如在 *MNIST* 手写数字识别任务中，可以达到 99% 以上的精度，这远高于传统模型的精度；
3. **比传统模型更好的泛化能力** 这意味着图像本身的形变（伸缩、旋转）以及图像上的噪音对识别的结果影响不明显，这正是一个良好的 *OCR* 系统所必需的。

5.2 训练数据

为了训练一个好的模型，必须有足够多的训练数据。幸运的是，虽然没有现成的数据可以用，但是由于我们只是做印刷字体的识别，因此，我们可以使用计算机自动生成一批训练数据。通过以下步骤，我们构建了一批比较充分的训练数据：

1. **更多细节** 由于汉字的结构比数字和英文都要复杂，因此，为了体现更多的细节信息，我使用 48×48 的灰度图像构建样本，作为模型的输入；
2. **常见汉字** 为了保证模型的实用性，我们从网络爬取了数十万篇微信公众平台上的文章，然后合并起来统计各自的频率，最后选出了频率最高的 3000 个汉字（在本文中我们只考虑简体字），并且加上 26 个字母（大小写）和 10 个数字，共 3062 字作为模型的输出；
3. **数据充分** 我们人工收集了 45 种不同的字体，从正规的宋体、黑体、楷体到不规范的手写体都有，基本上能够比较全面地覆盖各种印刷字体；
4. **人工噪音** 每种字体都构建了 5 种不同字号（46 到 50）的图片，每种字号 2 张，并且为了增强模型的泛化能力，将每个样本都加上 5% 的随机噪音。

经过上述步骤，我们一共生成了 $3062 \times 45 \times 5 \times 2 = 1377900$ 个样本作为训练样本，可见数据量是足够充分的。

5.3 模型结构

在模型结构方面，有一些前人的工作可以参考的。一个类似的例子是 *MNIST* 手写数字的识别——它往往作为一个新的图像识别模型的“试金石”——是要将六万多张大小为 28×28 像素的手写数字图像进行识别，这个案例跟我们实现汉字的识别系统具有一定的相似性，因此在模型的结构方面可以借鉴。一个常见的通过卷积神经网络对 *MNIST* 手写数字进行识别的模型结构如图 17。

经过充分训练后，如图 17 的网络结构可以达到 99% 以上的精确度，说明这种结构确实是可取的。但是很显然，手写数字不过只有 10 个，而常用汉字具有数千个，在本文的分类任务中，就共有 3062 个目标。也就是说，汉字具有更为复杂和精细的结构，因此模型的各方面都要进行调整。首先，在模型的输入方面，我们已经

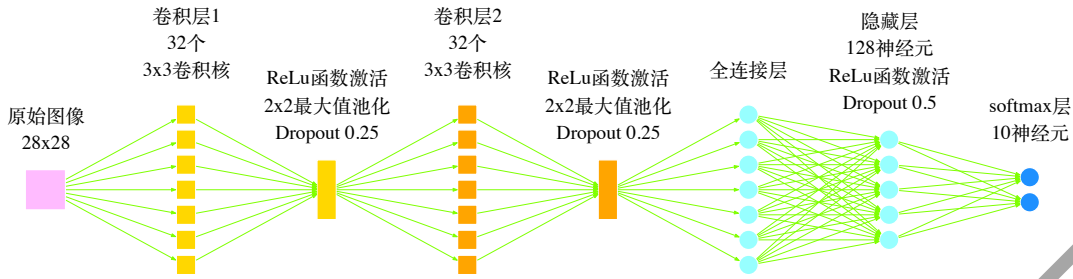


图 17: 一个用作 MNIST 手写数字识别的网络结构

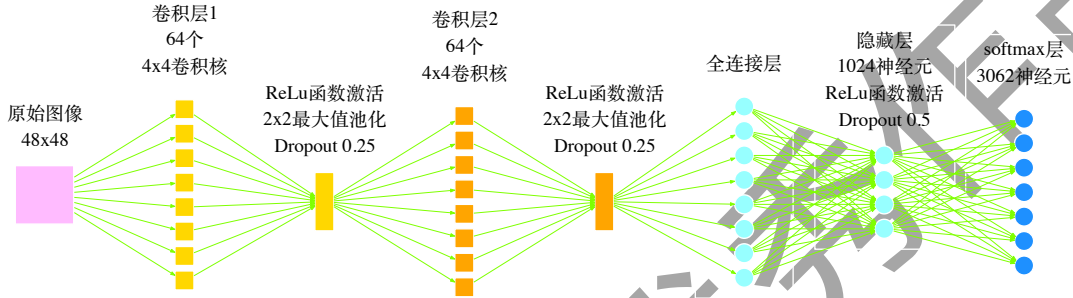


图 18: 本文用来识别印刷汉字的网络结构

将图像的大小从 28x28 提高为 48x48，这能保留更多的细节，其次，在模型结构上要复杂化调整，包括：增加卷积核的数目，增加隐藏节点的数目、调整权重等。最终我们的网络结构如图 18。

在激活函数方面，我们选取了 RuLe 函数为激活函数

$$ReLU(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \quad (13)$$

实验表明，它相比于传统的 sigmoid、tanh 等激活函数，能够大大地提升模型效果 [3][4]；在防止过拟合方面，我们使用了深度学习网络中最常用的 Dropout 方式 [5]，即随机地让部分神经元休眠，这等价于同时训练多个不同网络，从而防止了部分节点可能出现的过拟合现象。

需要指出的是，在模型结构方面，我们事实上做了大量的筛选工作。比如隐藏层神经元的数目，我们就耗费了若干天时间，尝试了 512、1024、2048、4096、8192 等数目，最终得到 1024 这个比较适合的值。数目太多则导致模型太庞大，而且容易过拟合；太少则容易欠拟合，效果不好。我们的测试发现，从 512 到 1024，效果有明显提升；而再增加节点效果没有明显提升，有时还会有明显下降。

5.4 模型实现

我们的模型在操作系统为 CentOS 7 的服务器（24 核 CPU+96G 内存 +GTX960 显卡）下完成，使用 Python 2.7 编写代码，并且使用 Keras 作为深度学习库，用 Theano 作为 GPU 加速库¹。

在训练算法方面，使用了 Adam 优化方法进行训练，batch size 为 1024，迭代 30 次，迭代一次大约需要 700 秒。

¹Tensorflow 一直提示内存溢出，配置不成功。

如果出现形近字时，应该是高频字更有可能，最典型的例子就是“日”、“曰”了，这两个的特征是很相似的，但是“日”出现的频率远高于“曰”，因此，应当优先考虑“日”。因此，在训练模型的时候，我们还对模型最终的损失函数进行了调整，使得高频字的权重更大，这样能够提升模型的预测性能。

经过多次调试，最终得到了一个比较可靠的模型。模型的收敛过程如图 19。

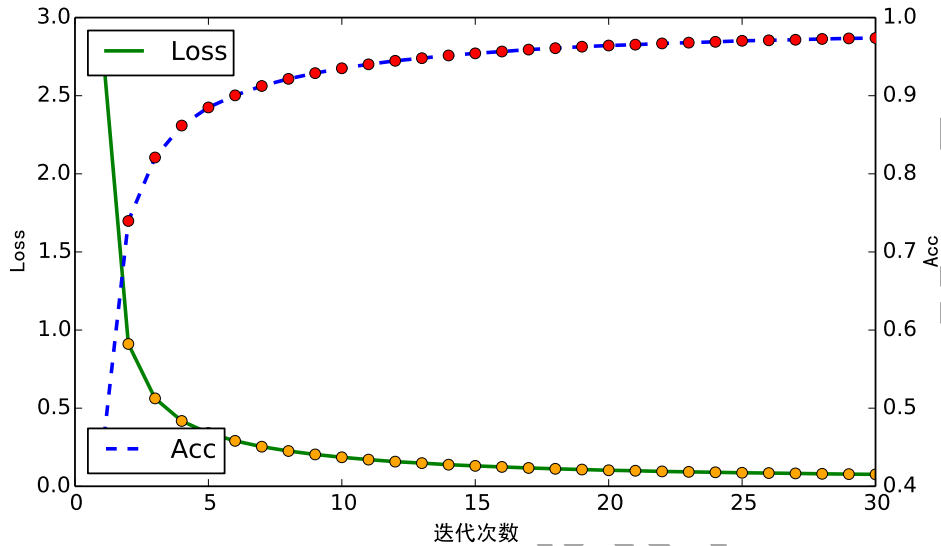


图 19: 训练曲线图 : Loss(损失函数) 和 Acc(精度)

5.5 模型检验

我们将从以下三个方面对模型进行检验。实验结果表明，对于单字的识别效果，我们的模型优于 Google 开源的 OCR 系统 Tesseract。

5.5.1 训练集检验

最终训练出来的模型，在训练集的检验报告如表 1。

数据集	正确率	样本数量
全部训练数据	99.70%	近 140 万样本
正规字体样本	99.85%	约 15 万样本

表 1: 模型训练结果汇总表

从表 1 可以看到，即便在加入了随机噪音的样本中，模型的正确率仍然有 99.7%，因此，我们有把握地说，单纯从单字识别这部分来看，我们的结果已经达到了 *state of the art* 级别，而且在黑体、宋体等正规字体中²，正确率更加高！

²正规字体样本是指所有训练样本中，字体为黑体、宋体、楷体、微软雅黑和 Arial unicode MS 的训练样本，这几种字体常见于印刷体中。

5.5.2 测试集检验

我们另外挑选了 5 种字体，根据同样的方法生成了一批测试样本（每种字体 30620 张，共 153100 张），用来对模型进行测试，得到模型测试正确率为 92.11%。五种字体的测试结果如表 2。






字体名称	八大山人手写体	岚竹风体	游狼软笔楷书	简超粗圆	造字工房情书常规体
字体样式					
正确率	82.83%	92.15%	92.65%	99.95%	92.97%

表 2: 模型在测试集中的结果 (5% 的随机噪音)

从表中可以看出，即便是对于训练集之外的样本，模型效果也相当不错。接着，我们将随机噪音增大到 15% (这对于一张 48×48 的文字图片来说已经相当糟糕了)，得到的测试结果如表 3。






字体名称	八大山人手写体	岚竹风体	游狼软笔楷书	简超粗圆	造字工房情书常规体
字体样式					
正确率	78.14%	85.34%	88.17%	99.81%	86.52%

表 3: 模型在测试集中的结果 (15% 的随机噪音)

平均的正确率为 87.59%，也就是说，噪音的影响并不明显，模型能够保持 90% 左右的正确率。这说明该模型已经完全达到了实用的程度。

6 语言模型

由于图像质量等原因，性能再好的识别模型，都会有识别错误的可能性，为了减少识别错误率，可以将识别问题跟统计语言模型结合起来，通过动态规划的方法给出最优的识别结果。这是改进 OCR 识别效果的重要方法之一。

6.1 转移概率

在我们分析实验结果的过程中，有出现这一案例。由于图像不清晰等可能的原因，导致“电视”一词被识别为“电柳”，仅用图像模型是不能很好地解决这个问题的，因为从图像模型来看，识别为“电柳”是最优的选择。但是语言模型却可以很巧妙地解决这个问题。原因很简单，基于大量的文本数据我们可以统计“电视”一词和“电柳”一词的概率，可以发现“电视”一词的概率远远大于“电柳”，因此我们会认为这个词是“电视”而不是“电柳”。

从概率的角度来看, 就是对于第一个字的区域的识别结果 s_1 , 我们前面的卷积神经网络给出了“电”、“宙”两个候选字 (仅仅选了前两个, 后面的概率太小), 每个候选字的概率 $W(s_1)$ 分别为 0.99996、0.00004; 第二个字的区域的识别结果 s_2 , 我们前面的卷积神经网络给出了“柳”、“视”、“规”(仅仅选了前三个, 后面的概率太小), 每个候选字的概率 $W(s_2)$ 分别为 0.87838、0.12148、0.00012, 因此, 它们事实上有六种组合: “电柳”、“电视”、“电规”、“宙柳”、“宙视”、“宙规”。

下面考虑它们的迁移概率. 所谓迁移概率, 其实就是条件概率 $P(s_1|s_2)$, 即当 s_1 出现时后面接 s_2 的概率. 通过 10 万微信文本, 我们统计出, “电”字出现的次数为 145001, 而“电柳”、“电视”、“电规”出现的次数为 0、12426、7; “宙”字出现的次数为 1980 次, 而“宙柳”、“宙视”、“宙规”出现的次数为 0、0、18, 因此, 可以算出结

$$\begin{aligned} P(\text{电}|\text{柳}) &= \frac{0}{145001} = 0 & P(\text{电}|\text{视}) &= \frac{12426}{145001} \approx 0.08570 & P(\text{电}|\text{规}) &= \frac{7}{145001} \approx 0.00005 \\ P(\text{宙}|\text{柳}) &= \frac{0}{1980} = 0 & P(\text{宙}|\text{视}) &= \frac{0}{1980} = 0 & P(\text{宙}|\text{规}) &= \frac{18}{1980} \approx 0.00909 \end{aligned}$$

果如图 20:

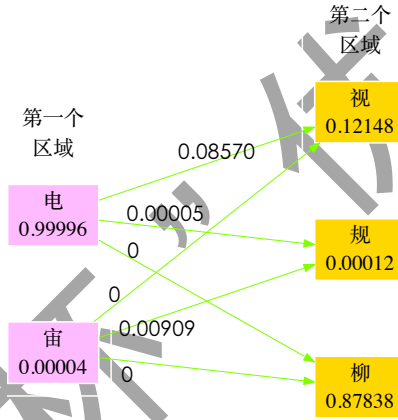


图 20: 考虑转移概率

从统计的角度来看, 最优的 s_1, s_2 组合, 应该使得式 (14) 取最大值:

$$f = W(s_1)P(s_1|s_2)W(s_2) \quad (14)$$

因此, 可以算得 s_1, s_2 的最佳组合应该是“电视”而不是“电柳”. 这时我们成功地通过统计的方法得到了正确结果, 从而提高了正确率.

6.2 动态规划

类似地, 如图 21, 如果一个单行文字图片有 n 个字 s_1, s_2, \dots, s_n 需要确定, 那么应当使得

$$f = W(s_1)P(s_1|s_2)W(s_2)P(s_2|s_3)W(s_3) \dots W(s_{n-1})P(s_{n-1}|s_n)W(s_n) \quad (15)$$

取得最大值, 这就是统计语言模型的思想, 自然语言处理的很多领域, 比如中文分词、语音识别、图像识别等, 都用到了同样的方法 [6]. 这里需要解决两个主要的问题: (1) 各个 $P(s_i|s_{i+1})$ 的估计; (2) 给定各个 $P(s_i|s_{i+1})$ 后如何求解 f 的最大值.

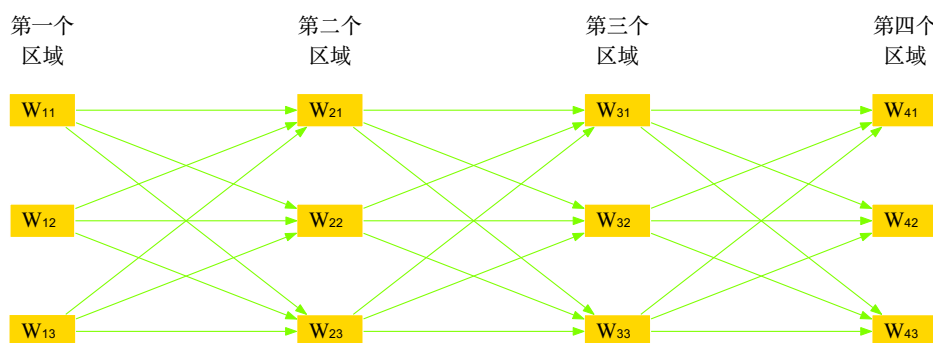


图 21: 多字图片的规划问题

6.2.1 转移概率矩阵

对于第一个问题,只需要从大的语料库中统计 s_i 的出现次数 $\#s_i$,以及 s_i, s_{i+1} 相接地出现的次数 $\#(s_i, s_{i+1})$, 然后认为

$$P(s_i|s_{i+1}) = \frac{\#(s_i, s_{i+1})}{\#s_i} \quad (16)$$

即可,本质上没有什么困难. 本文的识别对象有 3062 个,理论上来说,应该生成一个 3062×3062 的矩阵,这是非常庞大的. 当然,这个矩阵是非常稀疏的,我们可以只保存那些有价值的元素.

现在要着重考虑当 $\#(s_i, s_{i+1}) = 0$ 的情况. 在前一节我们就直接当 $P(s_i|s_{i+1}) = 0$, 但事实上是不合理的. 没有出现不能说明不会出现,只能说明概率很小,因此,即便是对于 $\#(s_i, s_{i+1}) = 0$, 也应该赋予一个小概率而不是 0. 这在统计上称为数据的平滑问题.

一个简单的平滑方法是在所有项的频数 (包括频数为 0 的项) 后面都加上一个正的小常数 α (比如 1), 然后重新统计总数并计算频率, 这样每个项目都得到了一个正的概率. 这种思路有可能降低高频数的项的概率, 但由于这里的概率只具有相对意义, 因此这个影响是不明显的³. 按照这种思路, 从数十万微信文章中, 我们计算得到了 160 万的词语的转移概率矩阵.

6.2.2 Viterbi 算法

对于第二个问题, 求解最优组合 s_1, s_2, \dots, s_n 是属于动态规划中求最优路径的问题, 其中最有效的方法是 Viterbi 算法 [6].

Viterbi 算法是一个简单高效的算法, 用 Python 实现也就十来行的代码. 它的核心思想是: 如果最终的最优路径经过某个 s_{i-1} , 那么从初始节点到 s_{i-1} 点的路径必然也是一个最优路径——因为每一个节点 s_i 只会影响前后两个 $P(s_{i-1}|s_i)$ 和 $P(s_i|s_{i+1})$.

根据这个思想, 可以通过递推的方法, 在考虑每个 s_i 时只要求出所有经过各 s_{i-1} 的候选点的最优路径, 然后再与当前的 s_i 结合比较. 这样每步只需要算不超过 l^2 次, 就可以逐步找出最优路径. Viterbi 算法的效率是 $\mathcal{O}(n \cdot l^2)$, l 是候选数目最多的节点 s_i 的候选数目, 它正比于 n , 这是非常高效率的.

³一个更合理的思路是当频数小于某个阈值 T 时才加上常数, 其他不加.

6.3 提升效果

实验表明，结合统计语言模型进行动态规划能够很好地解决部分形近字识别错误的情况。在我们的测试中，它能修正一些错误如下：由于用来生成转移矩阵的语料库不够大，因此修正的效果还有很大的提升空间。

电柳	→	电视
研友	→	研发
速于	→	速干
围像	→	图像
...		

表 4: 通过统计语言模型的动态规划能修正不少识别错误

不管怎么说，由于 Viterbi 算法的简单高效，这是一个性价比很高的步骤。

7 综合评估

7.1 数据验证

尽管在测试环境下模型工作良好，但是实践是检验真理的唯一标准。在本节中，我们通过自己的模型，与京东的测试数据进行比较验证。

衡量 OCR 系统的好坏有两部分内容：(1) 是否成功地圈出了文字；(2) 对于圈出来的文字，有没有成功识别。我们采用评分的方法，对每一张图片的识别效果进行评分。评分规则如下：

如果圈出的文字区域能够跟京东提供的检测样本的 *box* 文件中匹配，那么加 1 分，如果正确识别出文字来，另外加 1 分，最后每张图片的分数是前面总分除以文字总数。

按照这个规则，每张图片的评分最多是 2 分，最少是 0 分。如果评分超过 1，说明识别效果比较好了。经过京东的测试数据比较，我们的模型平均评分大约是 0.84，效果差强人意。

7.2 模型综述

在本文中，我们的目标是建立一个完整的 OCR 系统，经过一系列的工作，我们也基本完成了这一目标。

在设计算法时，我们紧密地结合基本假设，从模拟人肉眼的识别思路出发，希望能够以最少的步骤来实现目标，这种想法在特征提取和文字定位这两部分得到充分体现。同样地，由于崇尚简洁和模拟人工，在光学字符识别方面，我们选择了卷积神经网络模型，得到了较高的正确率；最后结合语言模型，通过动态规划用较简单的思路提升了效果。

经过测试，我们的系统对印刷文字的识别有着不错的效果，可以作为电商、微信等平台的图片文字识别工具。其中明显的特点是，我们的系统可以将整张文字图片输入，并且在分辨率不高的情况下能够获得较好的效果。

7.3 结果反思

在本文所涉及到的算法中，一个很大的不足之处就是有很多的“经验参数”，比如聚类时 h 参数的选择、低密度区定义中密度的阈值、卷积神经网络中的卷积核数据、隐藏层节点数目等。由于并没有足够多的标签样本

进行研究, 因此, 这些参数都只能是凭借着经验和少量的样本推算得出. 我们期待会有更多的标签数据来得到这些参数的最优值.

还有, 在识别文字区域方面, 还有很多值得改进的地方. 虽然我们仅仅是经过几个步骤就去掉了大部分的文字区域, 但是这些步骤还是欠直观, 亟待简化. 我们认为, 一个良好的模型应该是基于简单的假设和步骤就能得到不错的效果, 因此, 值得努力的工作之一就是简化假设, 缩减流程.

此外, 在文本切割方面, 事实上不存在一种能够应对任何情况的自动切割算法, 因此这一步还有很大的提升空间. 据相关文献, 可以通过 CNN+LSTM 模型, 直接对单行文本进行识别, 但这需要大量的训练样本和高性能的训练机器, 估计只有大型企业才能做到这一点.

显然, 还有很多工作都需要更深入地研究.

参考文献

- [1] 李萌；基于多尺度 Gabor 滤波器和 BP 神经网络的文本检测算法研究；计算机软件与理论；2007
- [2] 核密度估计；<https://zh.wikipedia.org/zh-cn/核密度估计>；维基百科
- [3] Xavier Glorot, Antoine Bordes, Yoshua Bengio；Deep Sparse Rectifier Neural Networks
- [4] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton；ImageNet Classification with Deep Convolutional Neural Networks
- [5] Dropout: A Simple Way to Prevent Neural Networks from Overfitting
- [6] 吴军；《数学之美》（第二版）；第 3 章
- [7] 吴军；《数学之美》（第二版）；第 26 章