Tesseract OCR引擎概述

雷·史密斯Googl
e Inc.
theraysmith@gmail.com

抽象

Tesseract OCR引擎和HP Research一样
全面描述了UNLV第四次OCR准确性年度测试的原型[1]。

概述。重点放在OCR引擎中新颖或至少不寻常的方面，尤其包括线查找，特征/分类
方法和自适应分类器。

1.简介–动机和历史

Tesseract是一个开源的OCR引擎，
它是由1984年至1994年在惠普公司开发的。像超新星一样，它在1995年的UNLV年度OCR准确性测试中从无处出现[1]，其结果闪闪发光，然后在与之相同的保密秘密下消失了现在，第一次可以揭示架构和算法的细节。

Tesseract最初是HP的博士研究项目[2]
布里斯托尔的Labs作为HP平板扫描仪产品线的一种可能的软件和/或硬件附件而获得了发展。当时的商业OCR引擎还处于起步阶段，但除最高质量的印刷品外，其他任何产品都惨遭失败，这为人们提供了动力。

经过HP Labs Bristol的联合项目，以及
惠普位于美国科罗拉多州Tesseract的扫描仪部门在准确性方面领先于商用发动机，但并未成为产品。开发的下一个阶段是回到HP Labs Bristol进行OCR压缩研究。工作

重点更多地放在提高拒绝效率上，而不是基本水平的准确性上。在该项目的最后，即1994年底，发展完全停止了。该发动机被送往UNLV进行1995年的OCR精度测试[1]，在当时的商用发动机中证明了它的价值。2005年底，HP发布了Tesseract开源。现在可以在http://code.google.com/p/tesseract-ocr上获得。

2.建筑

由于惠普拥有独立开发的页面布局
产品中使用的分析技术（因此尚未发布用于开源）Tesseract不需要其自己的页面布局分析。 Tesseract

因此，假定其输入是已定义可选多边形文本区域的二进制图像。

加工遵循传统的逐步步骤
管道，但其中的某些阶段在当时是不寻常的，甚至到现在仍可能如此。第一步是连接组件分析，其中存储了组件的轮廓。这在当时是一个计算上昂贵的设计决策，但具有显着的优势：通过检查轮廓的嵌套以及子轮廓和孙轮廓的数量，可以很容易地检测到反向文本并像黑白文本一样容易地识别它。 Tesseract可能是第一个能够如此轻松地处理黑白文本的OCR引擎。在此阶段，仅通过嵌套将轮廓聚集到Blob中。

Blob被组织为文本行，这些行和
分析区域的固定音高或比例文本。文本行根据字符间距的不同分为单词。固定的间距文本立即被字符单元斩断。

比例文本使用确定空间和模糊空间分解为单词。

然后，识别将作为两次通过过程进行。在
第一遍，尝试依次识别每个单词。每个满意的单词都将作为训练数据传递给自适应分类器。然后，自适应分类器就有机会更准确地识别页面下方的文本。

由于自适应分类器可能已经学习
一些有用的东西来不及在页面顶部做出贡献，为时已晚，将在页面上遍历第二遍，其中再次识别出未被充分理解的单词。

最后阶段解析模糊空间并检查
x高度定位小写字母文本的替代假设

0-7695-2822-8 / 07 25.00美元©2007 IEEE 629

3.行和词查找

3.1。寻线

寻线算法是

Tesseract以前已经发表过[3]。设计了寻线算法，以便无需偏斜即可识别偏斜，从而节省了图像质量。该过程的关键部分是斑点过滤和线构造。

假设页面布局分析已经

如果提供的文本区域的文本大小大致相同，则简单的百分位数高度过滤器会删除首字下沉和垂直触摸的字符。中位数高度接近该区域中的文本大小，因此可以安全地滤除小于中位数高度几分之一的斑点，这些斑点很可能是标点符号，变音符号和噪点。

过滤的斑点更可能适合以下模型

不重叠，平行但倾斜的线。通过x坐标对blob进行排序和处理，可以将blob分配给唯一的文本行，同时在页面上跟踪斜率，从而大大减少了在存在倾斜的情况下分配给错误文本行的危险。一旦将过滤后的斑点分配给线，就使用最小二乘方拟合[4]来估计基线，然后将过滤出的斑点重新拟合到适当的线中。

线创建过程的最后一步合并

至少水平重叠一半的斑点，将变音标记与正确的底色放在一起，并正确地关联了一些断字符的部分。

3.2。基线拟合

找到文本行后，基线

使用二次样条更精确地拟合。这是OCR系统的又一个先河，它使Tesseract能够处理具有弯曲基线的页面[5]，这是扫描过程中常见的工件，而不仅仅是装订书本。

通过划分斑点来拟合基线

对于原始笔直基线，以合理连续的位移分组。二次样条曲线以最小二乘拟合拟合到人口最多的分区（假定为基线）。二次样条曲线的优点是该计算是合理稳定的，但是缺点是当需要多个样条曲线段时，不连续性会增大。更多的传统三次样条[6]可能会更好。

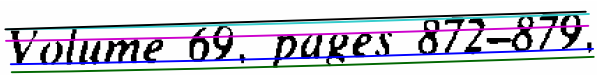


图1.曲线拟合基线的示例。

图1显示了一个带有

基线，下降线，中线和上升线。所有这些线都是“平行的”（y间隔在整个长度上是恒定的）并且略微弯曲。上升线是青色的（打印为浅灰色），其上方的黑线实际上是直行。仔细检查表明，青色/灰色线相对于其上方的黑色直线是弯曲的。

3.3。固定音高检测和斩波

Tesseract测试文本行以确定是否

他们是固定的音高。在找到固定音高文本的地方，Tesseract使用该音高将单词切成字符，并在单词识别步骤中禁用这些单词的斩波器和关联器。图2显示了固定音高单词的非典型示例。

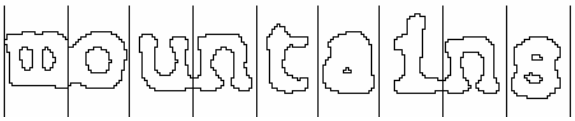
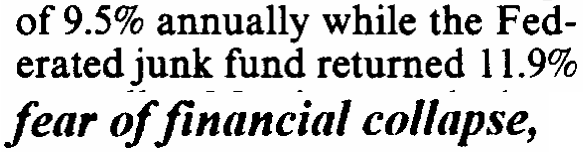


图2.固定音高的切碎单词。

3.4。比例词查找

非固定间距或成比例的文本间距是

高度不平凡的任务。图3说明了一些典型的问题。十个单位之间的差距为“ 11.9%”，与一般空间的大小相似，并且肯定比“竖起的”和“垃圾”之间的间距大。 “ of”和“ financial”的边界框之间根本没有水平间隙。Tesseract通过测量基线和平均线之间有限垂直范围内的间隙来解决大多数此类问题。在此阶段使接近阈值的空间变得模糊，以便可以在单词识别之后做出最终决定。



图。3.一些困难的单词间距。

4.单词识别

任何字符的识别过程的一部分

识别引擎将识别单词应如何细分为字符。来自线查找的初始分割输出首先被分类。单词识别步骤的其余部分仅适用于非固定音高的文本

4.1切合字符

虽然单词的结果（请参见第6节）是

不能令人满意的是，Tesseract尝试通过以字符分类器最差的置信度对Blob进行斩波来改善结果。候选切点可以从轮廓的多边形近似[2]的凹形顶点中找到，并且可以具有另一个相对的凹形顶点或线段。成功将连接的字符与ASCII集分开可能需要多达3对的截断点。

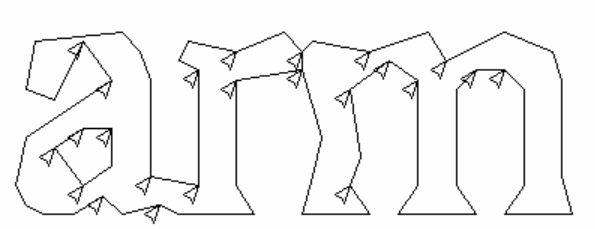


图4.候选排骨和排骨。

图4显示了一组候选斩波点，其中

箭头和选定的印章作为轮廓线，其中“ r”触摸“ m”。

排骨以优先顺序执行。任何印章

未能提高结果置信度的操作被撤消，但没有完全丢弃，因此如果需要，可以由关联人员稍后重新使用该印章。

4.2。关联残破字符

当潜在的印章用尽后，如果

这个词还不够好，它被赋予了联系者。关联者对可能的分割图进行A *（最佳优先）搜索

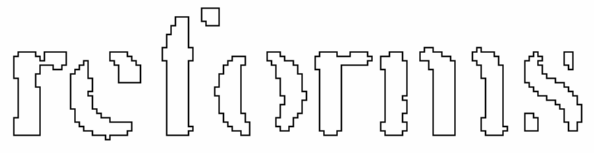
将最大切碎的斑点组合成候选字符。它无需实际构建细分图即可执行此操作，而是维护访问状态的哈希表。 A *搜索通过从优先级队列中拉出候选新状态并通过对未分类的类别进行评估来进行

片段的组合。

可能有人争辩说，这个完全砍掉然后关联的人

这种方法充其量是效率低下的，最坏的情况是容易丢失重要的印章，事实就是如此。好处是印后关联方案

简化了维护完整分割图所需的数据结构。



图。5.一个容易识别的词。

首次进行A *细分搜索时

Tesseract于1989年左右实施，其准确的残破字符远远领先于当今的商业引擎。图5是一个典型的例子。成功的关键部分是角色

可以轻松识别损坏字符的分类器。

5.静态字符分类器

5.1。特征

Tesseract的早期版本使用拓扑

功能由Shillman等人的工作开发而成。等[7-8]尽管很好地独立于字体和大小，但这些功能对真实图像中发现的问题并不健壮，如Bokser [9]所述。中间思想涉及使用多边形近似的线段作为特征，但是这种方法对于损坏的字符也不是很可靠。例如，在图6（a）中，轴的右侧为两个主要零件，但在图6（b）中，只有一个零件。

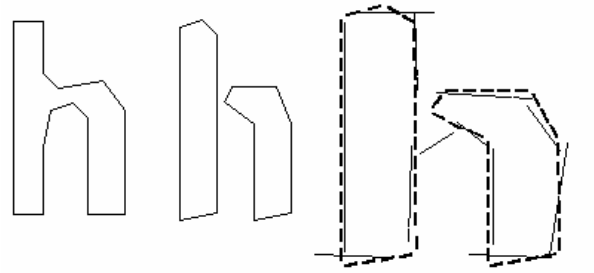


图6.（a）原始的“ h”，（b）破损的“ h”，（c）

与原型匹配的功能。

突破性的解决方案是

未知中的特征不必与训练数据中的特征相同。在训练过程中，使用了多边形近似[2]的这些特征作为特征，但是在识别中，从轮廓中提取了一个固定长度小的特征（以标准化单位表示），并将其与训练数据的聚类原型特征进行多对一匹配。在图6（c）中，短粗线是从未知数中提取的特征，细长线是用作原型的多边形近似的聚类段。桥接这两个部分的原型是完全无法比拟的。一侧的三个功能和另一侧的两个功能是不匹配的，但除此之外，每个原型和每个功能都匹配良好。该示例表明，这种由小特征匹配大型原型的过程可以轻松地应对受损图像的识别。其主要问题是计算未知物与原型之间的距离的计算成本非常高

因此，从未知中提取的特征是3-

尺寸（x，y位置，角度），通常在一个字符中具有50-100个特征，原型特征是4维（x，y，位置，角度，长度），在原型配置中通常为10-20个特征。

5.2。分类

分类过程分为两个步骤。在里面

第一步，类修剪器创建未知类可能匹配的字符类的简短列表。每个特征都从一个粗略量化的3维查找表中获取可能匹配的类的位向量，然后对所有特征求和。具有最高计数（在校正了预期数量的功能之后）的类将成为下一步的候选列表。

未知数的每个特征都查找一个

给定类的原型可能会匹配，然后计算它们之间的实际相似性。每个原型字符类都由一个逻辑乘积和表达式表示，每个术语都称为一个配置，因此距离计算过程会记录总相似度每个配置中的每个特征以及每个原型的证据。根据总的特征和原型证据计算得出的最佳组合距离是该类所有已存储配置中的最佳组合距离。

5.3。训练数据

由于分类器能够识别损坏

字符很容易，分类器没有针对损坏的字符进行训练。实际上，该分类器仅对来自8个字体的64个字符的20个样本进行训练，这些字体具有单个大小，但具有4个属性（正常，粗体，斜体，粗斜体），总共形成了60160个训练样本。其他已发布的分类器，例如具有超过一百万个样本的Calera分类器[9]和具有1175000个训练样本的Baird的100字体分类器[10]。

6。语言分析

Tesseract的语言相对较少

分析。每当单词识别模块考虑进行新的细分时，语言模块（错误命名为置换器）都会在以下各个类别中选择最佳的可用单词字符串：顶部常见单词，顶部字典单词，顶部数字单词，顶部大写字母，顶部小写字母单词（带有可选的初始大写字母），顶级分类器选择

字。给定分割的最终决定就是总距离评分最低的单词，其中上述每个类别均乘以一个不同的常数。

来自不同细分的单词可能具有

其中包含不同数量的字符。即使分类器声称产生了概率，Tesseract也不愿意直接比较这些词。在Tesseract中，通过为每个字符生成两个数字来解决此问题

分类。第一个称为置信度，是减去与原型的标准化距离。从更大的数字越好，但仍然是一个距离的角度，这使它成为“信心”，因为距离零越远，距离就越大。第二个输出（称为等级）将距原型的标准化距离乘以未知字符中的轮廓总长度。单词内字符的等级可以有意义地求和，因为单词内所有字符的总轮廓长度始终相同。

7.自适应分类器

有人建议[11]并证明[12]

OCR引擎可以从自适应分类器的使用中受益。由于静态分类器必须善于概括任何一种字体，因此其区分不同字符或不同字符之间以及非不同字符之间的能力会减弱。因此，通常使用由静态分类器的输出训练的对字体更为敏感的自适应分类器[13]，以在字体数量受到限制的每个文档中获得更大的区分度。

Tesseract不使用模板分类器，但是

使用与staticclassifier相同的功能和分类器。除训练数据外，静态分类器和自适应分类器之间的唯一显着区别是，自适应分类器使用各向同性基线/ x高度归一化，而静态分类器通过质心（第一矩）对字符进行归一化以进行位置尺寸和第二矩进行各向异性大小归一化。

基线/ x高度归一化使其更容易

区分大写和小写字符以及提高对噪声斑点的抵抗力。字符矩归一化的主要好处是消除了字体的长宽比和一定程度的字体笔划宽度。它还使子和上标的识别更加简单，但是需要附加的分类器功能来区分一些大写和小写字符。图7显示了以基线/ x高度归一化形式和矩归一化形式的3个字母的示例

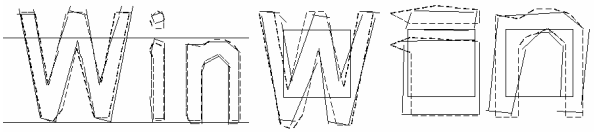


图7.基准线和矩归一化

字母。

8.结果

Tesseract被列入第四届UNLV年度测试
关于OCR准确性的文献[1]，称为“ HP Labs OCR”，但此后代码发生了很大变化，包括转换为Unicode和重新训练。表1将Tesseract的最新版本（显示为2.0）与1995年的原始结果（显示为HP）进行了比较。显示了1995年测试中使用的全部四个300 DPIbinary测试集，以及错误数（Errs），错误率（%Err）和相对于1995年结果的变化百分比（%Chg），包括字符错误和非字符错误。停用词错误。 [1]有关最新结果，请访问http://code.google.com/p/tesseract-ocr。

表1.当前和旧Tesseract的结果。

		字词					
Ver Set错误%Err%Chg错误%Err%Chg							
惠普总线5959 1.86 1293 4.27							
2.0总线6449 2.02 3.22 1295 4.28 0.15							
惠普doe 36349 2.48 7042 5.13							
2.0能源部29921 2.04 -17.68 6791 4.95 -3.56							
惠普mag 15043 2.26 3379 5.01							
2.0 mag 14814 2.22 -1.52 3133 4.64 -7.28							
惠普新闻6432 1.31 1502 3.06							
2.0新闻7935 1.61 23.36 1284 2.62 -14.51							
2.0总计59119 -7.31 12503 -5.39							

9.结论和进一步工作

休眠十多年后，
在准确性方面，Tesseract现在落后于领先的商用发动机。它的主要优势可能是其功能的非常规选择。它的关键弱点可能是它使用了多边形
近似值输入到分类器，而不是原始轮廓。

完成国际化后，准确性可以
明智地添加基于隐马尔可夫模型的字符n-gram模型可能会得到显着改善，并且可能会改进斩波器。

10.致谢

作者要感谢John Burns和Tom
Nartker为使Tesseract开放而做出的努力
消息来源是，UNLV的ISRI小组共享了他们的工具和数据，Luc Lucy，Igor Krivokon，Dar-Shyang Lee和Thomas Kielbus对本文的内容发表了评论。

11。参考资料

[1] S.V.赖斯詹金斯（T.A.） Nartker，OCR准确性的第四次年度测试，技术报告95-03，内华达大学信息科学研究院，拉斯维加斯，1995年7月。

[2] R.W. Smith，《从多媒体文档图像中提取和识别文本》，博士学位论文，布里斯托大学，1987年11月。

[3] R. Smith，“通过文本行累加的一种简单而有效的偏斜检测算法”，Proc.Natl.Acad.Sci.USA，87：3877-5。第三届国际会议文档分析和识别（第2卷），IEEE1995，第1145-1148页。

[4] P.J. Rousseeuw，A.M. Leroy，鲁棒回归和离群值检测，Wiley-IEEE，2003年。

[5] S.V.赖斯，纳吉，T.A。纳特克（Nartker），《光学字符识别：边疆图解指南》，KluwerAcademic Publishers，美国，1999年，第57-60页。

[6] P.J. Schneider，“自动拟合数字化曲线的算法”，A.S。 Glassner，Graphics Gems I，Morgan Kaufmann，1990，第612-626页。

[7] R.J. Shillman，基于字符的识别
现象学属性：理论和方法，博士学位，麻省理工学院。 1974年。

[8] B.A. Blesser，T.T。Kuklinski，R.J。 Shillman，“基于字符识别的心理学理论的特征选择的经验测试”，模式识别8（2），纽约爱思唯尔，1976年。

[9] M. Bokser，“ Omnidocument Technologies”，Proc。IEEE80（7），IEEE，美国，1992年7月，第1066-1078页。

[10] H.S. Baird，R. Fossey，“ A
100字体分类器”，Proc。第一国际Conf。关于文档分析和识别，IEEE，1991，pp 332-340。

[11] G. Nagy，“在OCR的前沿”，Proc。 IEEE 80（7），IEEE，美国，1992年7月，第1093-1100页。

[12] G. Nagy，Y。Xu，“自适应OCR的自动原型提取”，Proc。第四国际Conf。关于文档分析和识别，IEEE，1997年8月，第278-282页。

[13]我。Marosi，“工业OCR方法：体系结构，算法和适应技术”，文档
识别和检索XIV，SPIE 2007年1月，6500-01