PixelLink: Detecting Scene Text via Instance Segmentation

- 传统的做法
  - 水平投影:(确定行)每一行的上界限和下界限
  - 垂直投影:(确定字)每一个字符的左右边界
- 回归的做法
  - CTPN
  - TextBoxes
  - SegLink
  - EAST
  - 分成2步:text/non-text classification + Location regression
    后期处理:joining together + NMS etc.
- 本论文的做法
  - 具体思想
    - 直接通过实例分割进行文本/非文本的分类和边界框抽取
    - 利用了二分类+每个像素点与周围8个顶点的link分类
    - 边界框直接通过positive pixel组成Connected Components(CC)通过类似minAreaRect的方法获得
    - post-Filtering:利用图像处理的方法进行后处理(threshold,比如区域的面积和最短边要满足条件),不需要NMS
  - 具体做法
    - 提取featuremap: VGG-16(backbone)
    - text/non-text prediction
    - link prediction
  - 细节
    - link的规则
      - 两个像素都是positive=>link是positive的
      - 一个是text,另一个是non-text=>link是positive的
      - 两个都是non-text时=>link是nagative的
      - 给定pixel和8个临域点其中之一,两者都在实例内则他们之间的link是positive的,否则是negative
    - loss ≡
      - pixel loss
        - W:文字区域大小不同权重应该不同=>Instance-Balanced Cross-Entropy Loss
        - ==>W*Lpixel_ce/(1+r)*S
          Lpixel_ce:text/non-text prediction的交叉熵损失矩阵
          S:所有实例分割的面积和
      - link loss
        - positive links
        - negative links
      - lambda(value=2:加大对像素损失更重大)*pixel loss+link loss
  - 效果
    - 不需要在VGG-16上fine-tuning就可以得到很好的performance
    - 更少的数据,更少的训练迭代次数,更少的train时间
  - 其他问题
    - 为何不使用语义分割
      - text instances often lie close to each other, making them hard to separate via semantic segmentation
    - output
      - 语义分割结果,即像素是否文本
      - 和本像素与周围像素是否连接以构成文本

XMind:ZEN | 试用模式