# 电类工程导论 C 实验报告

韩冰 (组长)、陆冠东、毛咏

2018 年 1 月 13 日

## 目录

# 1 项目背景

各大书本平台 (京东、当当、豆瓣、亚马逊等) 提供的信息不完整，比如豆瓣提供更多的是读者对书本评价，京东提供更多的售价信息等. 这些问题使得我们想要购买一本书时需要一个一个网站去查询，比较耗费时间。另外，各个门户的搜索引擎都是基于标题来进行匹配，我们很难去根据内容来搜索一本图书，比如‘算法入门’，他们并不会给你匹配到《算法导论》这个算法入门的书籍。因此，我们做了一个搜索书本的平台'Book Search'，该平台具有以下功能:

1) 提供多平台书本信息 (价格、评分、封面、目录、简介等) 整合，利于你对该图书有一个直观的了解。

2) 提供通过书本封面搜索书籍功能，利于你通过随手的一张照片来搜索所要的图书。

3) 提供通过标题、类型、作者、内容等搜索书籍功能，并集成在一个搜索框中，利于你方便的搜索到自己想要的图书。

4) 提供相似书籍推荐，利于你探索更多对你口味的图书。

5) 提供京东、当当网的购买链接，学校图书馆借阅信息，利于你有快速的渠道获得该图书。

6)More...

# 2 项目技术细节

该项目以爬取的 800,000 本图书信息为基础，Lucene 文字搜索 ocr 图像识别为核心，html 和 css 网页为呈现形式，webpy 为主要框架。

注：下文只是实现大概步骤，详细显示请见各个代码中注释部分。

## 2.1 信息获取及整理

### 2.1.1 信息获取

**信息源** 信息源包括京东、当当、亚马逊、豆瓣和交大图书馆。

**多个库及工具** 以 urllib.request 为 post 方式，以 Firefox 为抓包工具，我们首先对网站进行抓包获得了所需信息的 headers，之后利用所需 headers 进行 request，得到所需的价格、简介、标题、作者、评论等等众多信息。

**多线程** 爬虫时利用多线程和队列，实现爬虫并线运行，提高爬虫效率。并且在爬虫中加入 random 时间间隔，来防止触发网站反爬机制。

**查错机制** 在网站的反爬机制中，我们遇到了网站会返回有误的信息，我们在爬虫中加入查错机制，如果返回信息不对，则隔段时间重新爬取。

**模拟浏览器** 在爬取时，遇到有些网页是动态网页，直接爬取并不能收到有效信息。我们采用了 PhantomJS 无头浏览器来模拟浏览器模拟运行 js，得到所需信息。

**动态 ip** 在有些网站，服务器反爬机制则是短时间内某 ip 访问次数，之前的 headers 和模拟浏览器并不能解决这个问题。我们解决这个问题采用了远程架设 ADSL 动态 ip 服务器，利用了 ADSL 庞大的 ip 池，来实现多 ip 爬取的效果。最终可以获得大量的数据。

**断点续爬** 因为数据量很庞大，需要很长时间。我们实现了断点续爬，防止中间网络不稳定或其他因素导致爬虫出问题。

### 2.1.2 信息整理

将所需信息从之前保存的本地文件中提取出来。

1) 采用 BeautifulSoup 包从网页信息中提取了其中的标题、简介、借阅信息等。

2) 采用 json 来解析 request 所得结果，得到评价、价格等等。

3) 采用 pickle 来储存提取信息，便于索引人员进行操作。

4) 实时从豆瓣和图书馆获取相关借阅信息。利于查看图书评价和查找订阅图书。

## 2.2 Lucene 文字索引

### 2.2.1 书本文字搜索

该部分索引主要分为 5 步，步骤如下：

1) 在所爬到的所有书籍中建立索引，在这之中建立了一份对书名分词的索引以及一份书名所有字符中间空开的索引，利用 whitespaceanalyser 建立，其中第二份索引就相当于利用 standardanalyser 建立。

2) 进行搜索之前对于搜索项赋予不同的权值，为了平衡 lucene 评分机制中文本越长分值越低的算法 (即 norm(t,d))，对于 command 中的每一条，赋予权重 (sqrt(len(item)))，即越长的匹配越有价值得分越高。

3) 由于 jieba 分词的分词颗粒程度较大，例如大学物理这样的词是不进行分词的，当上一条搜索没有结果时，会进行逐字匹配，即利用建立索引时的按照一个字一个字分开的书名进行匹配。由于这样的匹配方式不是很有说服力，因此赋予的权值较小 (0.8)。

4) 另一方面，在搜索书籍的时候，如果这一个词条在 comment 内部存在，则会进行附加分，用 (SHOULD) 实现。由于在 comment 出现的字段不一定在书名里出现，因此在这里解除了书名中 (MUST) 的要求。由于 comment 数量较大，对于评价性的文字出现的频率较高，为了防止其影响书名匹配的要求，在这里赋予权值 0.5。

5) 利用同样的方法对于标签也进行搜索，这里赋予的权值是 2.0，是一个非常大的数字。也就是说，如果在 command 内出现了标签名，那么这个结果的分数将会远超没有这一项的结果，会排在首位。

### 2.2.2 相关书本推荐

将搜索到的图书，重新迭代入搜索函数，对于书名进行重新检索。此时，书名搜索所分配的权值和分词方式不变，同样也进行逐字匹配。所得到的打分结果中，排名第一的当然是一切都符合的原书，那么第二、三、四本就是与原书最相近的三本书，这些书在书名上有很大的相似匹配度，也就是猜你喜欢返回的结果。针对当当京东图书的书名中要素过多，无关词语（如正版、包邮等等）会影响搜索结果，我先对书名进行拆分为不同要素的列表，然后将一些噪音词汇给剔除，之后再对剩下每一个要素进行搜索，这样就解决了这些词语的干扰。

## 2.3 图像搜索

### 2.3.1 OCR 方向 (主)

主要包括两部分，第一部分是文字位置检测，第二部分是文字识别。文字识别采用了 Github 开源项目 ChineseOCR，因此主要在于自己完成的文字位置检测环节，这一环节采用了两种方案。

**第一种思路** 来自【ECCV2016】Detecting Text in Natural Image with Connectionist Text Proposal Network(CTPN) 一文，主要步骤如下：

1) 使用 VGG16 的前 5 个 Conv stage(conv5) 提取图片特征，可以得到一个 W×H×C 的 feature map。

2) 在 feature map 上取 3×3×C 的滑窗的特征 (这些特征之后要用于预测该位置的 k 个 anchor 的信息，其中 vertical anchor 是宽度都为 16，高度从 11 到 273 以 0.7 为比例的等比数列)。

3) 将每一行的 3×3×C 的特征 (共 W 个) 输入到双向 LSTM(rnn) 中，得到 W×256 输出，并将其输入到 512d 的 fc 层。

4) fc 层特征输入到三个分类或者回归层中。第二个 2k scores 表示的是 k 个 anchor 的类别信息。第一个 2k vertical coordinate 和第三个 k side-refinement 是用来回归 k 个 anchor 的位置信息。2k vertical coordinate 表示的是 bounding box 的高度和中心的 y 轴坐标，k 个 side-refinement 表示的 bounding box 的水平平移量。

5) 得到的 text proposal(文本框) 平成一个水平的 sequence，这个 sequence 中的内容包含文字。

**该网络训练环境及其他信息** Ubuntu16.04 LTS，TensorFlow 1.4，Nvidia 388.59 WHQL。使用了单路 Geforce GTX 1070，耗时约 3 小时。

**优点** 识别横向文字精确度极高，由于没有可用的验证集，通过大致人工判断，正确率超过 95%。

**缺点** 由于论文【ECCV2016】Detecting Text in Natural Image with Connectionist Text Proposal Network(CTPN) 中的检测思路是根据英文而来的，因此对于检测竖向排列的中文是没有办法的 (英文没有竖向排列且文字正向)；尚未完成对于斜字的检测，得到的 text proposal 全部都是横向 sequence；检测时间大约在 3s 左右，要达到较好用户体验可能需要较强的服务器。

**第二种思路** 使用了 opencv，在光学角度进行文字探测，是基于形态学和边缘检测的算法，主要步骤如下：

1) resize 图像尺寸。降低图像分辨率，得到小尺寸图像，减少后续计算量。

2) 使用 sobel 算子进行边缘检测。包括文字在内，图像中所有边缘都被检测出来，比较多种算子，sobel 算子计算量小，在边缘检测要求不高的情况下，能取得快速的效果。

3) 将图像二值化。我采用的是 OTSU 最大类间方差法，这种方法相比于平均值法，更容易去除图像中的各种噪点。从而使输出图像更干净。

4) 平滑滤波。二值化之后，仍然存在一些影响结果的噪点。在此时仍然需要在空间域进行平滑滤波操作。我使用选择式掩模平滑滤波法，选取 5*5 的移动窗口，在窗口中以中心像素点为基准点，创建 9 个屏蔽窗口，分别计算灰度值平均值和方差。选择方差最小的窗口进行平均化，完成滤波平滑的目的，这种平滑避免了对题目的边缘进行平滑，具有选择性。

5) 连接文字区域。前面的处理，我得到干净、清晰的包含文字的二值图像，之后将利用形态学方法将文字边缘连接起来，得到文字块区域。第一步，对结果进行闭运算，消除图像中的长细的小沟和裂缝。之后，进行膨胀运算，将文字区域连接起来，考虑标题一般都是横向，所以采用 1*3 的元素模板。经过实验，使用 6 次反复的腐蚀和膨胀操作，就能将标题区域涂抹，将文字都连接起来。

6) 定位候选文字块区域。文字块已经形成，最后是讲文字块区域提取出来。连通域标记算法的原理是根据相邻像素之间的连通性。一、以一个值为‘0’的像素点为起点。向所有 8-邻域搜索‘0’像素点。二、对各个值为‘0’的像素点再进行同样的 8-邻域搜索，直到所有点的 8-邻域像素值都为‘255’为止。三、取出区域的四个顶点的坐标即可。

7) 筛选结果。标题一般只有一个，在最后取得的所有文字块区域，进行大小的筛选，标记出合适大小的区域。得到最终结果，过小的都被去除。

**优点** 速度快，相比较于第一种方法，神经网络，这个速度更快。计算要求低，在低配置的电脑上也能取得不错的效果。

**缺点** 准确率不能过度保证。和神经网络相比，准确率会低上一些。因为此方法是根据形态学进行处理的，有一些前提，就是图书中封面中最醒目的是标题，其他因素不会形成梯度很大，并且密度很大的结果。如果有一个封面图片中，有一个很复杂且很醒目的图片，就会出现误判，将此认定为标题。

### 2.3.2 LSH 方向 (辅)

LSH 缩小范围之后，使用 SIFT 对范围内图片匹配，得到和待搜索图片匹配 score 最高的 5 张图片，返回包含 5 个 ( label，score)tuple 的 list(label 为人工手动标注，网上没有这方面的数据)。将其 label 与 OCR 得到 string 匹配，如果存在的话就可以判定该识别为正确，若无匹配项，则把 score 最高的 label 和 OCR 得到 string 共同作为 keyword 进行搜索。

**优点** 数据集较小时，该方法的搜索效率和准确度很高。

**缺点** 随着数据集不断增大，准确度急剧降低。以 OCR 路线为绝对正确，可得该方法在数据量约为 1, 200 时的准确率仅为 32.14%。因此此方法只做辅助，去掉甚至可能整体搜索正确率会更高 (未做实验)

## 2.4 web 框架和 UI 相关

### 2.4.1 web 框架

我们采用 web 框架，来实现整个程序的交互。

1)/index 为文字搜索主页面。在此页面，可以跳转至图像搜索页面。当你把想搜的 keywords 输入在输入框中，点搜索，便会触发搜索函数，将结果存为全局变量并转到 result 页面。

2)/img 为图像搜索主页面。在该页面，实现文件选择和上传功能。每当选择某个文件的时候，在主程序中，会把该文件直接读入，然后在子目录/tmp 下写出，形成缓存文件，将路径传递给图像识别函数，最后利用 os 删除该文件。得到搜索结果后，将结果存为全局变量并转到 result 页面。

3)/result 为搜索结果显示页面。在该页面，展示所有的搜索结果的书名，类别和图片展示给用户，我将搜索结果存入一个全局变量中，然后将所需信息传入 webpy 的模板，展示出该页面。

4)/数字为搜索结果详情展示页面。在搜索结果页面，你点击感兴趣的书之后就会跳转到详情页面。我是利用 re 正则来实现知道你点击的是哪本书，每个框都会跳转到相对应的页面，我获得跳转页的网址就可以获得点击的书的序号，在之前搜索的过程中，已经将 result 定义为全局变量，从中取出相对应的书的信息传递给该页面展示出来即可。

5) 在实现网页和后台交互的时候，主要使用的是 form 表单来从前端获取信息传递给后端的。在 html 模板中，给不同的项目不同的 name 属性，就可以实现前端与后端交互，建立整个框架。

#### 2.4.2 UI

我采用 html 来写整体页面，以 css 还有 js 来美化模板。

1) 自适应网页设计。进行流动布局，字体和图片选择相对大小。让网页能适应不同宽度的屏幕分辨率，并且对内容进行从新缩放或排版。

2) 动态显示特效。我使用相关 js 代码，将动态特效加入网页之中。让搜索结果随着鼠标出发事件而淡入淡出。达到美观的效果。

3) 透明悬浮特效。利用 js 代码动态页面。让所有的图像和底色相同，制造出悬浮的效果。使页面更加美观。

4) 关键词自动补全。对于键入的关键词，在后台的 js 代码中，自动匹配相关的关键词。下拉框中会有备选词供你自动补全。该自动补全是模糊匹配，并不是绝对匹配，所以具有一定纠错能力。这个功能利于用户快速选择想要的图书。

# 3 更多改进

我们开发此项目花费了不少时间，但是仍然有很多令我们不满意的地方，下面是我们在接下来可以改进提升的地方，不仅限于此，会不断添加更多的新功能来完善我们的项目。

## 3.1 添加数据库

添加历史浏览记录。根据每个用户的历史查询来提供更加精准可靠的推荐，而不是单一地根据当前搜索书籍，这也意味着我们接下来将会有更加复杂的索引系统。

## 3.2 实现 OCR 的 classification 部分

之前曾尝试写了 classification 部分代码，但是由于训练所需时间成本较大，考虑到当时时间已经比较晚，因此我们选择放弃使用自己的代码，采用了开源项目的代码，但是我们会在后续添加并优化自己的程序。

## 3.3 更全面的数据

我们在爬虫的时候，因为在不断改进代码，以适应不同的网页。中间也让一些网页出了错。最后的数据库就没有采用。我们在之后可以利用更全的书籍网站，来构建更全的数据库。

# 4 如何使用

## 4.1 搜索图书标题

将想要搜索的图书标题键入搜索框。点击搜索即可得到与该关键词匹配度最高的书本。

## 4.2 搜索图书类型

将想要搜索的图书类型键入搜索框。点击搜索即可得到该类型标签最符合的书本。

## 4.3 搜索图书内容

对于想要搜索的某内容并且忘记书名的搜索。比如搜索：“算法入门书籍”，下面就会给你推荐有些评价和简介中包括‘算法入门’书籍，供你选择。

## 4.4 图片搜索

从主页面中进入图片搜索页面，在图片搜索界面中，选择待搜索图像文件，上传后，点击搜索即可。

## 4.5 详情

你可以查看该图书的评价，分数、简介、目录等信息，你还可以点击 JD 和当当的 logo，就可以转入相应购买页面。

你可以点击图书馆详情，就可以到图书馆相关图书的借阅页面。

## 4.6 更多

请查看 github 中的 demo 视频，其中包含了大部分的操作过程, 关于环境配置请详见项目 readme.md

# 5 项目分工

**韩冰 (组长)516030910523** 京东爬虫、图像识别 (光学)、界面框架、整合

**毛咏 516030910552** 豆瓣、图书馆信息获取、图像识别 (ctpn)

**陆冠东 516030910529** 当当爬虫、图书索引、图书推荐