任意网页正文内容主题词提取

原创 异尘 机器拾趣 2019-09-16

一般来说，给一段文本，要求使用NLP技术进行关键词抓取的需求是很常见的。但是，如何做到任意网页页面进行关键词抓取呢？

前言

网页正文内容主题提取，即任意给一个网页url，通过爬取网页内容和文本分析，得出该网页内容的关键词，作为网页的标签。这些关键词和标签在做流量分析_内容推荐方面有非常重要的意义。比如说我们做数字营销，用过页面来做用户引流，我们就可以知道吸引用户过来的点是什么，用户的潜在需求是什么；另外，针对内容社区的用户画像/推荐系统，关键点也是文章/页面的主题和标签。

这个任务涉及的技术点主要有以下几个：

1. 网页爬虫。做网页内容分析，首先得根据url把网页内容扒下来吧。

2. 正文提取。现在的web页面是非常复杂的，除了正文外，包含了大量的广告_导航_信息流等，我们需要去除干扰，只提取网页的正文信息。
3. 主题模型 。拿到正文文本后，就需要做NLP来提取主题关键字了。

网页爬虫

这里的网页爬虫和一般的爬虫还不太一样，会简单许多，主要是把原始网页的HTML抓下来即可，主要是为后续的分析挖掘打下基础，属于数据采集的阶段。这里我们采用了Python的 `requests` 包。 `requests` 相对于Python自带的 `urllib` 来说，API更为人性化，鲁棒性也更好。

```
import requests
r = request.get(url)
r.encoding='utf-8'
html = r.text
```

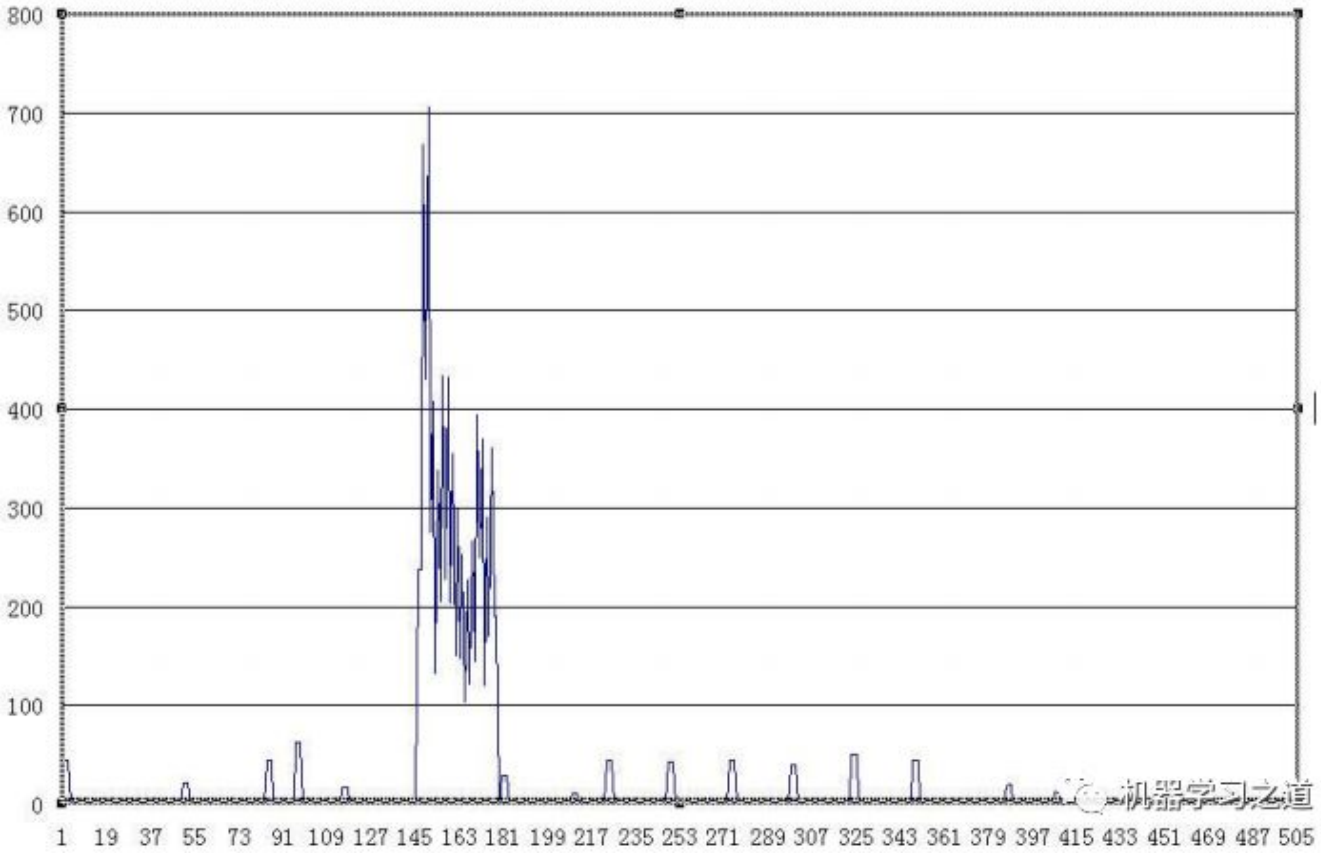
正文提取

通过研究爬取下来的原始HTML，我们可以看到是非常负责而且杂乱无章的，充斥着大量的js代码等。我们首先需要解析HTML，尽量过滤掉js代码，留下文本内容。这里我们采用了Python的 `BeautifulSoup` 包。这个包堪称Python一大神器，解析HTML效果非常好

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, features="html.parser")
for script in soup(["script", "style"]):
    script.decompose()
text = soup.get_text()
```

我们想要的是网页的正文内容，其他的诸如广告或者导航栏等干扰内容需要尽可能的过滤掉。通过 `BeautifulSoup` 可以解析出整个HTML的DOM树结构，但是每个网页HTML的写法各不相同，单纯靠HTML解析无法做到通用，因此我们需要跳出HTML的思维，使用其他的方法来提取网页的正文。这里有个很优雅的方式是“基于行块分布函数”的算法 `cx-extractor` 。

> 基于行块分布函数的通用网页正文抽取：线性时间、不建DOM树、与HTML标签无关
> 对于Web信息检索来说，网页正文抽取是后续处理的关键。虽然使用正则表达式可以准确的抽取某一固定格式的页面，但面对形形色色的HTML，使用规则处理难免捉襟见肘。能不能高效、准确的将一个页面的正文抽取出来，并做到在大规模网页范围内通用，这是一个直接关系上层应用的难题。
> 作者[1] 提出了 《基于行块分布函数的通用网页正文抽取算法》[2] ，首次将网页正文抽取问题转化为求页面的行块分布函数，这种方法不用建立Dom树，不被病态HTML所累（事实上与HTML标签完全无关）。通过在线性时间内建立的行块分布函数图，直接准确定位网页正文。同时采用了统计与规则相结合的方法来处理通用性问题。作者相信简单的事情总应该用最简单的办法来解决这一亘古不变的道理。整个算法实现代码不足百行。但量不在多，在法。



上图就是某个页面求出的行块分布函数曲线。该网页的正文区域为145行至182行，即分布函数图上含有最值且连续的一个区域，这个区域往往含有一个骤升点和一个骤降点，因此，网页正文抽取问题转化为了求行块分布函数上的骤升点和骤降点两个边节点。这里我们采用了这个算法的Python实现GitHub - chrislinan/cx-extractor-python: 基于行块分布函数的通用网页正文抽取算法的Python版本实现，添加了英文支持/ Web page content extraction algorithm, support both Chinese and English[3]

```
from CxExtractor import CxExtractor
cx = CxExtractor(threshold=40)
text = cx.getText(text)
```

```
texts = text.split('\n')
```

主题模型

拿到网页正文内容文本后，就需要提取正文主题关键词了。常见做法有3种：

1. TFIDF
2. Text-Rank
3. LSI/LDA 这里我们先采用TFIDF的方式来做。

> TFIDF(Term Frequency Inverse Document Frequency)是一种用于信息检索与数据挖掘的常用加权技术。词频（TF）=某个词在文本中出现的次数/该文本中总词数 逆向文档频（IDF）=log(语料库中所有文档总数/(包含某词的文档数+1)) 我们通过TF，也就是某个词在文本中出现的频度，来提升这个词在主题中的权重，然后我们通过IDF值，即逆向文档频来降低公共词的主题权重。TF*IDF也就得到了我们要的主题词权重。

做TFIDF，首先步骤是分词。分词的效果取决于词典的构建，且对后续关键词提取影响巨大。首先要基于分析的行业主题建立专用词典，然后还需要维护停用词的词典。、 有了词典后，我们就可以采用Python分词的神器 `jieba` 来处理分词。

```
import jieba
jieba.load_userdict('./dict.txt')    #自定义词典
stopwords = set([line.strip() for line in open('stopwords.txt', 'r', encoding=

word_lists = []
for text in texts:
    word_lists += (list(jieba.cut(text, cut_all=False)))
word_lists = [w for w in word_lists if not is_stop_word(w)]
```

分词完毕后，我们就可以计算TFIDF了。可以通过 `gensim` ， `scikit-learn` 等机器学习专用包来做， `jieba` 本身也提供这个功能，这里我们直接用 `jieba` 。

```
import jieba.analyse
keywords = jieba.analyse.extract_tags(' '.join(word_lists), topK=20, withWeigh
```

注意这里有个参数是 `allowPOS` ，按照词性过滤。这个需要根据实际的业务需求来设置。

> 词性标注(Part-Of-Speech Tagging, POS tagging)，是语料库语言学中将语料库内单词的词性按照其含义和上下文内容进行标记的文本数据处理技术。常见标注示例：
> n 名词
> nr 人名
> ns 地名
> nt 机构团体
> nz 其他专名
> a 形容词
> v 动词

服务

到这里，我们的关键词提取就结束了，为了方便其他同学来使用，我们可以用 `Flask` 做一个restful api，输入为网址url，输出为提取出的关键词并排序。

总结

在这篇文章里，我们完成了从任意网页url提取正文主题关键词的功能。在主题模型这块采用了常见的TFIDF的算法来解决，可以快速出一个原型提供给业务方使用。后续我们会继续优化，采用更多的算法来进一步提升效果。

References

[1] 作者: *http://weibo.com/cx3180*
[2] 《基于行块分布函数的通用网页正文抽取算法》: *http://cx-extractor.googlecode.com/files/%E5%9F%BA%E4%BA%8E%E8%A1%8C%E5%9D%97%E5%88%86%E5%B8%83%E5%87%BD%E6%95%B0%E7%9A%84%E9%80%9A%E7%94%A8%E7%BD%91%E9%A1%B5%E6%AD%A3%E6%96%87%E6%8A%BD%E5%8F%96%E7%AE%97%E6%B3%95.pdf*
[3] GitHub - chrislinan/cx-extractor-python: 基于行块分布函数的通用网页正文抽取算法的Python版

本实现，添加了英文支持/ Web page content extraction algorithm, support both Chinese and English: *https://github.com/chrislinan/cx-extractor-python*

阅读原文