体育新闻搜索引擎设计

——电类工程导论结课项目

纪喆	金勤鹏	刘文轩	庄轶哲 *
518030910303	518030910304	518030910308	518030910321

上海交通大学

日期：2020 年 1 月 14 日

摘　　要

本文为上海交通大学电类工程导论（EE208）课程结课项目报告，本组在该项目中完成了对新浪体育和网易体育相关新闻的定向爬取、信息索引、高效检索，成功构建了一个针对体育新闻的搜索引擎。本报告侧重讲述本搜索引擎后端搭建过程，详细的前端功能介绍将在配套的PPT中进行展示。用户可以访问http://101.132.133.247:8080/进行体验。

关键词：搜索引擎，网络爬虫，信息检索，图片搜索，聚类

1　实验准备

1.1　实验环境

本实验大致有网页爬取，构建索引，设计前端三个步骤，全程在Ubuntu系统中进行。为完成实验效果，本组使用的工具如下。

- Scrapy：具有爬取网站数据，提取结构性数据等功能的应用框架。
- pylucene：全文检索引擎的架构，提供了完整的查询引擎和索引引擎。
- jieba：中文分词工具。
- keras：由Python编写的开源人工神经网络库。支持现代人工智能领域的主流算法，包括前馈结构和递归结构的神经网络，也可以通过封装参与构建统计学习模型。
- h5py：用于处理HDF5文档，储存图片特征向量
- hnswlib：分层可通航小世界图算法，用于以图搜图。
- sklearn：数据分析挖掘工具，用于新闻聚类
- Web.py：轻量级Python web框架。
- bootstrap：用于快速开发网站的前端框架。

其中，scrapy需要pyOpenSSL, lxml等第三方包。 pylucene要求系统装有Java, python-dev, ant, JCC等。bootstrap可从官网进行下载。其他绝大多数库均可使用pip直接安装，pylucene推荐使用anaconda配置。

*按姓氏拼音排序

1.2 实验目的

本实验主要目的在于整合本学期的各次实验，将所学转为所用。具体的，本次实验要求包括：

1. 爬取不少于1万个新闻网页
2. 实现关键词检索以及新闻图片检索
3. 实现相关度排序以及时间排序功能
4. 进行相似新闻的自动聚类（optional）

通过对这些功能的实现，学生对信息检索有了更加深刻而立体的认知。其中，对新闻网页数量的要求保证了搜索引擎语料库的丰富性；关键词搜索、排序功能的实现完成了搜索引擎的必要功能；新闻图片检索相应当下图片搜索热潮；相似新闻自动聚类则是实现信息的模块化，区域化。

2 实验原理

所谓搜索引擎，就是根据用户需求与一定算法，运用特定策略从互联网检索出制定信息反馈给用户的一门检索技术。为完成这一目标，搜索引擎需要构建一定的信息库，对信息库中的内容进行有效的索引，最终实现高效检索的功能。由此，搜索引擎依托于众多技术，包括但不限于网络爬虫、检索排序、网页处理、大数据处理、自然语言处理。

2.1 Scrapy爬虫框架

由于本次实验爬取的网页数目较多，对网页的选择性较强，如果使用lab3中利用requests等库手动完成的小型爬虫进行爬取，效果不佳且结构繁琐。本着构建良好的代码生态，充分利用现有框架的原则，本实验采用了Scrapy完成网络爬虫模块。

Scrapy是一个为了爬取网站数据，提取结构性数据而编写的应用框架。可以应用在包括数据挖掘，信息处理或存储历史数据等一系列的程序中。

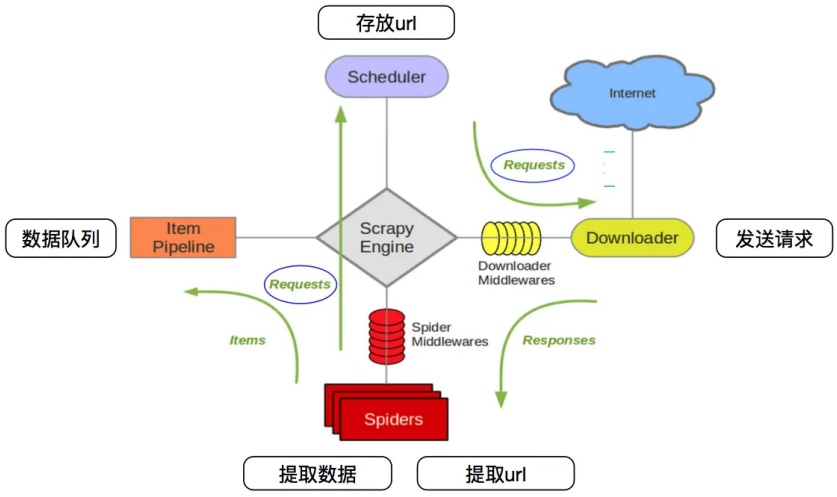

图 1: scrapy框架示意图

如图1所示，scrapy将实现网络爬虫的各个功能分为不同的模块：

1. Scrapy Engine，引擎，负责其他各个模块的通信，信号与数据的传递。
2. Scheduler，调度器，负责接受引擎发送过来的Request请求，并按照一定的方式进行整理排列，入队，当引擎需要时，交还给引擎。
3. Downloader，下载器，负责下载Scrapy Engine(引擎)发送的所有Requests请求，并将其获取到的Responses交还给Scrapy Engine(引擎)，由引擎交给Spider来处理。
4. Spider，爬虫，负责处理所有Responses,从中分析提取数据，获取Item字段需要的数据，并将需要跟进的URL提交给引擎，再次进入Scheduler(调度器)，
5. Item Pipeline，管道，它负责处理Spider中获取到的Item，并进行进行后期处理（详细分析、过滤、存储等）的地方。
6. Downloader Middlewares，下载中间件，可以自定义扩展下载功能的组件。
7. Spider Middlewares，爬虫中间件，可以自定扩展和操作引擎和Spider中间通信的功能组件。

本实验的具体使用见3.1网页爬取。

2.2 jieba分词

由于本次实验处理的是中文网页，因此中文分词必不可少。pylucene只提供对英文、德文两种语言的分词功能，因而对于中文的分词，我们必须额外使用中文分词工具。目前中文分词工具较多，本次实验使用应用较为广泛的jieba（结巴）分词工具。

jieba分词之所以应用广泛，主要因为有以下特征：

- 支持三种分词方式：

  全模式 把句子中所有的可以成词的词语都扫描出来, 速度非常快，但是不能解决歧义。

  精准模式 试图将句子最精确地切开，适合文本分析。

  搜索引擎模式 在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。
- 支持繁体分词
- 支持自定义词典
- MIT 授权协议

2.3 pylucene 文本索引建立

lucene是一个开放源代码的全文检索引擎工具包，它不是一个完整的全文检索引擎，而是一个全文检索引擎的架构，提供了完整的查询引擎和索引引擎，部分文本分析引擎（英文与德文两种西方语言）。lucene是当前以及最近几年最受欢迎的免费Java信息检索程序库。

pylucene是Java版lucene的Python版封装。这个工具的目标是让Python使用lucene的文本索引和搜索能力。它与Java版lucene的最新版本是兼容的。pylucene把一个带有JAVA VM的lucene嵌入到Python进程中。

利用pylucene，即可把爬取好的文本信息建立索引，达成网页搜索功能。具体实现见3.3。

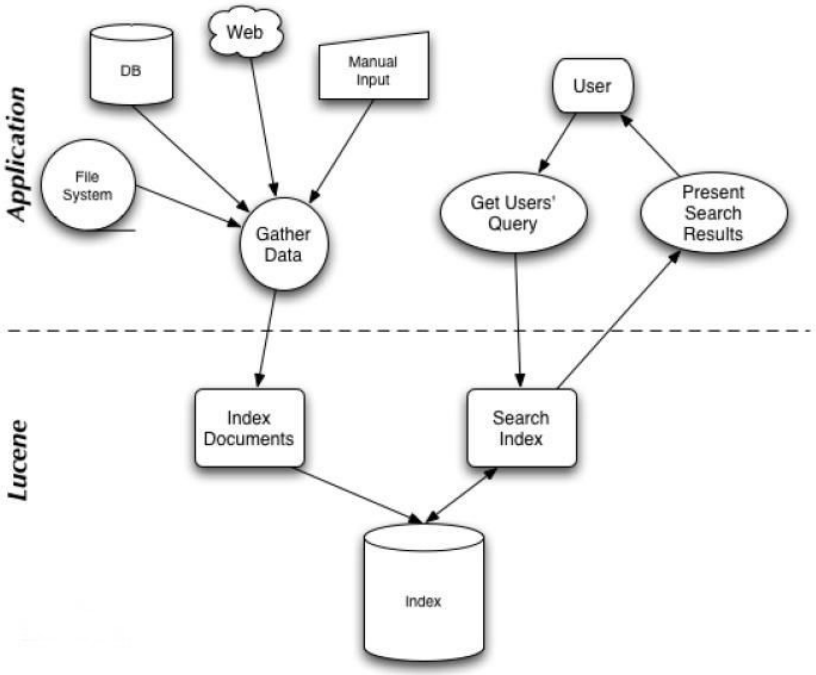

图 2: lucene的架构

2.4 VGG16 基于深度学习的图片特征提取

VGG是由Simonyan 和Zisserman在文献《Very Deep Convolutional Networks for Large Scale Image Recognition》中提出卷积神经网络模型，其名称来源于作者所在的牛津大学视觉几何组(Visual Geometry Group)的缩写。该模型参加2014年的 ImageNet图像分类与定位挑战赛，取得了优异成绩：在分类任务上排名第二，在定位任务上排名第一。其突出贡献在于证明使用很小的卷积（3*3），增加网络深度可以有效提升模型的效果，而且VGGNet对其他数据集具有很好的泛化能力。到目前为止，VGGNet依然经常被用来提取图像特征。

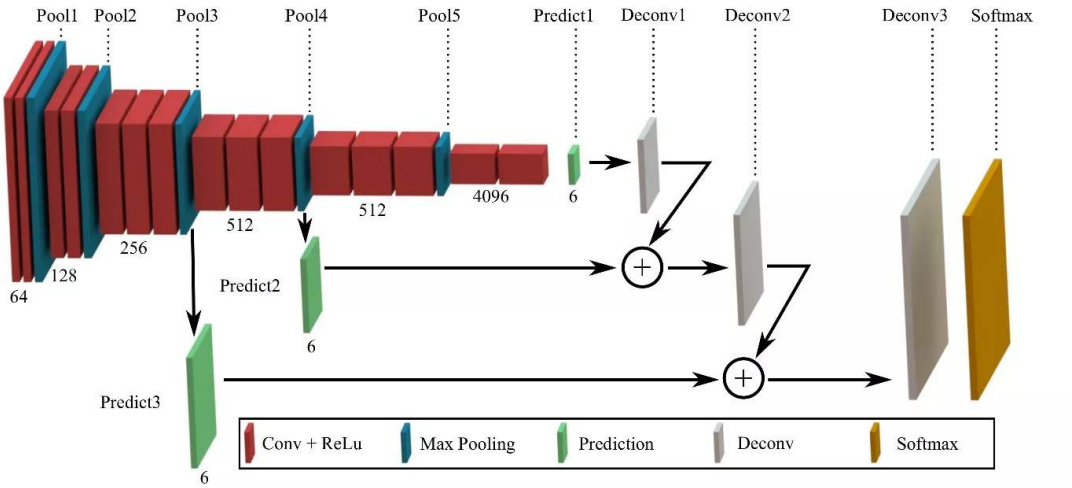

图 3: VGGNet结构示意

Keras是一个由Python编写的开源人工神经网络库，可以作为Tensorflow、Microsoft-CNTK和Theano的高阶应用程序接口，进行深度学习模型的设计、调试、评估、应用和可视化。

本实验利用keras提供的VGG网络模型提取图片特征，生成长度为512的特征向量，并以特征向量的欧氏距离作为相似度的判断标准。从而以图搜图的流程如下：

1. 用VGG网络生成目标图片的VGG特征向量
2. 与图片库中所有图片的特征向量（事先生成保存）进行距离计算
3. 选出特征向量距离较近的几张作为匹配对象

2.5 h5py 图片特征向量的储存

一个HDF5文件就是一个容器，用于储存两类对象：

datasets 类似于数组的数据集合

groups 类似于文件夹的容器，可以储存datasets和其它groups

使用准则 groups类似于字典（dictionaries），dataset类似于Numpy中的数组（arrays）。

本实验用HDF5文件储存图片的特征向量，作为以图搜图的数据库。

2.6 hnswlib 基于层次导航小世界图的近似最近邻搜索

由于维度高、数据规模大，直接应用最近邻方法并不可行，因此，最佳实践是使用逼近方法搜索最近邻。也就是近似最近邻方法，approximate nearest neighbor（ANN）。 ANN算法很多，当前最优的ANN算法基本上都是基于图（graph）的算法。根据相关技术人员的测评（如图4），本实验没有采用lab 14中的LSH，而选取了在时间效率和准确度上均占优的算法：HNSW

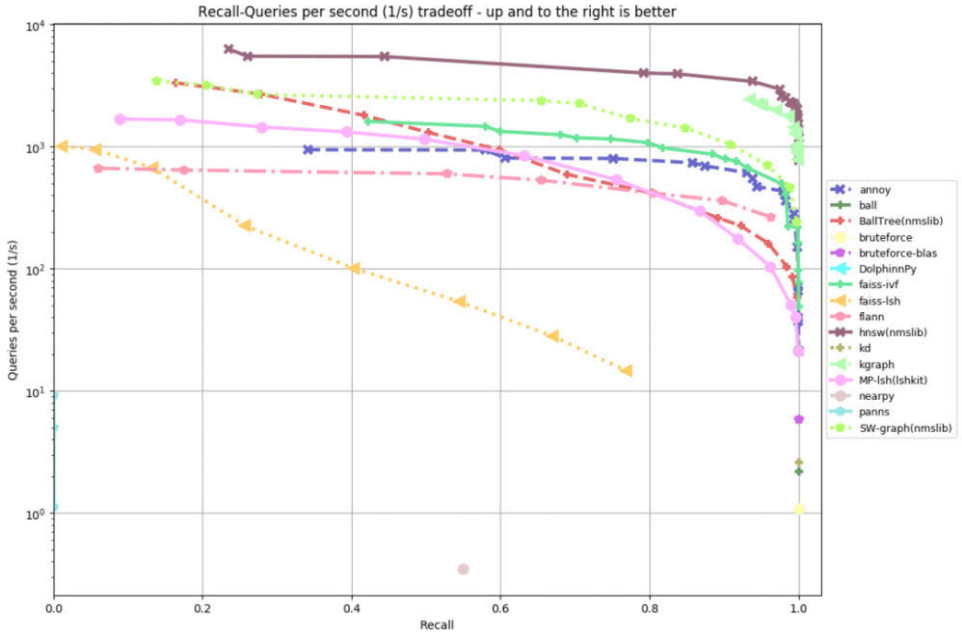

图 4: 各个算法在sift-128-euclidean测试中的表现

HNSW算法思想是根据链接的长度使用分层NSW图分离连接关系，查找每个元素需要访问的连接数和网络大小无关。算法通过递增地构建一个分层的邻接图索引结构加速查询，该图的上层是下层的子集，最底层包含全部元素。数据集中的元素在索引结构中出现的最高层是通过

一个指数衰减的概率分布随机选择的，如图5所示，查询时通过类似于跳表的结构从顶层开始使用贪心算法逐步往下层查询，每层选择最近邻点作为下一层的进入点，每个元素的连接数可以设置为常数，这样可以使得搜索过程的复杂性为对数级别。图5中黑色箭头表示贪心算法从进入点导查询点的路径，黑色方形点表示查询点，空心圆点表示数据集中的向量，可以看到从最高层到底层，节点之间连接的距离逐渐减小。

HNSW算法可以分为网络构建和查询两个阶段，实际上查询阶段类似于网络构建。网络构建阶段通过连续地将集合中的元素插入图结构实现，而检索过程就是在第0层的插入算法，区别是HNSW算法在该层找到的准备用来建立连接的候选最近邻元素被当作结果返回。

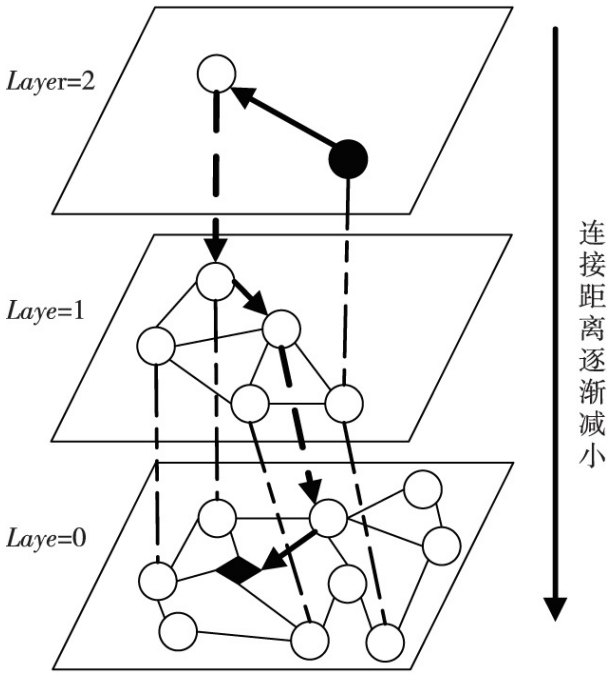

图 5: HNSW图结构示意图

2.7 K均值聚类算法新闻聚类

2.7.1 算法简介

聚类是一个将数据集中在某些方面相似的数据成员进行分类组织的过程，聚类就是一种发现这种内在结构的技术，聚类技术经常被称为无监督学习。

k均值聚类是最著名的划分聚类算法，由于简洁和效率使得他成为所有聚类算法中最广泛使用的。给定一个数据点集合和需要的聚类数目k，k由用户指定，k均值算法根据某个距离函数反复把数据分入k个聚类中。

本实验利用k均值据类算法，欲将一万条新闻分为约600个类别[1]，实现聚类，并以此为基础实现了相关新闻推荐。

[1]该数量综合考虑了新闻总条数以及相对相似程度。

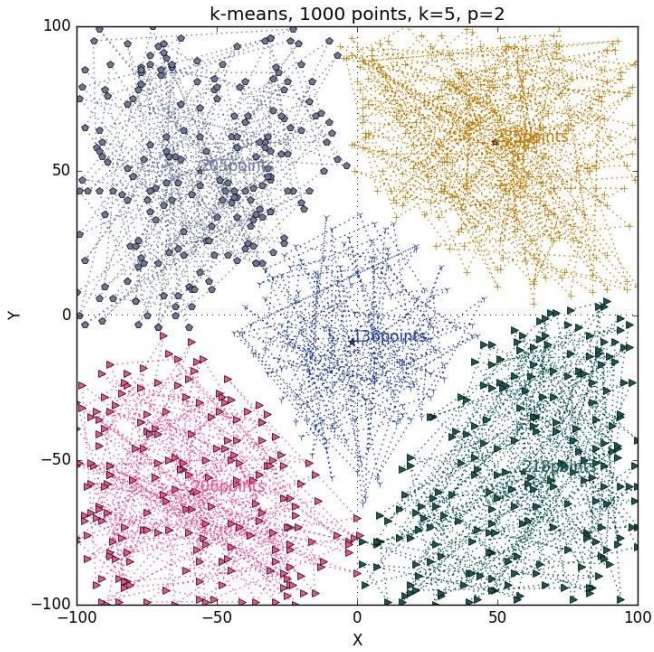

图 6: k均值聚类算法示例

2.7.2 算法流程

1. 随机选取K个对象作为初始的聚类中心。
2. 计算每个对象与各个种子聚类中心之间的距离
3. 把每个对象分配给距离它最近的聚类中心
4. 分配完毕后，重新计算每个聚类中心
5. 若不满足终止条件，回到第二步

终止条件：

- 没有（或数目达到容忍度）对象被重新分配给不同的聚类。
- 没有（或数目达到容忍度）聚类中心再发生变化。
- 误差平方和局部最小。

2.8 TF-IDF

TF-IDF（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。TF意思是词频(Term Frequency)，IDF意思是逆文本频率指数(Inverse Document Frequency)。

TF-IDF是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。TF-IDF加权的各种形式常被搜索引擎应用，作为文件与用户查询之间相关程度的度量或评级。

本实验采用TF-IDF作为词语在语料库中的重要程度，并以此作为数据对象进行聚类（k-means）

2.9 web.py 网页前端搭建

web.py 是一个轻量级Python web框架，它简单而且功能强大。在web.py的帮助下，先写好HTML模板，配套相应的py脚本，即可提供网页访问服务。

2.10 bootstrap 网页设计

Bootstrap 是由 Twitter 的 Mark Otto 和 Jacob Thornton 开发的。于2011 年八月在 GitHub 上首次发布的开源产品。该产品具有以下特点：

- 移动设备优先：自 Bootstrap 3 起，框架包含了贯穿于整个库的移动设备优先的样式。
- 浏览器支持：所有的主流浏览器都支持 Bootstrap。
- 响应式设计：Bootstrap 的响应式 CSS 能够自适应于台式机、平板电脑和手机。

本实验使用bootstrap框架进行前端网页模板设计，界面精致美观，用户体验极佳。

3 实验过程

3.1 scrapy 网页爬取

本部分对新浪体育和网易体育新闻进行定向爬取，下载相关新闻网页，为搜索引擎的构建提供信息库。在配置好各模块功能后，即可执行run.sh开始爬虫。run.sh中执行语句为

```
scrapy crawl wangyi -s LOG_FILE=all.log
scrapy crawl xinlang -s LOG_FILE=all.log
```

log将储存在all.log中。

3.1.1 items.py 新闻数据结构

根据本次搜索引擎项目的功能需求，对于每一条新闻，我们需要记录的内容包括：网址、发布时间、标题、主要图片地址、文字内容。该结构在items.py中定义，具体如下：

```
class NewsItem(scrapy.Item):
    url = scrapy.Field()
    time = scrapy.Field()
    title = scrapy.Field()
    imgs = scrapy.Field()
    content = scrapy.Field()
    ctime = scrapy.Field()
```

其中关于发布时间有两个Field：ctime和time。前者为浮点数，用于进行时间比较；后者则是字符串，用于时间显示。实际上只需要保存ctime即可，time可以经过运算直接得到，但为了简化后续工作，本次实验将两项全部保存。

3.1.2 spider 定向爬虫

对spider来说，爬取的循环类似下文:

1. 以初始的URL初始化Request，并设置回调函数。当该request下载完毕并返回时，将生成response，并作为参数传给该回调函数。
2. spider中初始的request是通过调用 start_requests() 来获取的。 start_requests() 读取 start_urls 中的URL，并以 parse 为回调函数生成 Request 。
3. 在回调函数内分析返回的(网页)内容，返回 Item 对象或者 Request 或者一个包括二者的可迭代容器。返回的Request对象之后会经过Scrapy处理，下载相应的内容，并调用设置的callback函数(函数可相同)。
4. 在回调函数内，可以使用选择器(Selectors)来分析网页内容，并根据分析的数据生成item。
5. 最后，由spider返回的item将被存到数据库(由某些 Item Pipeline 处理)或使用 Feed exports 存入到文件中。

不同的网站其网页结构也固然不同，在利用选择器分析网页内容时的具体做法也就不同。为了保证信息提取的准确性，需要为每个网站定义个性化的爬虫。以下以对网易体育新闻网页的爬虫为例进行说明。

针对网易体育新闻的个性化爬虫（类）在wangyi_spider.py中定义。该类继承了scrapy.Spider。在wangyi_spider的定义中，主要定义了几个参数以及一个解析函数parse。参数部分包括：

```
name = "wangyi"
allowed_domains = ["sports.163.com"]
start_urls = [ "https://sports.163.com/" ]
count=0
```

其中allowed_domains限制爬取域名，count用来记录爬取的网页数目。 parse函数接受response作为唯一参数。该方法负责解析返回的数据(response data)，提取数据(生成item)以及生成需要进一步处理的URL的 Request 对象。

在parse解析函数中，需要用XPath的方式选择出所需信息，包括新闻标题等。但在实验中发现，有不少网页并不是我们所期望的新闻内容网页。因此，需要一个标准来判断网页的性质。对于网易体育新闻，我们发现有如下特征：只有新闻内容网页才会有一个div专门显示新闻发布时间，且其class属性为"post_time_source"。基于此，我们以该div的存在与否判断网页性质。只有当通过

```
time=response.xpath('//div[@class="post_time_source"]/text()').extract()
```

可以提取到发布时间时，才进一步解析网页内容，并建立一个item通过yield方式返回。

无论该网页是否是新闻内容，由于主域名的限定，该网页至少是新浪体育下的内容。因此网页中的链接对我们依然是有用的。通过xpath提取网页中所有URL，对于其中的每一个URL，调用Request函数yield一个请求对象。

3.1.3 pipelines.py 信息的保存

爬取到的网页具体如何保存，在pipelines.py中以类的形式定义。为了更高效便捷地为python

脚本使用，本实验使用pickle库进行保存，而非直接保存文本。同时为了避免不兼容，没有直接dump得到的item，而是转成python自带的字典后进行存储。 dump函数的定义在my_dump.py中，主要就是调用pickle中的一个函数进行保存。

```
import pickle
def dump(filename,data):
    f=open('data/'+filename,'wb')
    pickle.dump(data,f)
    f.close()
```

相应的，为了后续操作方便使用，在my_load.py文件中定义了load函数，可以将存好的pkl文件读取加载。

```
def load(filename):
  f=open('data/'+filename,'rb')
  return pickle.load(f)
  f.close()
```

另外，为了方便计数，在该类中还定义了count变量，每保存一个网页就计数一次。

3.2 downloadpic.py 图片的下载

在scrapy爬虫阶段，对于图片只是找到了URL，但由于后续图片处理的需要，本实验需要将所有图片下载下来。这项工作由downloadpic.py进行。

本步骤看似简单，其实在实验中也遇到了一些问题。在网页中找到的一些imgurls，有太多是无用且重复的图片，包括但不限于：分享二维码，网站logo。本实验仍采取黑名单的方式进行筛选。通过黑名单筛选的真实图片将会通过requests进行下载。

具体地，先根据requests得到的header判断图片类型，然后根据图片类型进行命名，以二进制文件形式写入。

```
if 'jpeg' in r.headers['content-type']:
    Njpeg+=1
    filename=str(Njpeg)+'.jpg'
elif 'png' in r.headers['content-type']:
    Npng+=1
    filename=str(Npng)+'.png'
if filename:
    with open("pic/"+filename, "wb") as pic:
        for chunk in r.iter_content(chunk_size=1024):
            if chunk:
                pic.write(chunk)
    realimgs.append((i,filename))
```

3.3 IndexFiles.py 网页索引的建立

本部分主要使用pylucene作为检索引擎的架构。由于在scrapy爬虫阶段已经建立了结构化的

信息数据，本部分功能实现较为简单。

3.3.1 基本流程

首先利用pylucene建立JAVA VM，

```
if __name__ == '__main__':
  lucene.initVM(vmargs=['-Djava.awt.headless=true'])
```

后调用IndexFile函数。在该函数中，根据实际使用需求，个性化定义了两个字段类型。定义如下。

```
t1 = FieldType()
t1.setIndexed(False)
t1.setStored(True)
t1.setTokenized(False)

t2 = FieldType()
t2.setIndexed(True)
t2.setStored(True)
t2.setTokenized(True)
t2.setIndexOptions(FieldInfo.IndexOptions.DOCS_AND_FREQS_AND_POSITIONS)
```

t2类型被设置为Tokenized，并且将文档、词频、出现位置全部索引，以达到更精准的搜索结果。t1类型只做存储功能，在搜索到相应doc时可以返回内容即可。以此为基础，根据不同字段的作用， t1类型的使用者有：raw_title, filename, url, time, ctime, content, imgs, imgurls； t2类型的使用者有：title。

添加记录时，对于每一个doc，使用

```
doc.add(Field("title", title, t2))
```

进行添加。对于多值字段，只需要多次使用上述语句即可。

在IndexFile函数运行结束后，主函数输出count的值，显示成功索引的文档数量以确保所有文档均已成功索引。如果数量不对，则可以通过过程中的程序输出进行查看失败的索引记录。

3.3.2 细节处理

中文编码 在学期中的小实验中，常遇到的问题为中文编码乱码问题。不同的网页编码不尽相同，如果直接下载处理，经常会乱码。在之前的实验中，采取了统一编码保存的方式解决。但在本次实验中，由于爬取的网站固定，编码一致，因此不存在中文乱码问题。如果直接沿用小实验的做法，只会让实验过程变缓，浪费时间。因此本实验不再关注编码问题。

中文分词 本次处理的网页全部为中文网页，中文分词非常重要。在建立索引时，使用了jieba提供的精准模式分词，使用

```
title = ' '.join(jieba.cut(title))
```

即可将原始的标题经过搜索引擎模式分词后再通过空格拼接。[2]最终配合pylucene中的Whitespace Analyzer，以空格为唯一标志分词建立索引，即可实现中文文本的索引建立。

标题处理 由于经过搜索引擎模式的分词处理，不能便捷地通过分词结果直接得到原始标题[3]，因此在索引建立是，额外增加了一个raw_title的字段，用于储存原始标题。这样处理就保证了在搜索结果界面显示给用户的是原始的正常网页标题。

3.4 Search.py 网页搜索的实现

3.4.1 基本流程

1. 在命令行中接收用户的输入
2. 使用结巴的搜索引擎模式分词处理用户输入
3. 处理过后的目标传入lucene中在title字段中搜索
4. 将固定数目的结果返回，打印出用户所需要的信息。

3.4.2 细节处理

在这些字段中，除了title作为搜索对象的特殊性，imgs和imgurls也有其多值的特殊性。对于其他单值字段，使用doc.get即可获得内容，但对于imgs则会返回无效。需要使用doc.getValues得到字段的值（列表）。

3.5 index.py 图片特征提取

为了达到以图搜图的目的，需要将所有下载好的图片逐一转化为特征向量以进行比较。本次实验中使用表现优秀的VGG网络模型，将每个图片转化为一个特征向量。本网络模型在keras中有提供，可以调用其中的函数进行特征提取。

在extract_cnn_vgg16_keras.py中，利用keras提供的函数model_vgg.predict实现了VGG特征的提取函数，如下：

```
def vgg_extract_feat(self, img_path):
    img = image.load_img(img_path, target_size=(self.input_shape[0], self.
        input_shape[1]))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input_vgg(img)
    norm_feat = feat[0]/LA.norm(feat[0])
    return norm_feat
```

index.py文件通过遍历pic文件中的每个图片，通过上述代码产生特征向量，将图片的名称与特征向量分别放入一个列表中[4]。

[2]jieba.cut返回一个generator，通过字符串的join功能拼接。

[3]搜索引擎模式的分词会将长词再分，最终会有重复词语。

[4]共两个列表，feats和names。

```
norm_feat = model.vgg_extract_feat(img_path)
img_name = os.path.split(img_path)[1]
feats.append(norm_feat)
names.append(img_name)
```

在最后将两个列表存入vgg_featureCNN.h5中，用来后续的查询比较。

```
h5f = h5py.File(output, 'w')
h5f.create_dataset('dataset_1', data = feats)
h5f.create_dataset('dataset_2', data = names)
#h5f.create_dataset('dataset_2', data = np.string_(names))
h5f.close()
```

3.6 query.py 以图搜图

以图搜图的功能通过HNSW算法实现，算法原理已在2.6简述。本次实验使用hnswlib提供的函数直接调用。

在收到图片后，首先利用VGG网络模型计算VGG特征向量。

```
model = VGGNet()
queryVec = model.vgg_extract_feat(filename)
```

后建立HNSW索引，并导入先前保存好的数据图库vgg_featureCNN.h5。代码如下：

```
p=hnswlib.Index('l2',512)
p.load_index('hnswindex.bin')
p.set_ef(50)
```

其中在建立索引时，指定了用欧氏距离（l2）为指标，且特征向量维度为512。ef为hnsw算法中一个重要的参数。该参数含义为，在查找过程中，会存储每一层距离datapoint最近的ef个邻居。因此，ef越大，搜索效果也就越好，但相应的搜索时间会边长。本次实验在多次调整后选取了50作为ef的值，较好的平衡了时间效率与搜索准确度。

最后，返回k个特征向量最接近目标向量的图：

```
return p.knn_query(queryVec,k)
```

3.7 新闻聚类

3.7.1 cut.py 中文分词以及筛选

本步骤使用jieba中文分词中的精准模式进行语料分析。

首先我们在网上下载了一些常用停词，并且采取了人工选择的方法在停词库中添加了一些针对体育新闻的专用停词，从而增大了停词对文本分析的作用。停词库保存在stopword.txt中。

在cut.py中，首先读入停词库

```
with open('stopword.txt') as f:
    stopwords = [line.strip().decode('utf-8') for line in f.readlines()]
```

在此之后，遍历data文件夹中的pkl文件

```
for _,__,files in os.walk(path):
    for f in files:
        data=load(f)
```

使用jieba的精准模式进行中文分词，分词结果储存在word列表中。而后进行个性化的筛选：首先，利用正则表达式库re中的match排除掉英文或数字开头的非中文词语，然后又根据停词库中的停词，将对文章语义贡献不大的停词全部删掉，最终形成了文本分析的结果。

```
seglist = jieba.cut(content,cut_all=False)
words=list(seglist)
words = [w for w in words if len(w)>1 and not re.match('^[a-z|A-Z|0-9|.]*$',w
    )]
words = [w for w in words if w not in stopwords]
output = '␣'.join(words)
```

最后将结果存入res.txt中。与该新闻内容对应的url存入url.txt中。两个文本文件中不同新闻之间由一个\n间隔。

```
f=codecs.open('res.txt','w','utf-8')
f.write(res)
f.close()
f=codecs.open('url.txt','w','utf-8')
f.write(urls)
f.close()
```

3.7.2 k_means.py 聚类

本实验使用sklearn库提供的相关功能进行TF-IDF的计算，以及根据TF-IDF进行k-means聚类。

首先读取语料，存入corpus列表。根据上一过程中文分词的特点，在res.txt中每一行为一个文档。

利用sklearn中的CountVectorizer将文本中的词语转换为次品矩阵。其中矩阵元素a[i][j] 表示j词在i类文本下的词频。再通过sklearn中的TfidfTransformer统计每个词语的TF-IDF权重。具体代码如下：

```
vectorizer = CountVectorizer()
transformer = TfidfTransformer()
tfidf = transformer.fit_transform(vectorizer.fit_transform(corpus))
```

然后，通过`word = vectorizer.get_feature_names()`获取词袋模型中的所有词语，并将得到的TF-IDF权重转为array，最终把两者输入到tfidf.txt中保存备用。

使用sklearn.cluster中的KMeans进行聚类。

```
from sklearn.cluster import KMeans
clf = KMeans(n_clusters=500)
s = clf.fit(weight)
print(clf.inertia_)
```

其中n_clusters为类别总个数。该参数的设定应综合考虑新闻总数目以及最终的误差。itertia即可用来评估n_clusters的选取是否合适。

最后，将聚类结果写入cluster.txt。该文件中每行有两个数字，第一个为文档编号，第二个为所属类别的编号。

不幸的是，最终新闻聚类功能未能在网页中实现，详见问题部分4.2。

3.8 SearchFiles-web.py 前端构建

本次实验使用web.py进行前端搭建。模板文件存放在templates文件夹中。先根据web.py的语法写好HTML模板，后启动web.application。之后就可以访问指定端口（默认8080）进行访问。

在SearchFIles-web.py中共定义了两种页面：index与t。前者为索引页面，显示本网页的功能模块；后者为结果显示页面，包括文本搜索结果和图片搜索结果。

虽然结果页面全部在`host:8080/t`中，但处理方法显然不同。实现方法如下：文本搜索通过GET方法访问，而图片搜索则采用POST将被搜索图片传至web.py中，进行进一步处理。

在SearchFiles-web.py中的`class s`中定义了GET和POST两个成员函数，分别处理两种搜索。在GET函数中调用的搜索文本函数，以及在POST函数中调用的图片搜索函数，分别是在SearchFiles.py与query.py的基础上略做修改得到的，原理完全相同。通过处理得到搜索结果之后，即可使用位于templates文件夹中的预先写好的模板制作HTML结果，显示到前端，一次完美的搜索体验就此完成。

更多更详尽的网页功能介绍，请移步配套ppt中进行浏览。

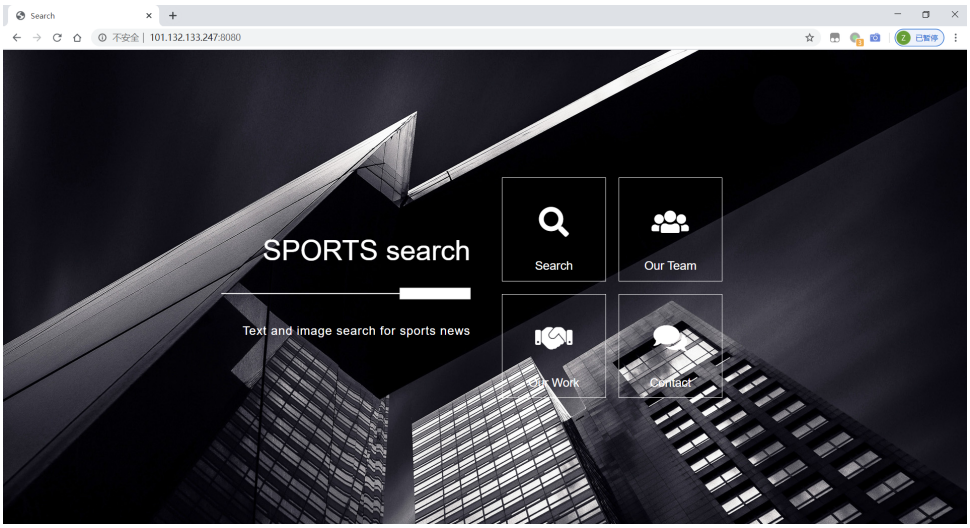

图 7: 索引页

4 实验总结

4.1 概述

本次实验在本组四位成员的合理分工、通力合作下圆满完成。从网络数据的爬取，到索引的建立，再到搜索功能的实现，最终设计并实现网页展示，整个实验有条不紊地进行。通过这次实验的锻炼，本组成员对搜索引擎的认识更加全面而深刻，从后端到前端，从一无所有到应有尽有。

4.2 问题与解决

问题 数据爬取的过程中，有时爬虫会长时间找不到符合要求的新闻网页。经过排查，发现网易体育新闻和新浪体育新闻的一些子域名下的内容几乎没有新闻，但会使爬虫陷入其中，长时间不能爬取到有效信息。

解决 因为本次爬取为定向爬取，形成干扰的子域名有限。本次实验采取手动添加黑名单的方式解决。即：在scrapy的parse函数中，对于每一个URL，在yield一个Request之前，先查询黑名单，确认是否生成请求。

问题 在使用SIFT提取图片特征以实现以图搜图的过程中，发现由于图片过多，运行生成SIFT的程序时会由于内存不足而失败。

解决1 改变参数设定，提高阈值，使每张图片产生的SIFT特征向量减少。但同时，如果参数调整不合适，SIFT特征向量过少，又会使图片匹配出现问题：有些相关图片却没有匹配到，召回率降低。平衡内存使用和搜索结果优劣之间的关系，又成为新的难题。

解决2 放弃了SIFT特征提取算法，使用深度学习的方式，利用目前来看非常优秀的VGG网络模型将每一张图片转换成一个长度为512的特征向量，并以此作为唯一特征值进行以图搜图功能实现。从而，特征向量大大减小，解决了内存使用问题；VGG有非常好的匹配效果，也解决了搜索内容准确性的问题。

问题 在使用LSH（局部敏感哈希）检索图片，等待时间太长（长达1 2分钟），已经远远超出正常用户的忍耐极限；而且逼近效果不佳，图片搜索结果经常与目标图片不太相关。

解决 放弃LSH，寻找其他更加高效的算法。通过对多种算法的比较（如图4），最终选择HNSW算法，最终搜索时间大大缩短，平均只需两秒。同时，逼近效果也较好，搜索结果令人满意。

问题 在搭建前端显示图片搜索结果时，只显示文件名（路径），不显示图片内容。

解决 将所有需要进行显示的图片全部转移到static文件夹中。原因为，web.py中只提供static一个静态文件夹，只有在静态文件夹中的文件才能直接在网页中访问。

问题 在运行k-means进行新闻聚类时，内存再一次爆满，导致程序运行失败。

尝试解决方案 增加停词库中的词语数量，尽量将无用词汇全部筛掉，仅保留对聚类有重大影响的关键词汇。但很不幸，即使删去了大量停词，由于网页数量太多，学生笔记本性能有限，只得放弃该选作功能。因为只是硬件限制问题，本组也将相关代码作为项目的一部分进行上传。

4.3 展望

EE208课程，立足实验学习，加以讲课辅助。通过这一个学期的学习，我们对网络爬虫、文本索引、检索评价、图像处理、分布计算等多方面内容有了实践性的理解。相信在不久的将来，这些知识都会成为推动我们研究的强大动力。

最后，感谢老师们和助教们的悉心指导与热心帮助。预祝老师们，助教们新年快乐！