清华新闻搜索

2015011308　唐适之

概要

本项目爬去清华新闻网下所有新闻，共计四万余条，并实现了以下拓展要求：

- 使用 CSS 美化页面；
- 将搜索结果分页显示；
- 显示包含关键词的正文摘要，并标红关键词；
- 可以按时间进行筛选；
- 使用 AngularJS 前端框架提升效率。

建立清华新闻搜索引擎分为以下若干步骤 :爬取新闻, 建立倒排表, 将数据插入数据库, 建立网页服务。爬取新闻于 spider.py 实现，建立倒排表于 invert.py 实现，插入数据库于 imports.py 实现。

为了使各个步骤相互分离，使每个步骤易于修改和调试，并且避免一个步骤的错误损坏所有数据, 每个步骤结束后都会将中间结果存入文件, 下一个步骤开始时再从文件提取数据。这个过程由 data.py 统一管理，使用 JSON 格式将爬取的新闻存入 data/pages.json，将倒排表存入 data/inverted.json。

爬取新闻 – spider.py

清华新闻网的“新闻合集”页面使用了一个 JSON API 来返回某年某月的新闻列表，并且该页面包含了所有清华新闻网中的新闻，可以很方便地以这个 JSON API 作为爬取新闻的索引。此 JSON API 的格式如下：URL 为

news.tsinghua.edu.cn/publish/thunews/newsCollections/d_<year>_<month>.json

的请求返回<year>年<month>月的所有新闻，其中<year>与<month>是变量。返回值为一个 JSON 对象，其中的“data”键对应的值为列表主体，也是一个 JSON 对象。“data”对象中每一个键对应该月的一日，其值为该日新闻的列表。列表中每个元素包含某新闻的 url、标题、访问量、栏目等。示例（节取）如下：

```
{
    "data": {
        "1" : [
            {
                "title" : "1日某新闻",
                "htmlurl" : "所对应URL"
            }
        ]
        "3" : [
```

```
            {
                "title" : "3日某新闻",
                "htmlurl" : "所对应URL"
            }
        ]
    }
}
```

获得此索引后，即可按顺序爬取每一新闻页面。由于索引中已包含标题和日期信息，爬取新闻页面时只需获得正文。清华新闻网页面中正文被直接包裹在<article></article>标签中，并且整个页面只存在这一处<article>标签。故只需使用正则表达式匹配出<article>标签，即可获得其内部的正文。接着再使用正则表达式将正文中的其余 HTML 标签一律清除，并使用 HTMLParser.unescape 对其中的 HTML 转义字符进行反转义，即可提取出纯文本的正文。

注意清华新闻网中很多页面已被删除，HTTP 返回 404，但标题仍可从“新闻合集”中找到。对于这样的页面，将其正文设为空字符串即可。

执行 spider.py 时，先执行 loadIndex 函数获得 JSON 新闻索引，再执行 loadPages 获取新闻正文，最后通过 data.py 将结果存入 data/pages.json 文件。data/pages.json 有 321MB，不方便提交，可查看 data/pages_sample.json 样例。

建立倒排表 – invert.py

维护一个 python 字典作为倒排表，其中键代表关键词，值为所对应的新闻列表。对于每篇新闻，对其标题与正文使用 jieba 进行分词，再将此新闻的 id 插入倒排表中。

执行 invert.py 时，首先通过 data.py 读取上一步骤获取的所有新闻页面，再通过 buildList 函数建立倒排表，最后再通过 data.py 将倒排表存入 data/inverted.json 文件中。data/inverted.json 有 108MB，不方便提交，可查看 data/inverted_sample.json 样例。

数据管理 – data.py

为了给以上两步骤的中间结果之存取提供一个统一的接口，实现了 data.py。data.py 提供了两个函数 load 和 save：save 将给定的数据以 JSON 格式存入 data 文件夹中的文件里，而 load 函数将 data 文件夹中某文件的内容取出。

插入数据库 – imports.py

data.py 管理的数据不适合直接由网页服务器读取，因为数据量过大，每次读取需要 5 秒左右，而且作为网页服务器的 django 框架不支持实现一个全局的数据管理 daemon，实现起来较为复杂。又由于执行倒排表不同关键字的合并需要消耗较大的运算量，不适合使用 python 这样的脚本语言处理，故须将 data.py 所管理的数据插入网页服务器所使用的 sqlite 数据库。

数据分为两部分：网页内容和倒排表。网页内容如原样插入 Pages 表，数据库每行表示一个新闻页面，其中包括标题、正文、日期、URL 等列。倒排表被拆分为若干键值对插入 Inverted 表，每个键值对表示某关键字在某页面中出现过，键即字符串表示的关键字，值即整数表示的页面 id。

由于将倒排表拆分成了键值对，总共产生了 17000000 以上的数据库行。插入数据库时，如果逐一插入，需要消耗无法容忍的时间。但如果将全部数据批量插入，又需要占用超过 PC 机所能提供的内存。所以 imports.py 以 10000 个条目为一组批量插入，兼顾了时间和内存。

由于 sqlite 数据库被 django 网页框架所管理，需要加载 django 框架才可以访问数据库，故须设定 django 相关的环境变量。因此实现了 imports.sh 作为辅助脚本，先设定相应的环境变量再执行 imports.py，同时 imports.sh 还在插入新数据前清空原有数据库，简化了导入过程中的人工步骤。

网页服务

网页服务采用了静态 HTML+JSON API 的架构，作为一种 UI 和数据分离的实现方式。根路径"/"返回一个静态的 HTML 页面作为客户端，客户端运行了基于 AngularJS 的 JavaScript 脚本。每次请求访问 URL 为/api/search 的 JSON API。客户端实现了翻页和日期筛选的功能，每次请求时通过 GET 方法向 API 提供关键词、页码、日期筛选模式三个参数，服务端从数据库获取信息，并以 JSON 格式返回，浏览器再通过 JavaScript 脚本将数据填入 HTML 以展示给用户。这样每次浏览器与服务端交互的数据量小，效率高，从而实现了搜索结果实时返回而不用用户点击"搜索"按钮的效果。

服务端 JSON API 获得浏览器发出的 GET 请求后，首先从数据库 Inverted 表（倒排表）中查询用户提供的关键词所对应的新闻 id 集合，再从 Pages 表（所有新闻）中查询所有在该 id 集合中的新闻。此操作通过 django 的 filter 函数实现，所有操作均在数据库而不是 python 脚本中进行，提高了运算效率。接着，服务端根据浏览器请求筛选出满足日期要求的新闻，并截取新闻列表中某一段的内容作为搜索结果的某页。最后，服务端找出这些新闻中出现的关键词，将其标为红色，并将无关部分转换为省略号，最后返回给浏览器。