Python 新闻爬虫大作业

姓名：嵇天颖

班级：计 64

学号：2016010308

一、功能实现

（一）基础功能

(1) 网络爬虫

1. 从清华新闻网上爬取了超过 5000 条新闻；

2. 实现了新闻数据的预处理与索引，抽取了新闻的内容，实现标题和正文分词处理，并为新闻建立了索引；

(2) Web 设计

1. 实现了首页新闻展示功能；

2. 新闻展示首页可以显示全部的新闻数量，并使用表格等方式显示新闻概要信息（标题、时间和正文摘要），并设计为分页显示；

3. 点击新闻展示首页中的新闻概要，可以跳转到新闻详情页，详情页中显示新闻全文；

4. 新闻主页都实现了一个查询表单，包括一个文本框和一个查询按钮；

5. 新闻查询也实现了支持时间查询的查询表单，包括一个文本框和四个时间选择项；

6. 点击查询按钮后跳转到查询结果页面；

7. 使用 css/javascript 等前端语言或前端框架对页面布局进行美化；

8. 搜索结果列表页中包含关键字的标题和正文部分予以高亮显示；

（二）加分项

1. **高级搜索**：在保留基础单关键字查询的基础上，支持下列几种高级搜索功能

(Ⅰ)关键字与时间范围的协同查询：支持查询给定时间段中带有关键字的所有新闻；

(Ⅱ)多关键字查询：支持输入空格分割的多个关键字或一句话，查询与输入的内容接近的若干条结果；

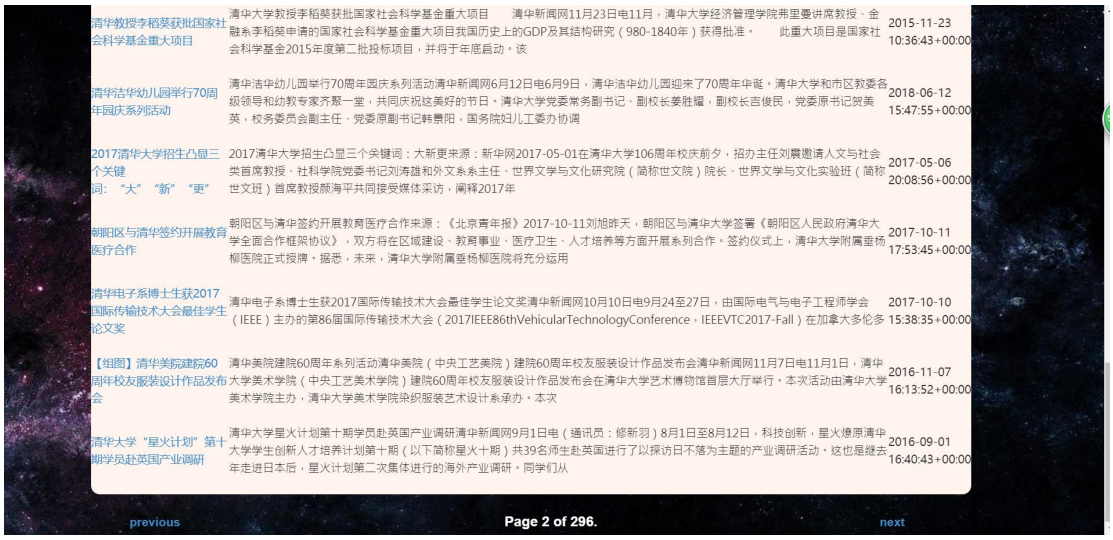
(Ⅲ)每次显示出查询耗时；

2. **新闻推荐**：在每条新闻详情页的最下方，增加"新闻推荐"部分，推荐 1-3 篇其他新闻，点击后可跳转到被推荐新闻的详情页

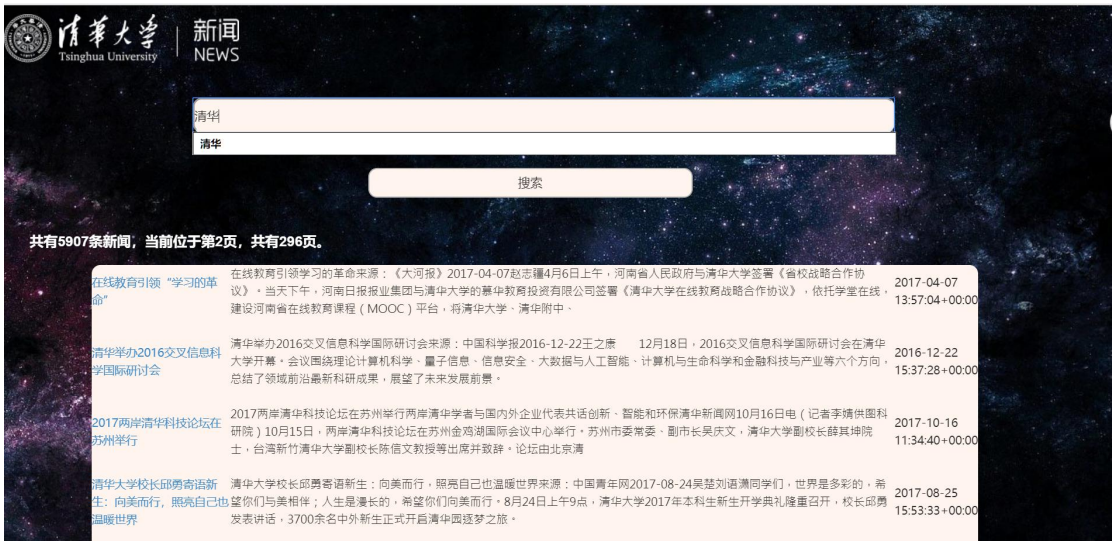
二、界面与功能展示

（一）新闻首页：新闻首页支持关键词查询，并展示了所有新闻，同时具备分页功能

点击'previous'可以跳转到上一页，点击'next'可以跳转到下一页，中间白色字体部分显示出当前所在页面：



首页支持关键词和多关键词查询：点击'搜索'按钮既可以显示在全体新闻中的全部搜索结果：



（二）新闻查询页

（1）新闻查询页支持关键词查询、多关键词查询、时间范围查询，点击‘搜索’按钮既可以显示在要求范围内中的全部搜索结果；

单关键词查询效果——

多关键词检查结合时间范围限定的查询效果——

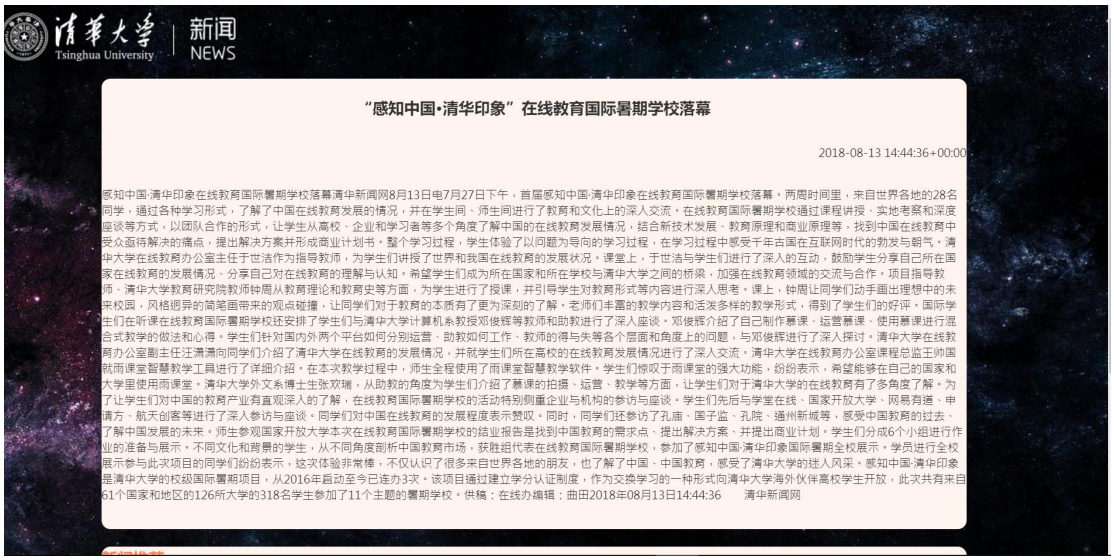
搜索整句效果——

（2）搜索栏目下方会展示出本次查询所消耗时间，以及结果总数目

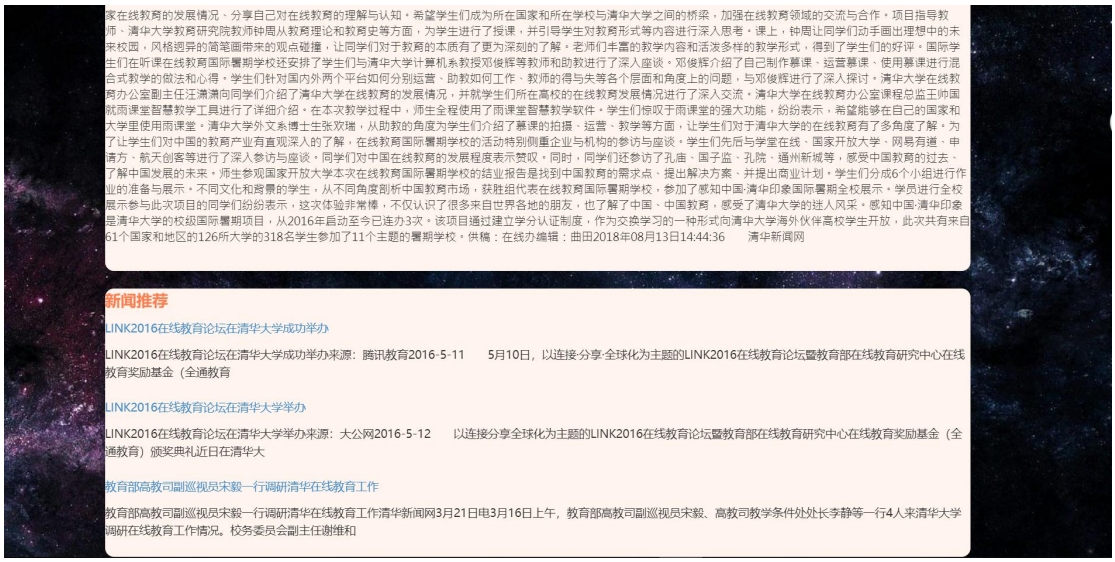
（3）检查结果显示具备分页功能

（三）新闻详情页

（1）新闻详情页展示出新闻的标题、正文和时间



（2）详情页有新闻推荐栏目，根据文本相似度判别，点击即可以跳转相应的详情页



三、查询算法与推荐算法

（一）查询算法

对搜索关键词和语句进行断开处理，存入 wordlist 后，设置时间限制的 filter，与数据库进行比对查询：

```
    for each_word in wordlist:
        curWord = Word.objects.get(pk = each_word)
        if articlelist:
            articlelist = articlelist&set(curWord.article.filter(article_time__gte=datetime.date.today()-datetime.timede
        else:
            articlelist = set(curWord.article.filter(article_time__gte=datetime.date.today()-datetime.timedelta(deltada
    articlelist = list(articlelist)
    time_start=time.time();
    print(time.strftime('%Y-%m-%d %H:%M:%S'))
    articlelist = sorted(articlelist, key=lambda x:x.article_time, reverse=True)
    dicts = [{'id':each.article_id,'url':each.article_url, 'title':redtitle(each.article_title, wordlist), 'text':redte
```

（二）推荐算法

利用包 gensim：对所有文章进行分词操作，去掉无用词后，生成词典，根据先前建立好的文章索引，获得相关文章的索引，存入矩阵之间，便于后面的查询

```
#对所有文章进行分词操作

texts = [jieba.lcut_for_search(document) for document in documents]

print("分词操作完成")

#去掉无用词
stoplist = list(', . ( ) 的 地 和 你 它 我 他 是 。 ， + - * / : ; [ ] = ! ~ ? < >'.split(" "))
texts = [word for word in texts if word not in stoplist]

print("无用词清除结束")

dictionary = gensim.corpora.Dictionary(texts)

print("词典生成完毕")

dictionary.save(fileaddr+"worddict.dict")

print("词典保存完毕 位置:%sworddict.dict" % fileaddr)

corpus = [dictionary.doc2bow(text) for text in texts]
corpora.MmCorpus.serialize(fileaddr+'mmcorpus.mm',corpus)
print("corpus保存完毕 位置:%smmcorpus.mm" %fileaddr)
```

四、目录结构

以下为项目目录的文件结构（只列出必要的文件与目录）

- project：项目文件夹
  - ArticleFile1.pickle：以页面 ID 为索引的分词数据（1）
  - ArticleFile1.txt：ArticleFile1.pickle 的文本形式
  - ArticleFile2.pickle：以页面 ID 为索引的分词数据（2）

– ArticleFile2.txt：ArticleFile1.pickle 的文本形式

– IDFile.pickle：以页面 ID 为索引的标题、正文、时间、路径

– IDFile.txt：IDFile.pickle 的文本形式

– crawler.py：爬取新闻网页并对其分析，将分析结果保存至 IDFile

– wordJieba.py：从 IDFile 读取分析结果，对正文与标题分词，分词结果保存至 WordFile

– articleJieba.py：从 IDFile 读取分析结果，对正文与标题分词，分词结果保存至 ArticleFie

- similarity.py: 获取文章文本相似度数据

– WordFile.pickle：以词组为索引的分词数据

– WordFile.txt：WordFile.pickle 的文本形式

- similarity4.txt: similarity4.pickle 的文本形式

-similarity4.pickle: 文章的相似度数据

– document：文档目录

* document：设计文档

– mysite：检索系统目录

* db.sqlite3：存储页面信息与分词数据的数据库

* inputarticle.py：将页面数据导入数据库

* inputconnect.py：将分词结果导入数据库

* inputword.py：将词组导入数据库

*inputsim.py: 将文本相似度导入数据库

* manage.py：Django 生成

* app：检索系统目录

· admin.py：Django 生成

· models.py：定义了两个 Model：Article、Word

· urls.py：Django 生成

· views.py：搜素功能函数以及其他函数

· static：存储 CSS、js 文件和图片

· templates：存储网页模板

* mysite

· settings.py：Django 生成

· urls.py：Django 生成

五、开发版本

Python :3.6.5　　Django:2.1.1

```
C:\Users\DELL\Desktop\News\mysite>python --version
Python 3.6.5
C:\Users\DELL\Desktop\News\mysite>python -m django --version
2.1.1
```