TUSK 校园搜索

搜索引擎技术基础 实验报告

章彦恺 计 22 班

宋佳铭 计 22 班

一、 摘要

我们的项目 TUSK，是 Tsinghua University Search Kit 的简称，它实现了以下的内容：

1. 基于**清华新闻和人人网数据**的搜索
2. 基于网络学堂数据的**文档搜索**
3. 基于计算机系教职工信息的**人物搜索**
4. 其他形式的搜索，例如**清华大学校历**
5. 准确，实时，高效的**语音搜索**
6. 准确，可扩展的搜索词**自动补全**
7. 不同**搜索内容的整合**
8. 用**机器学习方法**确定搜索参数
9. **简洁美观**的界面和搜索结果的显示

下面，我们会对这些功能的实现进行详细的说明。

二、 实验目的

1. 以清华新闻网为基础，抓取作为搜索源的网络数据

2. 开发一个用户友好界面整洁优美的网页前端
3. 设计合理的索引和搜索方式，设计后端模式可以针对不同类型的请求可以获得到相应类型的搜索结果
4. 用合适的方法设计搜索参数，优化搜索结果
5. 为搜索引擎添加文档、图片、人物等信息咨询搜索支持
6. 尝试添加其他用户交互模式，如语音搜索

三、 实验架构

实验由分离的前后端实现。前端由 html 静态网页组成，使用 Ajax 技术和后端服务器连接，接收来自后端的 Json 数据。

开发环境如下表所示：

项目	环境
后端	Jsp+Servlet
运行环境	Apaceh Tomcat 7.0.55
数据库	MySQL 5.6.20 Community(GPL)
开发环境	Eclipse Luna 4.4.0
操作系统	Microsof Windows 8 Enterprise Eng

后端提供的 API 及相应参数如下表所示：

API	参数	类型	用途
tusk/PageSearch	search	String	搜索的关键字

tusk/DocSearch	page	Int(optional)	搜索的页号
	search	String	搜索的关键字
tusk/AutoComp	page	Int(optional)	搜索的页号
	autocomp	String	需要进行自动补全的查询内容

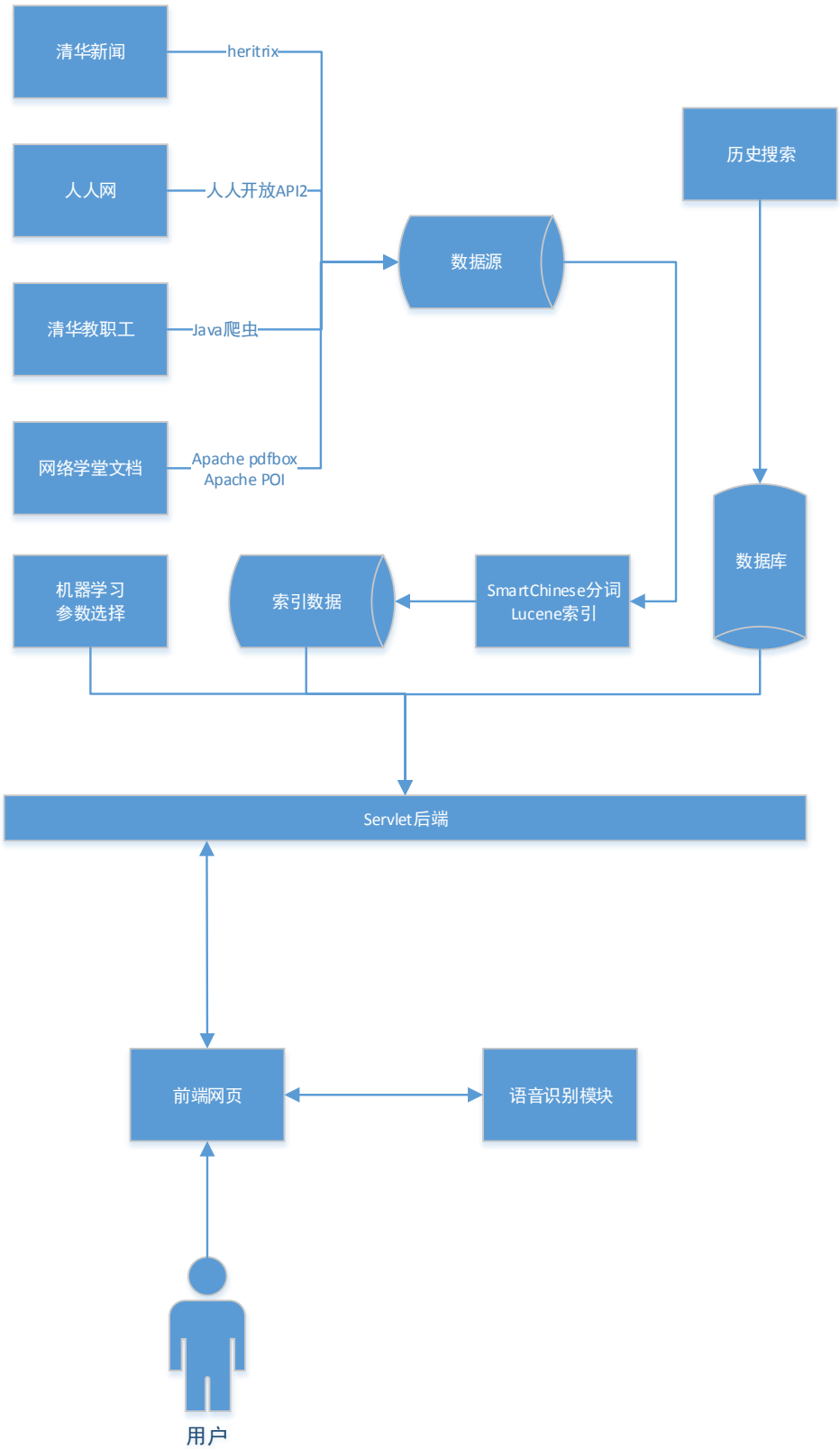


图 Tusk 项目整体架构流程图

提交的工程文件目录结构如下表所示：

目录	描述
db	用于创建数据库的脚本
index	索引目录
index/learn	对网络学堂文档的索引
index/news	对清华新闻和人人信息的索引
lib	使用到的第三方 jar 库
ml	机器学习工作目录
renrenCrawler	人人数据的工作目录
src	后端实现代码
tusk-frontend	前端实现代码

使用到的第三方库有：

- api-client-sdk-2.0.jar
- commons-codec-1.9.jar
- commons-lang3-3.4-javadoc.jar
- commons-lang3-3.4.jar
- commons-logging-1.1.3.jar
- jasper.jar
- jsoup-1.8.2-javadoc.jar
- jsoup-1.8.2.jar
- junit-4.10.jar
- juniversalchardet-1.0.3.jar
- log4j-1.2.17.jar
- lucene-analyzers-common-5.2.0.jar

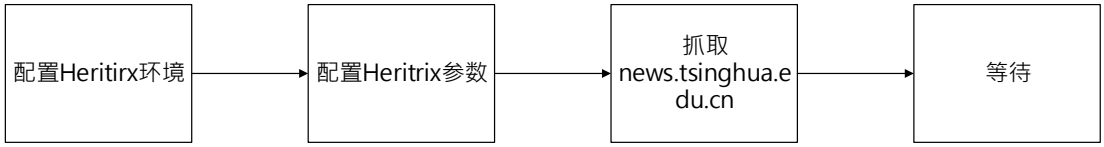
- lucene-analyzers-smartcn-5.2.0.jar
- lucene-core-5.2.0.jar
- lucene-queryparser-5.2.0.jar
- org.json.jar
- pdfbox-app-1.8.9.jar
- poi-3.12-20150511.jar
- poi-examples-3.12-20150511.jar
- poi-excelant-3.12-20150511.jar
- poi-ooxml-3.12-20150511.jar
- poi-ooxml-schemas-3.12-20150511.jar
- poi-scratchpad-3.12-20150511.jar
- servlet-api.jar
- xmlbeans-2.6.0.jar

四、 数据准备

1. 清华新闻

我们按照网上的方式配置好 Heritrix，用助教给的配置，对 news.tsinghua.edu.cn 执行抓取用的爬虫，经历约 2 天时间的爬取，抓取了超过 2 个 G 的文件。

开始时，我们发现不能执行抓取操作，通过查询网络得到的解决方法是先把正则表达式缩短，然后在开启爬虫后将正则表达式变成正确的，就可以解决问题了。如果手速够快，基本上不会多爬几个文件，手动删掉就可以了。



在使用抓取到的清华新闻进行索引的过程中，我们并没有遇到编码上的问题。这是由于我们使用了 juniversalchardet 工具包。这个工具包能够识别文本的编码。我们通过识别了编码，并将不同的网页编码统一转换成 UTF-8 编码，避免了对 html 页面进行解析得到编码的麻烦，也避免了不同编码造成的搜索结果乱码的问题。

2. 网络学堂文档

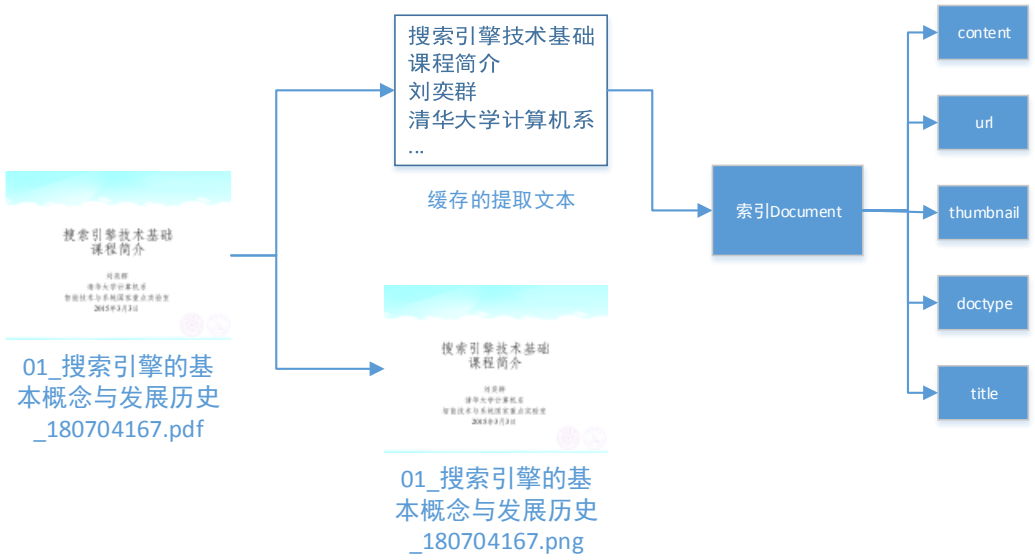
实验中使用到的网络学堂中的文档来自三年的学习过程中涉及和使用到的课件和文档，来自本地文件系统，因此无需使用网络爬虫进行抓取操作。

后端代码中 tusk.utils.docprec 包是用来处理网络学堂中文档的代码包。

其中 DocPrec.java 用于对于原始的文档进行文本内容的提取和缩略图提取，使用到 apache.poi 和 apache.pdfbox 库对文档进行解析。对于文档类型的支持程度如下表所示：

类型	文本提取	缩略图提取
Word(.doc)	✓	–
Word(.docx)	✓	–
PowerPoint(.ppt)	✓	✓
PowerPoint(.pptx)	✓	✓
PDF(.pdf)	✓	✓

在提取了文档的文本信息和缩略图信息后，通过 LearnIndexer.java 类对文档内容进行索引工作。数据的具体处理流程见下图所示：



3. 人人数据

人人在近期更新了 API 的访问方法。现在抓取人人网内的数据需要使用 OAuth 2.0 方法登陆获取 token 后才能够进行操作，且人人限制了测试应用在一段时间内访问 API 的次数。

为此我在人人开发平台中注册应用 SE 和 SE2，获得了两个应用的 APP ID、API KEY 和 Secret Key，并利用这些信息，利用人人网提供的 API 和 PHP SDK 进行数据的抓取。

图 应用开发平台

代码在 renrenCrawler 文件夹下。其中人人提供的 PHP SDK 不能直接使用，需要通过阅读 PHP SDK 代码，并修改出现 Bug 的部分才能够使得 PHP SDK 能够正常地运行和抓取数据。在通过 OAuth 2.0 协议登陆后，使用 blog/list API 抓取数据。

在抓取的过程中，遇到了两个主要的困难：

- 单次数据抓取过程限制了 30 秒运行时间
- 一定时间内访问 API 有次数限制

解决的方法是：

- 将一段连续的抓取过程拆分成独立的日志抓取过程，使用脚本修改参数不断调用抓取 php 程序，即可突破 30 秒运行时间限制
- 通过注册两个应用，并修改 config.php 的方法切换两个 App。用这种方法，在其中一个 APP 达到了 API 访问次数限制后，切换到另一个 App 的方法，解决了人人对 API 访问次数的限制

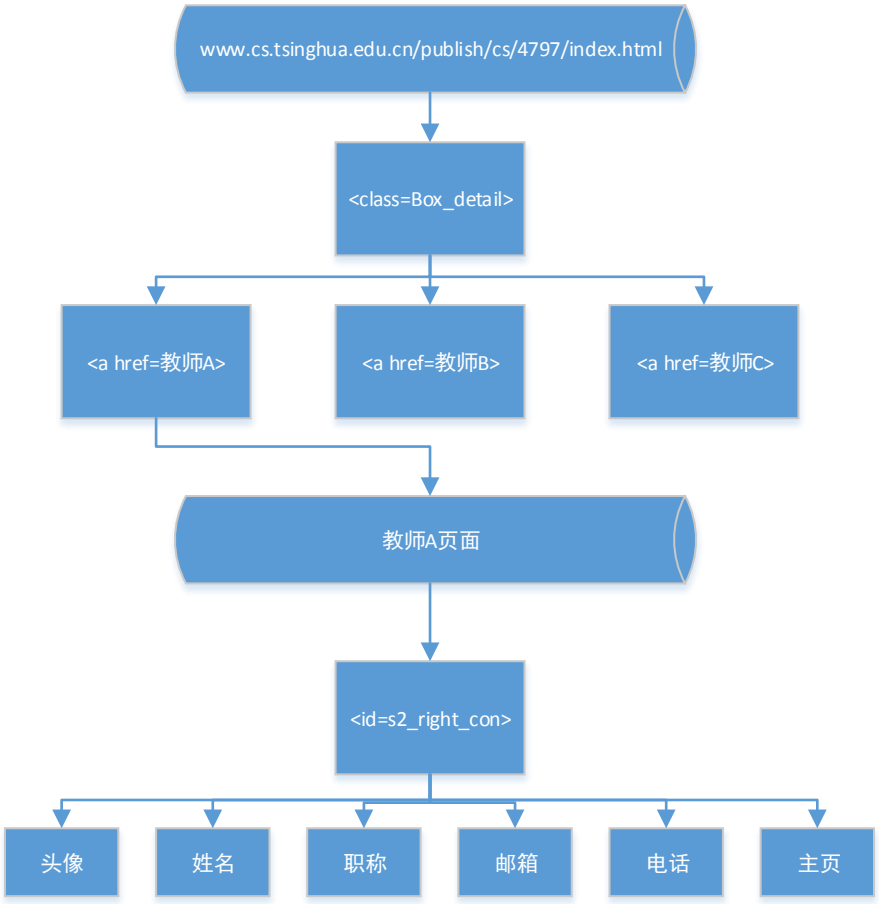
通过使用人人开放 API 2.0，我们成功获取了来自学生清华、清华社团、清华大学学生会、新清华学堂、清华大学清新时报、清华电视台、清华大学学生科协的一共 680 条日志。

4. 教职工数据

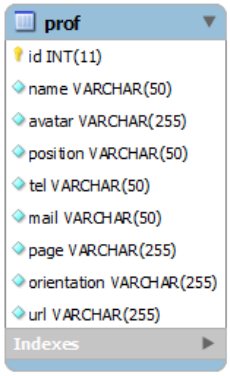
教职工的信息可以从 www.cs.tsinghua.edu.cn/publish/cs/4797/index.html 网站上抓取到。可以在后端代码的 tusk.utils.prof.ProfCrawler 类中看到用 java 实现的网页爬虫。爬虫使用到了 Jsoup 库用于解析 html 文本，使用 MySQL 提供的 JAVA Connector 连接 MySQL 数据库用来存储数据。

更具体的，各个教师的页面可以通过对于 index.html 中 class=box_detail 的容器，找寻其中的 a 链接获得。对于各个教师的页面，提取其中 id=s2_right_con 的容器，即可获得关于教授的头像、姓名、职称、邮箱、电话和主页信息。

对于本例中使用的抓取方法可以通过下面的图得到更直观的理解：



教职工数据在抓取后将存入到数据库中。教职工数据库的 ER 图如图所示：



五、 后端实现

1. 后端架构

后端的实现源码工程目录 src 文件夹下。每个文件及其描述如下表所示：

文件	描述	是否 SERVLET
autocomp.java	处理自动补全请求的 Servlet	✔
docsearch.java	处理搜索文档请求的 Servlet	✔
pagesearch.java	处理页面搜索请求的 Servlet	✔
tusk/learn/datapreparation.java	生成用于机器学习的训练数据	
tusk/renren/renrenindexer.java	对于抓取的人人网数据进行过滤和生成索引	
tusk/utils/tuskutil.java	公用的功能性类，内含共有功能方法函数	
tusk/utils/docprec/docprec.java	用于处理文档，提取文档中的文本信息和缩略图	
tusk/utils/docprec/learnindexer.java	将提取的文档文本和缩略图信息制成索引	
tusk/utils/docprec/pageindexer.java	将抓取的清华新闻的数据制成索引	

tusk/utils/prof/	抓取网页中的教职工信息，并添
profcrawler.java	加到数据库中

2. 自动补全

在搜索的过程中，用户会希望搜索引擎能够预测用户之后的输入，并给出可行的搜索的建议，因此自动补全功能是提升搜索引擎用户友好程度的一个十分重要的功能。

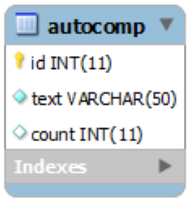
本例中实现搜索内容自动补全功能的原理是基于用户的搜索进行语料库的建立和维护。每当用户进行一次搜索，程序会将用户的搜索结果记录下来。下一次用户键入了之前出现过的搜索的一个前缀时，程序能够将所有可用的包含该前缀的自动补全结果按照被搜索次数排序，取出排序结果靠前的 10 个可用的自动补全结果，并返回给前端。

容易想到数据库能够高效地完成自动补全的功能，因此此处程序引入的 MySQL 作为数据库，通过 SQL 查询来高效地获得自动补全结果。实现代码如下：

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException
{
        try
        {
                response.setContentType("text/html;charset=utf-8");
                String s = request.getParameter("autocomp").replace('\'', ' ');
                ResultSet res =
                        stat.executeQuery(
                                "SELECT * FROM autocomp WHERE text LIKE '" +
                                s + "%' ORDER BY count DESC");
                JSONArray lst = new JSONArray();
                for(int i = 0; i < 10 && res.next(); ++i)
                        lst.put(res.getString(2));
                PrintWriter out = response.getWriter();
                out.print(TuskUtil.wrapJson(
```

```
                        lst.toString(), request.getParameter("callback")));
                out.close();
            }catch(Exception e)
            {
                e.printStackTrace();
            }
    }
```

自动补全使用到的数据库 ER 图如下：



由于这个功能在开发早期就已经实现，因此后期自动补全功能基于开发和测试时使用过的语料，已经有了较好的效果。

3. 搜索方法

本实验中使用的搜索方法是基于 Lucene 的 MultiFieldQuery，通过机器学习学习得到参数，作用于内容和标题，使得在搜索过程中，内容和标题的 Score 权重不同。即令$\mathrm{Score}_{\mathrm{content,i}}$为第 i 个项目的内容搜索得分，$\mathrm{Score}_{\mathrm{title,i}}$为第 i 个项目的标题搜索得分，那么第 i 个项目的最终得分为

$$\mathrm{Score_i} = \sum_{j\in\{content,title\}} \omega_j \times Score_{j,i}$$

机器学习的目标即为找到最适合的ω_j，使得得到的 Score 结果对于用户来说更为合理。

4. 参数学习

我们使用 Logistic Regression 做我们的参数学习方法。首先，我们自己手动标注了一些分值为 1-5 的数据（共 50 个，每个标记各 10 个），然后我们提取这些搜索结果的特征，包括<h>, <title>, <content>等共 7 个参数，对于这 7 个参数，我们使用逻辑斯蒂回归，得到最适合的参数值，应用到我们的模型中。经过试验，我们发现学习得到的参数值并不比原来有明显的提高，这可能是因为我们训练数据太少的缘故。

5. 语音识别

语音识别对于目前的搜索引擎而言有着重要的作用，它可以极大的方便用户使用搜索引擎，谷歌的 Google Now 和微软的 Cortana 都属于语音识别在搜索引擎中的重要应用。由于语音识别涉及到的技术非常复杂，我们考虑使用已经成型的工具，例如 Google 的语音识别 API。通过我们的实验，我们发现 Google 的 API 可以考虑到语境、人物、以及复杂的语义信息，和关键词过滤等等。

我们通过 js 实现 Chrome 上语音的录制，并将录制的语音 HTTPS POST 传输给 Google 的 API，接口地址如下:

https://www.google.com/speech-api/v1/recognize?xjerr=1&client=chromium&lang=zh-CN&maxresults=1

这样我们就可以得到返回的结果，以一个 json 形式给出：

“status”：结果代码

“id”：识别编号

“hypothesis”：结果和置信度的列表

返回的编码是 UTF-8，这样我们就可以在 javascript 中用 obj.hypothesis[0].utterance 得到第一个结果。之后，我们直接将这个文本覆盖在搜索框中，然后进行搜索即可。

我们发现 Google API 效果异常的好，它不仅可以识别一些简单的话例如“清华大学校历“，还可以识别一些知名人士的名字，例如”刘奕群“，”白晓颖“，”胡事民“等。

六、 前端设计

我们的前端和后端是完全分离的。在项目实现开始时，我们商量了一些 API，包括搜索页面，搜索文档，和自动补全等等。我们通过 AJAX 访问服务器端的信息，实现信息的传输，然后使用 javascript 处理收到的数据，进行渲染。我们的前端假设服务器端的位置是 http://grayluck.vicp.cc/tusk。如果要建立在本站上，或者使用不同域名的后端，只需要将这个域名修改即可。

1. 自动提示

我们的自动补全前端是自己实现的。我们每当搜索框中有文字变化的时候，会自动发送给 http://grayluck.vicp.cc/tusk/AutoComp 获得其返回的列表，并进行渲染。通过给列表增加 hide，我们的自动提示列表可以满足在搜索时、或

者搜索框为空时隐藏起来，达到交互逻辑上的正确性。搜索清华时会有如下结果：

2. 查询

我们将发送字符串使用 AJAX 的 HTTP GET 发送给 grayluck.vicp.cc/tusk/PageSearch 和 grayluck.vicp.cc/tusk/PageSearch ，如果是查询页面就用前者，如果是查询文档就用后者。返回结果后，我们就使用 javascript 对页面进行渲染。

我们实现的前端渲染有以下几种：

- 页面：包括标题，url，概述，来源

- 文档：包括缩略图，url，标题，概述，来源

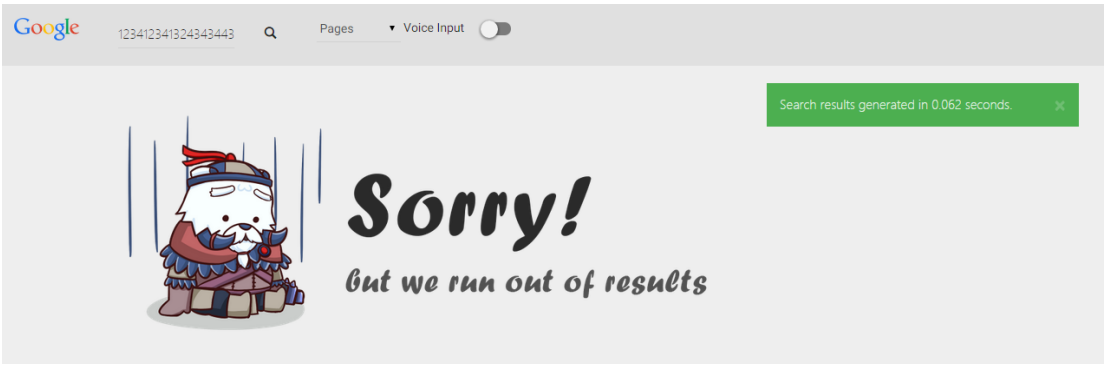
- 图片：包括标题，图片

- 人物：包括姓名，研究方向，职称，联系方式等

- 人人页面：包括标题，概述和全部正文

- 翻页支持如下图所示：

- 如果查询不到结果，我们会显示以下内容：



七、 总结与感想

通过这个项目，我们从抓取数据开始，到分析数据，建立搜索引擎网页前端后端，最后到整合成一个完整的校园搜索站点。从建立了 Github 私有源，在本地 clone 了 repository，到最后一次提交和上传，我们眼见了一个完整的工程从有到无，从单一简陋的功能逐渐完善，我们感触颇深。

工程的开发过程中遇到了很多的困难。在进行语音识别功能的开发过程中，我们发现现在在浏览器中访问麦克风并录下声音并不是一个非常普遍的应用。因此我们在查找资料的过程中遇到了很多麻烦。发现了 Chrome 浏览器对 WebRTC 能够进行较好的支持之后，尝试使用 RecordRTC 去录制音频信息，并将音频信息存储成 blob 上传到服务器转码成 Base64 交给百度 API 来处理。但是后来发现，这个过程过于繁琐，在研究 RecordRTC 的过程中也浪费了不少时间。最后我们找到了 Google API 来进行实现，省去了造轮子的过程，效果也非常理想。从这次教训中，我们学习到了在着手进行开发之前，要实现仔细进

行调查，不要在没有完备地调查的情况下贸然就开始开发，这样不仅会浪费开发时间，还会置相应的功能于“做不出来而被阉割”的境地。

此外，由于工程的架构决定了前后端的完全分离，导致了前后端处于不同的域下：前端是部署在本地的静态服务器，而后端部署在 Tomcat 服务器上，使用 Oral DNS 挂载在 grayluck.vicp.cc 上。这时使用 AJAX 直接向后端发起请求会被浏览器拒绝，原因是为了安全考虑拒绝跨域的 AJAX 请求。这个问题困扰了我们很长时间，直到我们找到了用回调方式，将返回的 Json 格式数据封装成函数返回来绕开跨域请求的方法，才成功解决了这个问题。但是这个方法并不是一个十全十美的方法，因为它的实现并不简洁，需要对前后端均进行修改。也许在之后的学习生活过程中我们会发现一个更好的解决方法。

在经历了这全部的开发过程和困难的解决过程后，我们感受到自己的身心得到了历练，代码能力也得到了显著的提高。像这样的系统架构能力，代码开发能力，问题解决能力和组员间沟通协调能力在我们今后的学习生活过程中十分重要也必不可少。为此我们觉得有必要要感谢老师和助教为我们提供的这样锻炼自己的机会。