网络爬虫和信息抽取——实验报告

软硕161 徐毅
2016213585

一、实验目的

本次实验主要目的是将网络上公开的文本数据集抓取到本地，并且抽取出文本中人物之间的对应关系。

实验中实现了一个简单的网路爬虫工具，从中国京剧戏考网(http://www.xikao.com)抓取了若干京剧名家的人物介绍，并通过第三方分词工具解析出人物介绍中出现的人名。在此基础之上，制定了若干词法分析规则，提取出这些人物的关系。

二、实验环境

操作系统：Windows 10 Education
CPU：Intel(R) Core(TM) i7-4790K
内存：8GB
开发IDE：Eclipse Java EE Neon Release(4.6.0)
Java版本：1.8

项目源代码地址：https://github.com/MyXOF/info-extraction，具体配置运行方法参见README.md

三、项目结构

项目按照包名组织如下：

- com.corp.myxof.config：存放程序运行时必要的常量和配置项
- com.corp.myxof.experiment：
    - Experiment1：爬取网页
    - Experiment2：找出正文中的人名
    - Experiment3：找出人物之间的关系
    - Experiment4：找出一个文件夹下所有文本文件中人物之间的关系(测试用以评价正确性，测试方法查看README.md)
- com.corp.myxof.relation：提取正文中的人名及人物之间的关系
- com.corp.myxof.utils：工具类
- com.corp.myxof.visualization：数据可视化(师徒关系)
- com.corp.myxof.web：网络爬虫工具
- com.corp.myxof.xml：读取/生成XML文件

四、实现思路

4.1网络爬虫

4.1.1 基本思路

由于本次实验只需要爬取指定网站上的人物介绍，并且爬取的网页数量大约在1000条左右，所以没有直接采用现有的开源爬虫工具，而是自己编程实现了一个简单的网络爬虫工具。

主要采用了http get请求抓取一个链接对应的网页，之后使用Jsoup工具(http://www.open-open.com/jsoup/)提取出html文本中指向人物的<a>标签，丢弃掉已经访问过的人物介绍，将没有访问过的链接放入到一个队列中。每次从队列中取一个链接重复上述操作，直到抓取到1000条介绍或者队列中没有链接为止。

实验中需要注意的一个小细节是，爬取下来的网页不是按“UTF-8”格式进行编码的，中文部分不能正常显示，所以需要先做一步编码转化，再用JSoup解析。

4.1.2 实验结果

实验结果保存在一个XML文件中，默认在项目根目录/data/web-content.xml，格式如下(正文介绍部分省略)：

<performer name="陈嘉梁" url=“http://history.xikao.com/person/陈嘉梁”>人物：陈嘉梁.....</performer>
<performer name="陈富瑞" url=“http://history.xikao.com/person/陈富瑞”>人物：陈富瑞.....</performer>
<performer name="陈盛泰" url="http://history.xikao.com/person/陈盛泰">人物：陈盛泰（陈文瑞） ……</performer>

4.2 命名实体识别

4.2.1 基本思路

这部分要实现从正文中提取出现的人名，采用了第三方分词工具Stanford NLP(http://stanfordnlp.github.io/CoreNLP/，以下简称NLP)，这个工具的优点是能够进行分词并且识别这个词的属性，如“梅兰芳”对应的属性为“PERSON”。缺点是分词的准确率不高，尤其是人名，经常把人名切分成两个词或者三个词，而且在初始化的时候需要一定的时间(10s左右)，不方便调试。

针对分词准确率不高的原因，找了一些测试样例分析了原因，主要是如果人名前面带一些介词或者动词，会让NLP认为人名仅仅是一个名词，而不是更具体的“PERSON”，当去掉前面的修饰词之后，再次使用NLP分析，就可以得到“PERSON”属性。

抽取人名的基本思路是：

第一次迭代：使用NLP工具逐句进行分词解析，当看到属性为“PERSON”的时候，即认为这是一个合法的人名，将他保存在结果集中。当看到属性为“NN”和“NR”的时候，这时表示这是一个名词，有可能是一个人名，也有可能和后面一个词连起来成为一个人名，这个时候将当前词语和后面的词语一起放到临时结果集合中。

第二次迭代：将第一次迭代得到的临时结果集再次使用NLP分析，当看到“PERSON”属性的时候，将词语加入结果结合中，否则直接丢弃。

实验的准确性主要依赖NLP工具对“PERSON”识别的正确性，采用上述方法之后，基本能够将文本中的人名准确的提取出来。但是仍然存在两个问题：

1）准确性，有些不是人名的词比如“战蒲关”、“望江亭”等，NLP不能正确处理，依旧会把他当做“PERSON”。

2）相关性，有些虽然是人名，但是和主人公不相关，比如“曹操”、“岳飞”等，这些都是主人公扮演过的角色，对本次实验要抽取的人物关系造成了干扰。

针对上述问题，人工设定一些规则进行过滤，但是仍然不能保证完全过滤到所有噪声，只能说中文博大精深，使用规则进行限制有一个极限，总有规则覆盖不到的地方。

4.2.2 实验结果

实验结果保存在一个XML文件中，默认在项目根目录/data/people.xml，格式如下(内容部分省略)：

```
<performer name="王艳"><name>田玉珠</name><name>张婵玉</name></performer>
<performer name="高盛虹"><name>毛盛荣</name></performer>
<performer name="张永远"></performer>
```

4.3 实体关系抽取

4.3.1 问题简化

本部分作为这次实验的重点，难点，直接抽取人物之间的关系存在困难，因此增加了以下前提条件，简化问题：

1. 对于每一条文本，可以看做是<主人公，介绍>这样一个二元组，其中“介绍”部分完全是关于主人公的，不会存在中间插了一段话介绍另外一个人的情况
2. 基于条件一，可以认为“介绍”中出现的人名，全部和主人公相关，不会和其他人相关

最后的结果应该是“实体 关系 实体”的形式，上述前提条件将问题简化为“实体 关系 主人公”，将变量控制在两个，因为主人公的名字在一开始就能够知道。

4.3.2 核心思路

具体实现的时候采用了 规则匹配+有限状态机模型，程序一共拥有六个状态，分别是{init, master, apprentice, family, partner, reference}，分别对应 {初始态，师傅态，徒弟态，亲戚态，合作态，引用态}。对于每一个词，它属于八种类型中的一个，分别是{人名，师傅关系，徒弟关系，亲戚关系，合作关系，句子结束，引用关系，无用词}。

对于输入的一个词，首先识别出它的类型，再根据程序对应的状态进行处理，程序的状态也随之发生变化。举个例子：

1. 在初始态下接收到类型为师傅关系的词语，这时程序进入到师傅态
2. 接收到类型为人名的词语，这时认为该人名是主人公的师傅，将该人名加入到“师傅集合”中，仍然保持师傅态
3. 接收到类型为徒弟关系的词语，从师傅态转为徒弟态
4. 接收到类型为人名的词语，这时认为该人名是主人公的徒弟，将该人名加入到“徒弟集合”中，仍然保持徒弟态
5. 接收到类型为无用词的词语，保持当前状态，不处理
6. 接收到类型为引用关系的词语，进入引用态
7. 接收到类型为人名的词语，因为在引用态中，认为这个人和主人公不具备我们所期望的关系，继续保持引用态，不处理这个人名
8. 接收到类型为句子结束的词语，状态转为初始态

程序将按照上述基本规则不断处理下去，直到所有文本处理完。简单的示意图如下：

4.3.3 特殊情况处理

上述规则不能正确处理像“梅兰芳是他的师傅”这样的语句，因为人名出现在人物关系之前，所以需要对上述规则进行修改，当“初始态”看到一个人名的时候，会先暂存这个人名在缓冲区，当看到“师傅关系”的词语后，在从“初始态”转移到“师傅态”的过程中，会将暂存的人名认为是主人公的师傅，将他们加入到“师傅集合”中，并清空缓冲区。

4.3.4 词语类型识别

要判断一个词语属于{人名，师傅关系，徒弟关系，亲戚关系，合作关系，句子结束，引用关系，无用词}集合中的哪一种，这是关系抽取的核心问题，这里采用了字符串模式匹配的方法，人工指定一些关键词集合，比如徒弟关系的关键词可以是：{ 弟子, 收, 徒, 招, 学生, 后人, 门生, 传人, 拜师, 共教}，也就是说当输入的词在这个集合中出现的时候，就认为这个词的类型是“徒弟关系”。

判断一个词的类型将按以下优先级进行：

1. 句子结束

2. 引用关系
3. 师傅关系
4. 徒弟关系
5. 合作关系
6. 亲戚关系
7. 人名
8. 无用词

特别地，在判断一个词是人名的时候，可以用4.2节中的实验结果，将得到的人名作为一个“词典”，每一次判断的时候可以用这个词典进行参考。

4.3.5 实验结果

结果保存在一个XML文件中，默认在项目根目录/data/relationship.xml，格式如下(内容部分省略)：

- <master>标签表示这个人是主人公的师傅
- <apprentice>标签表示这个人是主人公的徒弟
- <family>标签表示这个人是主人公的亲戚
- <partner>标签表示这个人和主人公有过合作

```
<performer name="梅兰芳">
  <master>吴菱仙</master>
  <master>叶春善</master>
  <apprentice>李湘君</apprentice>
  <apprentice>韩淑华</apprentice>
  <apprentice>李玉茹</apprentice>
  <family>梅葆玖</family>
  <family>朱小霞</family>
</performer>
```

```
<performer name="尚小云">
  <master>陈德霖</master>
  <apprentice>张蝶芬</apprentice>
  <apprentice>杨荣环</apprentice>
</performer>
```

```
<performer name="荀慧生">
  <master>齐白石</master>
  <apprentice>李薇华</apprentice>
  <apprentice>陆正梅</apprentice>
  <partner>冯子和</partner>
  <partner>舒舍予</partner>
</performer>
```

五、结果展示

针对4.3节得到的师徒关系，采用了ECharts(http://echarts.baidu.com)提供的可视化工具，将结果以关系图的形式展示出来，本次实验中师徒关系一共有四千多对，在做可视化的时候选取了最主要的一千多对，这里选取了徒弟最多的三个京剧名家的截图，更多师徒关系可以参见根目录下/webpage/index.html页面。

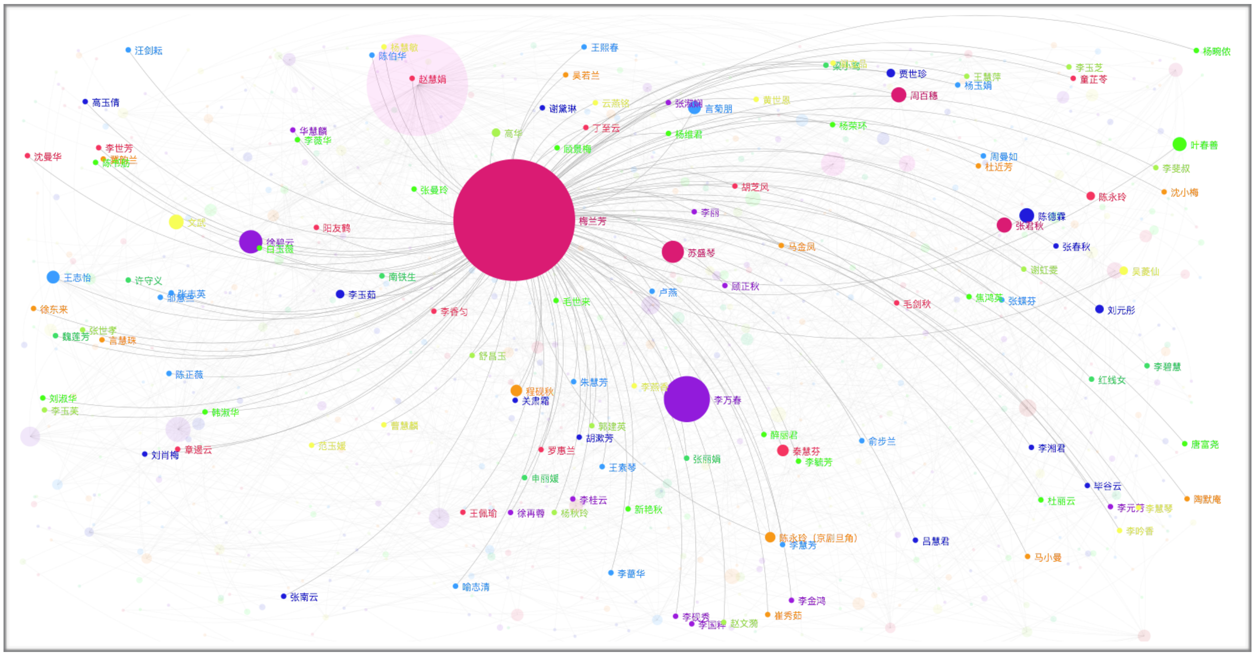

图1：梅兰芳—师徒关系

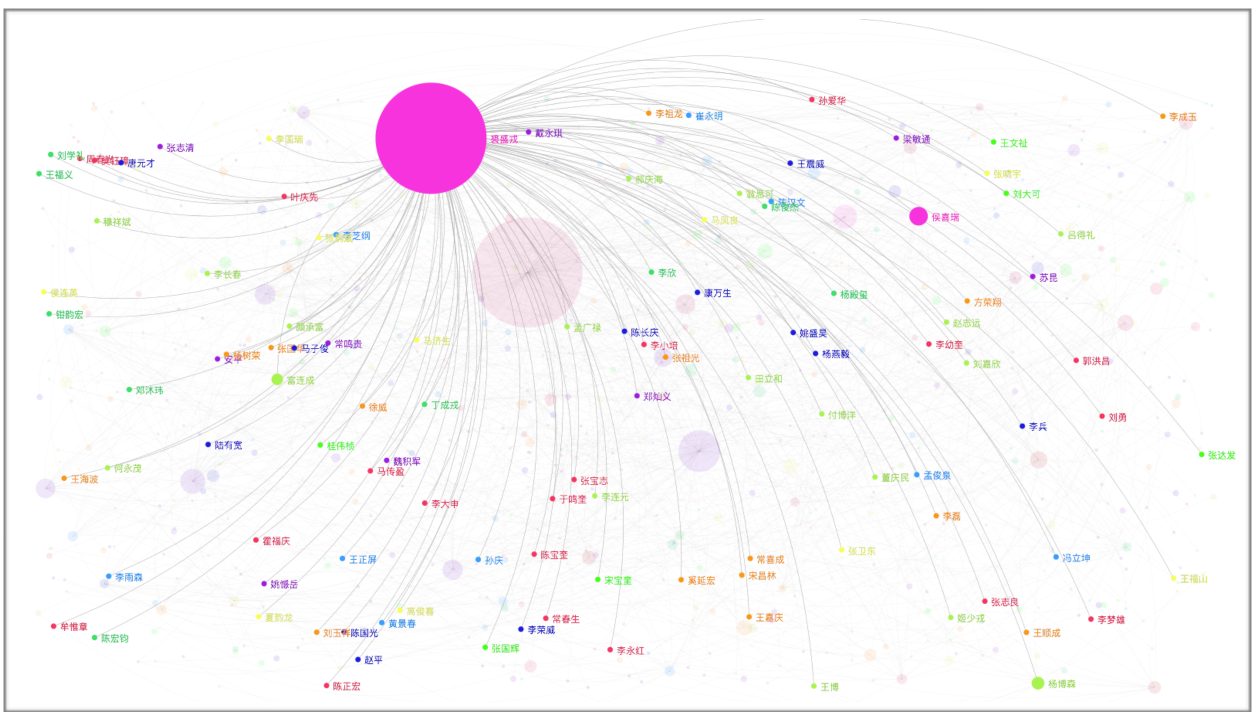

图2：裘盛戎—师徒关系

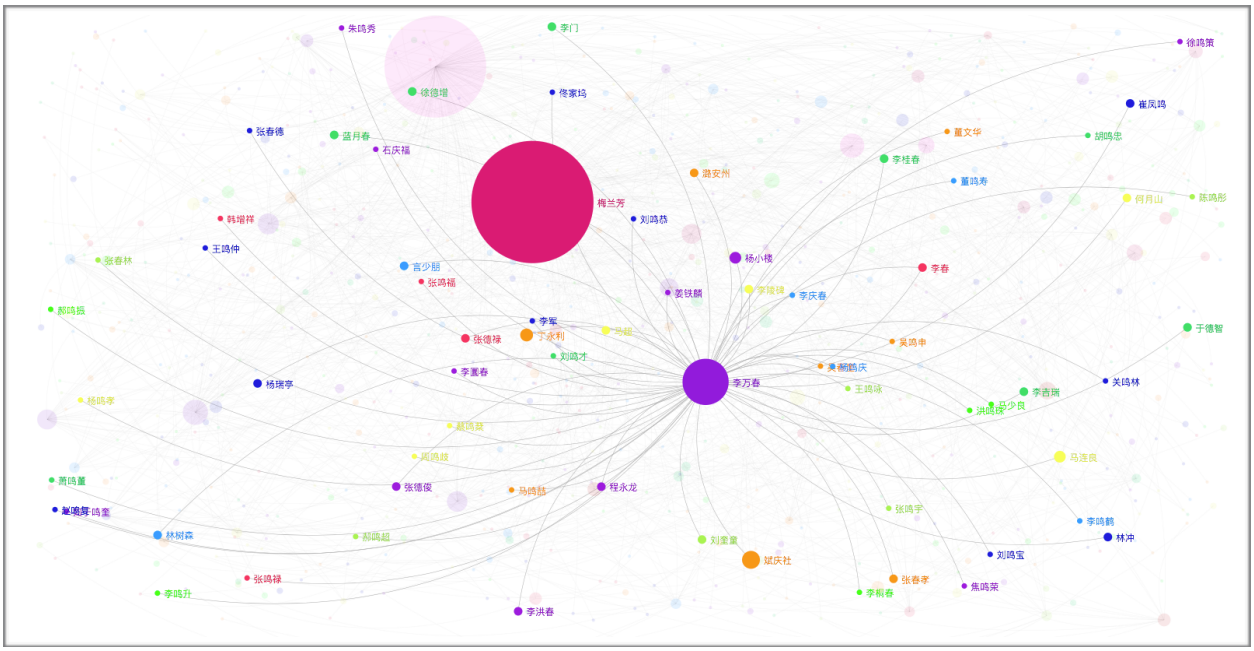

图3：李万春—师徒关系

六、额外加分点

1. 实现了多种关系的抽取(亲戚、合作)
2. 针对师徒关系，提供了大量关键词(具体可以查看/src/main/java/com/corp/myxof/config/InfoExtractionConfig.java文件)进行识别，提高了识别的准确率
3. 额外附加了人名字典，可以认为这个字典里面所有的人名都是正确的，用它检测一个名词是否是人名，提高准确率并剔除错误的人名
4. 在识别亲戚关系的，加入了姓氏识别，比如“梅兰芳，父明瑞”，可以将父亲梅明瑞识别出来
5. 加入了引用状态，将和主人公无关的人名全部过滤掉，降低了误识别率
6. 实体关系可视化

七、结论

本次实验主要分为三个部分，第一部分网络爬虫主要让我复习了以前写过的代码，第二部分让我学习了NLP工具如何使用。第三部分对我来说是全新的问题，这里采用了规则匹配和状态机解决这个问题，结果虽然不是100%正确，但是已经能够对标准的语法进行正确识别，但是对于一些只有人才能理解的表达方式，模式匹配做的不够好。我觉得解决这个问题的另一个思路应该是采用机器学习，用过大量的文本进行学习分析，但是这个京剧人物资料有限，学习的效果未必好，只有在数据集比较大的时候才能有一个好的效果。以后在做这类研究的时候应该试一试机器学习。