# 现代数据库系统概论

# 第一次作业报告

江天 2018312546

## 爬虫部分

## 爬虫框架

本次实验中使用的爬虫框架为WebMagic【1】。WebMagic是一款使用Java开发的轻量型爬虫框架。它具有如下特点：1.API简单，可以方便快速上手；2.结构模块化，方便扩展；3.提供多线程和分布式支持。下面将简单地介绍本实验中使用WebMagic进行爬取的流程。

在WebMagic，一个爬取实例称为一个Spider，一个Spider包含若干组件，例如Downloader，PageProcessor和Pipeline；其中Downloader决定如何从给定的url下载网页（本实验中使用默认的Downloader），PageProcessor负责对下载到的页面进行一些处理（例如在本实验中是对页面信息进行抽取并结构化）并且决定要将网页中的哪些链接加入到爬取队列，Pipeline用于获取PageProcessor的处理结果并进行进一步的操作（在本实验中是将其写入到一个XML文件）。具体的爬取流程如下：

1. 实现PageProcessor接口，定义如何对页面进行处理；
2. （可选）实现Pipeline接口，定义如何对PageProcessor的结果进行进一步处理；默认是将该结果输出到控制台；
3. 使用实现的PageProcessor构造一个Spider，并设置Downloader，Pipeline等组件，设置初始URL，线程数等参数。
4. 执行Spider。Spider会首先从初始URL下载网页，然后将网页交给PageProcessor进行解析并向爬取队列添加URL，再将解析结果交给Pipeline进行处理，之后从爬取队列取出新的URL，重复上述操作，直到爬取队列为空。

## 爬取对象以及处理

目标网站：新浪

话题：财经，军事，社会，体育，国际

数量：300

字段：ID（在本次爬取的该话题中第几个被爬取），URL，topic，title，date，content，editor（责任编辑，可能为空），source（新闻源），imgURLs（新闻包含的图片链接）

各字段的处理方式：

ID：由PageProcessor自己维护一个自增的整数。

URL：Spider会将下载到的页面包装为Page对象，通过Page.getUrl()方法可以获取。

Topic：Topic在创建PageProcessor时，作为一个参数传入。每一个PageProcessor（以及对应的Spider）只负责一个Topic，因此该Spider爬取到的所有网页都是该Topic。

Date：对爬取到的html页面使用XPath【2】（XPath是一种XML文档的元素选择方法，可以让使用者像使用文件路径一样访问XML中的元素）或正则表达式（Regexp）进行匹配，使用到的XPath或正则表达式有：

XPath "//\*[@id=\"top\_bar\"]/div/div[2]/span/text()"

"//\*[@id=\"pub\_date\"]/text()"

"//\*[@id=\"wrapOuter\"]/div/div[4]/span/text()"

Regexp ".\*published at ([^\\]]\*)\\].\*"

Content：通过XPath找到正文（article）所在的容器，然后通过WebMagic提供的smartContent函数提取其中的文本，典型的XPath有：

"//\*[@class=\"article\"]"

"//\*[@id=\"artibody\"]"

"//\*[@id=\"mainArticle\"]"

Editor：通过XPath "//\*[@class=\"article-editor\"]/text()" 进行选择。

Source：通过XPath "//\*[@class=\"source ent-source\"]/text()" 或 "//\*[@class=\"source\"]/text()"进行爬取

ImgURLs：通过以下XPath进行提取：

"//\*[@id=\"artibody\"]/\*/img/@src"

"//\*[@id=\"article\"]/\*/img/@src"

"//\*[@id=\"mainArticle\"]/\*/img/@src"

## 爬取结果

爬取结果存放在pages目录下，每一个子目录对应一个话题，每一个话题下是一个pages.xml文件，文件结构如下：

根元素是<pages>，它的子元素为若干个<page>元素，每一个对应一个页面，每一个<page>包含<id>,<URL>,<topic>,<title>,<date>,<content>,<editor>,<source>,<imgs>元素各一个，意义如1.2所述各个字段，示例如图1

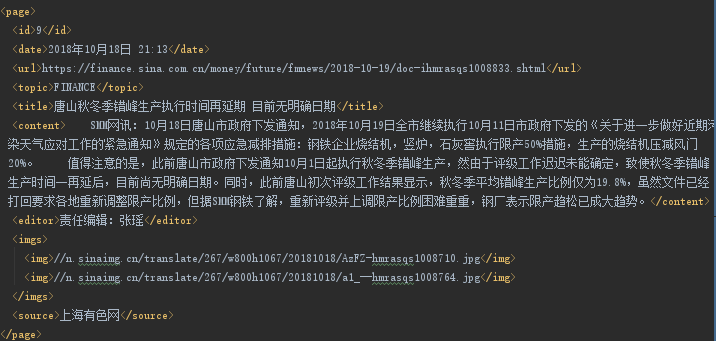


图1 <page>元素的结构示例

## 1.4 实践中遇到的问题

由于WebMagic并不支持设置爬取上限，我在实现中将Spider作为PageProcessor的一个字段，由PageProcessor自己对爬取到的有效页面（即包含具体新闻的页面）进行计数，当达到上限后不再产生新的处理结果并调用Spider的stop方法。

在开始时我对每个Spider给予了多个线程，结果发现不同线程同时写文件，使得文件结构被破坏。为了避免多线程竞争同一文件IO的情况，我对于每一个话题创建了一个Spider，并对每一个Spider只给予一个线程。

即使都是新浪新闻，网页的结构，尤其是html元素的命名也会有很大的不同。为了能够正确解析所有网页，我采取的方式是：观察某一个网页，使用该网页的结构编写规则；进行一轮爬取、将无法解析的网页通过异常抛出；分析这些异常网页的结构，补充新的规则，开始新一轮爬取。通过反复迭代来完善规则，从而正确识别所有网页。

当前很多网页采用的都是交互式设计，这使得网页上只会展示最新的新闻，想要查看往期的新闻需要点击相应的按钮，并且这些交互式多是通过Ajax而非修改网页URL来进行的，这就使得WebMagic这样的静态爬虫很难获取到往期的新闻，它们只能通过网页之间的静态链接来访问到往期新闻，而如果往期新闻和最新新闻处在不同的连通子集，就无法访问。

## 命名实体识别部分

## 2.1 实验工具

在本次实验中使用的自然语言处理工具是Stanford NLP。Standford NLP是斯坦福大学的自然语言处理团队开发的Java自然语言处理工具包（同时有其他语言版本），包含了统计模型，神经网络，规则模型等多种模型，支持诸如分词，分句，词性标记，命名实体识别，实体关系抽取等多种功能，并且支持中英西法德阿六种语言。

## 2.2 处理流程

在本实验中，我们的处理流程如下：

1. 使用配置文件"StanfordCoreNLP-chinese.properties"初始化StanfordCoreNLP(Pipeline)

对于之前爬取到的每一个网页：

2. 将标题和正文拼接为一个字符串，并构造一个CoreDocument；

3. 使用Pipeline对CoreDocument进行标注；

4. 遍历文档中的每一个词，按词的类别进行分类；

5. 将该网页的识别结果写到一个XML文件中

### 2.3 处理结果

Stanford NLP的处理结果和pages.xml放在同一个目录下，并命名为ner.xml，二者相互对应。ner.xml的结构如下：

根元素为<pages>，它包括若干个<page>元素，每一个<page>的结构为：第一个子元素<id>，即这个网页在pages.xml中的id；接下来是若干个子元素，每一个子元素的元素名为实体分类，元素文本是以逗号分隔的各个实体，按在文章中出现的顺序排列。一个示例如下：

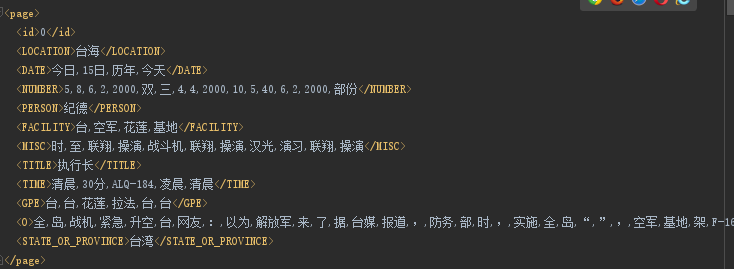


图2 命名实体识别的结构示例

观察这些结果，可以发现Stanford NLP对地点，时间，头衔，日期，数字等的识别还是比较准确的。

### 2.4 遇到的问题

通过Maven依赖导入Stanford NLP时，需要图3所示的三个依赖。我误以为第三个依赖包含了第二个依赖并进行扩展，所以一开始时只导入了第一、三两个依赖，发生了图4所示的错误，实际上第三个依赖并没有完整包含第二个依赖，因此所有三个依赖都是必须的。

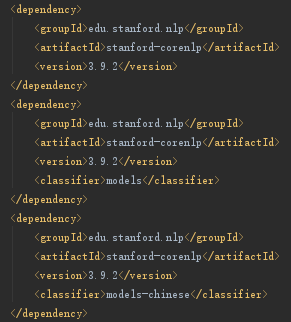


图3 使用Stanford NLP时所需的Maven依赖



图4 依赖不全时的错误

### 3. 实体关系抽取部分

### 3.1 处理流程

同样使用Stanford NLP，在本实验中，我们的处理流程如下：

1. 修改配置文件"StanfordCoreNLP-chinese.properties"，将其中的annotator改为”tokenize, ssplit, pos, lemma, ner, parse, depparse, relation”;

2. 使用配置文件"StanfordCoreNLP-chinese.properties"初始化StanfordCoreNLP(Pipeline);

对于之前爬取到的每一个网页：

3. 将标题和正文拼接为一个字符串，并构造一个CoreDocument；

4. 使用Pipeline对CoreDocument进行标注；

5. 遍历文档中的每一个句子，对于其中的每一个关系，按照关系种类进行分类。

6. 将一个网页中关系的抽取结果保存到XML中

### 3.2 处理结果

处理结果和pages.xml放在同一个目录下，并命名为rel.xml，二者相互对应。rel.xml的结构如下：

根元素为<pages>，它包括若干个<page>元素，每一个<page>的结构为：第一个子元素<id>，即这个网页在pages.xml中的id；接下来是若干个<relation>子元素，每一个<relation>包含两个元素，一个是<relation\_name>，它的内容是关系的名称；另一个是<relation\_participants>，它的内容为空格分割的二元组，每一个二元组形如“[<sub>#<obj>]”，其中<sub>,<obj>分别为宾语、主语。一个示例如下：

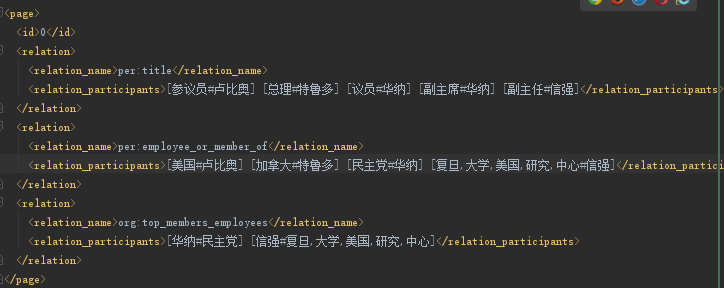


图5 关系抽取的一个示例

在这个例子中，成功识别出了“卢比奥”的“头衔”是“参议员”，“华纳”是“民主党”的“成员”，“信强”是“复旦大学美国研究中心”的“领导”等关系。尽管Stanford NLP对特定几类关系的抽取率尚可，但总的来说其支持的关系数较少，并且很容易对句子产生误解析。

3.3 遇到的问题

使用

额外点：

参考资料

1. WebMagic official site. <http://webmagic.io/>
2. BaiduBaike : XPath. <https://baike.baidu.com/item/XPath/5574064?fr=aladdin>
3. Stanford NLP official site. https://nlp.stanford.edu/