> 【*Data Science*】南京大学*2021*数据科学基础大作业

# 一. 总体介绍

这次大作业是实现分析“中国裁判文书网”的一个任务，主要采用了前后端分离设计，后端使用了`SpringBoot`来进行服务端的管理，因为不需要存储数据所以就没有使用数据库。

下面会简要介绍一下在此次大作业中的一些技术栈和遇到的困难和如何解决的。

# 二. 前端相关

## 1. 总体介绍

前端设计的初衷是想要设计出一个简洁美观，操作便捷，能够确实提升使用者使用效率的页面

借鉴网上模板，前端利用html，css，JavaScript等语言简单勾勒出页面的大体框架，整体页面以星空为背景，深蓝为基色，旨在希望帮助使用者使用时能够冷静专注，提升工作效率；此外页面内的边框大多采用圆角边框，使其看起来更加柔和舒适。而中间的主要部分从观感上可分为两大部分：功能区与标注区。

## 2. 功能区介绍

### (1) 功能介绍

功能区包括搜索框，以及“爬取文件”，“上传文件”，“开始分析”等按钮；主要实现的功能有：爬取文件，上传文件，分析文件。

#### 爬取文件

- 将搜索框内的关键字作为参数，在点击“爬取文件”按钮时将其传给后端的爬虫程序，得到其返回的内容将其呈现与输入文本框内；

#### 上传与分析文件：

1. 在点击“上传文件”的按钮后，能够将.txt文件内容同步到文本框内，同时在点击“开始分析”按钮后对将当前上传文件传输给后端，经NLP分析后能够生成根据名词，动词，形容词三类词性的分词，呈现在下方的标注区中；
2. 上传文件还提供了批量上传的功能，只需要在点击“上传文件”按钮后选中批量的txt文件，点击开始分析按钮后会先分析第一个文件的内容，在标注完成并保存后，NLP会自动分析文件流中的下一个文件，避免了重复上传文件的无效操作；
3. 如果不想要上传文件，还可以直接通过在文本框中输入内容，点击按钮进行NLP分析。

### (2) 研究过程

#### 上传文件按钮设计

在通过html与css完成功能区的基本样式设计后，前端方面通过JavaScript来获取页面对应元素，获取其属性内容，绑定监听对应事件以实现功能，如上传文件功能是利用了type='file'的input标签，

```
<input type='file'  value='上传文件' id="submitLocalFile"
style="position: absolute;" onchange="upLoadFile()" multiple>
```

但是由于该类标签已经固定好样式无法更改，于是我采用另外一个type='button'的input标签将其覆盖，

```
<input type="button" value="上传文件" style="position: relative;">
```

通过将第一个标签与对应上传文件事件uploadFile()相绑定以实现功能，设计第二个标签的样式使之与整体统一的方式实现整体样式协调与完成功能的双丰收

## 通信问题

在功能区的设计过程中前端遇到的难点主要是与后端的通信问题，在熟练运用JavaScript的基础上，我们通过学习Ajax通信方式，以及使用jQuery库，借localhost:8080端口传输彼此需要的参数与结果，再进行呈现与展示。

通信主要体现在分析的过程当中，在用户点击开始分析按钮后，当前页面需要根据用户上传的文件或是文本框的内容，将数据传输至后端程序，得到分词的结果呈现在下方的标注区中，


其过程是监听开始分析按钮的点击事件，给该事件绑定函数beginAnalyse()，在触发事件后将数据打包成FormData类型后通过jQuery库的Ajax传输给后端，并在成功通信后利用返回数据动态生成分词标注，而这则涉及到了标注区的相关功能，因此接下来会介绍一些标注区现有功能与研究过程

```
// ajax通信函数
 jQuery.ajax({
                type: 'post',
                url: url,
                data: formdata,
                contentType: false,
                processData: false,
                success: function (data) {
                    // alert("success");
```

```
                    words = data;
                    console.log(words)
                    formLabels(words)
                    afterAnalyse()
                }
            })
```

## 3. 标注区介绍

### (1) 功能介绍

标注区实现功能主要有：

1. 根据后端回传数据动态动态生成标签；
2. 根据选中标签标注在点击保存按钮后生成案件文本与案件标注文件。

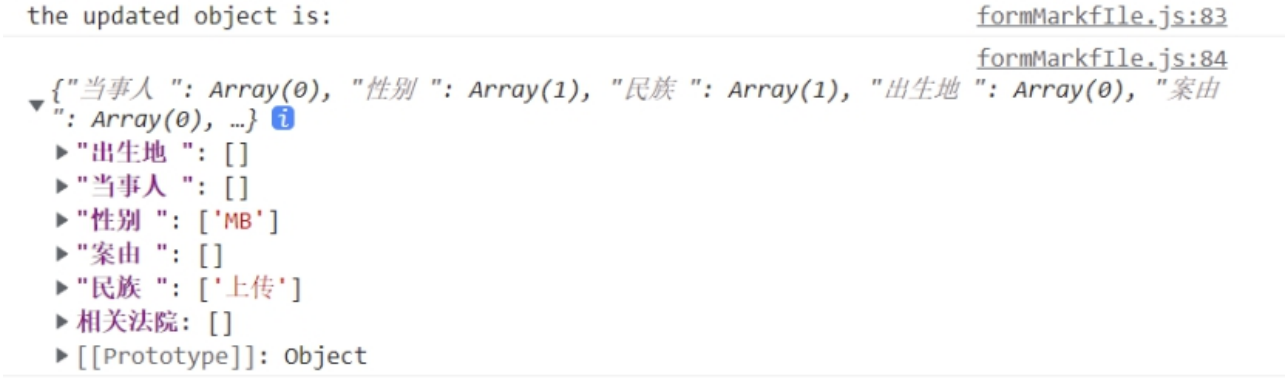
### (2) 研究过程

#### 动态生成标签

在根据词性动态生成标签的研究过程中，我们将分词结果的次性设置为3类：名词、动词、形容词。从NLP程序出得到回传数据，通过在得到数据后生成ul，li，以及挂载于其中的label，input等元素来装载对应的数据，并使之在页面中成功渲染实现效果；以及为了实现多文件传输自动分析，我们还设置了每一次生成对应文件后清空标签，初始化标注信息，使多文件传输得以正常进行

#### 记录信息保存到本地

根据选中标注与标签，我们会生成包含对应数据的markObject对象，如图：




在用户点击保存按钮后，能够将标注与案件文本对象，分别转化成json与txt文件保存到本地。

难点：与功能区的相互协作

在标注区实现过程遇到的难点主要是与功能区功能的衔接，如如何根据NLP结果动态生成标签，在多文件传输的情况下如何判断是根据文本框还是特定文件的内容进行分析得到结果，在合理分配二者函数类别，相互协作，最后合理实现！

# 三.后端相关

---

## 1. NLP

在NLP分词方面，主要使用的是`HanLP`来进行分词和划分词性，调用的是其中的标准分词器，如下所示：

```
List<Term> termList = StandardTokenizer.segment(“商品和服务”);
```

这里的`segment()`方法就是将参数进行分词且获得词性，下面会简要介绍一下这种分词的原理，用到的是中文分词中比较常见的词典和词图生成

# (1)算法详解

## 词典的生成

首先是词典的生成是在`HanLP`中带有了基础的词典之上添加了许多有关法律方面的专有名词，比如一些罪行的名字和一些判罚的条例等专有名词。当然，词典并不能够做到百分百完美的分词，我们可以举一个例子来说明词典可能出现的错误：

> 对于“川普”这个词，如果在词典中的话，那么在分词阶段其就极有可能会被分成结果（在只用词典的时候），但在不同的语境中，这个词的意思其实是不一样的，比如如下的两种情况
>
> 1. 你的四川普通话真好
> 2. 美国总统川普
>
> 很明显，在这两句话之中，“川普”这个词有着不同的含义，如果在第一个语境中将“川普”作为一个完整的词分在一起，显然就破坏了句子原本的含义。

针对这种情况，`HanLP`使用了多种方式来减少这种情况的出现。首先，`HanLP`使用了词性 + 频率的方式来记录词典，即对于相同的几个字，其中的组合可能有些的频数会比较大，有些频数会比较小。`HanLP`基于此来记录词典，可以比较有效的让某些出现次数较多的组合不会被误判。`HanLP`支持用户动态添加词典以及自定义词典，也可以自己确定优先级，这让我们要在特定领域使用特定的词典能够进行灵活的调整。

其次，`HanLP`使用了词图的生成方式来进行分析，这也是下面所要讲的

## 词图的生成

当分词系统有一份词典的时候，就可以生成词图了。所谓词图，指的是句子中所有词可能构成的图。如果一个词A的下一个词可能是B的话，那么A和B之间具有一条路径E(A,B)。一个词可能有多个后续，同时也可能有多个前驱，它们构成的图在`HanLP`中称作词图。

需要稀疏2维矩阵模型，以一个词的起始位置作为行，终止位置作为列，可以得到一个二维矩阵。例如：“他说的确实在理”这句话

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 始##始 | | | | | | | | |
| 1 | | | 他 | | | | | | | |
| 2 | | | | 说 | | | | | | |
| 3 | | | | | 的 | 的确 | | | | |
| 4 | | | | | | 确 | 确实 | | | |
| 5 | | | | | | | 实 | 实在 | | |
| 6 | | | | | | | | 在 | 在理 | |
| 7 | | | | | | | | | 理 | |
| 8 | | | | | | | | | | 末##末 |

那么如何建立节点之间的联系呢？也就是如何找到一个词A的后续B、C、D.....呢？有两种已实现的方法，一种是所谓的DynamicArray法，一种是快速offset法。

**DynamicArray**法

最直截了当的想法当然是用二维数组模拟这个模型了，很明显，其中有不少空洞，所以在ICT系列的分词器中定义了一个蹩脚的DynamicArray结构用来储存模型，DynamicArray结构的每个节点包含一个个词的row和col，待会儿看完offset法你就会明白我什么么说DynamicArray蹩脚。

在这张图中，行和列有一个非常有意思的关系：col为 n 的列中所有词可以与row为 n 的所有行中的词进行组合。例如“的确”这个词，它的col = 5，需要和它计算平滑值的有两个，分别是row = 5的两个词：“实”和“实在”。

连接词形成边的时候，利用上面提到的关系即可。

但是在遍历和插入的时候，需要一个个比较col和row的关系，复杂度是O(N)。

快速**offset**

虽然模型的表示用DynamicArray没有信息的损失，但问题是，真的需要表示模型吗？

当然不，可以将起始offset相同的词写到一行：

```
始##始
他
说
的/的确
确/确实
实/实在
在/在理
理
末##末
```

这个储存起来很简单，一个一维数组，每个元素是一个单链表。

怎么知道“的确”的下一个词是什么呢？“的确”的行号是4,长度是2,4+2=6，于是第六行的两个词“实/实在”就是“的确”的后续。就这么简单。

同时这种方法速度非常快，插入和查询的时间都是O(1)。

**reference**


1. HanLP官方文档
2. NLP中文分词原理及分词算法


地名的识别

地名识别的算法比较复杂，涉及到了HMM等，通过训练对地名进行标注然后统计词频得到词典，将多个HMM模型叠加可以发挥更加精准的效果。

> *HMM-Viterbi*角色标注地名识别

## 2.前后端通信

后端部分采用了`SpringBoot`来进行开发，几个重要的接口如下所示：

1. 获取文本分析的结果

```
/**
 * 返回词性分析的结果
 * @param text 待分析的文书
 * @return 词性分析结果，词性 - 对应的单词
 */
@RequestMapping(value = "/getResult", method =
RequestMethod.POST)
@ResponseBody
@CrossOrigin(origins = "*")
```

```
public Map<String, List<String>>
textAnalysis(@RequestParam(value = "text") String text) {
    // 清除之前的分析内容
    analysis.clear();
    analysis.setParagraph(text);
    analyse();
    return analysis.getRes();
}
```

2. 上传文件进行分析

```
/**
 * 接受一个文件的传输，返回词法分析的结果
 * @param uploadFile 用户上传的从前端接收的文件
 * @return 词法分析的结果
 * @throws IOException 文件接收异常
 */
@RequestMapping(value = "/uploadFile", method =
RequestMethod.POST)
@ResponseBody
@CrossOrigin(origins = "*")
public Map<String, List<String>>
uploadFile(@RequestParam("uploadFile") MultipartFile
uploadFile) throws IOException {
    if (uploadFile == null) {
        // 接收失败
        return null;
    }

    InputStream inputStream = uploadFile.getInputStream();
    BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream, StandardCharsets.UTF_8));
    StringBuilder temp = new StringBuilder();

    while (reader.ready()) {
        temp.append(reader.readLine());
    }

    // analyse
    String content = temp.toString();
    return textAnalysis(content);
}
```

3. 爬取文书内容

```
/**
 * 爬取文书内容
 * @param searchContent 搜索的关键词信息
 * @return 文本内容
 */
@RequestMapping(value = "/reptile", method =
RequestMethod.POST)
@ResponseBody
@CrossOrigin(origins = "*")
public String reptile(@RequestParam("searchContent") String
searchContent) throws InterruptedException {\
    reptile.clearContent();
    reptile.reptile(searchContent);
    return reptile.getContent();
}
```

`@RequestMapping`是`SpringMVC`中十分重要的一员，在`SpringBoot`中其实有着简化的`GetMapping`来代替，但是`GetMapping`在http中接受的是`get`请求，本项目中所有请求都是用`POST`请求来发送的，所以还是使用了`RequestMapping`。

`@CrossOrigin`是为了解决跨域问题的

`@ResponseBody`其实在这里是非必需的，因为控制类我用了`@RestController`进行标注，该标注是在`@Controller`的基础上进一步的优化，已经自带了`@ResponseBody`的annotation。

## 3. 爬虫

爬虫使用的是`Selenium`来进行模拟爬取，非本项目的研究重点，不过多介绍

## 4. 研究过程

司法数据分析的应用场景是目前较为火热的一个应用场景。近年来，人民法院深入贯彻习近平法治思想，加快智慧法院建设，推动大数据管理与服务平台建设，构建中国特色互联网司法新模式，积极为全球互联网法治发展贡献中国智慧中国方案。大数据已成为推动法院改革发展的重要力量，加快大数据同司法工作深度融合发展潜力巨大。

1. 大数据为审判执行赋能。运用司法大数据，有助于实现审判执行的自动化和精准化。大数据在众多司法活动领域如类案推送、量刑辅助、偏离预警、裁判文书自动生成、虚假诉讼识别、判决结果预测、诉讼风险评估等都有应用前景，给审判执行工作现代化带来新机遇。大数据可实现类案自动关联、法律法规推送，为法官工作提供个性化、精细化、智能化服务，便于统一法律适用标准，避免“同案不同判”。通过区块链技术统一证据标准，还能辅助案件办理过程中的证据采集和认定工作。
2. 大数据为审判管理赋能。大数据的价值在于提供科学的技术方法，助力科学决策，助推管理革命。大数据应用可以通过构建扁平化审判管理体系，实现事后管理向事前、事中管理转变，有效提升审判管理质效；有助于及时掌握办案情况，开展有效的流程管理、态势分析；可以科学测定工作量，健全和完善审判监督管理体系，提升审判管理自动化、精细化、智能化水平。
3. 大数据为诉讼服务赋能。大数据同诉讼服务工作深度融合，有助于推动诉讼服务更加便捷、透明、高效。已经建成的人民法院大数据管理与服务平台，可以提供审判运行态势数据、司法统计报表、律师律所信息、司法公开数据等综合数据查询服务。全国四级法院的信息化诉讼服务大厅，可为当事人提供诉讼引导、立案登记、诉前调解和救助等全方位信息服务。一些地方法院广泛应用无接触式诉讼服务，当事人足不出户就能参加诉讼。

本项目旨在对用户上传的文书进行分词和词性标注，辅助用户能够更加快速的对文书进行信息的获取和分析，通过分词和词性标注让用户在不同的词性当中选出“当事人”，“性别”，“民族”，“出生地”，“案由”，“相关法院”等信息，能够让用户更加快速获得想要获得的信息。在研究过程中，我们发现部分文书其实并不是十分的完整，并不一定能够获得完整的信息，比如说再审书或者其他的一些非裁判书，对于这些文书，我们所做的一样是提取其中有用的信息来帮助用户更好的对判决结果信息进行一个获取和汇总。