判决书处理流程（涉及计算机背景成员部分为黑色，其余标淡紫，重要步骤标红，illustration标蓝，举例解释小灰字|p.s.本文稿整理者为纯法学背景，如有表述不当之处还望海涵雅正）

- **STEP1【样本获取】**获取判决书
  - **目的**：获取样本
  - **样本容量**：1万份上下
  - **样本范围**：初拟为全样本分析，因此由法科生将裁判文书检索完毕之后交由计算机方向同学进行下载即可，计算机方向同学大概率不需要额外进行筛选
  - **获取样本步骤**：两种方法，二者取其一，由计算机方向负责人自行选用
    - 威科先行（一个裁判文书网站）手动集中下载：每点击一次可以下载100份判决书
    - 北大法宝（一个裁判文书网站）或威科先行（一个裁判文书网站）、无讼（一个裁判文书网站）使用爬虫访问提供判决书的网站，获取判决书
      - step1 确定提供判决书的网站作为爬虫目标（如上所述，北大法宝或威科先行或无讼，由于三个网站提供的样本容量及导出形式不同，具体网站选择由法科生与计算机结合专业角度沟通决定）了解其使用方法和技术参数

        【例】在威科先行法律信息库检索包含受贿罪关键词的裁判文书，浏览器上方显示的URL地址是——而通过抓包软件分析可发现请求判 决书 的 真实地址 是 ——使 用 爬虫 访问该地址才能正确获得判决书
      - step2 制定爬虫策略，编写爬虫程序

        【参考】一份爬虫代码，一周3000万份裁判文书+200万份法律文书：https://github.com/tsfnziy120/paper_crawler
        - 控制爬虫速度，防止被封禁
        - 涉及断点续传功能，提供程序的容错性
        - 选择适当的方式保存采集的数据：（或）
          - 写入磁盘
          - 存入数据库
      - step3 了解和规避目标网站的反爬虫措施→规避
        - 威科、北大法宝、无讼的反爬虫措施均较弱
- **STEP2【数据预处理】**对下载/爬取的裁判文书进行处理
  - **目的**：避免原始数据中存在的瑕疵，防止产生程序问题或者得出错误结论
  - **数据预处理步骤**：
    - step1文件格式统一

        - 原因：人工下载的判决书基本为word格式，便于人类浏览和编辑，不利于计算机程序访问和处理
            【问】不知爬虫下载的判决书一般为何种格式呢?
        - 处理方法：（或）
            - 将word文件转化为txt文件
            - 直接将判决书存入数据库
    - step2 文件编码
        【例】UTF-8编码——txt.文件
    - step3 文本内容清洗
        - 去除判决书中的商业标识（商业标志、标语、名称、二维码etc.）
        - 去除不可见字符，提高后续数据提取的准确性
            【例】不可见字符如换行符\n和回车符\r等
        - 标准化标点符号
            【例】统一中文标点和英文标点，全角标点和半角标点等
    - step4 数据去重：去除重复判决书
        - 原因：无法保证同一网站下载的裁判文书之间不重复
        - 处理方法：【1】提取判决书的案号→【2】根据案号判断出重复的判决书并删除
    - step5 筛选符合条件的判决书
        - 原因："脏数据"的存在
            【例】虽然在检索判决书时已经通过"高级检索"功能将案由限制为受贿罪，但初步采集的数据当中仍然混有大量的非国家工作人员受贿罪案件和其他罪名的案件
            【问】具体的操作步骤尚不确定→请教张建悦师兄
        - 处理方法：待定
    - step6 为每一份文书生成标识码
        - 原因：目的在于使得计算机程序依次有序读入判决书内容同时提取要素出错时能够快速定位出错的判决书
        - 处理方法：
            - 根据全部样本的数量
            - 生成一套序号，分配给每一份判决书
            - 序号从0开始
- **STEP3【提取要素】**从上述处理好的判决书当中通过字符匹配技术进行要素提取
    - 要素：提取程序将部分判决书内容转化为结构化的信息
    - **主要方法**：
        - 正则表达式：主要中的主要，直接定位判决书当中的相关语句和提取有关要素，适合处理判决书（判决书文本的特点见下一个节点主题）
            【参考文献】[美] Jeffrey E·F·Fridl：《精通正则表达式》（第三版），余晟译，电子工业出版社2007年版

【例】
①背景：裁判文书当中，判决结果经常使用“根据中华人民共和国……法……判决如下：……”的表述
②目标提取要素：A.援引的法条　B.判决的结果
③根据①与②，编写正则表达式对该语句进行定位，并提取出②中的目标提取要素，这一正则表达式可以表示为：“根据中华人民共和国（.+？），判决如下：（.+？）

- 自然语言分析：本研究当中不一定用得上，计算机负责同学自行斟酌即可
  - 根据中文词汇被使用的概率分布，结合中文语言本社的呢构词构句规律，对判决书文本进行分词、句法分析、语义分析等技术处理，提取语言所要表达的逻辑和意思

- **判决书文本作为要素提取范围的特征：**
  - 格式较为规范
  - 表述相对固定
- **正则表达式提取判决书要素使用工具：Python**
- **正则表达式提取判决书要素步骤及注意事项：**
  - step1 读入判决书内容

```
with open('C:\paper.txt','r',encoding='utf-8')as f:
    paper_content = f.read()
```

逗号分隔的要素分别为：①判决书的文件路径 ②读取文件 ③文本编码方式
paper_content指代读入判决书的全部内容

  - step2 按照文本结构分解文本：判决书分区
    - 原理：捕获对应部分开始和结束的标志，实现对文本的分割
    - 作用：每个要素提取逻辑的作用范围限制于特定的文书区域，根据分割的结果在各个部分开展要素提取任务

序号	区域名称	包含信息	通用开始标志	通用结束标志
1	标题	法院名称、文书类型、案号	—	“号”
2	诉讼程序	原被告主体信息、案件进入诉讼程序的流程信息	“号”	“现已审理终结”
3	控辩理由	原告起诉理由、被告抗辩理由	“现已审理终结”	“经审理查明”
4	事实认定	法院认定的事实和证据	“经审理查明”	“本院认为”
5	审理意见	法院审理意见和重要结论	“本院认为”	“判决如下”
6	判决结果	判决结果	“判决如下”	“审判长”
7	审判人员	合议庭成员姓名、宣判日期	“审判长”	—

【例】通常判决书文本分割方式

  - step3 设定待提取要素：文本型 数字型 时间型
    - 这一步骤主要关涉计算机背景成员的核心任务，即将A从B判决书当中提取出来，例如，需要从判决书当中识别出具体的量刑情节（自首、坦白etc.）
    - 这一步骤主要由法学生基于专业背景和研究目标完成
    - 但是也需要计算机方向同学参与，因为涉及到法学生所期待提取出的要素在计算机提取环节的可行性问题

- step4 编写正则表达式提取要素 这个步骤根据要素性质的不同提取方式和难度也有不同 *注意提高正则表达式的**兼容性** 注意判决书中可能存在的**错别字***
  - 直接提取位置相对固定的要素
    - 【例】正则表达式基础语法

序号	特殊符号	指代的内容	序号	特殊符号	指代的内容
1	\d	1 个数字(0～9)	7	法 +?	“法”出现 1 次或多次，但取最少的次数
2	.	1 个字符(数字、字母、汉字等)	8	法 {2,4}	“法”出现 2～4 次
3	[受贿罪]	“受”“贿”“罪”三个字符取 1 个	9	(?= 刑)法	“法”，同时“法”的前面是“刑”
4	法？	“法”出现 0 次或 1 次，取最多的次数	10	法(?=学)	“法”，同时“法”的后面是“学”
5	法 +	“法”出现 1 次或多次，取最多的次数	11	(?! 民)法	“法”，同时“法”的前面不是“民”
6	法 ??	“法”出现 0 次或 1 次，但取最少的次数	12	法(?!律)	“法”，同时“法”的后面不是“律”

    - ①确定要素在判决书忠出现的位置
    - ②根据经验和知识产生提取逻辑
    - ③提取逻辑融入用编程语言描述的正则表达式之中

      【例】
      #判决书中的原表述：“判决如下：一、被告人徐某某犯受贿罪，判处有期徒刑八年六个月”→在程序中记为“sentence”
      #功能逻辑中的表述：
      ①导入正则表达式功能模块：
      import re
      ②编写正则表达式，分别对应被告人姓名、罪名和刑罚：
      result_regex=re.compile（‘判决如下：一、被告人（.+？）犯（.+罪），判处（.+？）（’）
      ③正则表达式搜索sentence语句，搜索结果存入match：
      match=result_regex.search(sentence)
      ④从搜索结果中提取被告人姓名、罪名和刑罚，分别存入name,crime,penalty：
      name,crime,penalty=match.group(1),match.group(2),match.group(3)
    - ④由计算机程序执行和获取结果
  - 提取位置不固定、表述多变的要素
    - 处理方法：
      - 改进正则表达式：适当放宽提取条件，获取更多模糊要素，再从中筛选出符合条件的要素
      - 精确输入内容
      - 突破判决书分区，扩大检索范围
    - 特殊情况中的特殊情况：
      - 要素在不同判决书中表述不同，无法统筹在同一正则表达式当中
        - 处理方法：积累所有的表述，编写多个正则表达式同时提取
      - 要素在同一判决书中表述不一致（通常出现在判决书中的意见性表述当中）

- 处理方法：使用关联要素

  > 【例】当一篇裁判文书当中既会出现“构成自首”也会出现“不构成自首”时，可以通过“坦白、追缴、配合追缴等关联要素进一步确认辩方是否存在自首情节

- 提取间接要素
  - 根据要素提取
  - 根据要素组合

- step5 输出提取结果
  - CSV格式输出
    - 在phthon程序中，CSV格式可以由dandas类库中的DataFrame.to_csv函数生成

- step6 准确度检查和程序改进
  - 准确度检查
    - 人工抽样检查准确率

      > 提取要素的准确率定义：对每一要素，经过人工检查的正确的要素数量和要素总数之比

  - 程序改进