悬疑推理小说发展可视化分析

《数据可视化》课程期末项目报告

姓名： 俞辰杰　　学号：10192100571　　小组选题：文本可视化

1. 项目简介

此项目的初衷是为了展示自 1940 年至 2022 年为止，悬疑推理小说的热度变化趋势，以及推测这些变化是否存在着某些联系与共同点。根据豆瓣网对“悬疑推理小说”类别的好评榜单，推测在中国网民中的热度和偏好的变化，观察到悬疑小说的历史变化趋势：在千禧年之前，欧美悬疑小说更受网民欢迎；然而现代更受欢迎的悬疑小说却以日本作品为主。

基于这一结果，我们提出了四个疑问：

1、上个世纪的欧美悬疑推理小说和现代日本悬疑推理小说之间是否存在着某些区别？

2、不同类型的悬疑推理小说之间是否存在着某些区别？

3、近代欧美新式推理小说是否在尝试传统类型之外新的写作风格？

4、是否小说的读者对于推理悬疑的爱好迁移到了其他的领域？

为了进一步寻找其中是否包含某种联系，我们选取上世纪欧美悬疑作品的代表人物阿加莎·克里斯蒂的作品《尼罗河上的惨案》《无人生还》的作品，与现代日本推理悬疑作品《白夜行》进行悬疑推理角色关系复杂程度上的比较，分析得到结果传统欧美悬疑推理小说人物关系较为清晰，犯人的出场频率较为频繁；而现代日本的悬疑推理作品中人物关系更为复杂。

接着将《白夜行》与上世纪日本变格推理悬疑作品《脑髓地狱》进行比较，发现后者使用人物关系图无法清晰的阐释文章脉络和主题，采用了一种“关键词”的方法来暗示主旨与“精神”、“梦境”相关。而这一现象在现代欧美半推理作品《达芬奇密码》中也同样出现：大量出现的“圣杯”与“教会”也暗示了本文的剧情与宗教仪式相关。

在研究的最后，我们将目光放在外部因素上，在寻找中国网民对于悬疑作品的喜好是否体现在了其他地方，根据近年来对于“悬疑电影”“剧本杀”等关键词的查询频率，我们发现“悬疑小说”这一搜索词条的热度，在呈现持续下降的趋势：“悬疑小说”关键词的出现频率在近年来持续低于“悬疑电影”“剧本杀”等关键词。

2. 数据简介

此项目的数据分为三部分：

第一部分是从豆瓣，亚马逊等网站爬取到的“悬疑推销小说”的 Top250 推荐榜，用于判断哪些悬疑推理书籍受到欢迎、按国家进行分类统计推理小说的热度随着年份的变化情况、分析哪些类型的推理文章更加受到国内的读者的欢迎。同时分析哪些作者的热度更高，从此着手进行文本分析。

第二部分是悬疑小说的 txt 原文（《尼罗河上的惨案》，《无人生还》，《白夜行》，《脑髓地狱》，《达芬奇密码》），从图上用于在后续的单词分割制作词云统计，或是人物关系提取制作力导向图中使用。

第三部分是搜索引擎中对于“悬疑小说”、“悬疑电影”、“剧本杀”等关键词的搜索频率，通过这些数据，用于观察悬疑小说相较于其他娱乐方式，其本身的热度变化情况；以及《神探夏洛克》和《夏洛克福尔摩斯》这一原作和衍生剧的搜索对比，用于纵向比较同一 IP 在小说和电视剧上的不同热度

3. 数据处理

对于从豆瓣读书上爬取到的推理小说数据：

由于有些小说信息中未标明作者国籍，所以按照已知国籍的同名作家自动补齐却国籍作家的国籍，剩余作家则手动搜索国籍。

之后再按照小说出版年份对所有数据进行排序，并且对于 2000 年之前出版的图书，每十年统计一次各地理区域出版图书占比（主要统计中、日、美、欧洲 四个区域），对于 2000 年之后出版的图书，每五年统计一次。

对于作者出现的次数进行统计，分别找到出现最多的 10 位作者，再从源数据中找到该作者对应的分数最高的几部作品依次进行排序，按照作者的出现比例来排布作品的出现次数，呈现出旭日图的效果。

对于小说原本，本项目中使用了多种方法处理这些文字进行可视化：

第一种处理方法，为了能够呈现出推理小说中的人物联系关系，使用力导向图和关系直观呈现出人物之间的关系，用线段粗细(关系图)和空间距离（力导向图）表示人物之间的关系紧密程度。观察能否从人物关系图中分析出凶手在故事中位置？

对于《尼罗河上的惨案》,《无人生还》采用了关系图，对于《白夜行》采用了空间力导向图的结构。预先创建好名字文件和昵称对照字典，再使用 jieba，从文章中统计所有对应的人物出现的次数；同时对于每个人物，统计在出现的时候，上下文中是否出现了其他的角色，将此计入一次联系。

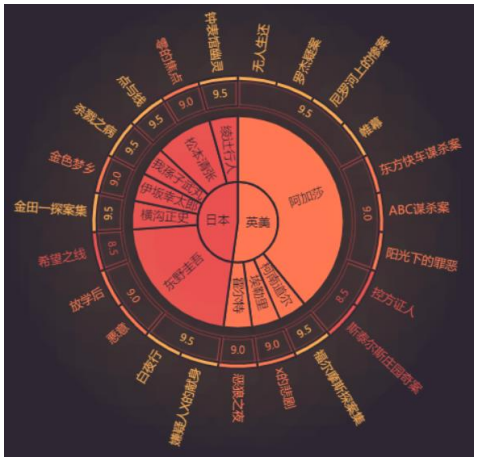
此方法可能会出现一些误差，例如“我”所对应的人称代词所对应的角色无法被统计，角色使用“我”的次数和出场次数成正比，所以此误差的影响并不大，可以在一定程度上统计人物之间的联系。

第二种处理方法，将文本进行直接语义拆分，将文本中的词语剥离进行统计，用于直接观看在文本中出现的最多的“关键词”是什么？使用“关键词”能否能够清晰地表达出这篇小说的主要内容与主旨，能否清晰的描述这篇小说是什么类型的小说？

对于《达芬奇密码》、《脑髓地狱》与《克苏鲁的呼唤》采用了此种处理方法，使用 jieba 得到预处理的词频统计文件；使用 MySQL 处理一些混乱的单词，优化数据结构，导出 json 脚本文件；最后使用 WordCloud 呈现出词云效果。

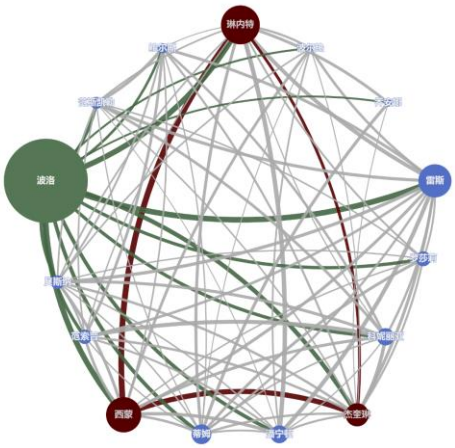
4. 可视化设计

对于作者的排行榜上的热度和其代表作品，可以使用旭日图来展现其中的信息，且旭日图具有饼图的性质，也可以对比的显示出不同区域间作者的占比差异，展现结果如下：

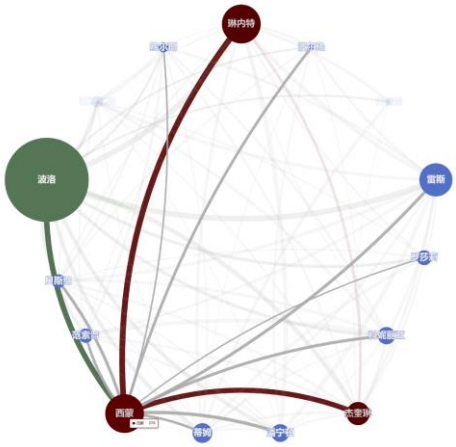


豆瓣“悬疑推理”排行榜 Top 作者

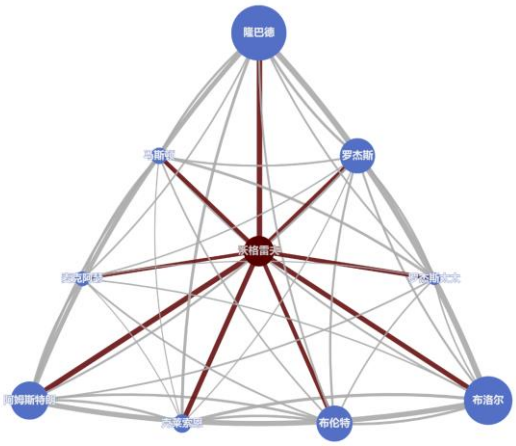
对于本格推理类小说，使用人物关系图可以较为清晰直观的表现出角色之间的联系，因此对于《尼罗河上的惨案》和《无人生还》，使用平面的关系图进行可视化。可视化图形采用正多边形，由于大多边之间存在着练习，如此进行展示的时候结构更加整齐紧密：



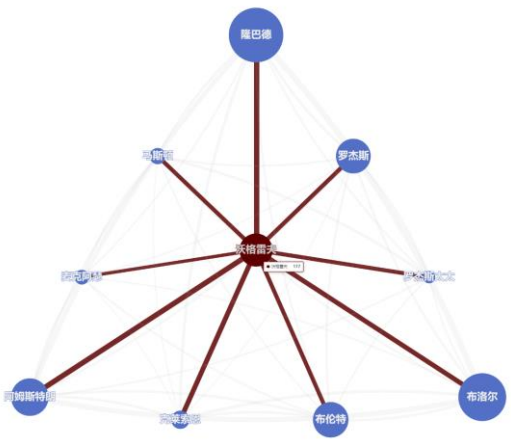
《尼罗河上的惨案》人物关系图



《尼罗河上的惨案》凶手关系图

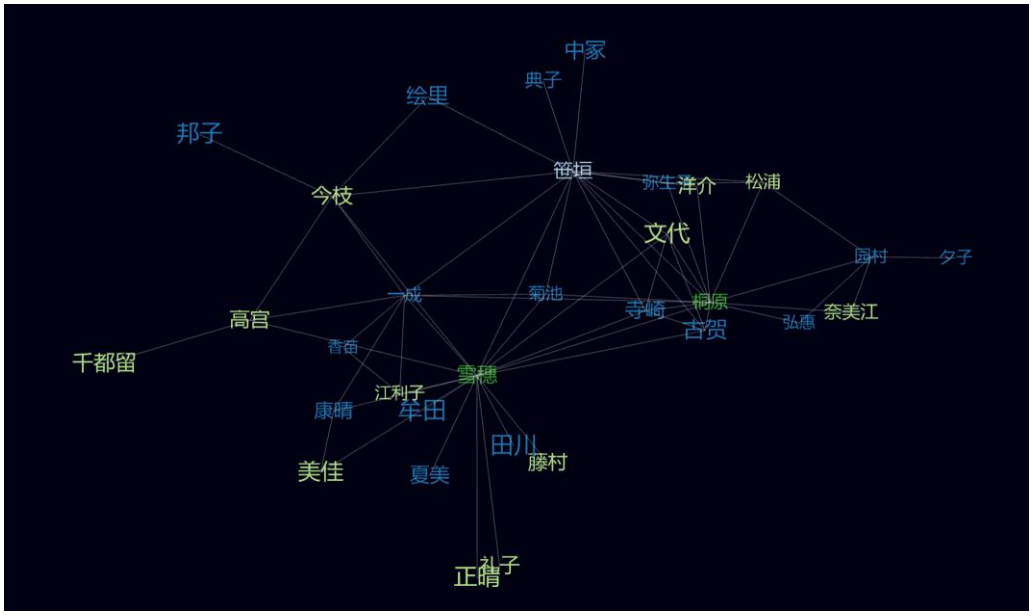


《无人生还》人物关系图



《无人生还》凶手关系图

对于社会类推理小说《白夜行》，由于出场角色相比于本格推理类小说多出许多，再使用平面关系图，会导致整个图混乱程度过高，可视化程度降低。所以本项目中，我们采用空间力导向图，可以交互式的拖动网页界面，选择式点击文字节点的观看：



《白夜行》人物关系图

对于变格推理类小说《脑髓地狱》、非传统推理小说《达芬奇密码》以及恐怖小说《克苏鲁的呼唤》，其本身的推理过程较弱，角色联系强度也较差，所以为了能够表达清晰这类文章，我们采用词云的方式，提取出关键词，通过关键词分析文章的表达内容。最终的呈现蒙版使用与主题相关的背景图像，这样在文字之外，还可以通过图像进行主题暗示：

《脑髓地狱》词云

《达芬奇密码》词云

5. 编程实现

文本人物关系统计 relationship_view.py

```
def synonymous_names(self):
    '''
    获取同义名字典
    :return:
    '''
    with codecs.open(self._synonymous_dict_path, 'r', 'utf-8') as f:
        lines = f.read().split('\n')
    for line in lines:
        self._synonymous_dict[line.split(' ')[0]] =line.split(' ')[1]
    return self._synonymous_dict

def get_clean_paragraphs(self):
    '''
    以段为单位分割全文
    '''
    new_paragraphs = []
    last_paragraphs = []
    with codecs.open(self._text_path, 'r', 'utf-8') as f:
        paragraphs = f.read().split('\n')
    for i in range(len(paragraphs)):
        if paragraphs[i] != '':
            new_paragraphs.append(paragraphs[i])
    for i in range(len(new_paragraphs)):
        new_paragraphs[i] = new_paragraphs[i].replace('\n', '')
        new_paragraphs[i] = new_paragraphs[i].replace(' ', '')
        last_paragraphs.append(new_paragraphs[i])
    print(len(last_paragraphs))
    return last_paragraphs

def count_person(self):
    '''
    统计人物出场次数，添加每段的人物
    '''
        paragraphs = self.get_clean_paragraphs()  # checked
        synonymous = self.synonymous_names()  # checked
        print('start process node')
        with codecs.open(self._dict_path, 'r', 'utf-8') as f:
            name_list = f.read().split('\n')  # 获取干净的 name_list
        for p in paragraphs:
            jieba.load_userdict(self._dict_path)
            # 分词，为每一段初始化新字典
```

```
            poss = jieba.cut(p)
            self._person_per_paragraph.append([])
            for w in poss:
                # 判断是否在姓名字典以及同义词区分
                if w not in name_list:
                    continue
                if synonymous.get(w):
                    w = synonymous[w]
                # 往每段中添加人物
                self._person_per_paragraph[-1].append(w)
                # 初始化人物关系，计数
                if self._person_counter.get(w) is None:
                    self._relationships[w] = {}
                self._person_counter[w] += 1
        return self._person_counter


    def calc_relationship(self):
        '''
        统计人物关系权值
        '''
        print("start to process edge")
        # 遍历每一段落，笛卡尔积形式，统计人物关系
        for p in self._person_per_paragraph:
            for name1 in p:
                for name2 in p:
                    if name1 == name2:
                        continue
                    if self._relationships[name1].get(name2) is None:
                        self._relationships[name1][name2] = 1
                    else:
                        self._relationships[name1][name2] += 1
        return self._relationships
```

3D 空间力导向图（JS）

```
    const Graph = ForceGraph3D()
      (document.getElementById('3d-graph'))
        .jsonUrl('./data/processedData.json')
        .nodeAutoColorBy('group')
        .nodeThreeObject(node => {
          const sprite = new SpriteText(node.id);
         // make sprite background transparent
          sprite.material.depthWrite = false;
```

```
            sprite.color = node.color;
            sprite.textHeight = 8;
            return sprite;
          })
          .onNodeClick(node => {
            // Aim at node from outside it
            const distance = 300;
            const distRatio = 1 + distance /
                                  Math.hypot(node.x, node.y, node.z);

            // special case if node is in (0,0,0)
            const newPos = (node.x || node.y || node.z) ?
                                  { x: node.x * distRatio,
                                    y: node.y * distRatio,
                                    z: node.z * distRatio }
                                : { x: 0, y: 0, z: distance };

            Graph.cameraPosition(
               newPos, // new position
               node, // lookAt ({ x, y, z })
               1000  // ms transition duration
            );
          })
          ;

      // Spread nodes a little wider
      Graph.d3Force('charge').strength(-120);
```

6. 运行环境配置及测试

项目用电脑搭载 Intel i5-10210U 处理器，NVIDIA MX350 集成显卡，操作系统为 Windows64 位。

使用 Visual Studio Code 进行前端代码的编辑；同样对于 Python 代码也同样使用 Visual Studio Code 进行编辑，Python 版本为 3.9 版本，对于实验项目中涉及到的包文件，均采用最新版本。

使用 Chrome 浏览器呈现网页界面，版本号为 100.0.4896.88；项目导入的 JS 框架除本地导入之外，额外导入的 js 来源于 fastly.jsdelivr.net 作为 CDN 框架。

由于网页编程的 debug 功能相较于 Python、Java 语言较弱，所以只能通过网页 console 的方法进行不断地运行查看。同时相较于高级语言，网页编程会遇到同步异步的问题，所以有时为了数据的同步加载，需要牺牲一部分的加载时间。此外，在 Echarts 官网中，可以实现 Echarts 模型的实时可视化修改，对于可视化图表的设计提供了清晰可见的测试能力。

截止至提交本报告为止，本项目所涉及的代码均可正常运行启动。

7. 课程小结

通过一学期的数据可视化课程的学习，让我对于网络编程以及可视化处理有了更深层次的了解。在这门课之前，我从未系统的学习过关于数据可视化表达的方法，之前接触到的可视化操作也仅限于柱状图表，饼图的 Execl 制作。在一个学期的学习以及完成了两次项目之后，不仅解除了更加多样化的数据可视化形式。

在制作 Kmeans 网页的项目中，第一次了解了将平常所使用的代码与日常所接触到的网页进行关联的方法，制作了交互式的界面。这也是我第一次接触到网页编程，对于 Html，Css，Js 所创建的网页前端框架有了初步的认识。

在期末项目中，通过 Python 方法，将文本处理为了关键词的统计，人物的统计，人物关系之间的链接强度。并且能够通过词云，力导向图将这些单纯的数字可以直观地表达出来，能够清晰地展现出在字里行间，作者留下的痕迹。

除了有些可以直接获取的数据，还有许多数据是不能够直接被我们所观察到的，这些数据往往也能展示许多趋势与暗含的信息。为了读取散落在网络之上的信息，可以合法合规使用爬虫，在庞杂的网络之中找到所需要的关键信息。

通过以上三次课程实践，我对于数据可视化方向有了一定的了解。此外，在老师的授课过程中，我也了解到了许多更加深奥的可视化领域，例如高维数据可视化中，如何将不同维度的数据映射到可以直观展现的图形图表中；许多更加复杂的可视化，例如地理空间可视化，利用地球作为图表，直观的展现洋流与季风，城市扩展与森林缩减的变化；亦或者是以及了解的领域的进一步探索，例如文本可视化中，也会涉及到使用自然语言处理，梳理出文章的文本流等等。

此外，数据可视化所学到的知识也可以和兴趣很好的结合。在第一次项目之后，我将课程中学到的方法运用在了游戏文本处理之中。在最近火热的《艾尔登法环》的游戏文本中，也隐藏着许多的世界观的隐藏剧情，通过文本摘取和角色统计的方式，对于游戏中的关键词以及主线剧情也有了更加直观清晰的了解，由于是独立制作，并且并没有现成框架可以使用，所以最终呈现的效果并不理想。但是这帮助我在第二次课程项目中选择文本可视化，对于其他的文本能够有更多效果更好，结构更清晰的展现。

课程结束之后，我对于数据可视化有了更多的了解，同时又发现了更多的未知。数据可

视化这门课程，相对于一些计算机科学系的其他专业课程，更加偏向于实际运用。在课程之外，但凡涉及到“数据”相关的领域，可视化方法都有其一席之地。并且可视化领域也不存在唯一的正确可视化方法，随着数据、场合、时间、目的不同，可视化方法都需要灵活变更。因而这就决定了数据可视化领域需要更多的实践，在课下也需要多尝试接触更多的方法，去主动了解更多的可视化研究。