数据挖掘和数据融合
——大数据视角下的东野圭吾

庄周 21610152 & 李奕达 21632103

一、摘要

东野圭吾是日本著名的推理小说家，与 1985 出道三十四年以来笔耕不辍，创作了 90 余本小说，其中不乏脍炙人口的精品。本次研究性学习，我们以东野圭吾小说为研究对象，利用自然语言处理工具与数据挖掘相关的算法，处理东野圭吾的作品集数据，探索他写作风格、人设变化、剧情安排变化，并利用爬虫技术在互联网上获取东野圭吾相关的资料，并对其进行分析。

二、概述

图 1 东野圭吾数据挖掘框架图为本次数据挖掘的技术路线图。本次针对东野圭吾小说文本挖掘的技术路线主要分为两个层面，一个层面为对小说文本的挖掘，主要分为先验知识采集、文本预处理、粗粒度文本分析、细粒度文本分析。采用百度自然语言处理接口与结巴分词接口，对原文本进行词频、句频、文本情感走向和全文相似度分析。进一步分析小说中常见的犯罪手法与犯罪动机。并推断东野圭吾本人的创作风格与特点。

另一个层面为针对东野圭吾的周边信息进行挖掘，包括利用爬虫技术获取知乎与豆瓣上的东野圭吾小说的书评和影视作品评价以及百度、百度百科、百度指数、谷歌和必应上的相关作品的搜索情况。分析如东野圭吾小说的排名、与东野圭吾相关的作家、与东野圭吾相

关的高票问题、文本的相关翻译者问题、小说的推理风格和相关的 IP 作品等信息。

最终上述挖掘的成果将采用文字、可视化的图表与词云的形式展现出来，并形成对东野圭吾小说全貌的概述与预测。

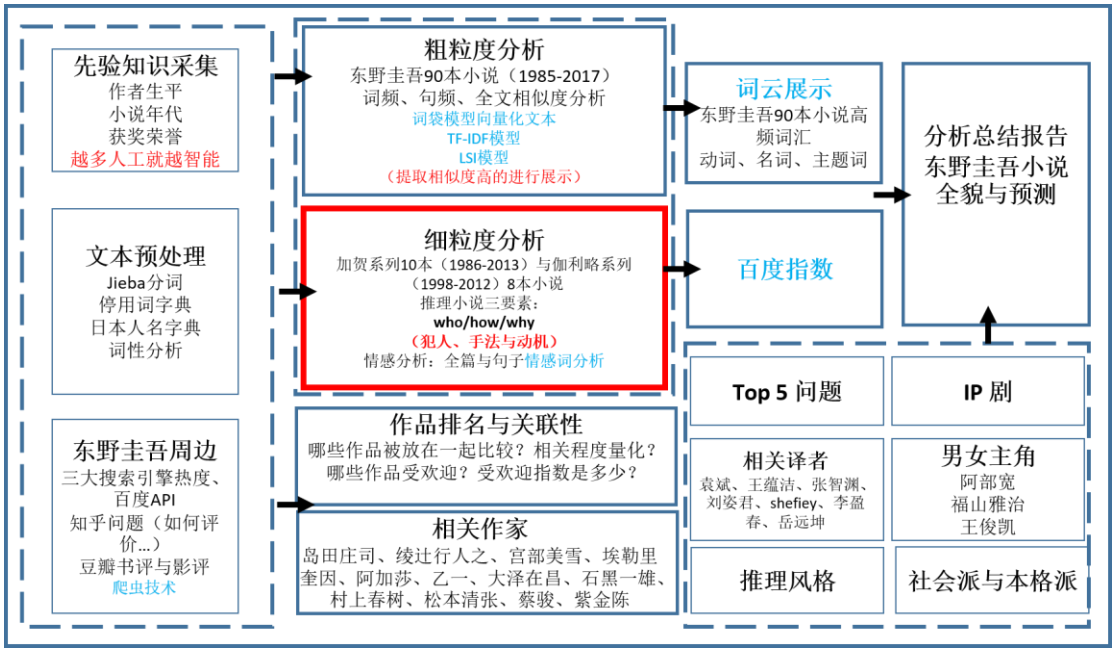


图 1 东野圭吾数据挖掘框架图

图 2 东野圭吾小说年表 展示了东野圭吾创作的作品年表，平均一年的作品数为 3 本，但是在 1997 年未产出一本作品，这一年东野圭吾与相守十四载的妻子离婚,婚姻的不幸显然影响了东野圭吾之后的创作风格，出现了《白夜行》中的唐泽雪惠与《幻夜》中的新海冬美这样的恶女形象，并且之后的作品《秘密》、《嫌疑犯 X 的献身》以及《红手指》等也都带有家庭不幸的色彩。出乎东野圭吾意料之外的是，这样地题材却受到了读者的追捧，东野圭吾的创作进入到了巅峰期，大量的影视作品也跟风改编东野圭吾这一时期的小说，其中备受人们关注的就是《白夜行》与《嫌疑犯 X 的献身》。

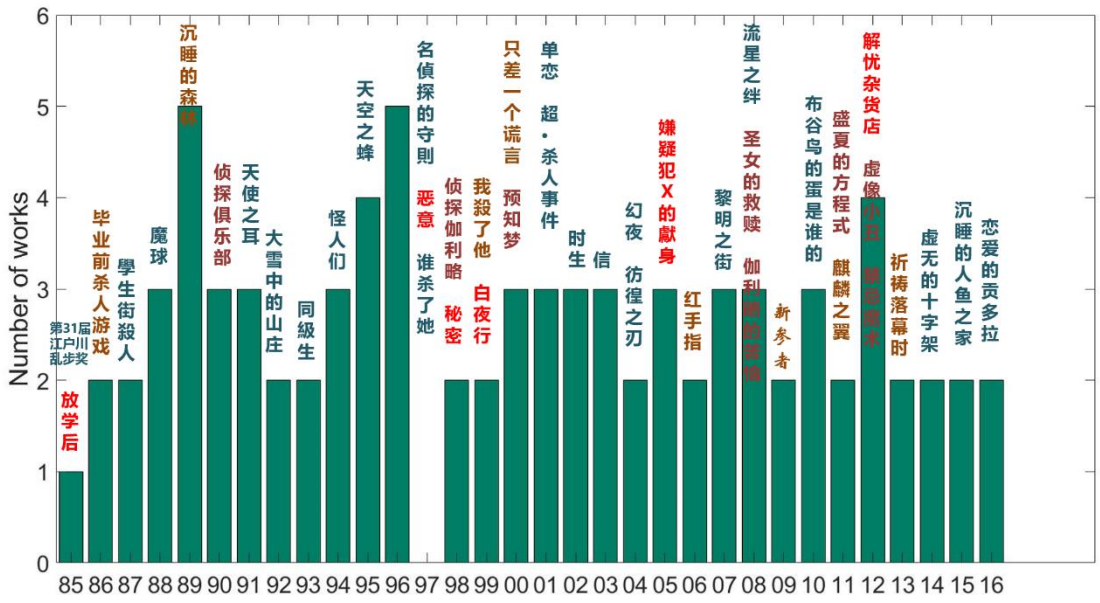


图 2 东野圭吾小说年表

三、文本预处理

在本次数据挖掘大作业中，所得到的东野圭吾系列作品的原始文本内容为.txt 格式的文件，文本内容语言为中文，编码格式为 Unicode 格式。为得到可程序化的特征，需从原始文本中提取字符串信息并进行量化操作，即文本预处理步骤。文本预处理工作通常包括以下几个步骤，去除停用词，文本分词以及词频统计。

3.1 分词工具

常用的中文分词工具有 Ansj、Paoding、盘古分词等多种，而最基础的分词器应该属于 jieba 分词器。近一段时间百度开放了其 AI 框架接口，其中包含了自然语言处理工具 Baidu-AipNlp，也提供了分词功能。各平台对比如下:

分词器	P	R	F
jieba	0.876	0.843	0.859
httpcws	0.903	0.896	0.899
盘古分词	0.921	0.906	0.913
IKAnalyzer	0.921	0.917	0.919
Ansj	0.931	0.912	0.921
imdict-chinese-analyzer	0.941	0.932	0.936
mmseg4j	0.954	0.932	0.943
Paoding	0.961	0.947	0.954

图 3 各分词平台对比

在本次大作业中，出于对上手效率、语义分割准确度以及分词工具其他功能接口的考虑，我们小组同时采用了 jieba 分词和 Baidu-AipNlp 两种分词工具。其中 jieba 分词支持人工导入词典，可以针对日本人名进行处理，同时也可以人工丰富停用词词典，进行更准确的分词。同时因为 Baidu-AipNlp 平台具有大量的文本分割调用经验，其分割结果准确度较高，可以作为 jieba 的验证和参考结果。同时 Baidu-AipNlp 包含了较高级 NLP 的函数接口，方便后续操作。

3.2 人名词典与停用词词典

在东野圭吾小说中，人物角色的设定和作者本人国籍一样多是日本人，其名字的汉语翻译和中国人的名字存在一定出入，比如长度、姓氏等存在较大区别，因此需要对日本人的人名进行特殊处理，即通过 jieba 分词导入日本人名词典。该词典为搜集到的日本人名词典文档（jpname.txt）。在小说中，人物名字出现的情况通常为对话中称呼、介绍等等，可能会出现提及某人时只称呼姓、只称呼名以及出现全名的情况。为保证区分人名的准确度（不将人物全名分割）需对人名词典进行补充，补充内容为文本中出现的人物全名。出于整理效率，首先对文本进行初步分词，提取出文本中出现的人物姓和名，然后将提取出的字符串进行排列组合，选取所有可能的姓名组合拼接为新字

符串来构建补充人物名称词典，可达到完全正确区分的效果。

预处理中需去除文本中的停用词，来保证粉刺的准确度，删除无关内容。该停用词词典采用某研究机构整理的中文停用词词典，经验证有较好的区分效果。

该部分人名词典以及停用词词典详见附件。

3.3 文本分词与词频统计

在完善词典后导入词典，并进行文本的去停用词和分词操作。该部分的分词选用 jieba 分词的精确模式，对分词效果不好的地方采用搜索引擎模式。在分词的过程中同时对词频进行统计，即每个词在文本中出现的次数。该部分通过构建词典存取，方便调用及排序。为了验证分词结果，在该部分也采用 Baidu-AipNlp 分词作为验证。

四、爬虫抓取

我们采用爬虫技术对知乎、豆瓣等网站的东野圭吾相关信息进行获取，爬虫代码见附录所示。

在爬取豆瓣的书评信息时，我们发现很多长评论都对作品的内容进行了复述，评论者的观点并不鲜明，为了避免干扰，去除冗余信息，我们仅仅选取了豆瓣中的短评，从而获得各个书籍的评论。

在爬取到知乎的信息之后，我们将内容按照单个作品评价、翻译者、女主角、文化背景、线索细节、相关作家、写作技巧、IP 剧、风格、男主角、作品排名、作者本人评价和作品关联性等主题进行分类。

知乎与豆瓣的爬取结果的例子如下图 4-7 所示。

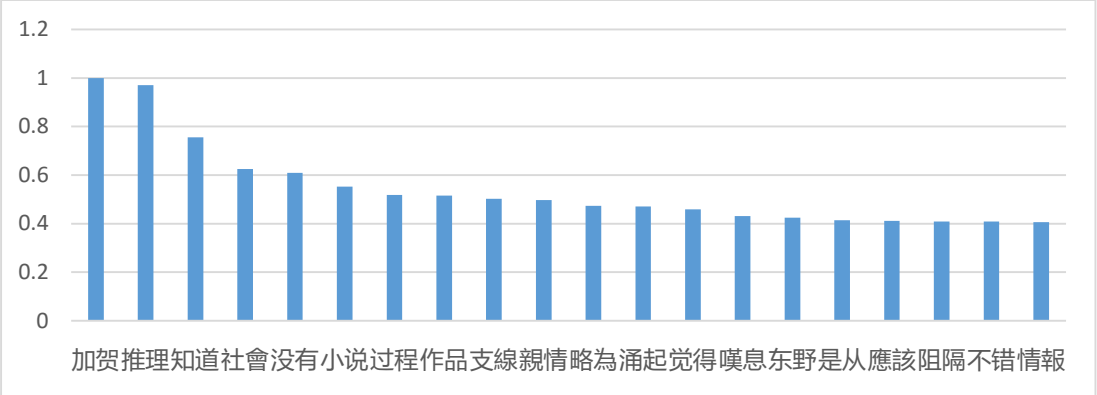


图 4 祈祷落幕时豆瓣评论关键词

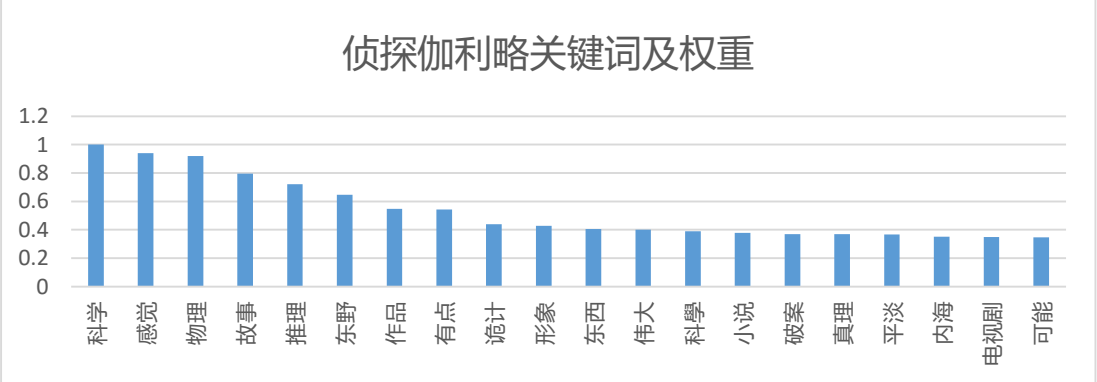


图 5 侦探伽利略关键词及权重

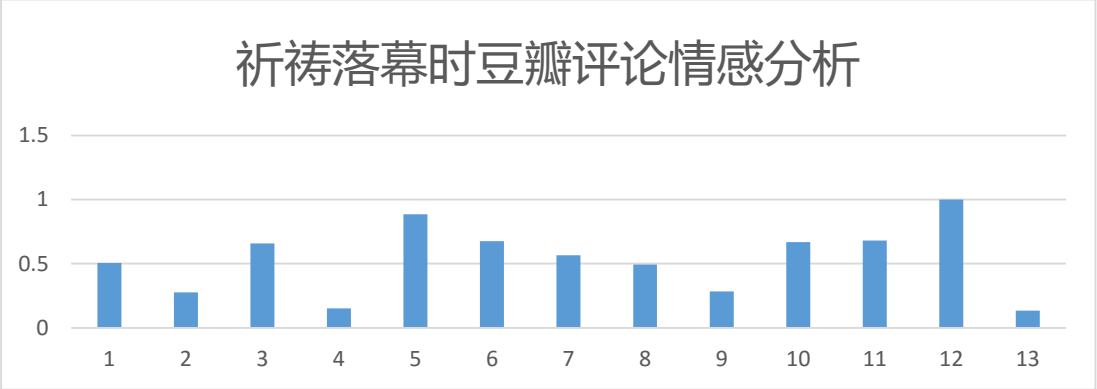


图 6 祈祷落幕时豆瓣评论情感分析

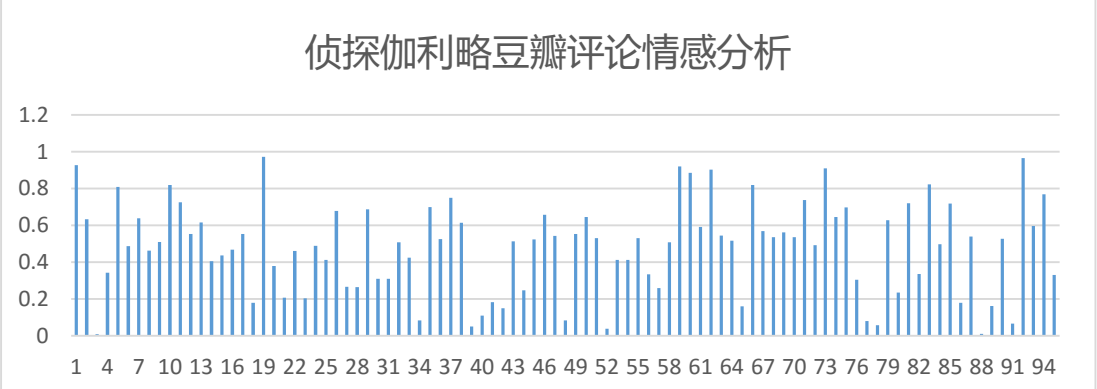


图 7 侦探伽利略豆瓣评论情感分析

为了研究东野圭吾作品的热度与排名，以及人们对于东野圭吾小说关注的热点，我们对东野圭吾小说在三大搜索引擎和百度百科上的搜索次数进行了统计与可视化处理。其中解忧杂货铺、白夜行、麒麟之翼、嫌疑犯 X 的献身和加贺系列小说是搜索量排名前 5 的作品，这些可以归功于以上述五本小说改编的影视作品的广泛流传。

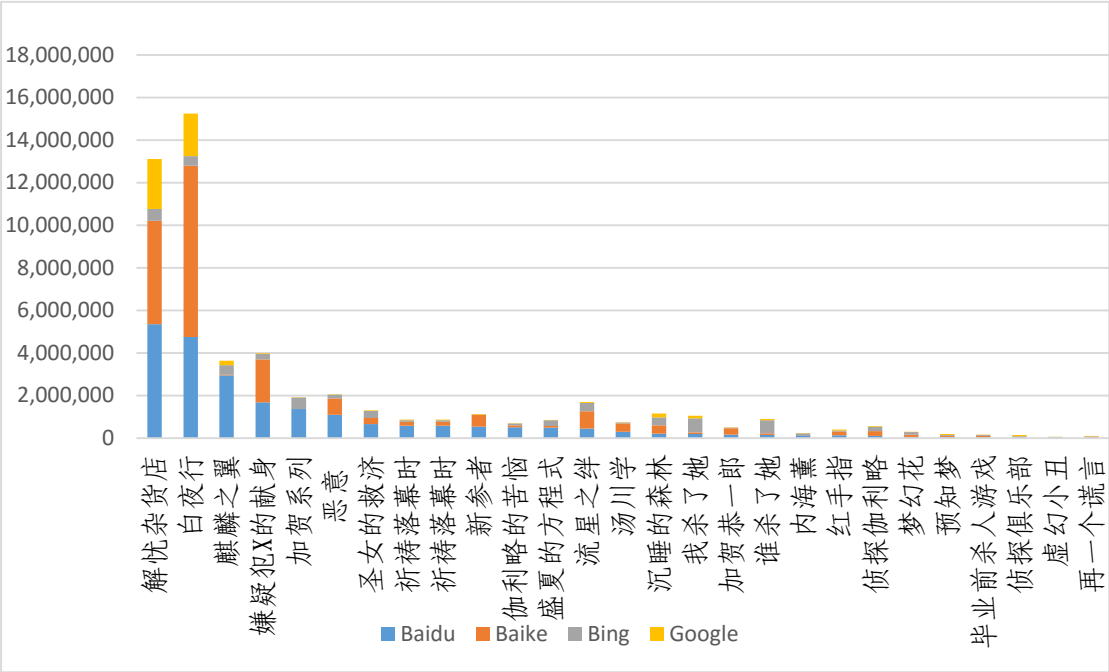


图 8 东野圭吾小说搜索热度展示

同时，如图 9 关注东野圭吾小说的人群分布和图 10 东野圭吾相关搜索趋势 我们利用百度指数，对关注东野圭吾作品的人群和关注对象进行了描述，我们发现对东野圭吾作品进行搜索的人群 30 到 39 岁最多，针对东野圭吾作品最多的搜索是文本和影视资源的资源。同时，我们发现一般小说存在改编的影视作品的情况下，会受到人们的广泛的关注，并且一般东野圭吾小说的女主角都是年轻貌美的人设，从影视作品的女主角的选择，如图 11 东野圭吾 IP 剧明星 也可以得出东野圭吾小说角色设定的偏好。

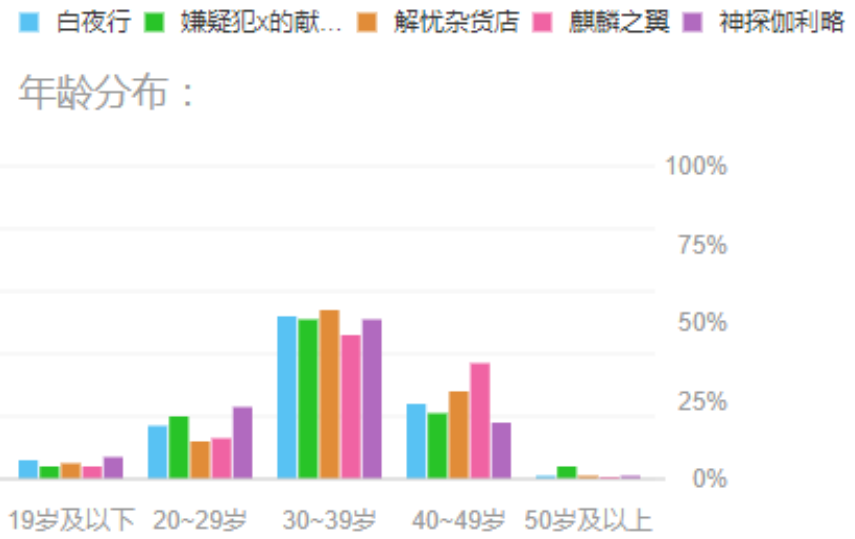


图 9 关注东野圭吾小说的人群分布

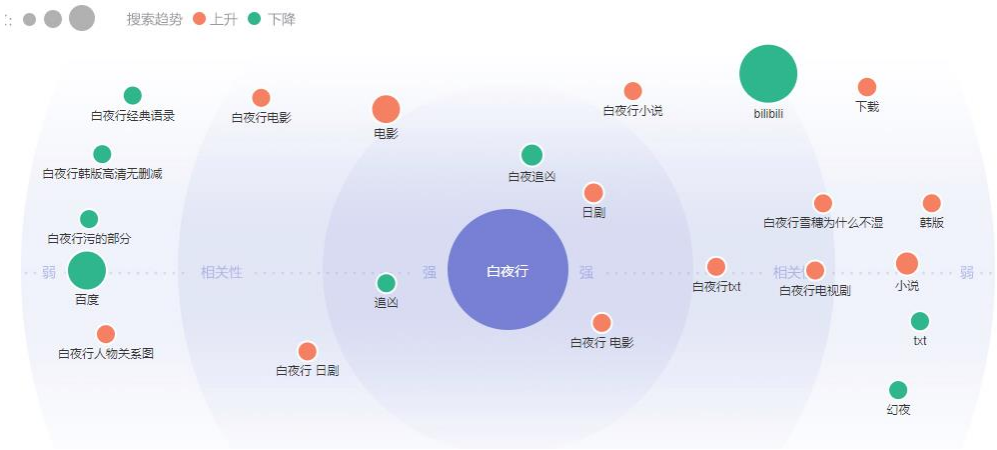


图 10 东野圭吾相关搜索趋势

图 11 东野圭吾 IP 剧明星

读者关注的东野圭吾小说 5 大问题。

1. 小说《白夜行》里的唐泽雪穗是否深爱桐原亮司？
2. 如何评价东野圭吾？
3. 如何评价中国版的电影《嫌疑人 X 的献身》？
4. 东野圭吾的哪几本小说值得优先选读？
5. 如何评价东野圭吾的《恶意》？

五、粗粒度文本挖掘

在粗粒度文本挖掘中，我们主要对东野圭吾的 90 本小说的词频、句频、全文相似度进行了分析。下面主要讲述了如何对文本的相似度进行分析。

在计算中文文本的相似度的时候，需要将中文文本转化为向量，通过计算向量之间的相似度来比较文本的相似度，其中向量的相似度可以采用欧氏距离、余弦相似度等方法。图 12 文本相似度比较流程图展示了文本相似度比较的流程。

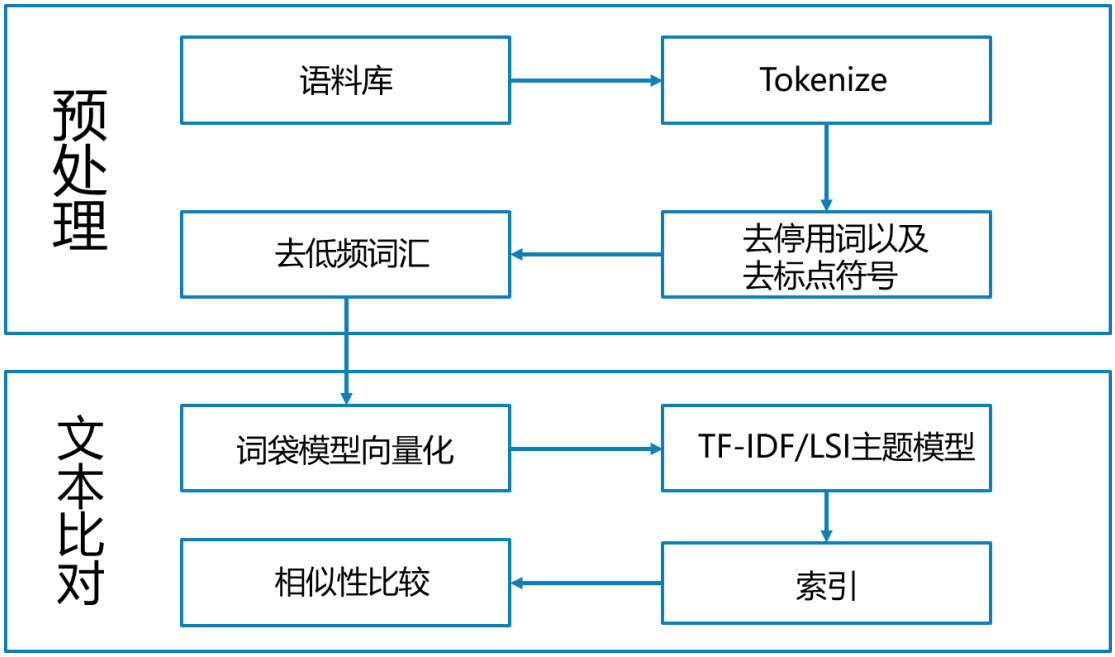


图 12 文本相似度比较流程图

预处理步骤与之前类似不做赘述，这里如何将文本表示成为向量，

最简单的方式就是建立词袋模型，将一篇文本当做是由一个个词汇构成，所有词放入一个袋子中，没有先后的顺序，没有语义。

词袋模型简单易懂，但是存在问题。中文文本里最常见的词是“的”、“是”、“有”这样的没有实际含义的词。一篇关于足球的中文文本，“的”出现的数量肯定多于“足球”。所以，要对文本中出现的词赋予权重。

一个词的权重由 TF * IDF 表示，其中 TF 表示词频，即一个词在这篇文本中出现的频率；IDF 表示逆文档频率，即一个词在所有文本中出现的频率倒数。因此，一个词在某文本中出现的越多，在其他文本中出现的越少，则这个词能很好地反映这篇文本的内容，权重就越大。

回过头看词袋模型，只考虑了文本的词频，而 TF-IDF 模型则包含了词的权重，更加准确。文本向量与词袋模型中的维数相同，只是每个词的对应分量值换成了该词的 TF-IDF 值。

TF-IDF 模型足够胜任普通的文本分析任务，用 TF-IDF 模型计算文本相似度已经比较靠谱了，但是细究的话还存在不足之处。实际的中文文本，用 TF-IDF 表示的向量维数可能是几百、几千，不易分析计算。此外，一些文本的主题或者说中心思想，并不能很好地通过文本中的词来表示，能真正概括这篇文本内容的词可能没有直接出现在文本中。

因此，这里引入了 Latent Semantic Indexing（LSI）从文本潜在的主题来进行分析。LSI 是概率主题模型的一种，另一种常见的是 LDA，核心思想是：每篇文本中有多个概率分布不同的主题；每个主题中都包含所有已知词，但是这些词在不同主题中的概率分布不同。LSI 通过奇异值分解的方法计算出文本中各个主题的概率分布，严格的数学证明需要看相关论文。假设有 5 个主题，那么通过 LSI 模型，文本向量就可以降到 5 维，每个分量表示对应主题的权重。

在本次研究中我们采用了 TF-IDF 和 LSI 模型对于东野圭吾小说的相似性进行分析，成功将同一系列的小说进行了归类。

六、细粒度文本挖掘

6.1 词性划分

Jieba 分词中包含了 jieba.analyse 包，该部分提供了词性分析接口，通过该部分可对分词后的结果进行词性划分，以方便根据词性对文本内容进行区分。同时构建数据结构，根据词性进行存储，方便调用。同时将各词构建成词向量，对不同文本进行相似度比较。

对单句中出现过两个及以上人物的句子单独提取出来，再抽取其中的人物以及谓语动词，重新按照顺序拼接成短句以提取关键信息，并梳理人物之间的关系。同时将这部分内容存储成字典的格式，key 为谓语动词，检索文本中是否出现敏感性动词，比如：“杀”、“死”、“打”等可明显指代嫌疑人与被害者身份的句子，并将此存储起来，作为推理结果。

事情(n)发生(v)四月(m)十六日(m)星期二(t)

那天(r)下午(t)三点(m)家里(s)出发(v)前往(t)日高邦彦(nr)住处(n)
日(m)高家(n)距离(n)住(v)地方(n)隔(v)一站(m)电车(n)路程(n)到达(v)车站(n)改搭(v)巴士(ns)走上(v)一小(d)段(q)路(n)时间(n)二十分钟(m)

平常(a)没什么(l)事常(d)到(v)日(m)高家(n)走走(v)那天(r)却是(d)特别(d)事(n)要(v)办(v)
说好(v)错过(v)那天(r)再也(d)不到(v)

家(q)座落在(i)美丽(ns)整齐(d)住宅区(n)里(f)区内(s)清一色(i)高级(b)住宅(n)称之为(v)豪宅(n)气派(n)房子(n)
杂树林(n)住家(n)依然(d)庭院(n)里(f)招(v)原本(n)林木(nr)
围墙(n)山毛榉(nr)砾树(n)长(a)茂盛(a)浓密(a)树荫(n)覆满(v)整条(m)巷道(n)里(f)

说(v)路(n)狭窄(a)一律(d)规划(n)成(n)单行道(n)
讲究(vn)行走(v)身分(n)地位(n)一种(m)表徵(n)

几年(m)前(f)听到(v)日高(nr)买房子(n)时想果(n)不出所料(i)
地区(n)长大(ns)少年(m)家(m)买(v)人生(n)梦想(n)

日(m)高家(n)称不上(v)豪宅(n)夫妻俩(n)住(v)说(v)绰绰有余(i)宽敞(a)
主(n)屋(n)采(zg)屋顶(n)形式(n)虽(zg)是(v)日本(ns)风边(n)窗(n)拱型(n)玄关(ns)二楼(n)窗际(n)花坛(n)全(a)是(v)西式(n)设计(vn)
想必(d)夫妻俩(n)一半(m)主意(n)
砖造(n)围墙(n)夫人(n)占上风(nr)
透露(v)想(v)住(v)欧洲(ns)古堡(ns)家里(s)

更正(d)夫人(n)说(v)前(f)夫人(n)

砖造(n)围墙(n)走(v)终于(d)来到(v)方形(n)红砖(n)砌(zg)门前(s)按(p)下(f)门铃(n)

久(a)没人来(l)应门(v)我(r)往(p)停车场(n)一(m)看(v)日高(nr)SAAB(eng)车(n)出门(v)

这(r)下(v)打发(v)时间(n)
想起(v)那(r)株(q)樱花(n)
日高(nr)家庭(n)院里(s)一株(m)八重(m)樱(nr)上次(t)三分(m)开(v)算算(v)十天(m)不知(v)

家(m)仗(n)主人(n)朋友(n)份(q)请(v)自入(v)

图 13 词性划分

6.2 关键词提取

在分析的过程中，文本中的一些关键词可以表达出关键性的信息，比如人物间的关系、重要的时间、案发现场和作案手法等等。利用 jieba 分词通过构建 TextRank 模型来对文本中词汇的权重进行划分，并且可以根据词性分别进行统计。该权值划分可支持对单句和长文本的权值统计，其权值划分原理类似于 PageRank 的思想，将文本中的语法单元视作图中的节点，如果两个语法单元存在一定语法关系（例如共现），则这两个语法单元在图中就会有一条边相互连接，通过一定的迭代次数，最终不同的节点会有不同的权重，权重高的语法单元可以作为关键词。节点的权重不仅依赖于它的入度结点，还依赖于这些入度结点的权重，入度结点越多，入度结点的权重越大，说明这个结点的权重越高；TextRank 迭代计算公式为：

$$WS(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} * WS(V_j)$$

节点 i 的权重取决于节点 i 的邻居节点中 i-j 这条边的权重除以 j 的所有出度的边的权重*节点 j 的权重，将这些邻居节点计算的权重相加，再乘上一定的阻尼系数，就是节点 i 的权重；阻尼系数 d 一般取 0.85。

在本次数据挖掘中，设置统计词性为‘ns’、‘n’、‘v’、‘vn’、‘vd’、‘a’、‘nt’、‘nr’、‘t’的词的权值，并分别统计出全文中各词性中权值前 20 的词语和单句中权值前 5 的词语，排序后保存输出到.csv 文件中，方便统计和结果处理。

6.3 句式统计

在东野圭吾的创作历程上，是否存在文风的变化进行探究。文风的体现之一在于句式结构的统一性，这表现了作者的创作习惯。在本次数据挖掘中，对东野圭吾的加贺系列和伽利略系列的文本中几种句

式结构进行统计，并将统计结果保存在.csv 格式文件中。

在本次探究中，共统计主动句式、被动句式两种类别，以及统计陈述句、疑问句和反问句三种句式。其中主动句式和被动句式统计结果相加后与全文总句子数接近，而陈述句、疑问句和反问句三种句式统计结果也和全文总句子数接近。并以此作为统计结果置信验证。

句式结构的区分依据关键字和标点符号进行区分。其中被动句包含“被”、“受…所…”、“为…所…”等结构，其余划分为主动句式。结尾带有“？”划分为问句和反问句，其中反问句带有“难道”、“怎么”、“怎能”等关键字词。统计结果人工验证后可用于下一步分析。

为了观察两个系列中文风的变化，将每本书的五种句式结构占比视作特征值，通过 Kmeans 的方法进行聚类分析，观察其正确率。如果正确率较高，则说明两个系列在文风上存在一定区别。

6.4 句子情感分析

Baidu-AipNlp 提供了对文本情感倾向进行分析的接口。针对带有主观描述的中文文本，可自动判断该文本的情感极性类别并给出相应的置信度。情感极性分为积极、消极、中性。本次数据挖掘中记录文本中每句话的 Positive Probability 和 Negative Probability，并绘制趋势，进行文本情感分析。

七、结果展示

7.1 词性词频及文本关键词分析结果

对两个系列的数进行词性划分和词频统计后进行可视化，部分结果如下:

部分名词的词频统计如下:

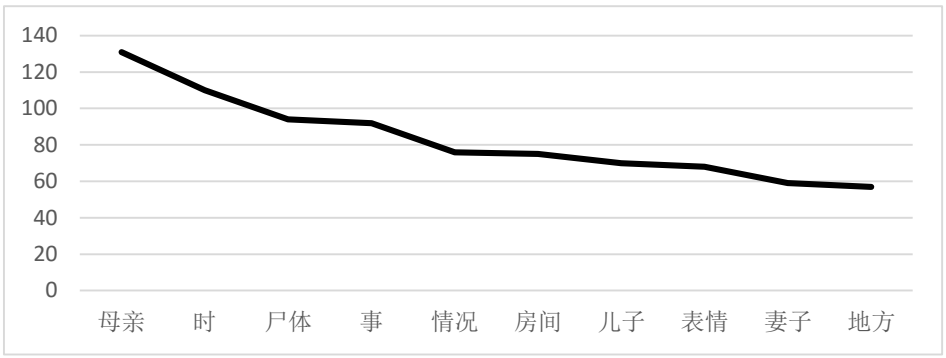


图 14《红手指》名词词频分析

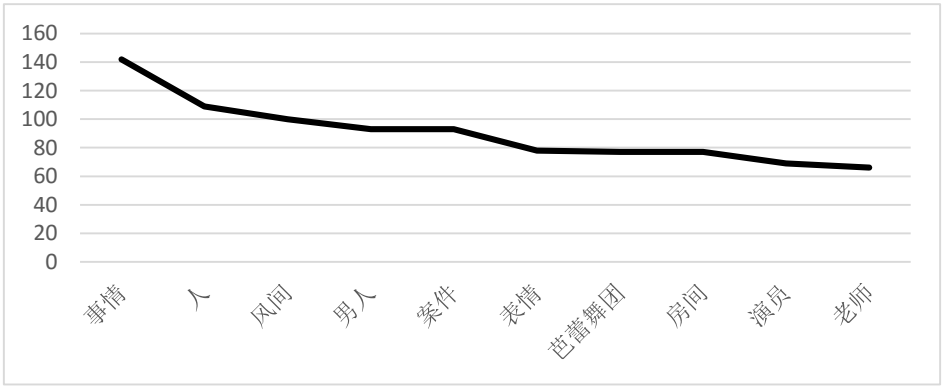


图 15 沉睡的森林名词词频统计

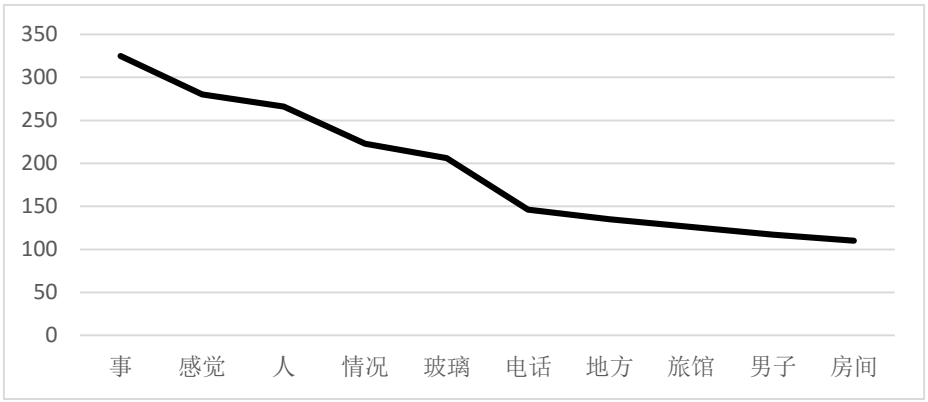


图 16 盛夏方程式名词词频统计

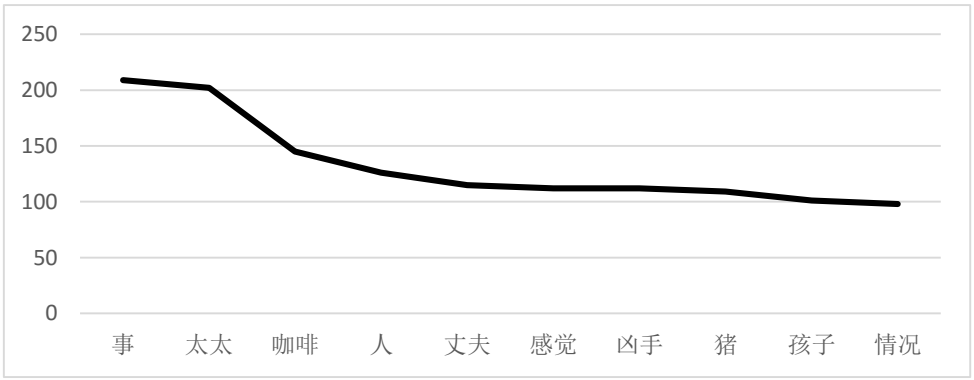


图 17 圣女的救济名词词频统计

从名词词频统计结果可初步看出，常出现的人物、物品等意向多

和文章中主题相关，而对美系列所有书进行词频统计后，可以看出常用的意象为“男人”、“妻子”、“孩子”等等，可初步断定东野圭吾的作品常常和家庭关系相关，或者人物的环境设置或者作案动机和家庭环境有较大联系。

部分形容词的词频统计如下:

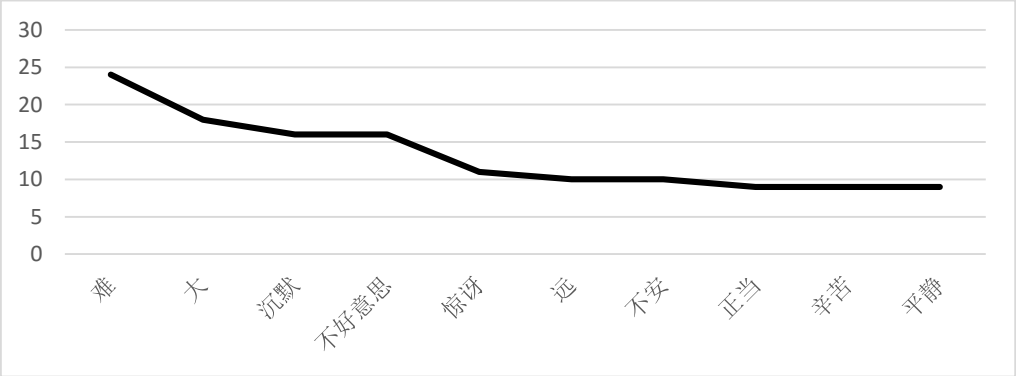


图 18 红手指形容词词频统计

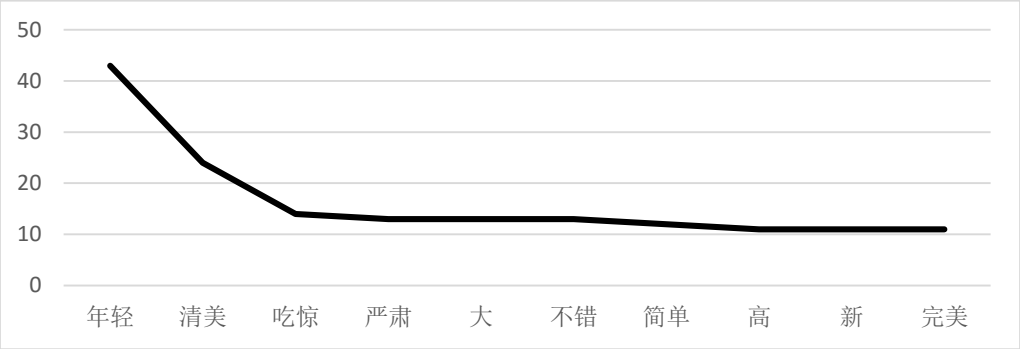


图 19 沉睡的森林形容词词频统计

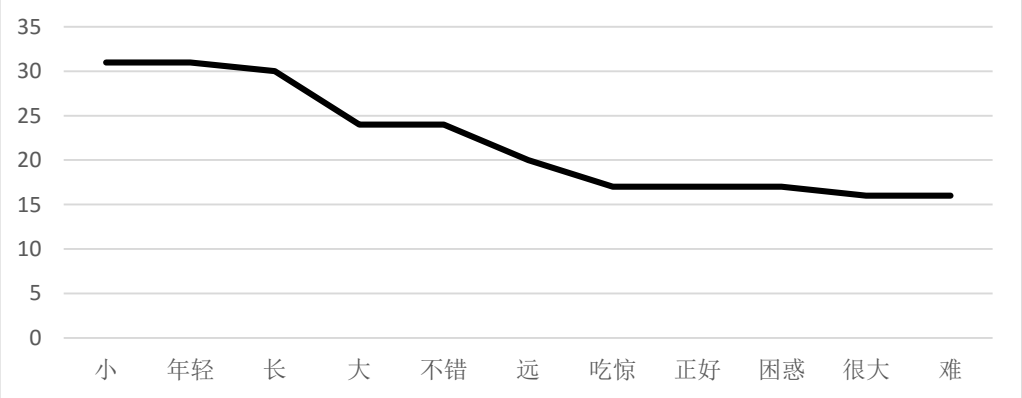


图 20 盛夏方程式形容词词频统计

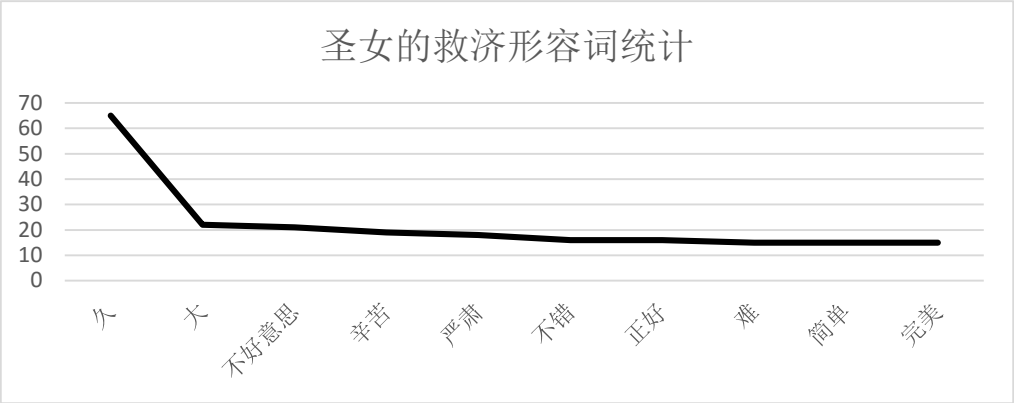


图 21 圣女的救济形容词词频统计

通过形容词词频统计结果可以看出，高频的形容词为“难”、“大”、“年轻”、“完美”等，并且多为负面情绪。可以初步断定情感偏向负面，并且出现了很多描写女性的形容词，比如“年轻”等等，可以认为东野圭吾作品中主要的女性角色，或者说是女主角多为年轻貌美的女性形象。

关键词分析部分，选取部分书籍结果展示如下:

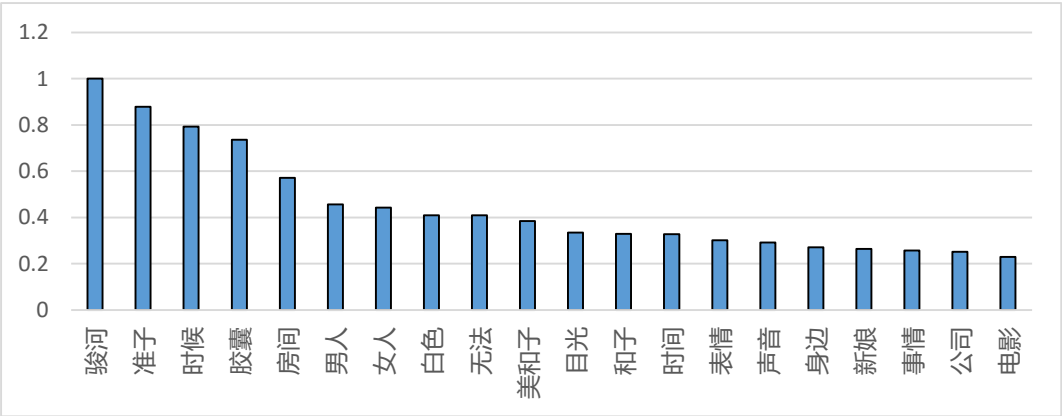


图 22 我杀了他名词权重统计

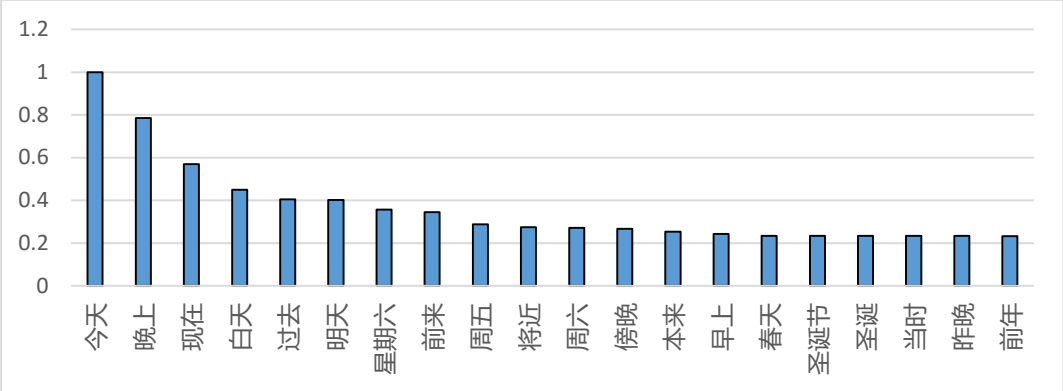


图 23 我杀了他时间副词权重统计

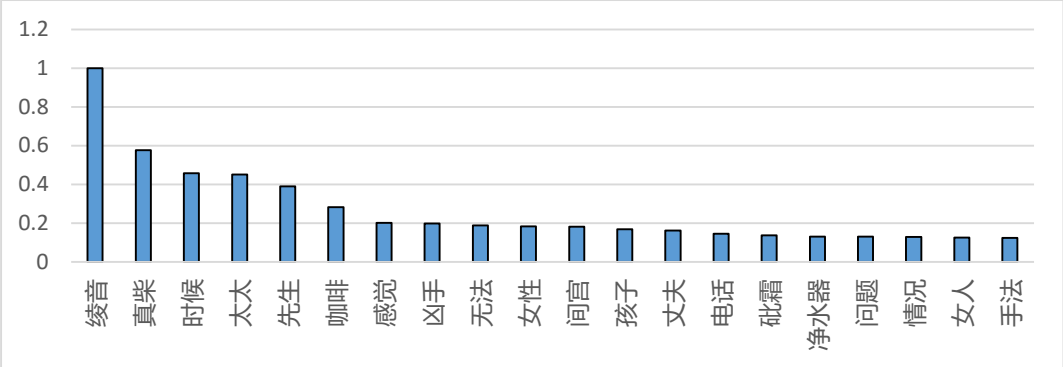


图 24 圣女的救济名词权重统计

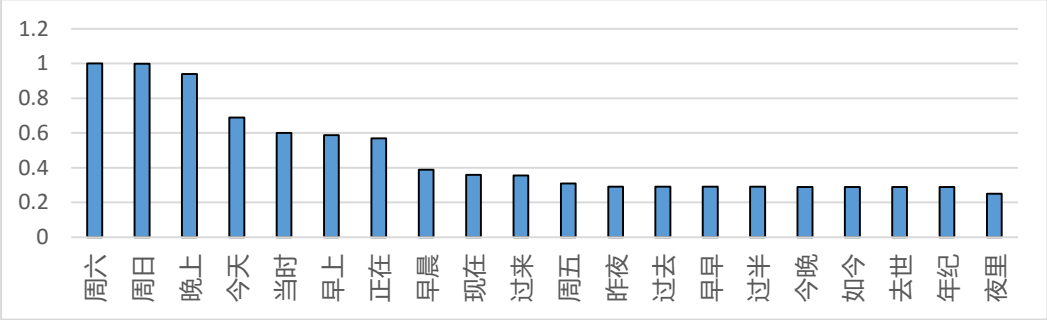


图 25 圣女的救济时间副词权重统计

可以看出，高权重词汇中包含的名词多和主题相关，比如围绕着芭蕾舞团而展开的《沉睡的森林》一书。但是该部分每本书的主题不一，未能很好的量化从而找到共性的规律。其次高权重词汇中多描述了男性和女性及其相关关系，可以看出这是东野圭吾做平中常常涉及到的内容，联系到东野圭吾自身经历（1997 年离婚），这点的影响不难理解。并且该部分也是作案动机的诱因之一。再次，在高权重名词中可以找到凶器或者关键性线索，比如可以找到关键词“砒霜”，这部

分词汇包含了东野圭吾常用的杀人手法的设计方式，并且多为下毒、利刃、钝器和枪杀，同时可以看出其手法设计上多包含较强的科学性，这也与其工程硕士出身相关，情节包含科学性并且逻辑严谨缜密。关键的场所也多为迷失、学校和滑雪场，可见其场景设置的偏好。

提取人物及动作未能得到和好的效果，只有部分作品可以直接提取出嫌疑人、北韩人以及作案手段等信息，但是可以辅助了解故事的大致梗概。因此可以初步认为人物关系铺陈比较含蓄，不直接表明关系。

因此可以得出初步推论，东野圭吾的小说设计多和家庭关系相关、多和情感纠葛相关，并且多为学校题材的故事。

7.2 句式分析结果

部分句式结构的统计结果如下：

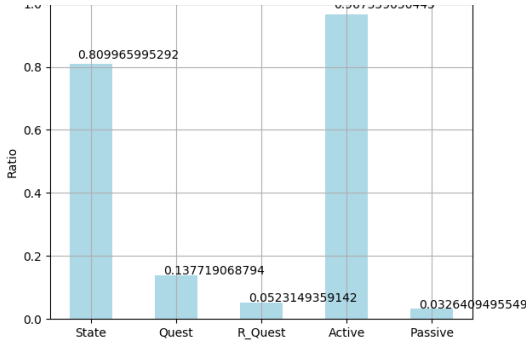


图 26 嫌疑人 X 的献身句式统计结果

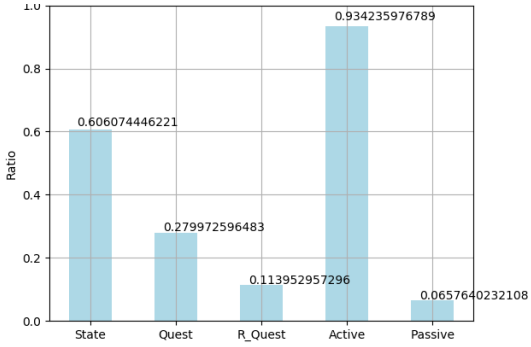
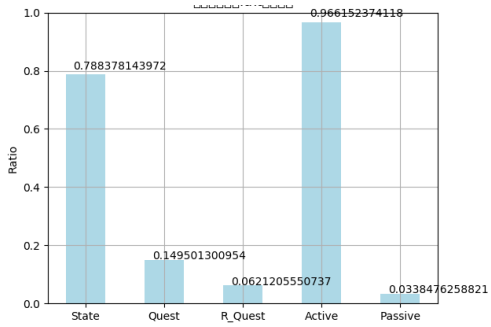


图 27 圣女的救济句式统计结果



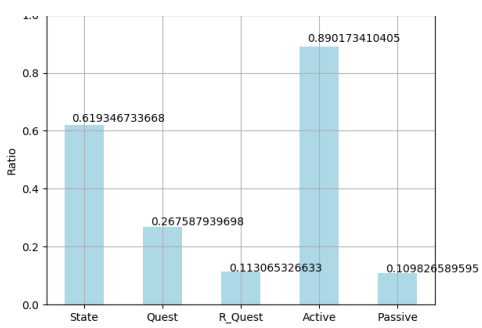


图 28 盛夏方程式句式统计结果　　图 29 虚像的小丑句式统计结果

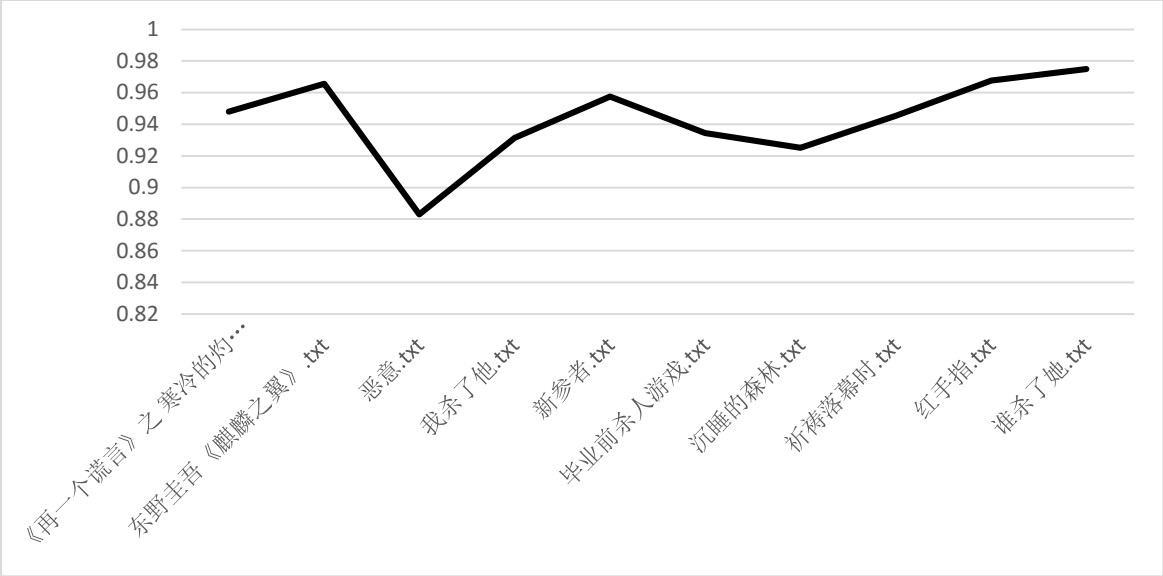


图 30 加贺系列主动句统计

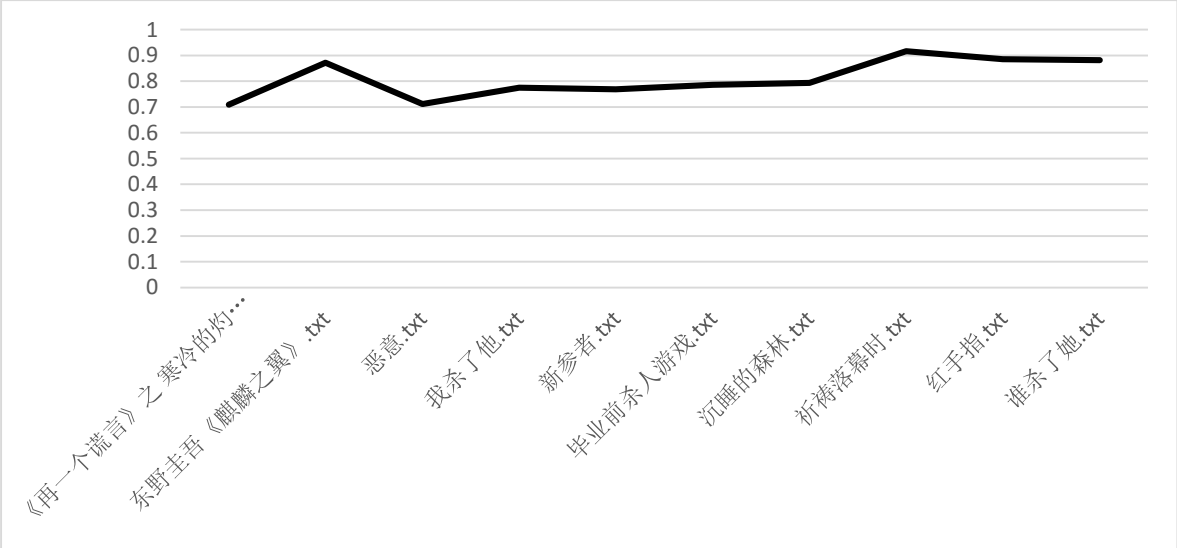


图 31 加贺系列陈述句统计

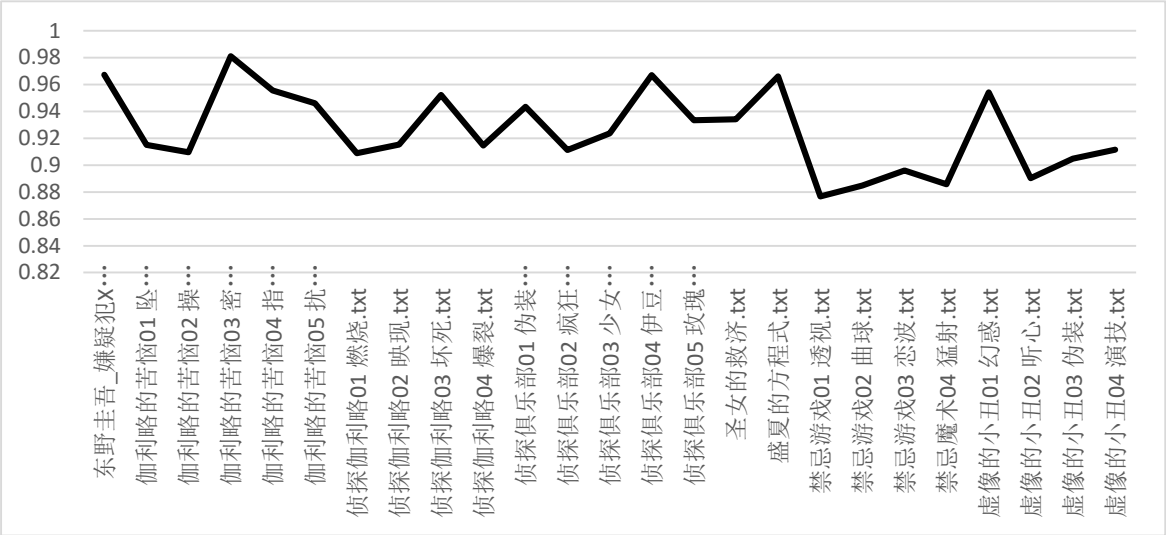


图 32 伽利略系列主动句统计

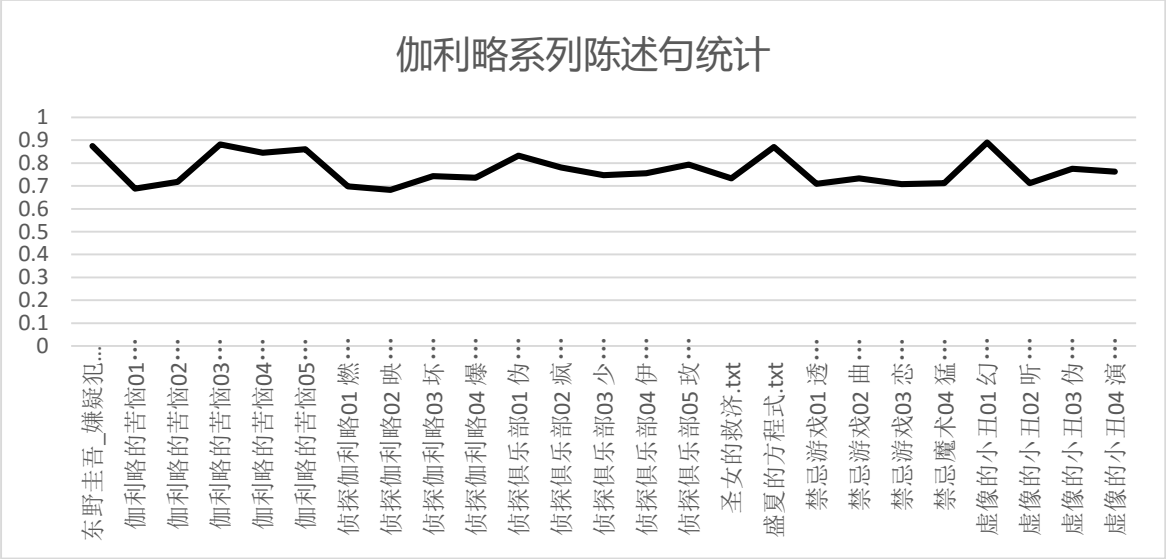


图 33 伽利略系列陈述句统计

通过句式统计结果可以看出，东野圭吾的作品中陈述句占多数，疑问句和反问句占比之和不超过 50%，但是不同系列作品中陈述句所占比例有一定差别，其中加贺系列的陈述句较高。其次主动句也占了绝大多数，但是不同系列也存在一定差别。为了验证这种差异的存在，对该部分进行聚类分析，采用的聚类方法为 Kmeans，其聚类准确率达到了 78.58%，可以说不同系列作品之间存在一定的文风区别。

7.3 情感走势分析

通过 Baidu-AipNlp 对文本情感进行分析，部分绘制的趋势图如下:

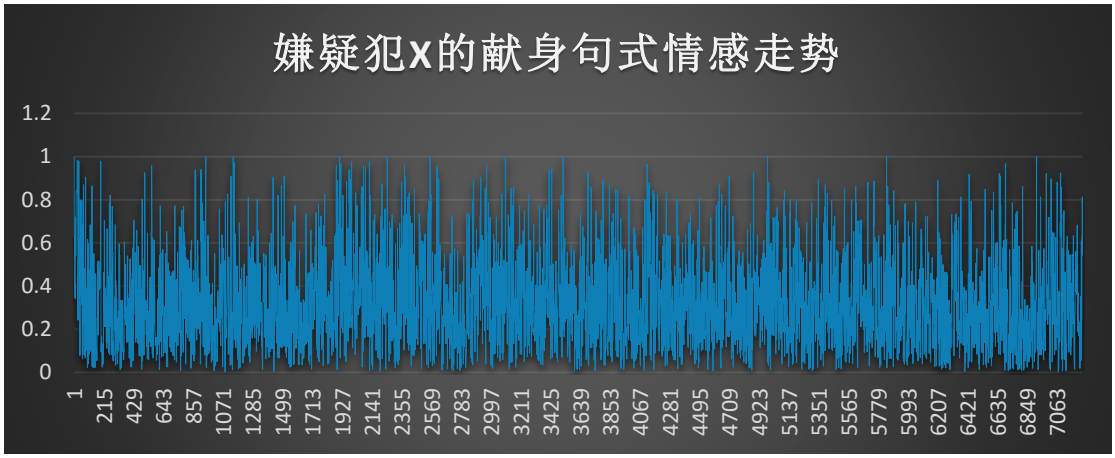


图 34 嫌疑犯 X 的献身句式情感走势分析，均值=0.326731

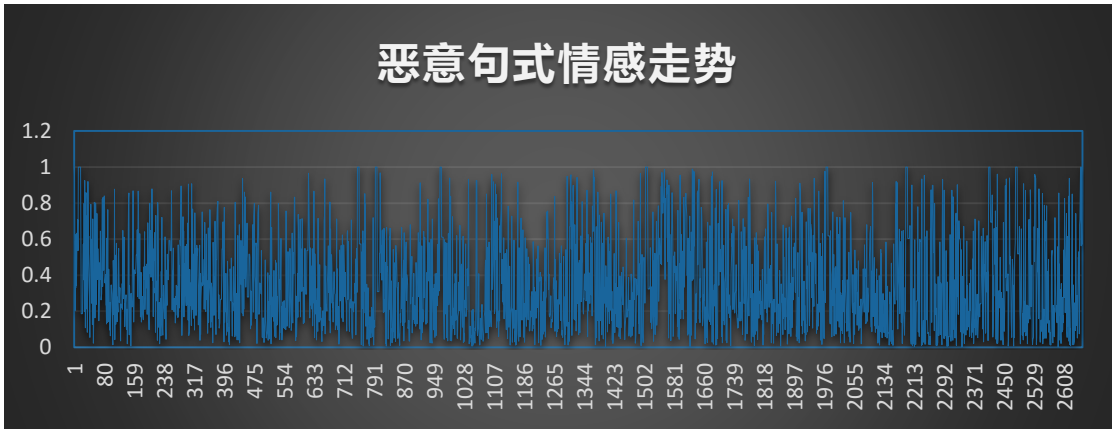


图 35 恶意句式情感走势分析，均值=0.349677

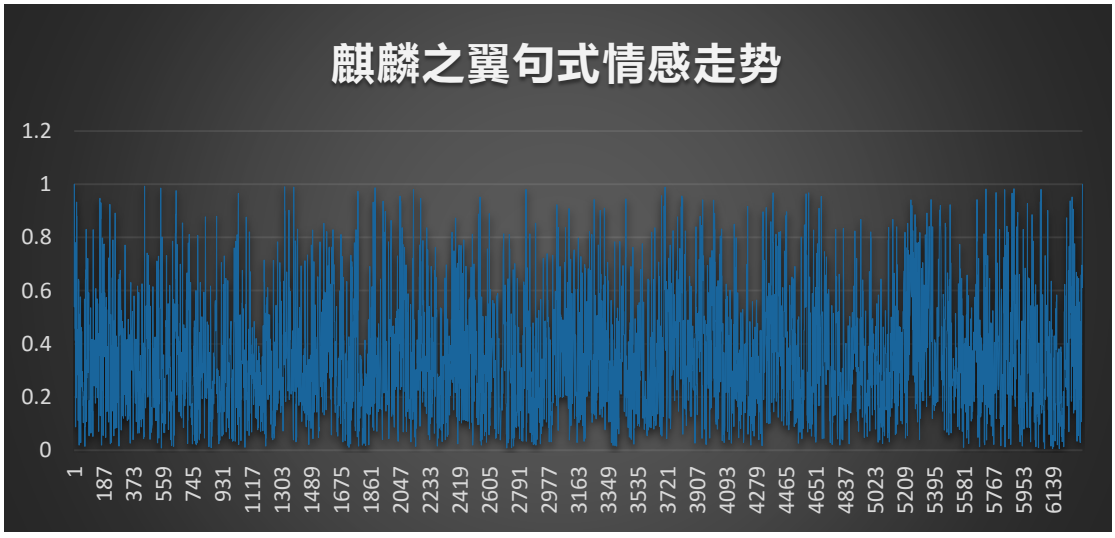


图 36 麒麟之翼句式情感分析，均值=0.355682

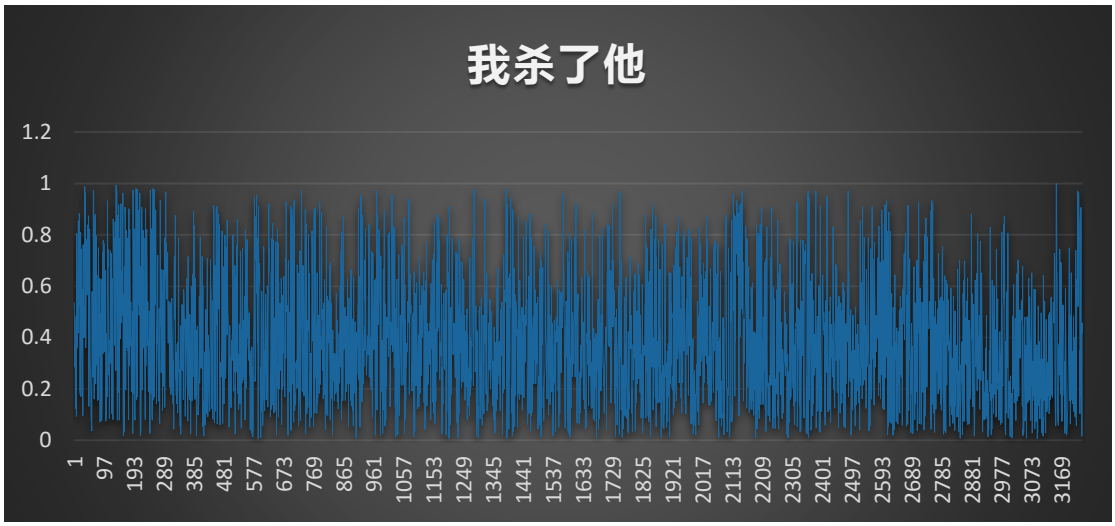


图 37 我杀了他句式情感分析，均值=0.388721

通过分析句子情感走势，得到以下结论：两个系列句式情感走势相似，没有很大的区分效果，Positive probability 均值几乎无区别，都集中在 0.3-0.4 之间，计算后，伽利略系列整体情感均值略高于加贺系列情感均值。并且文章的中后段句式情感变化十分剧烈，整体走势也出现起伏变化。

因此可以做出初步推论推理类的作品情感基调偏向负面情绪，在全书后半段情感起伏较大，应该涉及情节反转。

7.4 共性结论分析

综合上述结果，可初步得到以下结论

- 文章情感基调偏向负面情绪
- 与婚姻或男女关系相关，倾向于婚姻中发生了不幸
- 女性角色年轻貌美
- 诡计内容有较高科学性（物理、化学）
- 包含对人性的探讨

- 作者对滑雪场、学校有偏好
- 场景设置大概率为室内
- 包含支线剧情和情节反转
- 大概率包含杀人事件

图 38：东野圭吾词云-综合

图 39 东野圭吾词云-名词

图 40 东野圭吾词云-动词

图 41 东野圭吾词云-形容词

八、总结与展望

在本次数据挖掘大作业中，通过学习，快速入门了文本处理的基本方式和方法，了解到了数据挖掘的魅力。并且在这个过程得到了一些粗浅的成果，虽然未曾读过东野圭吾的著作，但是已然对其作品和

人格有了一定了解。在挖掘信息的过程中掌握了一些 NLP 的方法和模式，上手使用了一些数据分析的框架，并且拓展了思路。作为外行人士，着实产生了数据挖掘的兴趣。

但是作为非研究此领域的学生，在这个过程中也认识到了很多不足之处，比如文本的量化程度不够，缺乏进一步挖掘的数学、程序手段，所得到的结论也比较直观和粗浅，并且通过整理发现，整体化的思路不够明显，还需锻炼和加强，如果可以应该接受系统化的训练。

九、参考文献

[1]芮伟康. 基于语义的文本向量表示方法研究[D].中国科学技术大学,2017.

[2]赵妍妍,秦兵,刘挺.文本情感分析[J].软件学报,2010,21(08):1834-1848.

[3]杜朋朋. 基于改进 TF-IDF 特征提取的文本分类模型的设计与实现[D].华中科技大学,2016.

[4]牛萍. TF-IDF 与规则结合的中文关键词自动抽取研究[D].大连理工大学,2015.

[5]安子建. 基于 Scrapy 框架的网络爬虫实现与数据抓取分析[D].吉林大学,2017.

[6]百度指数爬虫 https://www.cnblogs.com/TTyb/p/6051366.html

[7]文本相似度计算 https://www.cnblogs.com/liaojiafa/p/6287314.html

[8]孙爽. 基于语义相似度的文本聚类算法的研究[D].南京航空航天大学,2007.

十、附录

爬虫代码

```
# -*- coding: utf-8 -*-
works = ['雪月花杀人游戏','谁杀了她 东野圭吾', '我杀了她 东野圭吾','虚幻小丑 东野圭吾','预知梦 东野圭吾','放学后','白马山庄杀人事件','学生街的杀人事件','十一字杀人','浪花少年侦探团','魔球',\
        '以眨眼干杯','布鲁特斯的心脏','东野圭吾《鸟人计划》','空中杀人现场','十字屋敷的小丑','假面山庄杀人事件','没有凶手的杀人夜','宿命 东野圭吾',\
         '变身 东野圭吾','回廊亭杀人事件','天使之耳','大雪中的山庄','美丽的凶器','分身 东野圭吾','同级生','怪人们 东野圭吾','过去我死去的家','造彩虹的人 东野圭吾',\
         '怪笑小说','平行世界的爱情故事','天空之蜂','我的晃荡的青春-东野圭吾','毒笑小说','恶意','名侦探的守则','名侦探的诅咒','秘密','白夜行','只差一个谎言 东野圭吾',\
         '单恋','东野圭吾_超·杀人事件','绑架游戏','湖边凶杀案','时生','家信 东野圭吾','酷酷的代课老师','杀人之门','幻夜','彷徨之刃','黑笑小说','使命与心的极限','濒死之眼','东野圭吾最后致意',\
        '黎明之街 东野圭吾','流星之绊','悖论 13','布谷鸟的蛋是谁的','雪地杀机 东野圭吾','假面饭店','解忧杂货店','禁忌魔术 东野圭吾','歪笑小说','梦幻花','假面前夜 东野圭吾','虚无的十字架','沉睡的人鱼之家',\
         '拉普拉斯的魔女','风雪追击 东野圭吾','恋爱的贡多拉 东野圭吾']
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.84 Safari/537.36',\
    'Cookie': 'd_c0="AGCChmnOxwuPTo8DCwb9s3I8GQVd2dJcaNw=|1495167334"; _zap=6eba6f3c-001f-4f30-b653-6d3b8be0f55b; q_c1=9a9e83a1796747229eae2d265ae8b7fe|1506512064000|1498009306000; \
    z_c0=Mi4xRXpRLUFBQUFBQUFBWUlLR2FjN0hDeGNBQUFCaEFsVk5rMkh0V2dCMmhSdWp4ekZwUEttX0JCbVg4cWdFcjdhYjNn|1509954451|fc684b27315a5d08a2820e1c56bd9a780772a6eb; __utmv=51854390.100-1|2=registration_date=20140308=1^3=entry_date=20140308=1;\
    q_c1=9a9e83a1796747229eae2d265ae8b7fe|1514467662000|1498009306000; _xsrf=8753454accc4f6d083e983fa23c0f0cd; __utma=51854390.337173148.1513947974.1514167661.1514543641.3; __utmz=51854390.1514543641.3.3.utmcsr=baidu|utmccn=(organic)|utmcmd=organic; \
    aliyungf_tc=AQAAAPz0qm6PDAsAyaKdtwyF+BKotAHV; _xsrf=8753454accc4f6d083e983fa23c0f0cd',\
```

```
    'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8',\
    'Accept-Language': 'zh-CN,zh;q=0.9,zh-TW;q=0.8',\
    'Cache-Control': 'max-age=0'}
para = {'t':'general','q':'解忧杂货铺','correction':'1','search_hash_id':'4f6c83f37717c6bd8a4f339f08fbac89','offset':'5','limit':'10'}
para2 = {'include':'data[*].is_normal,admin_closed_comment,reward_info,is_collapsed,annotation_action,annotation_detail,collapse_reason,is_sticky,collapsed_by,suggest_edit,comment_count,can_comment,content,editable_content,voteup_count,\
reshipment_settings,comment_permission,created_time,updated_time,review_info,question,excerpt,relationship.is_authorized,is_author,voting,is_thanked,is_nothelp,\
upvoted_followees;data[*].mark_infos[*].url;data[*].author.follower_count,badge[?(type=best_answerer)].topics',\
        'offset':'2',\
        'limit':'20',\
        'sort_by':'default'}
testDict={'解忧杂货铺里你明白什么道理？': 'https://www.zhihu.com/question/46397802', '《解忧杂货铺》的主线到底是几条呢？': 'https://www.zhihu.com/question/29138975'}
articleLogPath='G:/数据挖掘大作业/输出文档/知乎爬虫/articleLog.txt'
questionLogPath='G:/数据挖掘大作业/输出文档/知乎爬虫/question'
projectPath='G:/数据挖掘大作业/输出文档/知乎爬虫/'
def collectQ(name,url,filename,limit=20):
    with open(filename,'a') as f:
        try:
            f.write(name+'\r\n')
        except:
            f.write(url+'\r\n')
    r = requests.get(url, headers=headers)
    soup=bs4.BeautifulSoup(r.content,'lxml')
    answers=soup.find_all('span',{'itemprop':'text','class':"RichText CopyrightRichText-richText"})
    pat_html=r'<[\s\S]*?>'
    answer2=''
    for answer in answers:
        answer=''.join(str(answer.contents))
        answer=re.sub(pat_html,'',answer)
        answer2+=answer.replace(',','').replace("'",'')
        answer2+='\r\n====================\r\n'
    with open(filename,'a') as f:
        try:
```

```
            f.write(answer2)
        except:
            logging.warning(name+'      '+url+'1')
    for i in range(int(limit/20)):
        para2['offset']=i*20+2
        r = requests.get('https://www.zhihu.com/api/v4/questions/'+url[-8:]+'/answers?', headers=headers, params=para2)
        try:
            new_question=r.json()
            for answer in new_question['data']:
                answer=re.sub(pat_html,'',answer['content'])
                answer2=answer+'\r\n====================\r\n'
                with open(filename,'a') as f:
                    try:
                        f.write(answer2)
                    except:
                        # logging.warning(name+'      '+url+answer['content'])
                        pass
        except:
            pass
def collectZL(name,url,LogPath):
    r = requests.get(url, headers=headers)
    soup=bs4.BeautifulSoup(r.content,'lxml')
    article=soup.find_all('textarea',{'id':'preloadedState'})
    pat_content=r'"content":"[\s\S]*?","updated"'
    pat_html=r'<[\s\S]*?>'
    # print(article)
    try:
        result=re.findall(pat_content,article[0].contents[0])
        tmp=re.findall(pat_content,result[0][11:])
        if len(tmp)!=0:
            result=tmp
        result=result[0][11:-11].replace('\\u003C','<').replace('\\u003E','>')
        result=name+'\r\n'+re.sub(pat_html,'',result)+'\r\n==============\r\n'
        # print(re.sub(pat_html,'',result))
        with open(LogPath+'\\article.txt','a') as f:
            try:
                f.write(result)
            except:
                logging.warning(name+'      '+url)
                pass
```

```
    except:
        logging.warning(name+'===>null')
def findresult(work,num=10):
    questionDict={}
    articleDict={}
    url = 'https://www.zhihu.com/search?type=content&q=' + work

    r = requests.get(url, headers=headers)
    content=r.content
    pat=r'hash_id=.{32}'
    hashID=re.findall(pat, str(content))
    # print(hashID[0][8:40])
    para['search_hash_id']=hashID[0][8:40]

     soup = bs4.BeautifulSoup(content, 'lxml')
     questions=soup.find_all('div',{'itemprop':'zhihu:question'})
     pat_content=r'content="(.*?)"'
     for question in questions:
         soup2=bs4.BeautifulSoup(str(question),'lxml')
         title=soup2.find_all('meta',{'itemprop':'name'})
         title=re.findall(pat_content,str(title))
         title=title[0].replace('<em>','').replace('</em>','')
         title_url=soup2.find_all('meta',{'itemprop':'url'})
         title_url=re.findall(pat_content,str(title_url))
         questionDict[title]=title_url[0]
         # questionDict.setdefault(title,title_url[0])
         # print(title[0].replace('<em>','').replace('</em>',''),'    ',title_url[0])
     articles=soup.find_all('div',{'itemprop':'article'})
     # print(articles)
     for article in articles:
         soup2 = bs4.BeautifulSoup(str(article).replace('<em>','').replace('</em>',''), 'lxml')
         title=soup2.span
         title=title.contents
         title_url=soup2.a
         title_url=title_url['href']
         articleDict[title[0]]='https:'+str(title_url)
     for i in range(int(num/10)):
         para['offset']=i*10+5
         r = requests.get('https://www.zhihu.com/api/v4/search_v3?t=general&q='+work, headers=headers, params=para)
         new_question=r.json()
         # print(new_question)
```

```
        for question in new_question['data']:
            try:
                title=question['object']['question']['name'].replace('<em>',''
).replace('</em>','')
                title_url=question['object']['question']['url'].replace('ap
i','www').replace('questions','question')
                # print(title,title_url)
                questionDict[title]=title_url
            except:
                title=question['highlight']['title'].replace('<em>','').rep
lace('</em>','')
                title_url=question['object']['url'].replace('api','zhuanlan
').replace('articles','p')
                # print(title,title_url)
                articleDict[title]=title_url
                pass
    # print(new_question['data'][0][])
    return questionDict,articleDict
def main():
    for work in works[:]:
        print(work)
        workPath=projectPath+work
        os.mkdir(workPath)
        questionDict, articleDict = findresult(work,80)
        with open(workPath+'\\dict.txt','w') as f:
            for name,url in articleDict.items():
                try:
                    f.write(name+'     '+url+'\r\n')
                except:
                    logging.warning(url)
            f.write('==========================\r\n')
            for name,url in questionDict.items():
                try:
                    f.write(name+'     '+url+'\r\n')
                except:
                    logging.warning(url)
        # print(questionDict)
        # print(articleDict)
        for name,url in articleDict.items():
            collectZL(name,url,workPath)
            time.sleep(0.1)
        questionNum=0
        for name,url in questionDict.items():
            filename=workPath+'\\question'+str(questionNum)+'.txt'
```

```
            print(questionNum)
            collectQ(name,url,filename,60)
            questionNum+=1
            time.sleep(0.1)

if __name__ == '__main__':
    import requests
    import bs4
    import re
    import logging,os,time
    logging.basicConfig(filename='G:/数据挖掘大作业/输出文档/知乎爬虫/failLog.log', level=logging.DEBUG, filemode='w')
    main()
```