目录

一、爬虫

本次作业是基于豆瓣书评进行文本分析。因此，本节则通过分析网页结构，在 15 分钟内从网页上爬取了 524 条关于《活着》的书评。

首先，我们登录豆瓣网站，搜索《活着》，进入该页面，找到“书评”，如下图：

图 1：书评链接

点击上述页面链接，进入书评页面，发现共计 10818 条书评，541 个翻页，每次翻页都可以看到新的若干条书评。每次执行翻页操作时，可以发现每个页面的网址变化十分有规律，每次都以 20 的步长改变链接地址中 reviews?start 的最后一个数值，可以发现，实际上 20 的步长也是每个页面中书评的数量，如第一页网址为 https://book.douban.com/subject/4913064/reviews?start=0，第二页网址则是 https://book.douban.com/subject/4913064/reviews?start=20，如下图所示：

图 2：书评翻页

因此，在后续的文本爬取中，我们可在设置适当的迭代停止条件后，循环爬取每个页面中的书评。

接着，我们检查此时页面中的每个书评，发现绝大部分书评的内容在此页面中都没完整展示出现，只有点击“展开”或者点击书评标题链接后才能看到完整内容，如下图所示。

图 3：书评内容

考虑到如果我们在上述页面中爬取文本信息，那么此时的书评内容则是不完整的。因此，我们可以通过 BeautifulSoup 解析网页，在每个页面中依次点击每个书评的标题链接，再进入该书评页面采集该书评的完整内容。每个页面有 20 个书评，也就是说此时在每个页面中，我们会依次采集 20 个书评标题的链接，再依次爬取该链接里的书评内容。故我们可以维护一个书签标题链接列表，在每个页面中采集书评标题链接，最后再循环采集链接列表里的书评内容。因此，下一步我们考虑如何采集书签标题链接。通过 BeautifulSoup 解析网页，检查页面元素发现，每个书签标题链接都是 h2 标签下的 a 标签里的 href 属性，如图 4 所示。

图 4：书评标题链接

因此，我们可通过 find()方法找到对应标签，再通过 get('href')的方法采集到指定链接的目标地址，并存储在一个书签标题链接列表中。下一步的内容则是根据链接列表中的每个指定书评链接爬取书评内容。检查页面元素发现，每个指定标题链接下的书评内容大多数都是 p 标签下的文本，少数不是 p 标签，如图 5 所示。

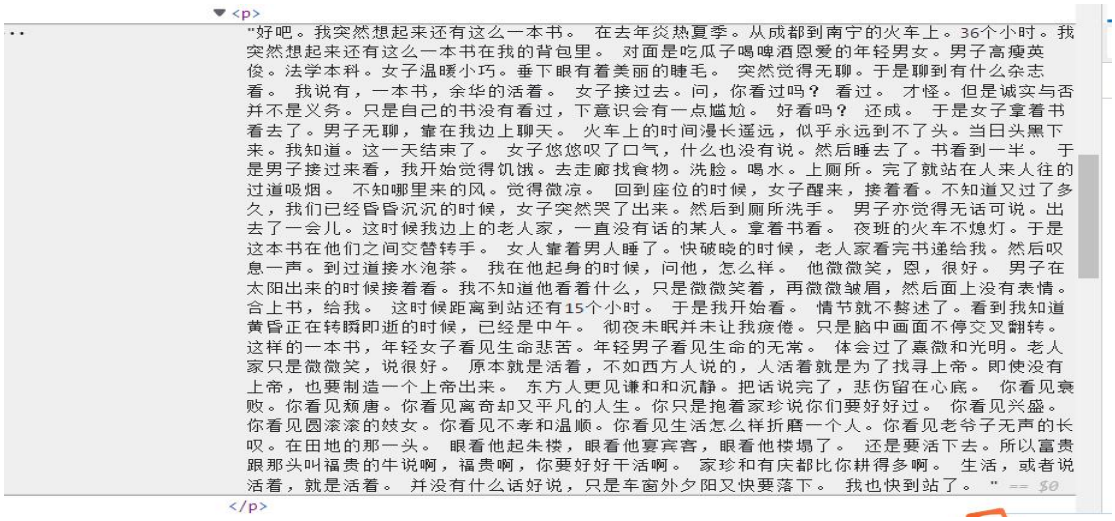

图 5：书评内容

因此，我们通过 get_text()方法采集 p 标签下的文本信息就可以得到绝大多数的书评。值得注意的是，p 标签下还有部分内容并不属于书评，如'扫码直接下载'，'这篇书评可能有关键情节透露'等内容，我们在采集时需要排除此类标签里的内容。

至此，我们便可以顺利爬取到书评里的内容。对于上述的翻页爬取操作，若连续由同一请求头进行爬取，存在着 IP 被封的风险，因此我们可以在网上搜索多个用户请求头，然后每次执行翻页时再随机选择一个请求头访问即可。值得注意的是，我们再爬取内容时也要注意遵守国家法律法规和网站的规定，因此，在具体实现时，在设置翻页休眠时间的同时，也要设置爬取时间限制和爬取数量限制，当爬取书评数量达到指定数量或爬取时间达到指定上限时，则停止爬取。

本节的爬取豆瓣书评思路总结如下：第一步，依次翻页通过采集每个书评标题的指定链接地址，并维护在一个列表中；第二步，根据列表里的指定链接地址，依次采集书评内容；第三步，判断当前是否达到循环终止条件时，是的话则停止请求，并将现有爬取的文本信息保存到本地。

二、文本预处理

（一）分词

文本预处理，或者说是特征提取，首先是要分词，分词是训练词向量的前提。分词，是指将文本分解成标记的过程。中文由于没有空格，分词是一个需要专门去解决的问题。

首先，我们先通过 pandas 库中的 read_csv()函数读取上述我们保存书评内容的本地文件，共计 524 条书评内容，用 id 进行唯一标识。

接着，我们再通过 jieba 库进行中文分词。为了将句子最精确地切开，我们通过调用 jieba 库的 lcut 函数，并设置 cut_all 参数为 False，即精确模式，最终返回一个长度为 94565 的分词列表。

值得注意的是，经过上述分词操作后，此时我们的中文词列表出现了许多诸如‘吧’、‘了’等毫无意义的停用词以及类似空格、‘u3000’等占位符，因此，下一步我们再通过外部导入 stop_words 的文本去除停用词以及过滤占位符，最终得到了一个长度为 32966 的关于《活着》书评的分词列表。

然后我们通过 wordcloud 库中的 Wordcloud()函数生成该书评分词列表的词云，从而对书评中出现频率较高的关键词予以视觉上的突出。我们设置 max_words 参数为 100，即词云显示的最大词语数量为 100，该书评的词云图显示如图 6 所示。

图 6：书评词云图

同时，我们也通过 collections 库中的 Counter()方法获取分词列表中每个分词的频次，通过 most_common(10)的方法获取前十大关键词，并通过直方图将其直观展示，如图 7 所示。

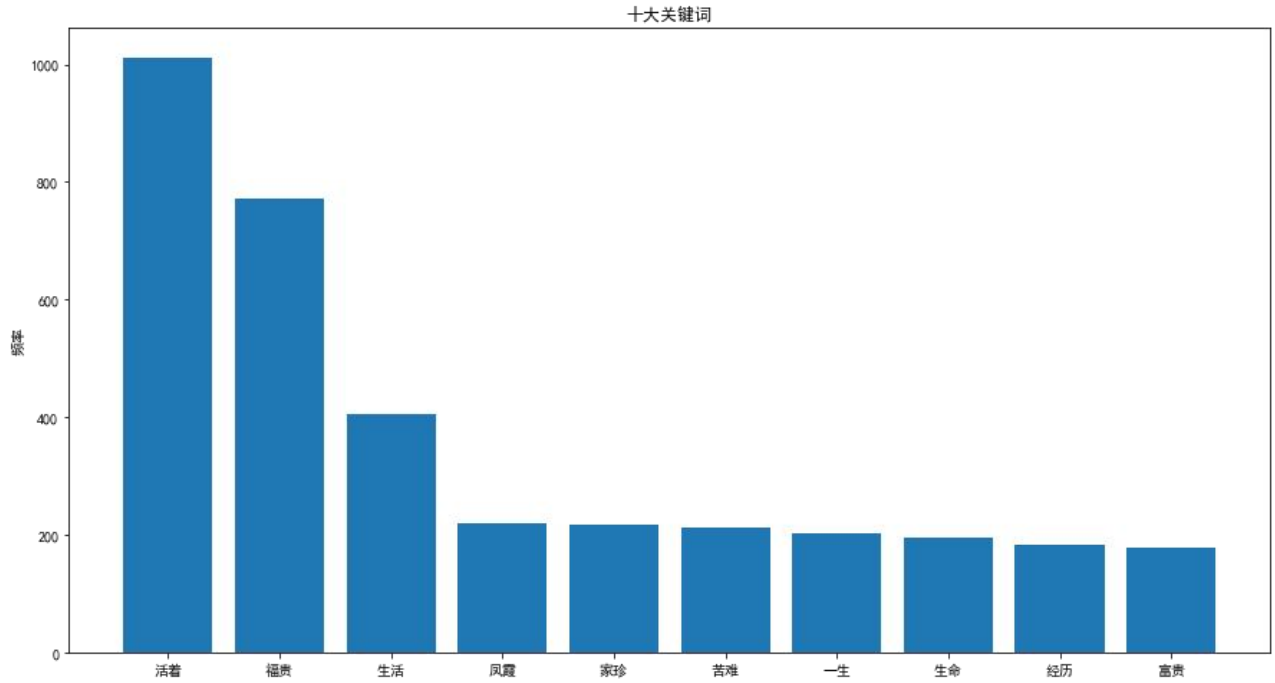

图 7：书评词频直方图

可以发现，无论是词云图，还是直方图，我们都可以直观地观察到“活着”最显眼或出现的频率最高，而“活着”恰好也是该书的标题和主题。在我们的词频统计中，“活着”在书评中共计出现了 1012 次，其次是“福贵”出现了 773 次，“生活”出现了 405 次，“凤霞”出现了 219 次，“家珍”出现了 218 次，“苦难”出现了 213 次。其中“福贵”、“凤霞”和“家珍”是作品的主人公，而“生活”和“苦难”则是该作品着重表现的东西。

（二）向量化

在构建好语料库，即完成分词的基础上（此时我们的语料库过滤了停用词），本节我们则进行文本的向量化。文本的向量化是一个将文本转换为数值张量的过程，即训练词向量，将词和词频对应起来，做成矩阵向量的形式。文本向量化是后续用于机器学习模型的前提和关键。

关于词向量训练，我们可以采用较为简单的词袋模型，也可以采用词嵌入的方法。考虑到后续向仅用于聚类分析，没有其他复杂的诸如情感分析的工作，因此我们可以考虑采用简单的词袋模型。

词袋模型，我们可简称为 BOW 模型，其假设我们不考虑文本中词与词之间的上下文关系，仅仅考虑所有词的权重，而权重与词在文本中出现的频率有关。BOW 模型首先会进行分词，之后，通过统计每个词在文本中出现的次数就可以得到该文本基于词的特征，再将各个文本的词和对应的词频放在一起，就是我们所说的向量化。具体而言，BOW 模型会将我们书评中的中文分词通过 one-hot 编码与一个唯一的整数索引相关联，然后将该整数索引转换为长度为分词列表大小的二进制向量。我们可以通过调用 Scikit-learn 库的 CountVectorizer()函数实现 BOW 模型，并通过索引的切片获取指定书评的向量，如图 8 所示。

```
第一个评论的向量为 :[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 3), (9, 1), (10, 1), (11, 1), (12, 1), (13, 1), (1
4, 1), (15, 1), (16, 1), (17, 1), (18, 1), (19, 1), (20, 1), (21, 1), (22, 1), (23, 1), (24, 1), (25, 1), (26, 1), (27, 1), (28, 1), (29,
2), (30, 1), (31, 1), (32, 1), (33, 1), (34, 1), (35, 1), (36, 1), (37, 1), (38, 1), (39, 1), (40, 1), (41, 1), (42, 1), (43, 1), (44, 1),
(45, 1), (46, 7), (47, 2), (48, 1), (49, 1), (50, 1), (51, 2), (52, 1), (53, 1), (54, 1), (55, 1), (56, 2), (57, 1), (58, 1), (59, 1), (60,
3), (61, 1), (62, 1), (63, 1), (64, 1), (65, 1), (66, 4), (67, 1), (68, 1), (69, 1), (70, 1), (71, 1), (72, 1), (73, 1), (74, 1), (75, 1),
(76, 1), (77, 2), (78, 1), (79, 1), (80, 2), (81, 1), (82, 1), (83, 1), (84, 1), (85, 1), (86, 5), (87, 1), (88, 1), (89, 1), (90, 1), (91,
1), (92, 1), (93, 1), (94, 1), (95, 1), (96, 1), (97, 1), (98, 5), (99, 1), (100, 1), (101, 1), (102, 3), (103, 1), (104, 1), (105, 1), (10
6, 1), (107, 2), (108, 2), (109, 1), (110, 1), (111, 1), (112, 6), (113, 1), (114, 1), (115, 1), (116, 1), (117, 1), (118, 3), (119, 3), (12
0, 1), (121, 1), (122, 1), (123, 2), (124, 1), (125, 1), (126, 1), (127, 1), (128, 1), (129, 3), (130, 1), (131, 1), (132, 1), (133, 1), (13
4, 1), (135, 1), (136, 1), (137, 1), (138, 1), (139, 1), (140, 1), (141, 1), (142, 1), (143, 1), (144, 1), (145, 1), (146, 1), (147, 1), (14
8, 1), (149, 1), (150, 2), (151, 1), (152, 2), (153, 1), (154, 1), (155, 1), (156, 1), (157, 1), (158, 1), (159, 1), (160, 1), (161, 1), (16
2, 1), (163, 1), (164, 1)]
```

图 8：第一个书评的向量

但 BOW 模型有很多缺点，前面提到它没有考虑单词之间的顺序，其次它无法反应出一个句子的关键词，因为有的时候频次高并不能反映问题，因此我们使用 TF-IDF 进行特征修正。

TF-IDF 包括两部分 TF 和 IDF，其中 TF（词频）的公式为：

$$\text{tf(word)} = \frac{\text{单词 word 在文档中出现的次数}}{\text{文档的总词数}}$$

如果某个词很重要，它应该在这篇文章中多次出现，即 tf 值越高，表示该词越关键。

而 IDF（逆文本频率）的公式为：

$$\text{idf(word)} = \log\left(\frac{\text{语料库中文档总数}}{\text{包含 word 的文档数} + 1}\right)$$

其中分母之所以加 1 是为了防止分母为 0。idf 值强调的是词的普遍性。假设在《活着》的书评中，“中国”和“福贵”出现的频率一样，那么此时这两个词的 tf 值一样大，但是我们知道“福贵”比起“中国”显然更符合文章的主题。因此，idf 值衡量一个词是不是常见词，充当一个重要性调整系数。如果某个词比较少见，但是它在这篇文章中多次出现，那么它很可能就反映了这篇文章的特性，正是我们所需要的关键词。

而 tf-idf 值则表示为 tf 值和 idf 值的乘积。TF-IDF 值越大说明这个词越重要，也可以说这个词是关键词。一般情况下， TF-IDF 模型就能满足关键词提取要求，从而能够针对 BOW 模型的缺点进行特征的权重修正。

我们通过调用 Scikit-learn 库的 TfidfVectorizer()函数训练 TF-IDF 模型，并设置 min_df 参数为 0.2 用于过滤极少出现的词条，max_df 参数为 0.9 用于过滤太频繁出现的词条，再通过 fit_transform()方法将特征标准化，最终得到一个 shape 属性为(524, 8640)的词频矩阵。我们可以通过 get_feature_names()方法获取特征名，也可以通过索引切片获取指定的 tf-idf 特征值，如图 9 所示。

```
第 5 个书评的 TF-IDF ：
 [(312, 0.1924056780155172), (708, 0.3568849806976042), (760, 0.23265034763538145), (761, 0.201100872492618), (762, 0.19657191502252278), (7
63, 0.44925355529716776), (764, 0.20606178410826062), (765, 0.2687287802482401), (766, 0.33996670299080395), (767, 0.1924056780155172), (76
8, 0.224626777648583888), (769, 0.20606178410826062), (770, 0.25375487779869893), (771, 0.201100872492618), (772, 0.18854834503572526)]
```

图 9：第五个书评经 TF-IDF 修正后的向量

（三）主题分析

在上述向量化工作中，我们简要介绍了 TF-IDF 模型，并表示 tf-idf 值也可以揭示文章的主题词。除了 TF-IDF 模型外，我们可以通过 LDA 模型进行更加细致的主题分析工作。

LDA 由 Blei, David M.、Ng, Andrew Y.、Jordan 于 2003 年提出，用来推测文档的主题分布。它可以将文档集中每篇文档的主题以概率分布的形式给出，从而通过分析一些文档抽取出它们的主题分布后，便可以根据主题分布进行主题聚类或文本分类。

在上述向量化的工作中，我们已经构建好了语料库，过滤了停用词，此时我们再通过调用 gensim 库的 LdaModel 接口训练 LDA 模型，n_components 参数设置为 6，表示划分为 6 个主题。我们可以通过 fit_transform()函数查看评论属于某主题的概率，通过 components_属性查看词汇权重。

```
书评属于某主题的概率： [[0.0143521  0.01435989 0.01435973 0.0143541  0.92822448 0.0143497 ]
 [0.01075443 0.01075439 0.01076044 0.01075573 0.94622424 0.01075077]
 [0.01041668 0.01040596 0.01043275 0.21958366 0.7387487  0.01041224]
 ...
 [0.01593299 0.01595359 0.01591022 0.01594817 0.92031876 0.01593626]
 [0.02820741 0.79588125 0.02824782 0.02824199 0.09116368 0.02825785]
 [0.01788005 0.01786393 0.01787128 0.01787007 0.46871095 0.45980371]]
词汇权重： [[0.16667002 0.28329087 0.16842164 ... 0.16666989 0.16666917 0.33573006]
 [0.16666975 0.16667326 0.16713637 ... 0.16666956 0.1666689  0.1666723 ]
 [0.1666697  0.16667326 0.16666872 ... 0.33414901 0.16666893 0.16667254]
 [0.16666986 0.16667356 0.16666882 ... 0.16666968 0.16666901 0.16667246]
 [0.34933622 0.23873427 0.57384708 ... 0.1666686  0.21567883 0.17116267]
 [0.16666985 0.16667354 0.16666884 ... 0.16666975 0.16666908 0.16667239]]
```

图 10：书评主题概率和词汇权重

此外，我们可通过结合 components_属性和 argsort()方法查看每个主题下的前 N 大关键词汇，这里我设置了 N=6，最终输出如图 11 和图 12 所示。

```
第1个主题的主要词汇：['历史', '一阵子', '读者', '惊喜', '活着', '真诚地']
第2个主题的主要词汇：['全是', '召唤', '福贵', '有时候', '黑夜', '想想']
第3个主题的主要词汇：['历史', '活着', '难过', '小说', '时间', '人物']
第4个主题的主要词汇：['意义', '疯婆子', '时代', '活着', '生活', '福贵']
第5个主题的主要词汇：['活着', '福贵', '生活', '富贵', '苦难', '凤霞']
第6个主题的主要词汇：['时期', '文学', '足够', '历史', '余华', '时代']
```

图 11：主题关键词

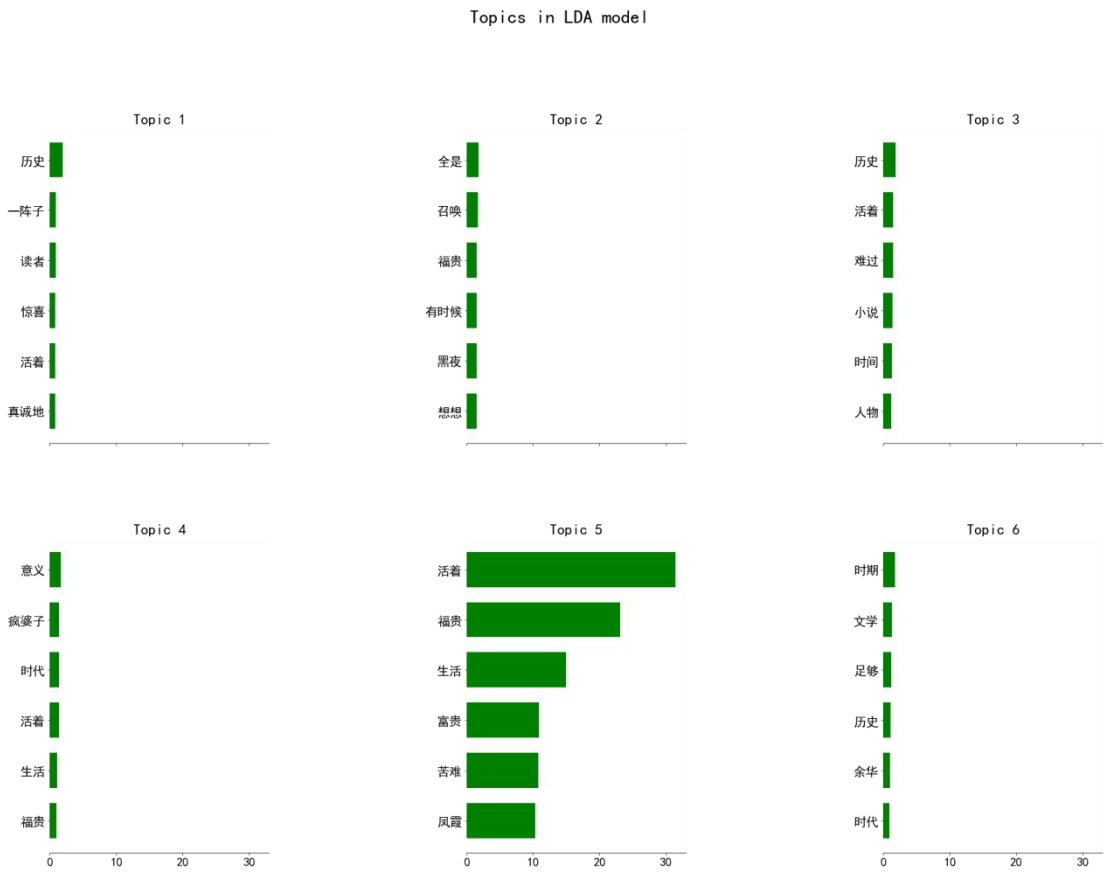

图 12：主题关键词分布

通过图 12，我们可以直观地发现第五个主题的概率相比其它五个主题的概率明显较高。第五个主题的主要词汇包括活着、福贵、生活、富贵、苦难和凤霞，其中活着是作品的名称和主题，福贵和风险则是作品的主人公，生活、苦难和富贵则是作品着重描述的内容。

三、聚类

在完成上述文本预处理的工作后，我们已经得到了关于书评的特征矩阵，可以采用任何合适的机器学习模型进行训练和预测。在本节中，我们将通过调用 Scikit-learn 库中的 KMeans 接口实现 KMeans 模型，从而实现聚类分析的无监督学习。

KMeans 算法比较简单，首先，随机取 K 个样本，将其位置作为重心；然后，不断地标记样本，更新重心，直至重心不再移动。我们设置 KMeans 模型中的 n_clusters 参数为 3，表示聚类成 3 簇，再通过计数神奇 Counter()函数得到每

个簇中书评的数量，其中第一个簇的数量为 88，第二个簇的数量为 300，第三个簇的数量为 136。

我们为每条书评添加 id 标识，并设置每个簇有 6 个关键特征，最终聚类结果输出如图 13 所示。

```
Cluster 0 details:
Key features: ['活着', '意义', '福贵', '生活', '苦难', '生命']
id in this cluster:
1, 3, 10, 12, 24, 32, 36, 52, 55, 62, 63, 70, 79, 83, 84, 91, 102, 123, 124, 125, 130, 136, 152, 153, 154, 164, 169, 178, 184, 185, 195, 19
8, 199, 205, 221, 233, 237, 241, 245, 250, 251, 261, 262, 279, 285, 288, 293, 296, 316, 323, 330, 333, 338, 340, 349, 360, 363, 365, 372, 37
8, 381, 393, 395, 400, 404, 406, 416, 418, 436, 438, 442, 451, 452, 456, 460, 461, 463, 464, 478, 486, 487, 503, 510, 515, 516, 517, 521, 52
3
Cluster 1 details:
Key features: ['活着', '福贵', '历史', '余华', '时代', '生活']
id in this cluster:
0, 4, 5, 6, 7, 9, 13, 14, 15, 16, 18, 19, 20, 22, 23, 25, 26, 27, 29, 30, 34, 35, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 50, 56, 5
7, 58, 59, 60, 61, 64, 65, 66, 69, 71, 72, 73, 74, 75, 78, 81, 82, 85, 87, 92, 93, 94, 96, 97, 98, 99, 100, 101, 104, 105, 106, 107, 109, 11
0, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 128, 129, 131, 132, 137, 138, 139, 140, 141, 142, 143, 144, 145, 148, 150, 15
1, 156, 157, 160, 165, 167, 168, 170, 171, 172, 175, 177, 179, 183, 186, 187, 188, 189, 190, 191, 192, 193, 194, 196, 197, 200, 202, 203, 20
4, 206, 207, 209, 210, 211, 213, 214, 215, 216, 217, 218, 220, 222, 223, 224, 225, 226, 227, 229, 231, 232, 234, 235, 236, 238, 239, 240, 24
2, 243, 244, 247, 248, 249, 252, 254, 255, 256, 257, 258, 259, 260, 264, 265, 266, 269, 270, 273, 274, 275, 276, 278, 281, 282, 284, 286, 28
7, 289, 290, 291, 295, 297, 299, 300, 301, 302, 303, 304, 308, 311, 312, 313, 314, 315, 318, 320, 321, 324, 325, 329, 331, 332, 334, 335, 33
6, 337, 339, 341, 343, 345, 346, 347, 351, 354, 355, 356, 361, 364, 367, 368, 369, 370, 371, 373, 375, 377, 379, 380, 384, 385, 386, 387, 38
8, 389, 390, 391, 394, 399, 401, 407, 409, 412, 414, 417, 419, 420, 422, 425, 428, 429, 431, 432, 434, 435, 439, 440, 443, 444, 445, 446, 44
8, 450, 453, 458, 467, 469, 472, 474, 475, 476, 477, 480, 481, 484, 485, 488, 491, 492, 493, 494, 496, 497, 499, 500, 501, 504, 505, 508, 50
9, 512, 513, 514, 519, 520, 522
Cluster 2 details:
Key features: ['福贵', '富贵', '凤霞', '生活', '家珍', '苦根']
id in this cluster:
2, 8, 11, 17, 21, 28, 31, 33, 49, 51, 53, 54, 67, 68, 76, 77, 80, 86, 88, 89, 90, 95, 103, 108, 126, 127, 133, 134, 135, 146, 147, 149, 155,
158, 159, 161, 162, 163, 166, 173, 174, 176, 180, 181, 182, 201, 208, 212, 219, 228, 230, 246, 253, 263, 267, 268, 271, 272, 277, 280, 283,
292, 294, 298, 305, 306, 307, 309, 310, 317, 319, 322, 326, 327, 328, 342, 344, 348, 350, 352, 353, 357, 358, 359, 362, 366, 374, 376, 382,
383, 392, 396, 397, 398, 402, 403, 405, 408, 410, 411, 413, 415, 421, 423, 424, 426, 427, 430, 433, 437, 441, 447, 449, 454, 455, 457, 459,
462, 465, 466, 468, 470, 471, 473, 479, 482, 483, 489, 490, 495, 498, 502, 506, 507, 511, 518
```

图 13：聚类结果

我们可以发现第一个簇和第二个簇的关键特征还是非常相似的，都包括了“活着”，“福贵”和“生活”，在这的基础上，第一个簇更注重感性层面的认识，如“意义”，“苦难”和“生命”，而第二个簇更注重从历史和作者的角度出发，如“余华”，“历史”和“时代”。而第三个簇的关键特征则基本上是作品中的人物。