Film Analysis Based on NLP and Spider

向兴旺

2022 年 12 月 3 日

1 Abstract

摘　要　2022, 中国电影市场处于疫情的特殊关头。本文将通过爬虫获取豆瓣上最热门的十部电影的基本信息、粉丝分布、评分数据，评论数据，并对这些数据利用 python 进行数据处理和分析，特别的，对于评论数据，本文将使用 NLP 技术对其进行分词和情感分析。最终，得到每一部电影的正负面评价比率、粉丝分布、票房属性，对电影作出分析和推荐报告，更进一步的，我们试图通过这些数据，分析疫情下的中国电影市场。

关键词　爬虫　NLP　电影　分词　情感分析　数据处理　电影市场　票房属性

2 Film Recomending Module

当前主要的电影分析手段，仍然是人工分析，而传统人工分析:

1. 如果对于影片的评论作完善分析，则逐条逐字查看太过消耗时间。

2. 如果仅仅对于影片的评分进行分析，则难以避免水军刷评论的现象。

针对这一情形，构建一个自动化电影分析框架是有其必要价值的。本文主要利用了两种技术: 爬虫，NLP。均采用 python 实现。

在数据获取，即爬虫阶段，本文主要将其分为两个步骤: 1)http 交互：即通过程序，向远程服务器发送 web 请求，并接受 respose 报文，实现与 web 服务器的交互。主要使用 requests module 实现 2) 数据处理和保存。即对于接受到的 html 报文进行解析，提取关键数据，并进行格式化的存储，便于数据分析时使用。主要使用 BeautifulSoup module 和 json module 实现。

我们主要获取获取的信息包括: 影片的基本信息，影片的评论内容，影片评论人的地址分布，影片评论的分数分布。

在数据分析，也包含两个步骤: 1) 图表绘制，即对于所收集的数据，以各种图表做可视化处理。主要基于 matplotlib module 和 seaborn module 实现。2)NLP 分析，主要采用 SnowNLP module 和 jieba module 实现。[1]

首先，对于地址和评分的分布，可以考虑用扇形图直观的描述。但是，对于评论信息，难以直观的表述和呈现，所以需要使用 NLP 的技术加以分析。考虑以下方法，首先，一部电影 i , 对于其每一条评论 k, 进行情感分析，得到情感分数 s_{ik}，（分数越大，情感越积极）。然后，对于所有评论，综合考虑，得到情感分数的均值 $\overline{s_i}$ 以及情感分数的分布图。紧接着，分别通过分词和关键词提取技术，分别获取负面评论和正面评论中出现频率较大的词，过滤无价值的 stop words (主要基于代码目录下 stop_word.txt)。得出好评观众和差评观众各自的关注点和影片的关键词。对评论形成完善的分析模型。

总体而言。本文完成的工作:

- 通过爬虫和 NLP 技术对于电影进行评价性分析
- 完成了一个电影分析的代码框架并将相应代码和数据开源

3

本项目的完整代码开源在 https://github.com/Du-Mu/film_recomending

3.1 Data Collection

首先，本文以豆瓣电影新片榜 (https://movie.douban.com/chart) 界面为爬虫的首先访问界面，这一界面列举了最近热映的新片。对于此界面的 html 代码进行分析，可以发现电影的基本信息都在 class=‘nbg’ 的元素中。爬取此页面后，我们可以遍历此界面的 nbg 元素，获取每一个对应电影的 link，进一步，请求访问对应电影的界面和对应电影评论的界面，后面的这一步骤主要由 $get\_basic\_data()$,$film\_reviews\_spider()$ 实现。

```
    film_url = 'https://movie.douban.com/chart'
    film_html = requests.get(url=film_url, headers=headers).text
    film_soup = BeautifulSoup(film_html, 'html.parser')
    num = 0
    for i in film_soup.find_all('a', 'nbg'):
        # 解析每一个电影对应的元素，获取对应的电影专业的link
        rating_data = {}
        addr_data = {}
        basic_data = {}
        num+=1
        print('[-]now are processing film ' + str(num))
        film_name = i.get('title')
        film_info = {}
        film_reviews_spider(film_link=i.get('href'), film_info=film_info,
            rating_data=rating_data, addr_data=addr_data)
        basic_data = get_basic_data(i.get('href'))
```

在 $get\_basic\_data()$ 中, 针对性的对电影主界面进行分析，在 script 中 type 为’application/ld+json’ 中发现电影信息的 json 格式字符串。于是获取对应内容，再将之存储在名为 basic_iofo 的 dict 中。

```
def get_basic_data(film_url):
    time.sleep(1)
    film_main_html = requests.get(url=film_url, headers=headers).text

    film_main_soup = BeautifulSoup(film_main_html, 'html.parser')

    basic_info = film_main_soup.find('script', type='application/ld+json').text

    return json.loads(basic_info, strict = False)
```

在 film_reviews_spider() 中，由于一个界面，不能显示过多的评论，于是迭代获取对应界面，同时，为了防止触发反爬虫措施，我们通过 $sleep()$ 降低请求的频率。在获取到评论界面后，将获取到的评论，分条目存入 dict 中，同时统计评分和地理位置信息。

```
def film_reviews_spider(film_link, film_info, rating_data, addr_data):
    start_num = 0;
    count = 0
    while (start_num < 1000):
        # 迭代获取评论界面
        print('[*]now are loading '+str(start_num)+' comments')
        start_num += 100
        review_url = film_link+'comments?start='+str(start_num)+'&limit=100&
            status=P&sort=new_score'
        time.sleep(5)
        try:
            review_html = requests.get(url=review_url, headers=headers).text
        except Exception:
            print('[*]fail to parse:'+review_url)
            return False
        review_soup = BeautifulSoup(review_html, 'html.parser')

        for i in review_soup.find_all('div', 'comment'):
            count += 1
            # 找到每一条评论
            addr = i.find('span', 'comment-location').text
            rating = i.contents[1].contents[3].contents[5].get('title')

            comment_info = {
                'id':i.contents[1].contents[3].contents[1].text,
                'content':i.find('span', 'short').text,
                'addr':addr,
                'rating':rating
            }

            if (rating_data.get(rating) == None and len(rating) == 2):
                rating_data[rating] = 1
            elif (len(rating) == 2):
                rating_data[rating] += 1
```

```
            # 统计分数信息
            if (addr_data.get(addr) == None):
                addr_data[addr] = 1
            else:
                addr_data[addr] += 1
            # 统计地址信息
            film_info[count] = comment_info
            # 存入评论信息

    return True
```

最后，将获取到的数据，分别存放到不同下的 json 文件中，方便后续分析和读取

```
try:
    os.mkdir('./film_data/film'+str(num))
except Exception:
    continue

file = open('./film_data/film'+str(num)+'/comments.json', 'w')
json.dump(film_info, fp=file, sort_keys=True, separators=(',', ': '), indent=4,
    ensure_ascii=False)
file.close()

file_addr = open('./film_data/film'+str(num)+'/addr.json', 'w')
json.dump(addr_data, fp=file_addr, sort_keys=True, separators=(',', ': '), indent
    =4, ensure_ascii=False)
file_addr.close()

file_rating = open('./film_data/film'+str(num)+'/rating.json', 'w')
json.dump(rating_data, fp=file_rating, sort_keys=True, separators=(',', ': '),
    indent=4, ensure_ascii=False)
file_rating.close()

file_basic = open('./film_data/film'+str(num)+'/basic.json', 'w')
json.dump(basic_data, fp=file_basic, sort_keys=True, separators=(',', ': '),
    indent=4, ensure_ascii=False)
file_basic.close()
```

3.2 Data Analysis

首先我们对电影基本信息进行分析, 将基本各个电影的基本信息以 csv 表格的形式列出。

```
def analysis_basic_info(csv_file_name):
    header = ['No','name','data','genre','data','rating_count', 'rating_value']
    csv_file = open(csv_file_name, 'w')
    writer = csv.writer(csv_file)
    writer.writerow(header)

    for i in range(1, 11):
        file = open('./film_data/film'+str(i)+'/basic.json', 'r')
        basic_info = json.load(fp=file)
        row = [
            i,
            basic_info['name'],
            basic_info['datePublished'],
            basic_info['genre'],
            basic_info['aggregateRating']['ratingCount'],
            basic_info['aggregateRating']['ratingValue'],
        ]
        writer.writerow(row)
        file.close()
    csv_file.close()
```

其次，我们处理地理分布信息，根据之前得到的地理分布人数的 json 文件，得到地理分布人数的 dict，然后，根据分布人数，作出对应分布图。

```
def analysis_addr_info(file_name, film_no):
    file = open(file_name, 'r')
    addr_info = json.load(fp=file)
    file.close()

    addr_info['else'] = 0
    for key in list(addr_info.keys()):
        if (addr_info[key] < 8):
            addr_info['else'] += addr_info[key]
            del addr_info[key]

    plt.rcParams['font.sans-serif']=['KaiTi','SimHei','FangSong']
    plt.figure(figsize = (8,8))
```

```
    explode = []
    for i in range(0, len(addr_info)):
        explode.append(i*0.01)
    plt.pie(addr_info.values(), labels=addr_info.keys(), explode = explode,
        autopct = '%1.2f%%',
        pctdistance = 0.8, labeldistance = 0.9)
    plt.title('addr_info')
    plt.savefig('./paper_writing/images/addr'+str(film_no)+'.jpg')
```

类似的，我们处理分数分布信息:

```
def analysis_rating_info(file_name, film_no):
    file = open(file_name, 'r')
    addr_info = json.load(fp=file)
    file.close()

    plt.rcParams['font.sans-serif']=['KaiTi','SimHei','FangSong']
    plt.figure(figsize = (8,8))
    explode = []
    for i in range(0, len(addr_info)):
        explode.append(i*0.01)
    plt.pie(addr_info.values(), labels=addr_info.keys(), explode = explode,
        autopct = '%1.2f%%',
        pctdistance = 0.8, labeldistance = 0.9)
    plt.title('rating_info')
    plt.savefig('./paper_writing/images/rating'+str(film_no)+'.jpg')
```

最后，处理评论信息。对于评论信息，主要进行了两项处理，情感分析和关键词提取。

关键词提取：分词和关键词主要基于 snownlp 和 jieba 实现的 TextRank 算法，对于一段文本，构建一段基于各种词与词和词之间联系的无向图，再根据图的连接，更新每个词的权重。[2]

情感分析: 对于提取出的关键词，分析其情感因子，再结合其在图中的位置和关键程度，给予其相应的系数，综合而言，得到整段评论的情感因子。[3]

```
def analysis_comments_info(file_name, film_no):
```

```
stopwords = [line.strip() for line in open('./stop_word.txt', encoding="utf-8
    ").readlines()]
pos_word = {}
neg_word = {}

file = open(file_name, 'r')
comments_info = json.load(fp=file)
file.close()
com_sentiments = []
for i in range(1, len(comments_info)+1):
    comment = comments_info[str(i)]['content']
    comment = ' '.join(re.findall('[\u4e00-\u9fa5]+', comment, re.S))
    if comment == '':
        continue
    comment_snow = SnowNLP(comment)
    com_sentiments.append(comment_snow.sentiments)
    # 获取情感因子
sns.distplot(com_sentiments, bins=30)
plt.savefig('./paper_writing/images/comments'+str(film_no)+'.jpg')

mean = np.mean(com_sentiments)

for i in range(1, len(comments_info)+1):
    comment = comments_info[str(i)]['content']
    comment = ' '.join(re.findall('[\u4e00-\u9fa5]+', comment, re.S))
    if comment == '':
        continue
    comment_snow = SnowNLP(comment)
    if comment_snow.sentiments > mean:
        now_word = pos_word
    else:
        now_word = neg_word

    words = jieba.lcut(comment)
    for word in words:
        if word in stopwords:
            continue
        if len(word)<2:
            continue
        if now_word.get(word):
            now_word[word]+=1
        else:
            now_word[word] = 1
        # 过滤无价值词
```

```
    pos_item = list(pos_word.items())
    pos_item.sort(key=lambda x: x[1], reverse=True)
    print("pos_word:")
    for i in range(30):
        p_word, word_count = pos_item[i]
        print(p_word+':'+str(word_count))

    neg_item = list(neg_word.items())
    neg_item.sort(key=lambda x: x[1], reverse=True)
    print("neg_word:")
    for i in range(30):
        p_word, word_count = neg_item[i]
        print(p_word+':'+str(word_count))
```

4 Film Analysis

4.1 Overview:

No	name	data	genre	rating_count	rating_value
1	西线无战事 Im Westen nichts Neues	2022-09-12	['剧情', '动作', '战争']	70433	8.6
2	名侦探柯南：万圣节的新娘 名探偵コナ	2022-04-15	['动作', '动画', '悬疑']	65087	7.5
3	危笑 Smile	2022-09-22	['恐怖']	26922	6.3
4	6/45 육사오	2022-08-24	['剧情', '喜剧']	31548	7.7
5	共助2：国际 공조2:인터내셔날	2022-09-07	['动作']	19006	7.1
6	黑亚当 Black Adam	2022-10-19	['动作', '科幻', '奇幻', '冒险']	26491	5.8
7	火山挚恋 Fire of Love	2022-01-20	['纪录片']	11039	9
8	福尔摩斯小姐：伦敦厄运 Enola Holme	2022-11-04	['剧情', '动作', '犯罪', '悬疑', '冒险']	8441	6.6
9	珀尔 Pearl	2022-09-03	['恐怖']	25287	7.3
10	新神榜：杨戬	2022-08-19	['动作', '动画', '奇幻', '冒险']	218984	7

图 1: 电影基本信息统计

4.2 西线无战事

4.2.1 图表分布

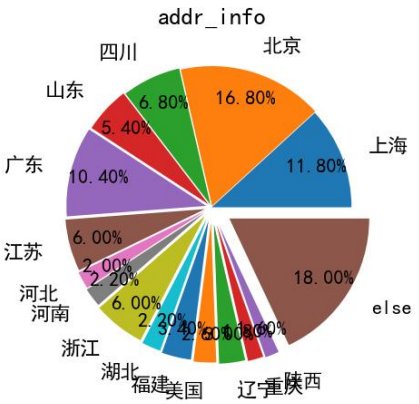

图 2: 观众评论地址分布

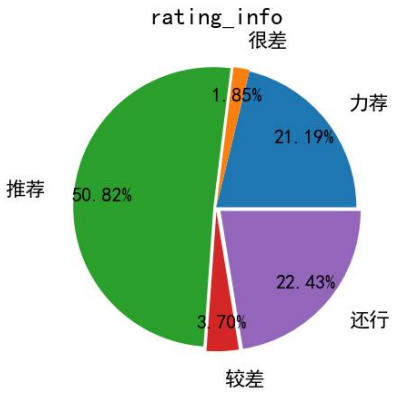

图 3: 观众评论分数分布

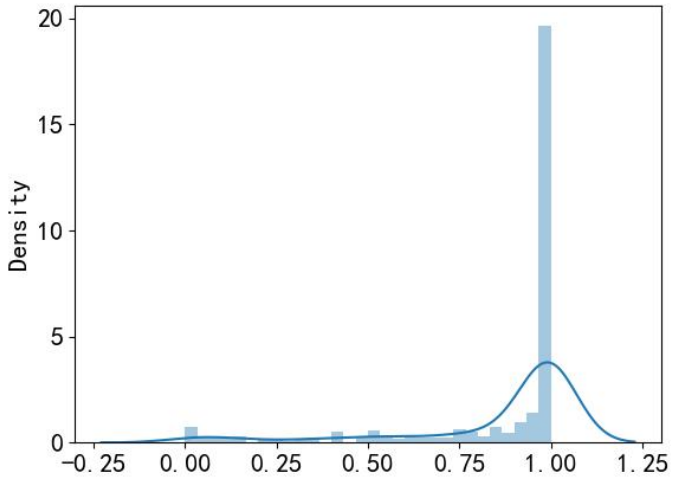

图 4: 影片评论情感分数分布

4.2.2 Keywords

正面评论和负面评论关键词

pos_word: 战争:295, 电影:88, 战场:74, 反战:70, 士兵:60, 残酷:53, 战争片:51, 配乐:43, 原著:43, 最后:38, 死亡:35, 一战:33, 生命:32, 西线:32, 故事:31, 觉得:31, 保罗:30, 镜头:30, 摄影:28, 无战事:28, 人类:27, 震撼:27, 非常:26, 和平:26, 真的:26, 视角:25, 停战:25, 反思:24, 世界:24, 虚无:24

neg_word: 战争:9, 国家:7, 士兵:6, 最后:6, 棋子:5, 戰場:5, 地方:5, 原著:5, 一战:5, 原版:5, 不會:4, 戰爭:4, 殘酷:4, 真的:4, 西线:4, 无战事:4, 现在:4, 有点:4, 节奏:4, 反战:4, 和平:4, 导演:3, 他們:3, 一個:3, 生命:3, 有人:3, 知道:3, 震撼:3, 主角:3, 青年:3

4.2.3 综合分析

综合来看，本片整体分数较高，力荐和推荐 (好评) 占比例 72.02%, 还行 (中评) 占比 22.43%, 差评占比 5.55%，粉丝主要分布在北京、上海、四川、山东等地区。正面评价主要关键词包含战争、反战、残酷、配乐、镜头。差评关键词主要包含节奏、原版等。说明其作为战争电影，在叙事、配乐、镜头上，很好地表现了战争的残酷，集中体现了反战的价值观，但是情节和原著有一定差别。

总体而言，较为推荐。

4.3 名侦探柯南 万圣节的新娘

4.3.1 图表分布

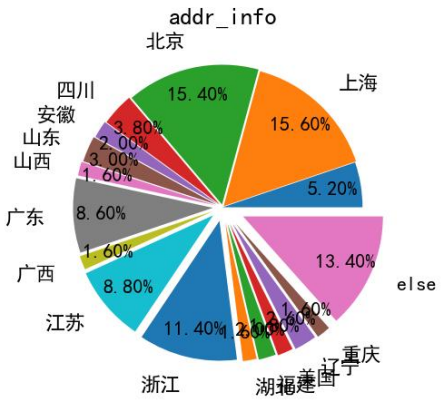

图 5: 观众评论地址分布

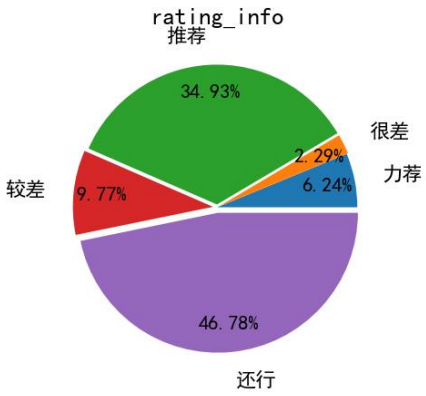

图 6: 观众评论分数分布

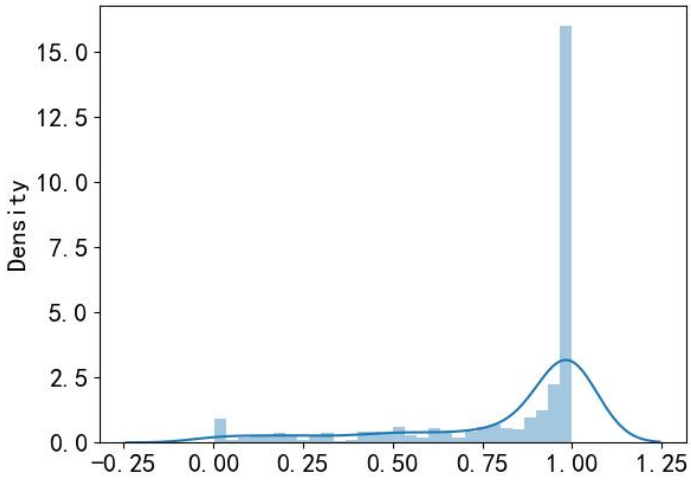

图 7: 影片评论情感分数分布

Keywords

正面评论和负面评论关键词

pos_word: 柯南:158, 剧场版:143, 推理:83, 警校:70, 剧情:54, 动作:54, 一部:54, 真的:54, 五人组:52, 最后:49, 好看:45, 已经:41, 故事:40, 感觉:36, 不错:36, 松田:35, 这部:34, 电影:32, 安室:29, 确实:28, 柯学:28, 场面:27, 毛利:25, 有点:25, 情怀:25, 部分:25, 足球:25, 现在:24, 炸弹:23, 电影院:22

neg_word: 柯南:20, 松田:14, 真的:14, 剧场版:14, 推理:13, 离谱:11, 反派:10, 阵平:8, 难看:8, 安室:8, 剧情:8, 一部:8, 下水道:7, 涩谷:7, 最后:7, 佐藤:6, 最佳:6, 只能:6, 足球:6, 几年:6, 有点:6, 竟然:6, 五人组:6, 故事:6, 警校:6, 劇場:6, 俄语:5, 实在:5, 高木:5, 涉谷:5

综合评价

综合来看，本片整体水平比较中庸，力荐和推荐 (好评) 占比 41.71%, 还行 (中评) 占比 46.78%，差评占比 11.08%，粉丝主要分布在北京、上海、浙江等地。正面评价主要关键词包括警校、动作、剧情、五人组等，负面评论主要包含推理、离谱、反派、松田、难看等，说明其作为一部柯南电影，仍然带有很大粉丝向的成分，对于警校五人组的刻画，以及剧情和动作戏，都很好地满足了粉丝的需求。负面评论主要集中吐槽了反派的塑造和推理仍旧有一些离谱。

总体而言，是一部合格的粉丝向电影。

危笑

图表分布

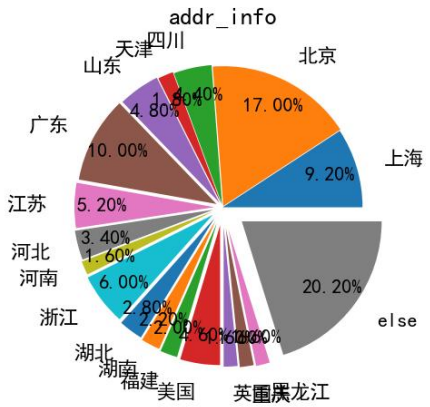

图 8: 观众评论地址分布

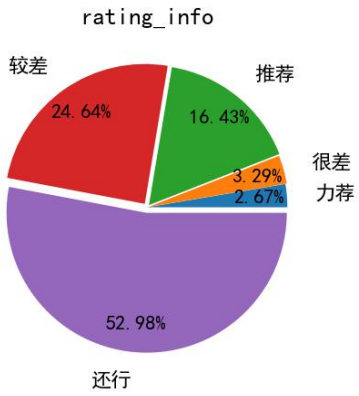

图 9: 观众评论分数分布

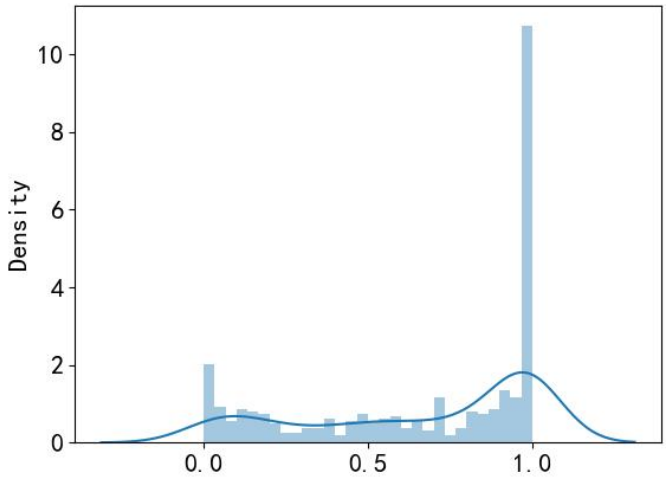

图 10: 影片评论情感分数分布

Keywords

正面评论和负面评论关键词

pos_word: 恐怖片:79, 女主:55, 微笑:53, 恐怖:53, 最后:50, 诅咒:50, 不错:44, 电影:42, 故事:40, 真的:39, 镜头:34, 吓人:34, 有点:34, 结尾:31, 剧情:29, 怪物:28, 创伤:26, 结局:24, 觉得:23, 心理:22, 身后:22, 氛围:22, 音效:22, 一惊:22, 诡异:21, 其实:20, 这种:20, 惊吓:20, 午夜凶铃:20, 影片:19

neg_word: 吓人:29, 恐怖片:27, 恐怖:20, 午夜凶铃:18, 有点:15, 剧情:12, 真的:12, 最后:12, 诅咒:11, 女主:11, 无聊:11, 电影:10, 预告片:9, 看到:9, 身后:8, 不会:8, 一惊:8, 觉得:8, 吓死:8, 知道:7, 镜头:7, 这种:7, 老套:7, 专业:7, 微笑:6, 设定:6, 电影院:6, 低级:6, 地方:6, 前男友:6

综合评价

综合来看，本片整体水平中等偏下，力荐和推荐 (好评) 占比 19.10%, 中评占比 52.98%, 差评占比 27.92%, 粉丝主要分布在北京、上海、广东，正面评价主要关键词包括恐怖片、女主、微笑、恐怖、诅咒等。负面关键词主要包括恐怖、午夜凶铃、剧情、老套等。说明，作为一部恐怖片，对于大多数差评观众而言，有一点惊吓，但是对于一部分恐怖电影受众来说，有点老套。剧情略有瑕疵。

总体而言，对于想要体验恐怖电影的人来说，可以一看，但是对于其他观众，则不是很推荐。

图表分布

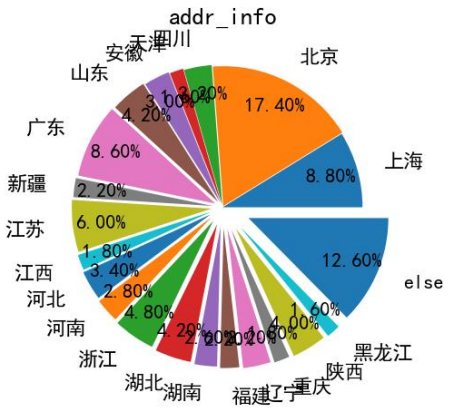

图 11: 观众评论地址分布

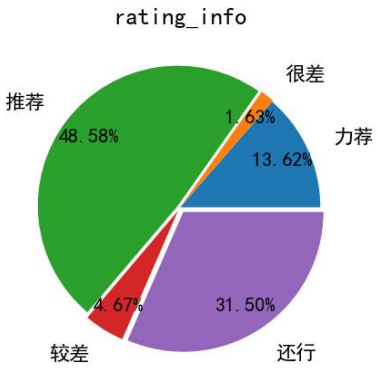

图 12: 观众评论分数分布

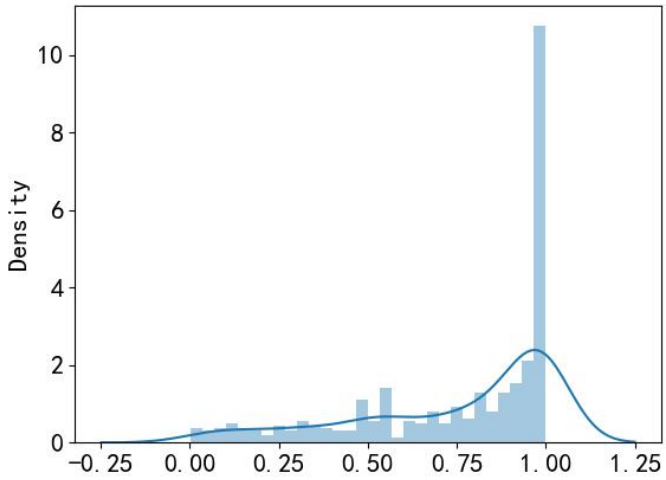

图 13: 影片评论情感分数分布

Keywords

正面评论和负面评论关键词

pos_word: 喜剧:62, 哈哈哈:56, 搞笑:50, 南北:48, 电影:39, 不错:38, 真的:30, 统一:29, 好笑:29, 故事:29, 笑点:22, 哈哈哈哈:22, 共同:20, 朝鲜:20, 有点:20, 结尾:19, 剧情:18, 彩票:18, 韩国:18, 和平:17, 题材:17, 很多:17, 喜剧片:16, 爆笑:15, 一部:15, 警备区:15, 轻松:15, 看到:14, 荒诞:14, 这种:13

neg_word: 搞笑:27, 南北:24, 好笑:23, 喜剧:14, 真的:13, 共同:11, 警备区:11, 有点:11, 统一:11, 彩票:10, 韩国:9, 电影:8, 无聊:7, 女团:7, 笑点:6, 知道:6, 剧情:5, 可能:5, 好看:5, 喜欢:5, 野猪:5, 故事:4, 设定:4, 觉得:4, 三星:4, 真是:4, 无厘头:4, 地方:4, 好久没:4, 确实:4

综合评价

综合来看，本片水平中等偏上。好评比例 62.20%, 中评比例 31.50%, 差评比例 2.30%，粉丝主要分布于北京、上海，正面评价主要关键词包括哈哈哈、喜剧、南北、搞笑。负面评价主要关键词包括搞笑、南北、好笑、彩票、韩国等。作为一部喜剧电影，完成了其搞笑的使命，同时其中涉及到了一些关于朝鲜和韩国的关系的话题，有些人可能无所谓，有些人可能较为排斥。

总体而言，如果想看一部能让你放松的喜剧电影，那么这部电影非常推荐。

共助 2: 国际

图表分布

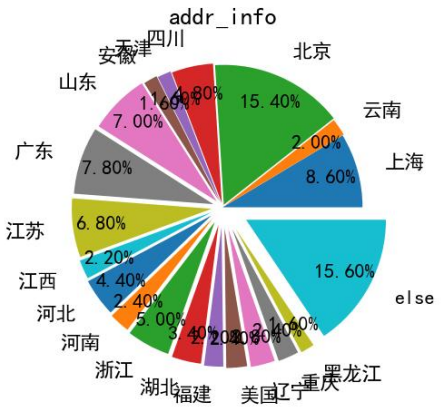

图 14: 观众评论地址分布

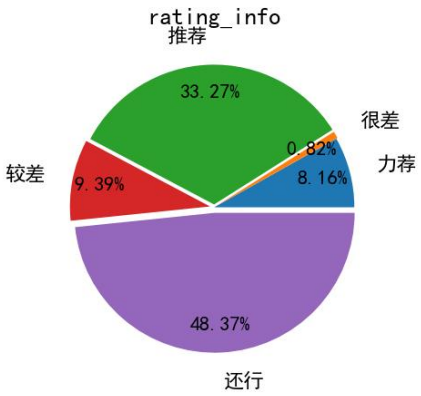

图 15: 观众评论分数分布

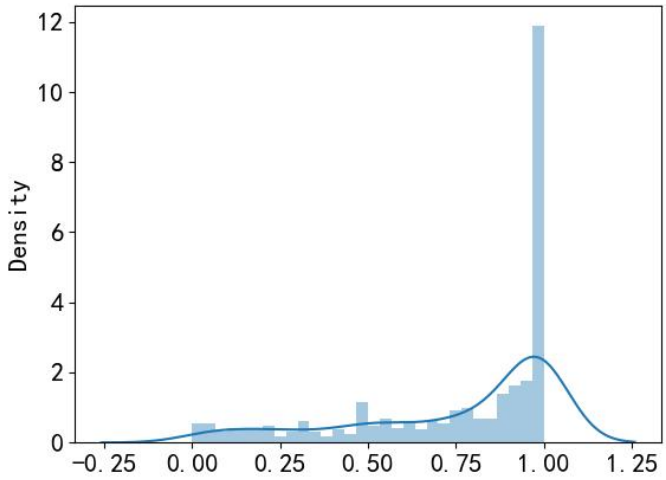

图 16: 影片评论情感分数分布

Keywords

正面评论和负面评论关键词

pos_word: 玄彬:99, 动作:63, 电影:60, 第一部:53, 韩国:51, 剧情:42, 搞笑:34, 喜剧:33, 林允儿:30, 真的:28, 一部:28, 好看:27, 允儿:25, 不错:25, 最后:24, 共助:23, 角色:22, 美国:22, 爆米花:21, 花痴:20, 感觉:20, 丹尼尔:20, 帅哥:20, 动作片:19, 商业片:19, 场面:18, 故事:17, 看到:16, 商业:16, 小姨子:16

neg_word: 玄彬:33, 韩国:21, 电影:13, 第一部:11, 爆米花:11, 剧情:10, 允儿:9, 真的:9, 搞笑:9, 丹尼尔:8, 动作:8, 无聊:8, 不能:6, 娱乐性:6, 反派:6, 合格:6, 一部:6, 好看:5, 帅哥:5, 南北:5, 商业片:5, 林允儿:5, 有点:5, 美国:5, 帅气:4, 没意思:4, 好笑:4, 感觉:4, 好帅:4, 差不多:4

综合评价

总体而言,评价中等,好评率 41.43%, 中评占比 48.37%, 差评占比 10.21%。粉丝主要分布于北京。正面评价主要关键词包括玄彬、动作、电影、韩国、剧情、搞笑。负面评论关键词主要包括玄彬、韩国、电影、爆米花。是一部一部动作类型的爆米花电影，主演关键词占比高，主要卖点是演员。

总而言之，是一部中规中矩的电影，适合主演的粉丝观看。

黑亚当

图表分布

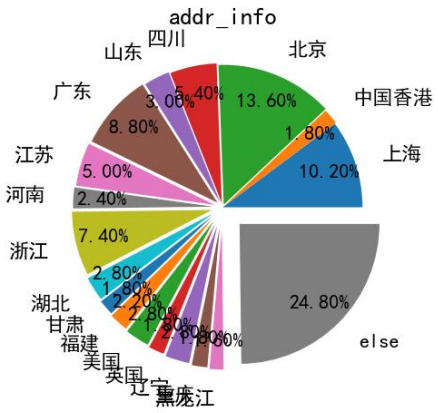

图 17: 观众评论地址分布

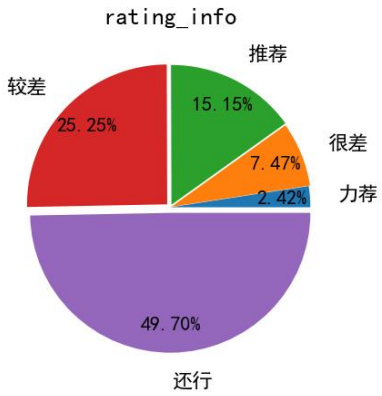

图 18: 观众评论分数分布

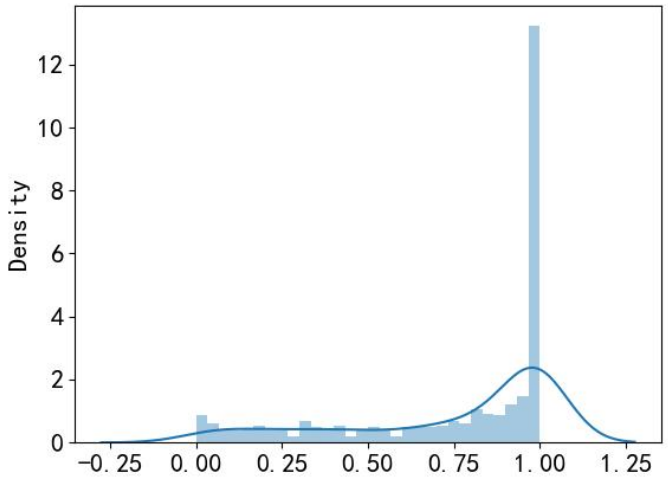

图 19: 影片评论情感分数分布

Keywords

正面评论和负面评论关键词

pos_word: 电影:129, 英雄:103, 亚当:102, 博士:73, 特效:68, 剧情:63, 角色:60, 正义:55, 命运:54, 超英:52, 爆米花:49, 强森:45, 最后:39, 巨石:38, 超级:38, 漫威:36, 故事:33, 真的:33, 有点:30, 协会:29, 超人:29, 看到:27, 已经:26, 不错:26, 彩蛋:26, 这种:25, 布鲁斯南:23, 一部:23, 感觉:22, 这部:21

neg_word: 特效:25, 剧情:22, 电影:19, 超英:15, 真的:12, 漫威:10, 这种:9, 有点:9, 巨石:8, 好看:8, 最后:7, 感觉:7, 彩蛋:7, 无聊:7, 亚当:7, 强森:6, 反派:6, 觉得:6, 知道:6, 毫无:6, 爆米花:6, 电影院:5, 超级:5, 唯一:5, 超人:5, 意思:5, 片子:5, 编剧:5, 演技:5, 鬼子:4

综合评价

整体评价中等偏下, 好评占比 17.57%, 中评占比 49.70%, 差评占比 32.73%, 粉丝主要分布于北京、上海。正面评论中主要关键词为英雄、亚当、博士、特效、剧情、角色，负面评论中主要关键词为特效、剧情、电影、超英。是一部典型的超级英雄影片，有着比较充足的特效，但是剧情存在一定问题。

总体而言，不是很推荐观看，仅仅适合想要放松心情的人群。

火山挚恋

图表分布

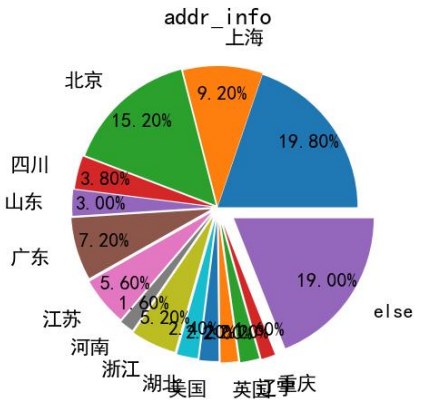

图 20: 观众评论地址分布

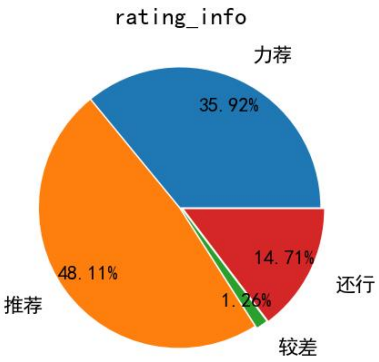

图 21: 观众评论分数分布

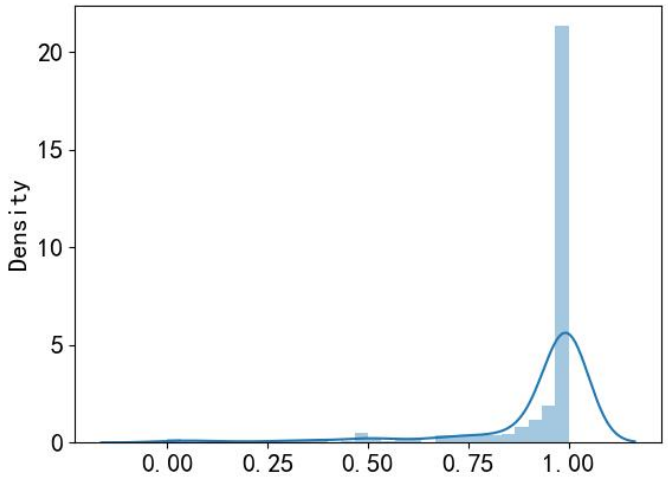

图 22: 影片评论情感分数分布

Keywords

正面评论和负面评论关键词

pos_word: 火山:329, 人类:110, 浪漫:92, 纪录片:68, 影像:59, 震撼:47, 热爱:46, 人生:46, 岩浆:43, 自然:43, 生命:41, 素材:41, 地球:38, 爱情:35, 电影:34, 真的:34, 两人:33, 一生:31, 渺小:31, 极致:31, 两个:30, 科学家:30, 看到:29, 剪辑:27, 莫里斯:27, 一起:26, 夫妇:25, 危险:25, 一种:25, 火山爆发:24

neg_word: 火山:16, 浪漫:8, 北影:6, 旁白:5, 震撼:4, 時間:4, 影像:4, 极致:3, 无聊:3, 喜欢:3, 故事:3, 他們:3, 今年:3, 觉得:3, 四星:2, 科学:2, 知道:2, 野兽:2, 完全:2, 一点:2, 热爱:2, 素材:2, 纯粹:2, 真的:2, 配乐:2, 两个:2, 跳舞:2, 活着:2, 这部:2, 内容:2

综合评价

整体评价非常好，好评率达到了 84.03%, 中评占比 14.71%, 差评仅占 1.26%, 主要观影人群在北京，好评非常多，在这些好评中，大多都提到了浪漫的关键词，纪录片的手法和影像都给人很大的震撼，也能给人以关于人生和自然的思考。差评中，主要是部分人会觉得有一些无聊。一部富有启迪性的纪录片，献给火山爱好者和浪漫主义者的情书，

整体而言，非常推荐，可以给人以很深刻的思考。

福尔摩斯小姐: 伦敦厄运

图表分布

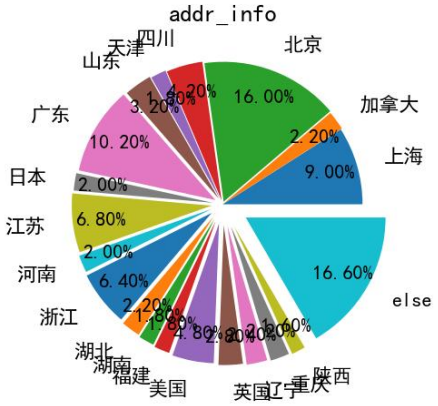

图 23: 观众评论地址分布

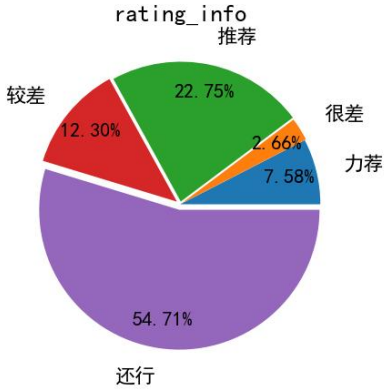

图 24: 观众评论分数分布

通过 zotero-better-bibtex 插件来设置

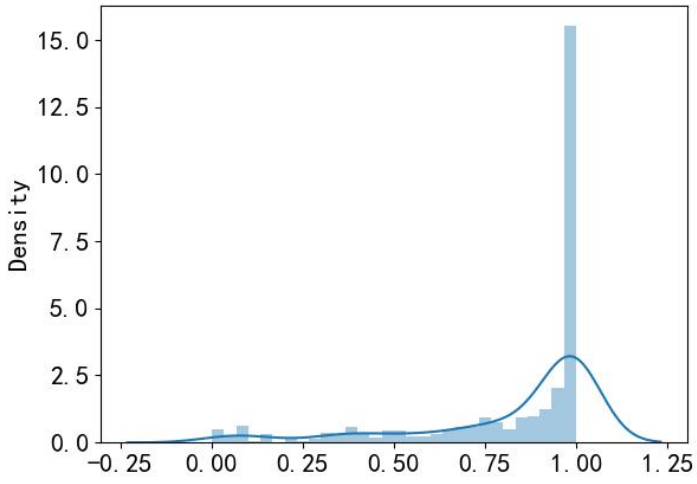

图 25: 影片评论情感分数分布

Keywords

正面评论和负面评论关键词

pos_word: 福尔摩斯:104, 第一部:97, 女性:85, 推理:76, 电影:60, 华生:52, 好看:51, 里亚蒂:49, 黑人:46, 剧情:43, 故事:39, 真的:38, 最后:37, 有点:35, 女权:32, 喜欢:32, 正确:32, 感觉:30, 可爱:28, 很多:27, 小姐:27, 一部:26, 政治:26, 部分:25, 角色:25, 不错:24, 女主:24, 夏洛克:22, 女工:21, 系列:19

neg_word: 第一部:20, 黑人:16, 里亚蒂:14, 好看:13, 华生:12, 无聊:9, 真的:7, 爆米花:6, 一部:6, 夏洛克:6, 最后:6, 女权:5, 一点:5, 火柴:5, 剧情:5, 亨利:4, 女工:4, 有点:4, 黑女:4, 女性:4, 還是:4, 很多:4, 三星:4, 可爱:3, 无法:3, 接受:3, 看看:3, 事件:3, 小妹:3, 星星之火:3

综合评价

总体评价中等, 好评占比 30.33%, 中评占比 54.71%, 差评占比 14.96%, 观众主要分布在北京、广东。正面评价主要关键词包括福尔摩斯、女性、推理，负面评价主要关键词包括黑人、莫里亚蒂等。综上看来，作为一部福尔摩斯电影，喜欢福尔摩斯和悬疑、推理的人群可以一看，但是出于政治正确添加的角色设定一定程度上影响了本片的评价

总体而言，对于福尔摩斯爱好者值的一看。

珀尔

图表分布

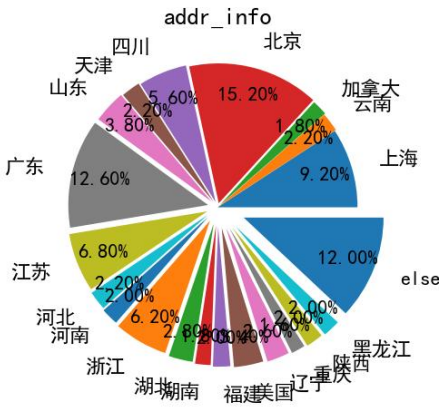

图 26: 观众评论地址分布

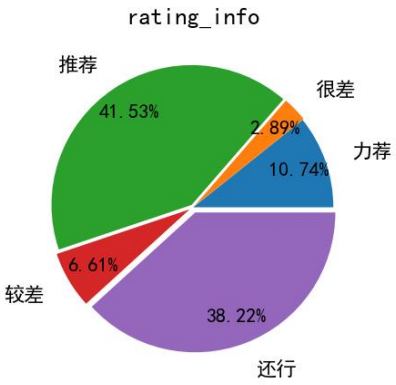

图 27: 观众评论分数分布

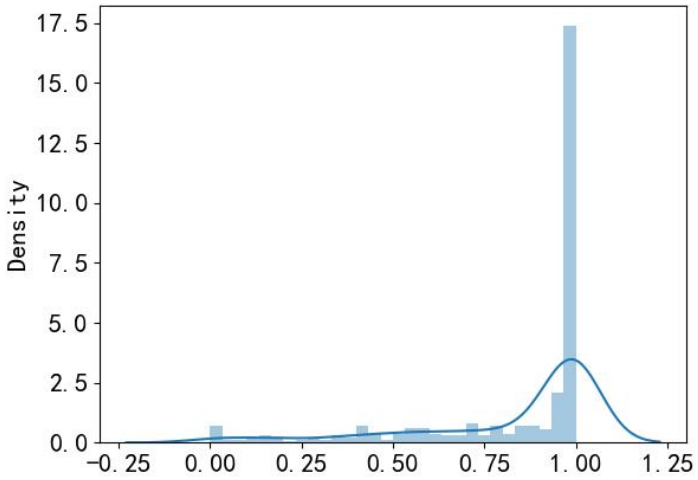

图 28: 影片评论情感分数分布

Keywords

正面评论和负面评论关键词

pos_word: 女主:66, 电影:65, 独白:58, 最后:58, 演技:54, 米娅:52, 高斯:50, 恐怖片:45, 真的:42, 恐怖:40, 表演:39, 压抑:37, 复古:34, 人物:34, 角色:33, 故事:32, 感觉:30, 长镜头:30, 这部:29, 家庭:28, 结尾:28, 镜头:28, 珀尔:27, 母亲:26, 农场:25, 血腥:24, 导演:23, 主演:23, 一部:23, 这种:22

neg_word: 演技:12, 最后:10, 恐怖:9, 电影:8, 真的:7, 好看:6, 高斯:5, 女主:5, 可能:5, 变态:5, 一点:4, 太好了:4, 知道:4, 疯批:4, 疫情:4, 一部:4, 不要:4, 故事:4, 有点:4, 镜头:3, 危笑:3, 剧情:3, 神经病:3, 喜欢:3, 这部:3, 已经:3, 恐怖片:3, 家庭:3, 完全:3, 开关:3

综合评价

综合评价值得一看, 好评率 52.27%, 中评占比 38.22%，差评占比 9.50%, 观众主要分布于北京、广东。正面评论关键词为女主，独白，最后，演技，负面评论主要关键词为演技，恐怖，变态等。作为恐怖片，好评主要集中在女主最后的独白的演技上，相比之下恐怖的成分却要逊色很多。

总体而言，评价较为可观，值得一看。

新神榜：杨戬

图表分布

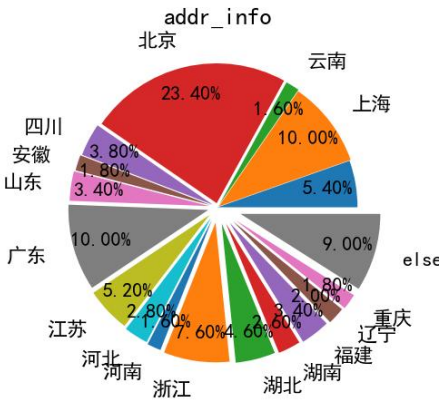

图 29: 观众评论地址分布

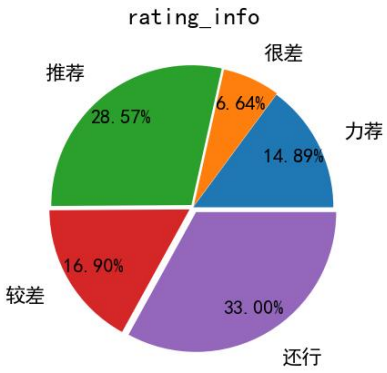

图 30: 观众评论分数分布

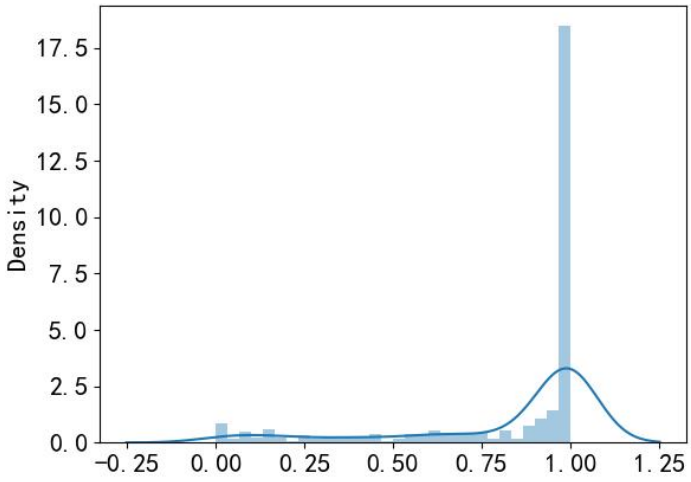

图 31: 影片评论情感分数分布

Keywords

正面评论和负面评论关键词

pos_word: 杨戬:226, 故事:154, 剧情:143, 追光:113, 人物:101, 电影:88, 画面:86, 真的:86, 沉香:86, 动画:79, 特效:75, 朋克:61, 角色:59, 劈山:56, 编剧:54, 美术:51, 哪吒:50, 最后:50, 设定:48, 申公豹:45, 剧本:45, 觉得:44, 感觉:44, 中国:43, 场景:43, 有点:39, 蒸汽:38, 救母:38, 不能:37, 神话:37

neg_word: 剧情:28, 特效:23, 编剧:22, 故事:18, 真的:18, 杨戬:17, 追光:17, 画面:12, 沉香:11, 不能:11, 美术:10, 有点:9, 感觉:8, 觉得:7, 最后:7, 无聊:7, 剧本:7, 知道:6, 人物:6, 劈山:6, 好看:6, 求求:5, 希望:5, 东西:5, 恶心:5, 朋克:5, 逻辑:5, 浪费:5, 两个:5, 不好:5

综合评价

整体评价中等偏上,好评占比 43.46%, 中评占比 33.00%, 差评占比 23.54%。观众主要分布在北京、上海、广东，正面评论主要关键词为杨戬、故事、剧情、追光、人物，负面评论主要关键词包括剧情特效。作为一部动画电影，人设到位，主人公杨戬极为突出，特效和美工都受到了欢迎。但是剧情相对有所欠缺。

总体而言，忽略一些剧情上的缺点，值得一看。

总结

综上所述，上述电影可分为以下几个类别:

非常推荐：西线无战事，火山挚恋比较推荐或特定人群可看：名侦探柯南: 万圣节的新娘，6/48, 共助 2：国际, 福尔摩斯小姐：伦敦厄运, 新神榜：杨戬不推荐观看: 危笑，黑亚当

同时，从影片粉丝地域分布可以看出，国内电影业受疫情影响较大，除了几大超一线城市外，其他城市人群的观影热情相对而言受到了较大影响。

5 Future Work

本文仍然存在以下缺陷:

- 由于豆瓣对爬虫的限制，每个电影只能爬取到 600 条左右的评论数据，数据规模不是很大。
- 由于缺乏数据集，使用的是预训练好的模型，缺乏对于电影评论的针对性
- 对于一段评论，可能同时存在多个正负面不同的观点，对一段评论用统一的一个情感因子评价，难以确切的衡量对于评论中某些特定关键词的态度，因此，对于关键词的分析，还不够精确。

所以，在时间充足的情况下，我会考虑如下优化:

- 广泛收集多个网站的数据，自己进行模型的训练
- 对于一段评论的情感分析，考虑上下文的建模，对评论进行细粒度的分析，使得提取出的负面和正面关键词能更具有代表性。[4]

6 Referance

参考文献

[1] Kun Wang, Chengqing Zong, and Keh-Yih Su. Which is More Suitable for Chinese Word Segmentation, the Generative Model or the Discriminative One? In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 2*, pages 827–834. City University of Hong Kong.

[2] Rada Mihalcea and Paul Tarau. TextRank: Bringing Order into Texts. page 8.

[3] Tomoki Ito, Kota Tsubouchi, Hiroki Sakaji, Tatsuo Yamashita, and Kiyoshi Izumi. Word-Level Contextual Sentiment Analysis with Interpretability. 34(04):4231–4238.

[4] Christopher Potts, Zhengxuan Wu, Atticus Geiger, and Douwe Kiela. DynaSent: A Dynamic Benchmark for Sentiment Analysis.