



# 廣東工業大學

## 《文本信息挖掘概论》

### 课程设计

学 院 \_\_\_\_\_ 计算机学院 \_\_\_\_\_

专 业 \_\_\_\_\_ 软件工程 \_\_\_\_\_

班 级 \_\_\_\_\_ 一班 \_\_\_\_\_

学 号 \_\_\_\_\_ 3119005028 \_\_\_\_\_

学生姓名 \_\_\_\_\_ 魏耀辉 \_\_\_\_\_

授课教师 \_\_\_\_\_ 杨易扬 \_\_\_\_\_

2022 年 07 月

## 目录

1 绪论 .....	1
2 数据集抓取 .....	1
2.1 数据抓取方法 .....	1
2.2 数据抓取步骤 .....	1
1. 爬取 B 站弹幕 .....	1
2. 爬虫模拟浏览器发送请求 .....	2
3. 使用正则表达式来过滤弹幕内容 .....	2
2.3 将内容通过词云展示出来 .....	3
3 定制化词典 .....	4
3.1 为何定制化词典 .....	4
3.2 如何定制化词典 .....	4
4 对数据集进行 LDA 降维操作 .....	6
4.1 进行 LDA 降维操作 .....	6
4.2 数据可视化 .....	7
5 对数据进行 word2vec 操作 .....	7
5.1 word2vec 操作 .....	7
6 通过词向量进行人物分析 .....	9
6.1 阿尼亚/阿妮亚人物分析 .....	9
6.2 约尔人物分析 .....	10
6.3 黄昏人物分析 .....	11
6.4 总体分析 .....	12
7 总结与心得体会 .....	12
4.1 总结 .....	12
4.2 心得体会 .....	12
参考文献 .....	13

# 1 绪论

当今环境，日本动漫番剧收到越来越多人的喜爱，B 站作为国内最大的动漫/二次元弹幕网站，吸引了当下许多年轻人的观看，而优秀的漫改番如何吸引到更多的人观看成为了当下的一个问题，本次实验决定以 B 站热门番《间谍过家家》的弹幕为切入点，通过爬取弹幕并对弹幕进行分词分析，来更好的通过关键词来吸引年轻人地观看。通过弹幕进行词向量分析，可以大致得出角色在弹幕的刻画下的样子，比官网的百科给出的人物性格特点要更加真实可信，也可以从侧面反映出该部作品的刻画人物水平。

## 2 数据集抓取

### 2.1 数据抓取方法

本次实验使用 Python 作为实验语言，因为 Python 提供了许多应用于爬虫和数据集处理的库函数，方便调用。本次实验主要爬取 B 站《间谍过家家》的前五集的六月每日弹幕作为基本数据集，并将其汇总称为总弹幕数据集。

### 2.2 数据抓取步骤

#### 1. 爬取 B 站弹幕

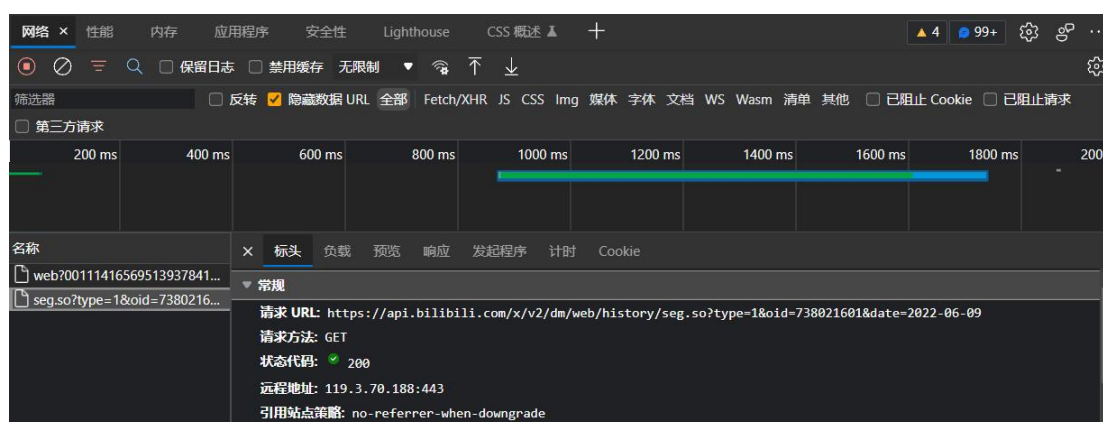


图 1 B 站弹幕存放位置

通过 F12 开发者工具进行分析, B 站弹幕可以通过该请求 URL 以 GET 的方式获取。因此, 本次实验导入 requests 包来进行爬取数据

## 2. 爬虫模拟浏览器发送请求

为了多次爬取数据, 我们需要爬虫模拟浏览器发送 GET 请求来爬取 B 站的弹幕, 所以我们通过 F12 的开发者工具, 来获取头部中的 cookie 等信息来模拟浏览器发送请求。

```
cookie: innersign=0; buvid3=557A3846-3228-F2B1-FE61-7087810E927F94539infoc; i-wanna-go-back=-1; b_ut=7; b_lsid=577E3131_181C9FE4E70; _uuid=515BF210B-355E-85EA-132E-9DC5599941BA95605infoc; buvid4=6A21E434-95CE-D927-F2D7-2FEA9A372C4295168-022070500-Z0rzTPPKU1m+0CTJGajq/Q%3D%3D; fingerprint=fc45c92d75a934cc7e98ed3c39fbe474; sid=5ed6gsgx; buvid_fp_plain=undefined; DedUserID=79988877; DedUserID__ckMd5=c3bf93be6438be9e; SESSDATA=107ffffbc%2C1672503316%2Cfc6c2%71; bili_jct=0eec244e4c66bfadc88ed96bcb859854; bp_video_offset_79988877=679039539361087500; CURRENT_BLACKGAP=0; nostalgia_conf=-1; buvid_fp=fc45c92d75a934cc7e98ed3c39fbe474; CURRENT_FNVAL=4048; b_timer=%7B%22ffp%22%3A%7B%22333.1007.fp.risk_557A3846%22%3A%22181C9FE5270%22%2C%22333.42.fp.risk_557A3846%22%3A%22181C9FE682E%22%2C%22333.337.fp.risk_557A3846%22%3A%22181C9FF2C85%22%2C%22666.25.fp.risk_557A3846%22%3A%22181C9FF3578%22%7D%7D; rpidid=(umm)lm~mu|0J'uYlYmluJl1
origin: https://www.bilibili.com
referer: https://www.bilibili.com/bangumi/play/ep508404?spm_id_from=333.337.0.0
```

图 2 请求头部包含的信息

## 3. 使用正则表达式来过滤弹幕内容

请求返回的内容包含了弹幕的发送时间, 发送的帐号等等信息, 我们只需要发送的弹幕内容, 所以使用正则表达式来进行过滤信息。并写入到文件中。

```
data = re.findall('.*?([\u4e00-\u9f5a]+).*', response.text)
for index in data:
    print(index)
    with open('./弹幕/弹幕1.txt', mode='a', encoding='utf-8') as f:
        f.write(index)
        f.write('\n')
```

图 3 正则过滤信息写入文件

## 2.3 将内容通过词云展示出来

为了将内容以更好的方式展示出来，本次采用了词云来进行展示，词云可以作为一种新型的技术可以作为番剧的封面来吸引年轻人的观看点击

```
import jieba
import wordcloud
import imageio

img = imageio.imread(r'D:\project\python\example.png')
f = open(r'D:\project\python\弹幕.txt', encoding='utf-8')
text = f.read()
text_list = jieba.lcut(text)
string = ''.join(text_list)

wc = wordcloud.WordCloud(
    width=800,
    height=600,
    background_color='white',
    font_path='msyh.ttc',
    mask=img,
    scale=15
)
wc.generate(string)
wc.to_file('wordcloud.png')
```

图 4 词云代码



图 5 词云背景图



图 6 弹幕词云图

## 3 定制化词典

### 3.1 为何定制化词典

在做文本挖掘的时候，首先要做的预处理就是分词，现代分词都是基于字典或统计的分词，而字典分词依赖于词典，即按某种算法构造词然后去匹配已建好的词典集合，如果匹配到就切分出来成为词语。通常词典分词被认为是最理想的中文分词算法。不论什么样的分词方法，优秀的词典必不可少。

基于字典（Dictionary）的方法的优劣：

- 优势：
  - 高效、简单、易解释
- 劣势：
- 歧义，无法识别新词
- 其他解决方案…
  - 定制化字典

为了后续更好地进行主题选取操作以及对比分析，我们需要对弹幕进行准确的分词，而分词依赖于词典，准确的词典能够对弹幕句子进行准确的划分操作，所以针对本次爬取弹幕的定制化词典是必不可少的。

### 3.2 如何定制化词典

基础代码展示，用于生成自定义词典，如表 1 所示：

表 1 生成自定义词典

```
for num in range(1, 6):  
    f = open(f'./弹幕/弹幕{num}.txt', encoding='utf-8')  
    text = f.read()  
    file = open('./词典/自定义词典.txt', mode='a', encoding='utf-8')  
    file.write(text)  
for i in range(1, 20):  
    content = open(f'./词典/自定义词典.txt', encoding='utf-8').read()
```

```

tags = jieba.analyse.extract_tags(content, topK=10000)
for index in tags:
    with open('./词典/dict.txt', mode='a', encoding='utf-8') as f:
        f.write(index)
        f.write('\n')

dict1 = open('./词典/自定义词典.txt', encoding='utf-8')
text1 = dict1.read()

f = open('./词典/dict.txt', mode='a', encoding='utf-8')
f.write(text1)
f.close()

f = open('./词典/自定义词典.txt', encoding='utf-8')
text2 = f.read()

tags = jieba.analyse.extract_tags(text2, topK=10000)
for index in tags:
    with open(f'./词典/dict{i}.txt', mode='a', encoding='utf-8') as f:
        f.write(index)
        f.write('\n')

```

词典作用展示：

```

word = jieba.cut('阿尼亚女鹅是唧唧的好女鹅，今天也是优雅的一天，哇酷哇酷')
print("|".join(word))
jieba.load_userdict('./词典/自定义词典.txt')
word_list = jieba.cut('阿尼亚女鹅是唧唧的好女鹅，今天也是优雅的一天，哇酷哇酷')
print("|".join(word_list))

```

图 7 比较句子分词

首先使用 jieba 自带的词典进行分词操作，然后倒入我们自己自定义的词典来进行句子的分词操作。结果如图 8 所示：

```

阿|尼亚|女鹅是|唧唧|的|好|女鹅|，|今天|也|是|优雅|的|一天|，|哇酷|哇酷
阿尼亚|女鹅|是|唧唧|的|好|女鹅|，|今天|也|是|优雅|的|一天|，|哇酷哇酷

```

图 8 句子分词比较结果

分析：阿尼亚为人名，录入到了自定义词典中，女鹅也录入到字典当中，哇酷哇酷同样录入到字典中，划分正确，但由于弹幕存在随机性与特殊性，所以也会错误录入像优雅的一这种错误的单词，这一点有待改进。

## 4 对数据集进行 LDA 降维操作

### 4.1 进行 LDA 降维操作

基础代码展示，用于对数据进行降维处理以及可视化操作，如表 2 所示：

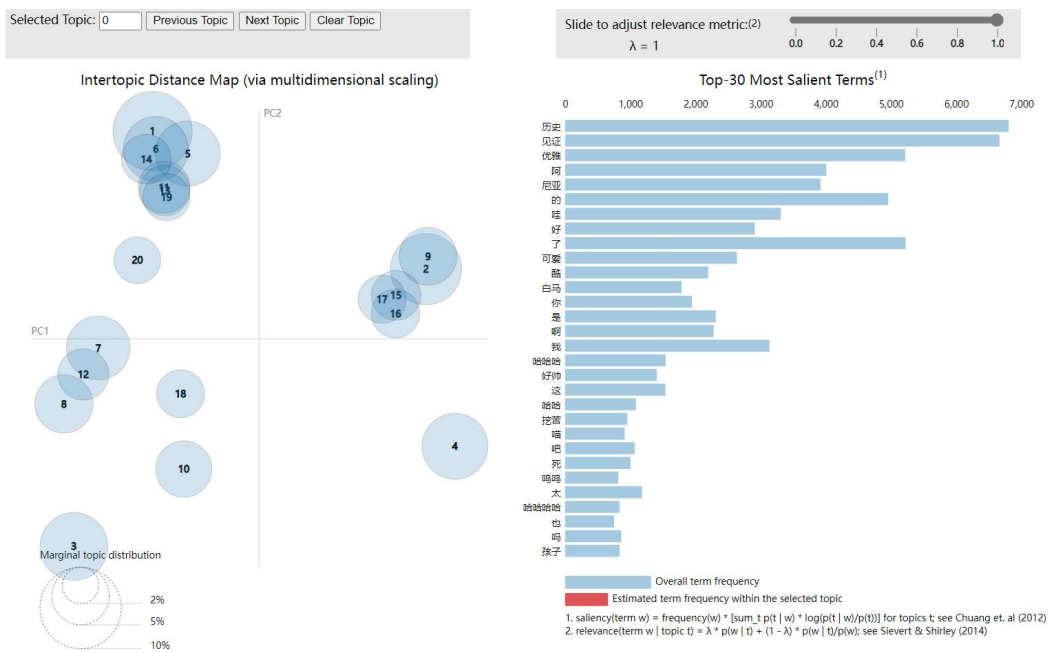
表 2 LDA 降维基础代码

```
from gensim import corpora, models
import jieba.posseg as jp, jieba
import pyLDAvis.gensim_models
f = open('./弹幕/弹幕 1.txt', encoding='utf-8')
texts = []
for line in f:
    texts.append(line.strip())
word_list = []
for text in texts:
    words = [w.word for w in jp.cut(text)]
    word_list.append(words)
dictionary = corpora.Dictionary(word_list)
corpus = [dictionary.doc2bow(words) for words in word_list]
lda = models.ldamodel.LdaModel(corpus=corpus, id2word=dictionary,
num_topics=20,passes=60)
d = pyLDAvis.gensim_models.prepare(lda,corpus,dictionary)
pyLDAvis.save_html(d, 'lda.html')
```



4.2 数据可视化

数据可视化如图 9 所示：



通过该可视化分析文件可以观察这些数据的频率情况和主题的相关度，为数据的观察提供了更加优秀的方法。

5 对数据进行 word2vec 操作

5.1 word2vec 操作

基础代码展示，用于将自定义词典作为语料库进行模型训练，如表 3 所示：

表 3 word2vec 代码展示

```
import gensim.models
from gensim.models.word2vec import Word2Vec
file = open('./词典/自定义词典.txt',encoding='utf-8')
sss=[]
while True:
    ss=file.readline().replace("\n","").rstrip()
```

```

        if ss=="":
            break

        s1=ss.split(" ")
        sss.append(s1)
file.close()
model = Word2Vec(vector_size=200,workers=5,sg=1)
model.build_vocab(sss)
model.train(sss,total_examples=model.corpus_count, epochs=model.epochs)
model.save('./gensim_w2v_sg0_model')
new_model = gensim.models.Word2Vec.load('gensim_w2v_sg0_model')
sim_words = new_model.wv.most_similar(positive=['哇库哇库'])
for word,similarity in sim_words:
    print(word,similarity)
print(model.wv['哇酷哇酷'])

```

训练输出‘哇库哇库’相近的词语和概率如图 10 所示：

```

休 0.24723123013973236
我不知道为什么我感觉这个服装店老板的眼神有点怪异 0.24366602301597595
突然发现这个女的声音像宝可梦里的武藏 0.23884637653827667
这可是专家号 0.23601117730140686
可爱捏 0.23522867262363434
呜呜呜呜我女儿 0.232460156083107
神里 0.23079296946525574
这句话让我想到了天皇 0.2227657288312912
好羞耻 0.22261261940002441
衣服跟 0.22077405452728271

```

图 10 输出‘哇库哇库’相近的词语和概率图

## 6 通过词向量进行人物分析

### 6.1 阿尼亚/阿妮亚人物分析

阿妮亚相近词语和概率代码如图 11 所示：

```
sim_words = new_model.wv.most_similar(positive=['阿妮亚'])  
for word, similarity in sim_words:  
    print(word, similarity)
```

图 11 阿妮亚相近词语和概率代码

阿妮亚相近词语和概率如图 12 所示：

```
坑爹了属于是 0.2263168841600418  
优雅至极 0.21524634957313538  
看得我血糖都高了 0.2149854451417923  
棋逢对手 0.21193112432956696  
用完就扔 0.21138536930084229  
您太优雅了 0.2113761007785797  
经费燃烧 0.2080463021993637  
公主 0.2078900784254074  
阿尼亚啃大瓜 0.20625081658363342  
太 0.20598632097244263
```

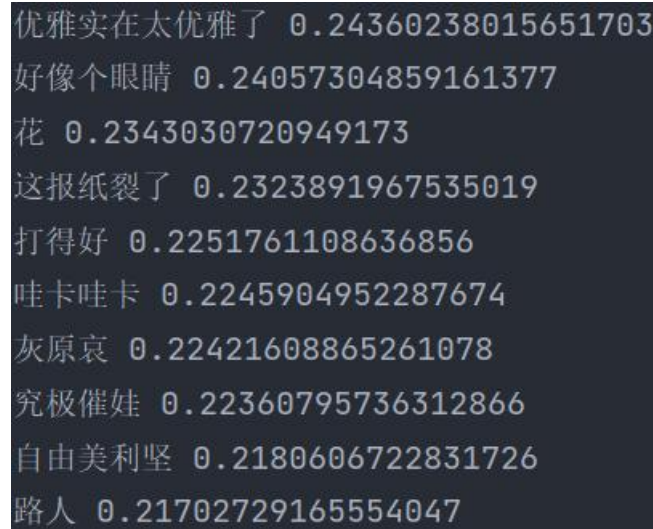
图 12 阿妮亚相近词语和概率

阿尼亚相近词语和概率代码如图 13 所示：

```
sim_words = new_model.wv.most_similar(positive=['阿尼亚'])  
for word, similarity in sim_words:  
    print(word, similarity)
```

图 13 阿尼亚相近词语和概率代码

阿尼亚相近词语和概率如图 14 所示：



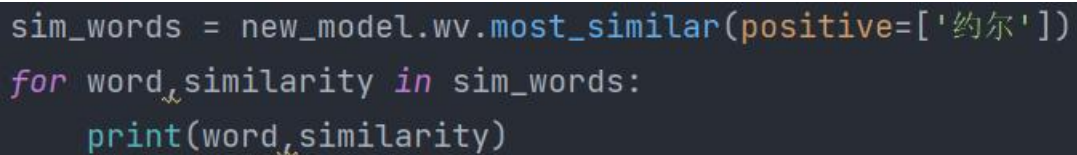
优雅实在太优雅了	0.24360238015651703
好像个眼睛	0.24057304859161377
花	0.2343030720949173
这报纸裂了	0.2323891967535019
打得好	0.2251761108636856
哇卡哇卡	0.2245904952287674
灰原哀	0.22421608865261078
究极催娃	0.22360795736312866
自由美利坚	0.2180606722831726
路人	0.21702729165554047

图 14 阿尼亚相近词语和概率

分析：可以由相近词语来推断主角阿尼亚人物的分析①坑爹了属于是，阿尼亚因为多次的犯错导致父亲需要帮其背锅，引起了观众的共鸣，所以坑爹的形容十分贴切②公主，阿尼亚作为黄昏和约尔的女儿，像公主一般被宠爱。③阿尼亚啃大瓜，阿尼亚作为超能力者，能够听到别人的心声，所以也有了瓜王的称号。④灰原哀，阿尼亚设定为刚上伊甸学院（小学）的学生，样子类似灰原哀也是符合形象的。

## 6.2 约尔人物分析

约尔相近词语和概率代码如图 15 所示：



```
sim_words = new_model.wv.most_similar(positive=['约尔'])
for word, similarity in sim_words:
    print(word, similarity)
```

图 15 约尔相近词语和概率代码

约尔相近词语和概率如图 16 所示：

```
我在想什么 0.25201648473739624
可爱 0.2451586127281189
出门 0.23578770458698273
尼古拉斯 0.23159077763557434
南宫阿尼亚 0.22530390322208405
大声密谋 0.22008460760116577
桌子 0.21944595873355865
地板上有移动的痕迹 0.2076694667339325
吾 0.20662455260753632
舰 0.20625202357769012
```

图 16 约尔相近词语和概率

分析：可以由相近词语来推断主角约尔人物的分析①我在想什么，约尔经常在剧中会想乱七八糟的东西，引起了观众的共鸣，所以形容十分贴切②可爱，约尔同事作为杀手和阿尼亚“母亲”，间谍“黄昏”的妻子，在生活中表现的极为可爱动人。

### 6.3 黄昏人物分析

黄昏相近词语和概率代码如图 17 所示：

```
sim_words = new_model.wv.most_similar(positive=['黄昏'])
for word, similarity in sim_words:
    print(word, similarity)
```

图 17 黄昏相近词语和概率代码

黄昏相近词语和概率如图 18 所示：

```
Traceback (most recent call last):
  File "D:\Project\python\demo04.py", line 18, in <module>
    sim_words = new_model.wv.most_similar(positive=['黄昏'])
  File "D:\Project\python\venv\lib\site-packages\gensim\models\keyedvectors.py", line 842, in most_similar
    mean = self.get_mean_vector(keys, weight, pre_normalize=True, post_normalize=True, ignore_missing=False)
  File "D:\Project\python\venv\lib\site-packages\gensim\models\keyedvectors.py", line 519, in get_mean_vector
    raise KeyError(f"Key '{key}' not present in vocabulary")
KeyError: "Key '黄昏' not present in vocabulary"
```

图 18 黄昏相近词语和概率

分析：主角黄昏相近词语在自定义词典中未出现

## 6.4 总体分析

词向量对人物性格的描写对于约尔和阿尼亚是较为突出的，而对于黄昏来说的是十分不明显的，我猜测原因是因为①女性角色与孩子角色相较于男性角色更受到观众的青睐以及弹幕留言②黄昏在目前番剧中的描述只有优雅等是较为让人印象深刻的，所以不如阿尼亚和约尔的突出。

# 7 总结与心得体会

## 4.1 总结

本次课程设计主要使用了 Python 作为开发工具，使用 jieba 库来进行自定义词典以及分词，使用 gensim 来进行 LDA 以及 word2vec 操作，本次课程设计主要通过爬取 B 站《间谍过家家》的弹幕，以及对弹幕进行分词处理，并对数据降维，向量化操作来寻找人物性格的刻画与观众弹幕的描述的相关性

## 4.2 心得体会

通过本次的课程设计，让我学习到了文本信息挖掘的基础理论知识与实践知识，初步了解掌握了如何进行文本信息的处理与分析，通过自选命题，发现了可以通过弹幕来反映一部番剧对人物的刻画是否深刻且容易令观众产生共鸣的。

## 参考文献

- [1]用 Python 一招爬取 B 站弹幕，源码已附文末  
<https://zhuanlan.zhihu.com/p/449193667>
- [2]Python jieba 分词（使用默认词典，自定义词典，对文件内容分词并统计词频）  
[https://blog.csdn.net/qq\\_44331100/article/details/109531971](https://blog.csdn.net/qq_44331100/article/details/109531971)
- [3]LDA 主题模型简介及 Python 实现  
[https://blog.csdn.net/weixin\\_41168304/article/details/122389948](https://blog.csdn.net/weixin_41168304/article/details/122389948)
- [4]Word2vec 原理及其 Python 实现  
[https://blog.csdn.net/huacha\\_\\_/\\_/article/details/84068653](https://blog.csdn.net/huacha__/_/article/details/84068653)