廣東工業大學

《文本信息挖掘概论》

课程作业

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

作业 1 数据收集与预处理

一、摘要：

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。

二、实验介绍

1.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。

2.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，并将其汇总整。

3.为了展示，本次实验使用词云来展示数据集的内容

三、作业展示：

1. 爬取 B 站弹幕

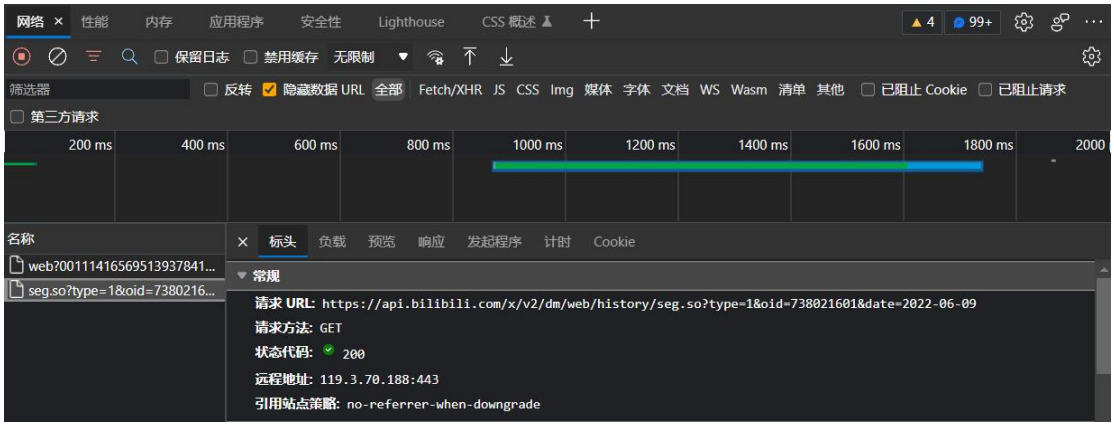


图 1 B 站弹幕存放位置

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。

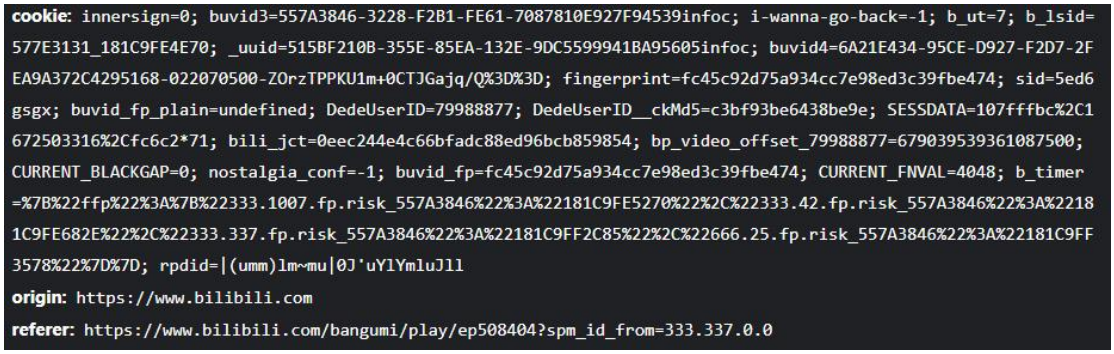


图 2 请求头部包含的信息

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

4. 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。

```python
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

四、作业心得：

通过该次作业，学习了如何爬取 B 站弹幕以及如何绕过 B 站的反爬虫机制来进行持续爬虫。

五、引用：

[1]用 Python 一招爬取 B 站弹幕，源码已附文末 https://zhuanlan.zhihu.com/p/449193667

作业 2 定制化词典

一、摘要：

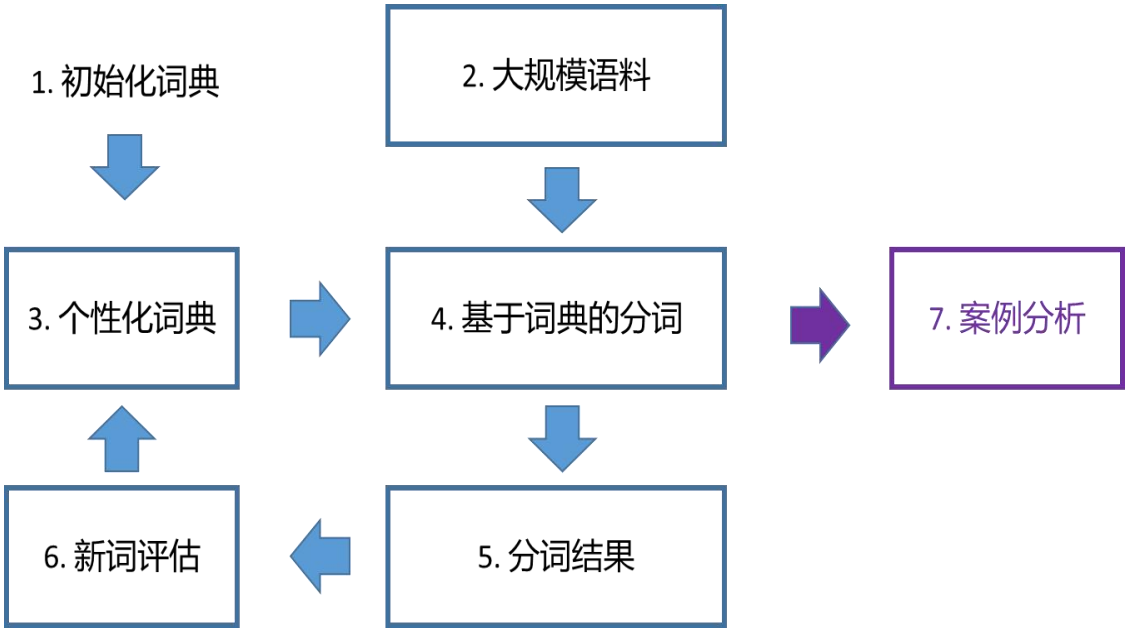
在做文本挖掘的时候，首先要做的预处理就是分词，现代分词都是基于字典或统计的分词，而字典分词依赖于词典，即按某种算法构造词然后去匹配已建好的词典集合，如果匹配到就切分出来成为词语。通常词典分词被认为是最理想的中文分词算法。不论什么样的分词方法，优秀的词典必不可少。

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

二、实验介绍

定制自己的词典，也能采用如下流程：



本次实验计划使用作业 1 爬取的 5 份 B 站弹幕，将第一份弹幕的高频关键词加入到第二份 B 站弹幕文件中，再筛选出第二份弹幕的高频关键词加入到第三份 B 站弹幕文件中，直到生成《间谍过家家》的弹幕词典。

三、作业展示：

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，后续是需要想办法去清除该数据的。

四、作业心得：

通过该次作业，学习了通过 jieba 分词来制作自定义词典，并不断对自定义词典进行优化迭代与数据清除。

五、引用：

[1]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

作业 3 文档的隐含义/含义/主题挖掘

一、摘要：

在做文本挖掘的时候，降维（Dimension Reduction）技术是文本挖掘中常用手段。常用的技术手段有：

1. LSA(Latent semantic analysis)/LSI(Latent semantic index)
2. LDA（Latent Dirichlet Allocation）
3. Word2vec
4. …….

本作业以 LSA/LSI/LDA 等降维为出发点，设计一套基于降维技术的推荐系统框架。

LSA/LSI 的优劣：

- 优势：
  - 高效、简单、易解释
- 劣势：
  - .....

二、实验介绍：

LSA/LSI 的理论基础是 Singular Value Decomposition (SVD), 而 SVD 的基础公式如下（假设 n ≤ m）：

$$A = U\Sigma V^T$$

$$A: n \times m$$

$$U: m \times m$$

$$\Sigma: n \times n$$

$$V: n \times n$$

根据 SVD，谱分析等概念，A 矩阵会有 r 个 rank（序），而利用这 r 中的前 k 个概念➔ low rank 会很好地得到 A 逼近（approximation）。LSI/LSA 的核心思路就是通过 SVD 得到这 k 个概念（降维）。

PLSA/LDA: 算法构建不同，核心思路都是：假设有 k 个概念，使用生成的概念去得到 k 个概念与文档、词之间的关系（EM 算法）。

UMAP[3]：我是谁？我在哪？我为啥会出现在这里。

1) The data is uniformly distributed on Riemannian manifold（黎曼几何）;
2) The Riemannian metric is locally constant (or can be approximated as such);
3) The manifold is locally connected.

核心思路高维 ➔ 低维，不断地优化（EM 算法+负采样）

本次作业通过使用 Gensim 中的 LDA 对数据集进行数据处理，并对数据进行降维处理，采用 pyLDAvis 来对数据进行可视化。

三、作业展示：

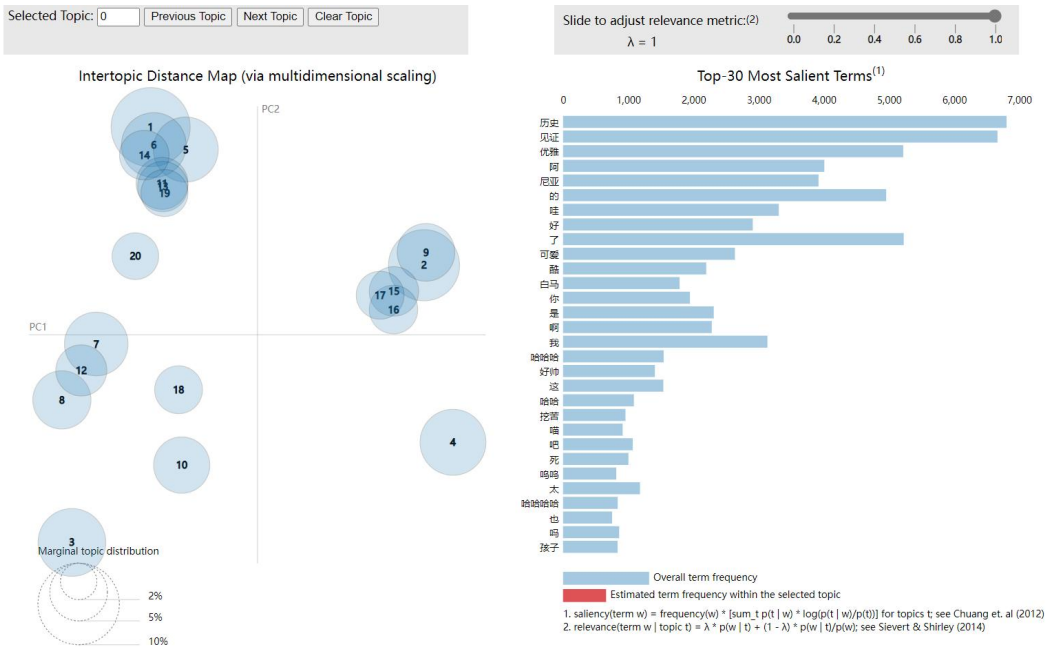
基础代码展示，用于对数据进行降维处理以及可视化操作，如表 2 所示：

表 2 作业 3 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')
```

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

四、作业心得：

通过本次作业让我了解了 LDA 这种让数据集降维的方法，并成功地将弹幕数据集降维，并通过可视化的 html 文件展示出来。

五、引用：

[1]LDA 主题模型简介及 Python 实现 https://blog.csdn.net/weixin_41168304/article/details/122389948

作业 4 随机游走在时代的嗨点上

一、摘要：

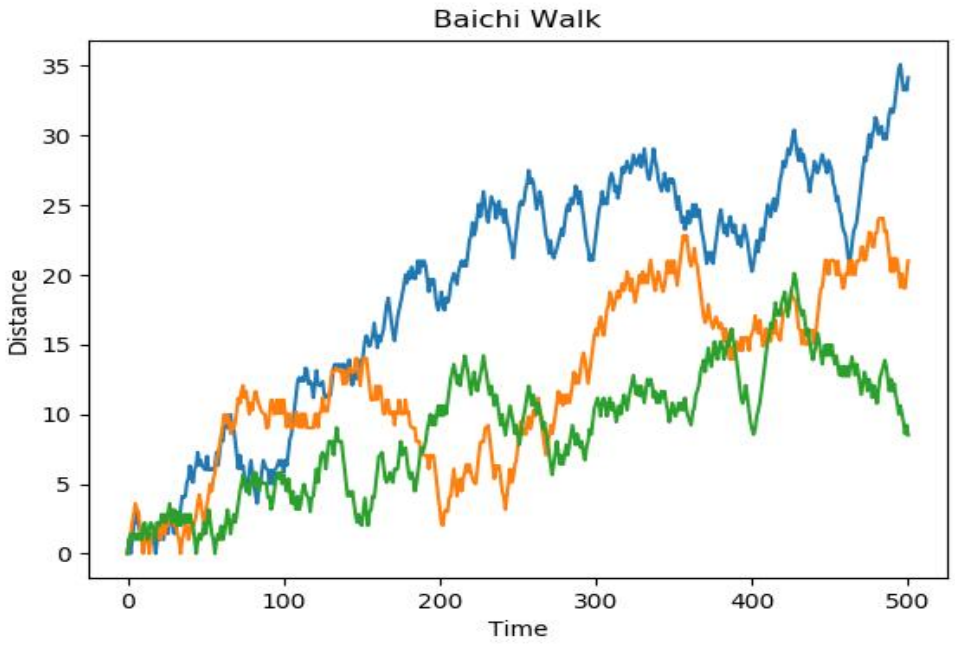
随机游走（random walk）也称随机漫步，随机行走等是指基于过去的表现，无法预测将来的发展步骤和方向。核心概念是指任何无规则行走者所带的守恒量都各自对应着一个扩散运输定律，接近于布朗运动，是布朗运动理想的数学状态，现阶段主要应用于互联网链接分析及金融股票市场中[4]。

随机游走理论:

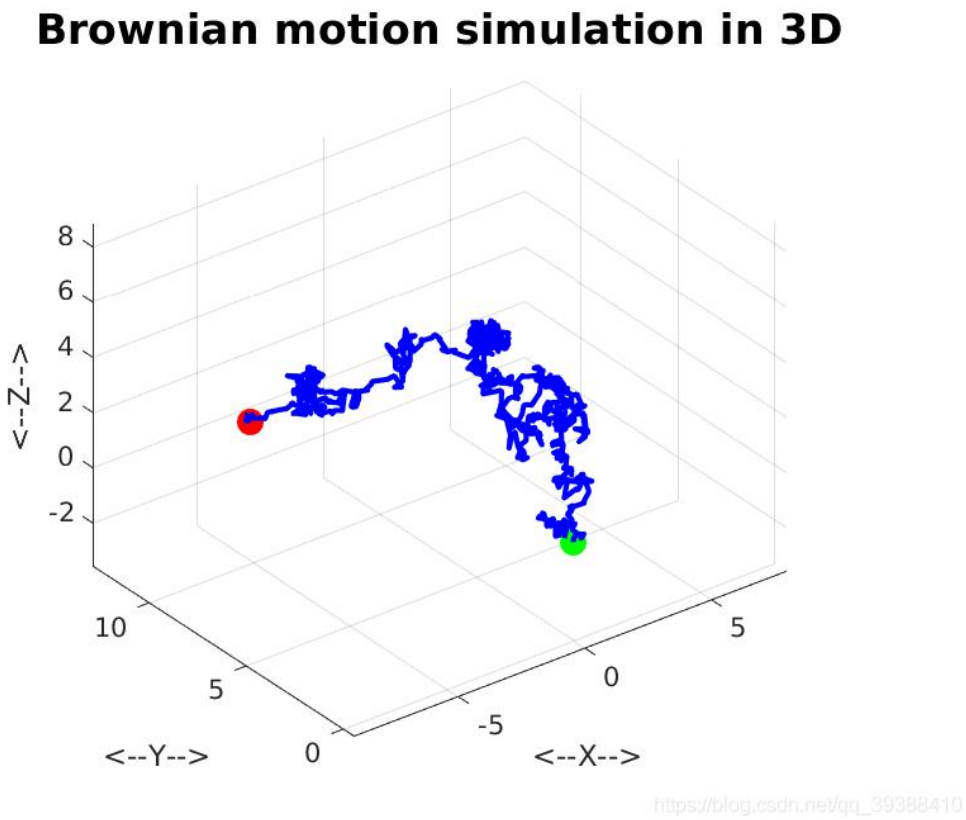
1905 年，英国统计学家 Pearson 在《自然》杂志上公开求解随机游走问题（Random Walk Problem）：如果一个醉汉走路时每步的方向和大小完全随机，经过一段时间之后，在什么地方找到他的可能性最大？

二维简化情况：

假设某地有一个醉汉，每一秒钟会朝“东”，“南”，“西”，“北”中的一个方向走一步，那么这个醉汉在走了 500 步之后会在什么地方？1000 步呢？是不是随着时间的增长，醉汉离原点越来越远呢？用电脑程序来模拟，那么就可以很容易地把醉汉行走的轨迹，醉汉离原点的距离展现出来。

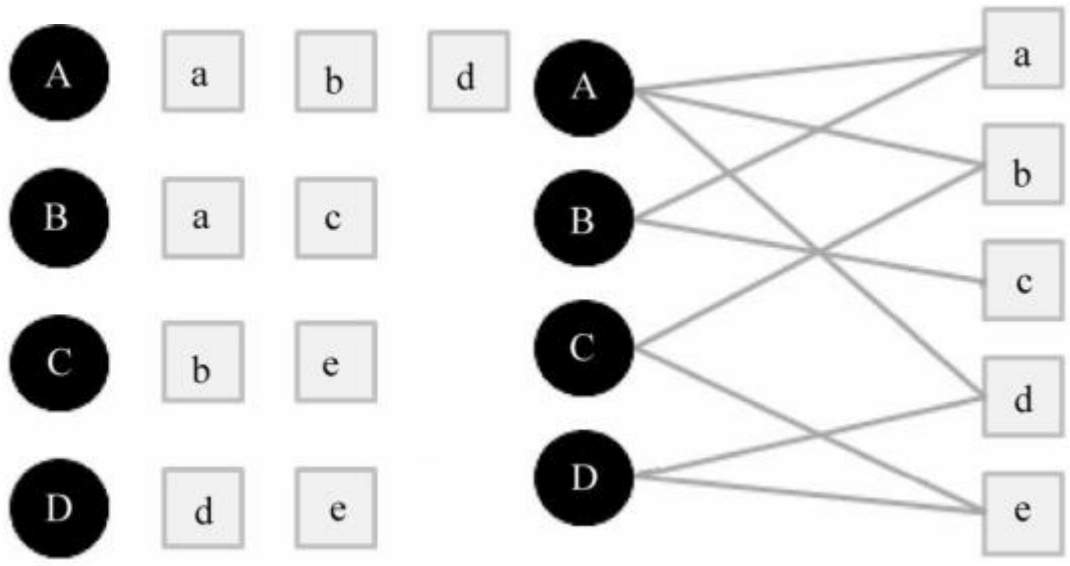


图表 1：醉汉（小白）的游走路线



图表 2：3D 游走

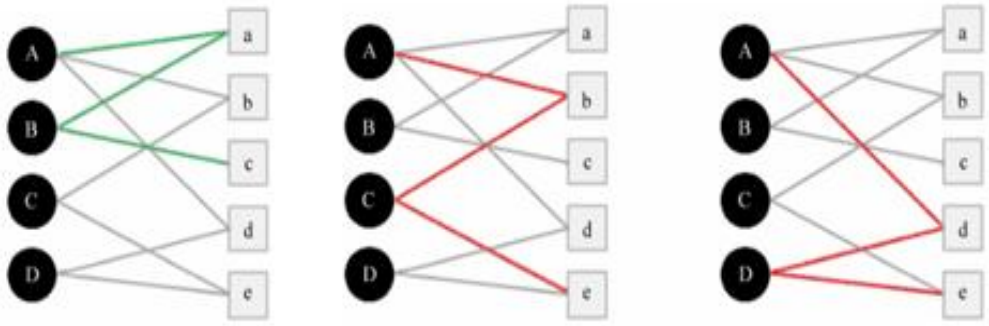
随机游走算法：



图表 3：商品示例

整理用户 A、B、C、D 对于商品 a、b、c、d、e 的喜爱列表，得到二部图（bipartite-graph）。对于上图，A 到达不了 c 和 e(如：A 没买过/点击过 c 和 e)，那么我们如何判断 A 更有可能喜欢 c 还是 e 呢？

1 统计 A 到达 c 和 e 的最短路径，见下图：



图表 4：SVD 的随机游走解释

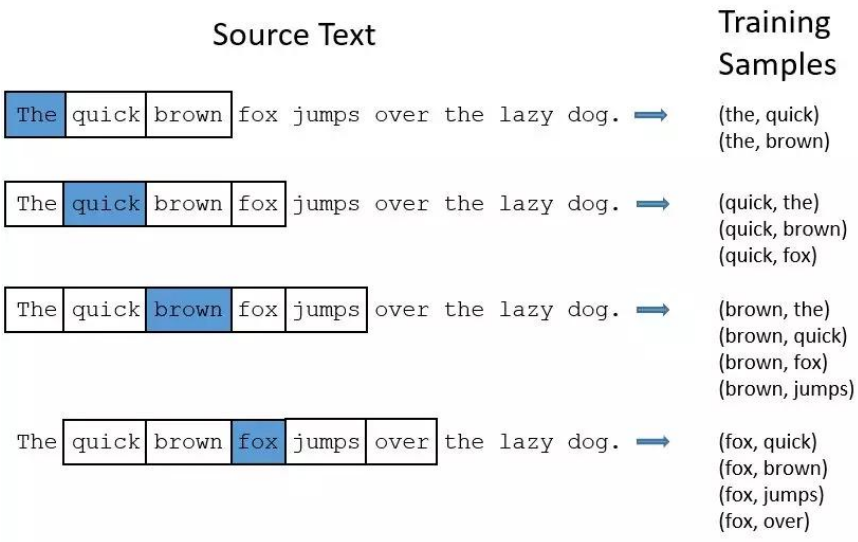
发现 A 到达 c 和 e 的最短路径都是 3，于是进一步判断。

2. 因为 A 到达 c 的最短路径只有 1 条，而到达 e 的有 2 条，那我就有理由判断：A 可能更喜欢 e。这就是随机游走算法的直观表现。那随机游走算法的实现方式是什么呢？谱聚类、 SVD。

Word2vec:本质上也是降维技术，开启了万物皆可 vector 的时代。将上下文关系引入嵌入. 例子：

1. 大家都夸 YYY 帅
2. 大家都夸吴彦祖帅
3. 大家都夸王一博帅
4. 大家都夸于和伟帅

⇨ 嵌入（真理）空间：YYY = 吴彦祖 = 王一博 = 于和伟



图表 5：word2vec 的图示例

其他技术忽略，请参考原文或者知乎等网站自行学习技术细节。

本质上上下文是图中的一种边（edge)

二、实验介绍：

DeepWalk 是基于 word2vec 发展而来的技术，是文本挖掘在网络表示学习领域的一种应用。 作者 Perozzi 等人发现文本语料中词语出现的次数，和在网络中随机游走节点被访问的次数，两者都服从指数分布，因此作者认为在语言模型中表现优异的 Skip-gram 模型(word2vec 中的一种模型)也能移植到网络节点表示学习中。DeepWalk 模型在网络中进行随机游走，可以得到一系列游走节点序列，将这些节点视为单词，节点序列视为句子，就可以将这些数据直接作为 word2vec 模型的输入，从而学习到节点的向量表示。一句话：万物皆可图，万物皆可游走，从而万物皆可 vector。

本次作业使用了 word2vec 来对数据进行操作处理，使用 skip-gram 模型

三、作业展示：

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()
```

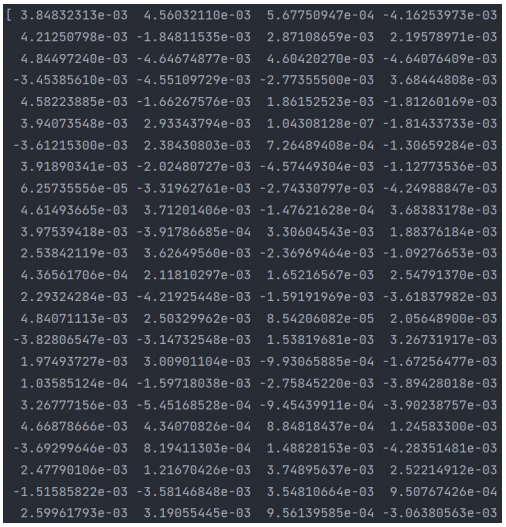
```
model = Word2Vec(vector_size=200,workers=5,sg=1)
model.build_vocab(sss)
model.train(sss,total_examples=model.corpus_count, epochs=model.epochs)
model.save('./gensim_w2v_sg0_model')
new_model = gensim.models.Word2Vec.load('gensim_w2v_sg0_model')
sim_words = new_model.wv.most_similar(positive=['哇库哇库'])
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 0.247231230139732 36
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

输出’哇酷哇酷’的词向量如图 11，12 所示



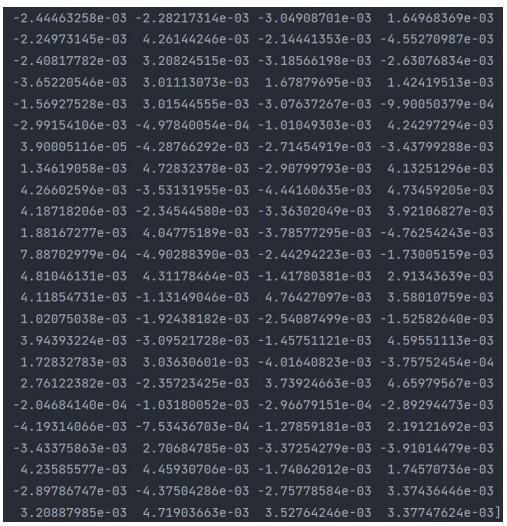


图 11 词向量 1　　图 12 词向量 2

四、作业心得：

本次作业使用了 gensim 中的 models 中的 word2vec 来对数据进行向量化操作，训练了自己的 word2vec 词向量，从词向量和相近词语概率图可以看出，结果有待继续优化，也可能是弹幕的特殊性导致的数据集并不能很好的进行处理

五、引用：

[1]Word2vec 原理及其 Python 实现 https://blog.csdn.net/huacha__/article/details/84068653