



**《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 1](#_Toc32555)

[一、摘要： 1](#_Toc16500)

[二、实验介绍 1](#_Toc20204)

[三、 作业展示： 1](#_Toc27194)

[四、 作业心得： 3](#_Toc19234)

[五、 引用： 3](#_Toc15142)

[作业2 定制化词典 4](#_Toc32555)

[一、摘要： 4](#_Toc29012)

[二、实验介绍 4](#_Toc16588)

[三、 作业展示： 5](#_Toc11851)

[四、 作业心得： 6](#_Toc31823)

[五、 引用： 6](#_Toc4141)

[作业3 文档的隐含义/含义/主题挖掘 7](#_Toc32555)

[一、摘要： 7](#_Toc22145)

[二、 实验介绍： 7](#_Toc19115)

[三、 作业展示： 8](#_Toc29167)

[四、 作业心得： 9](#_Toc9547)

[五、 引用： 9](#_Toc12910)

[作业4 随机游走在时代的嗨点上 1](#_Toc32555)0

[一、 摘要： 10](#_Toc27989)

[二、 实验介绍： 13](#_Toc19933)

[三、 作业展示： 13](#_Toc4088)

[四、 作业心得： 15](#_Toc6208)

[五、 引用： 15](#_Toc26254)

## 

## 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

### 作业展示：

1. 爬取B站弹幕

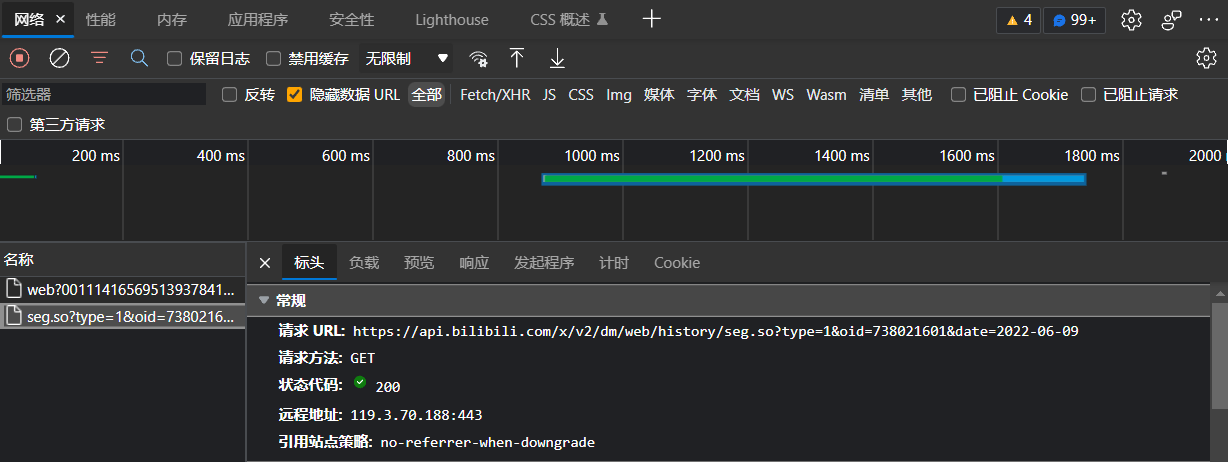


图1 B站弹幕存放位置

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

1. 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。

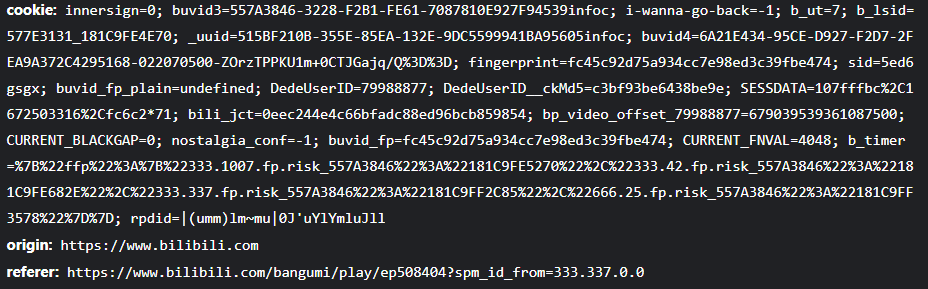


图2 请求头部包含的信息

1. 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。

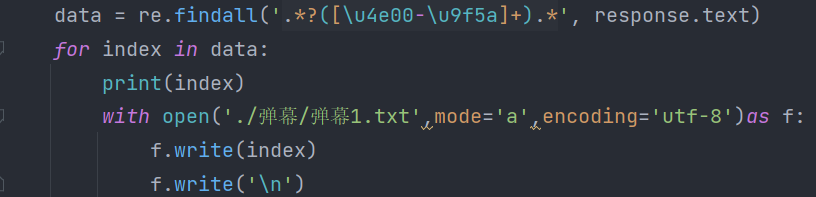


图3 正则过滤信息写入文件

1. 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。

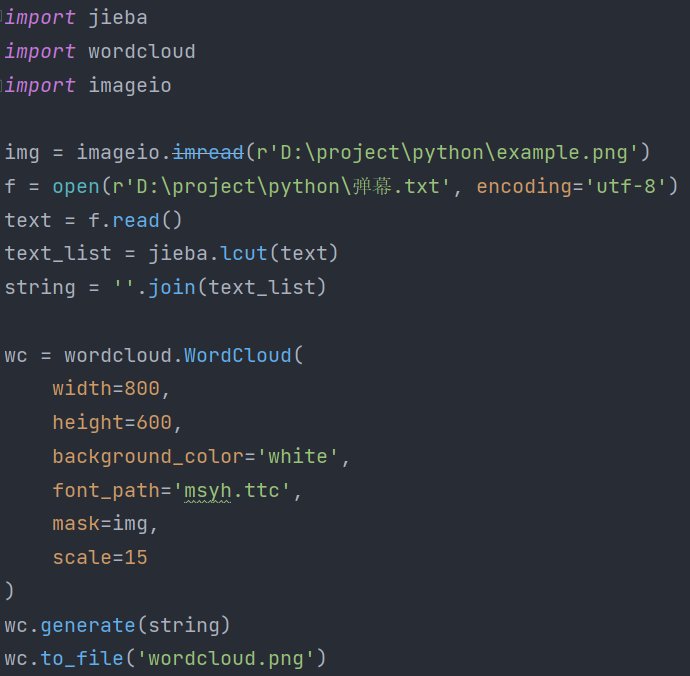


图4 词云代码

图5 词云背景图 图6 弹幕词云图

### 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。

### 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：

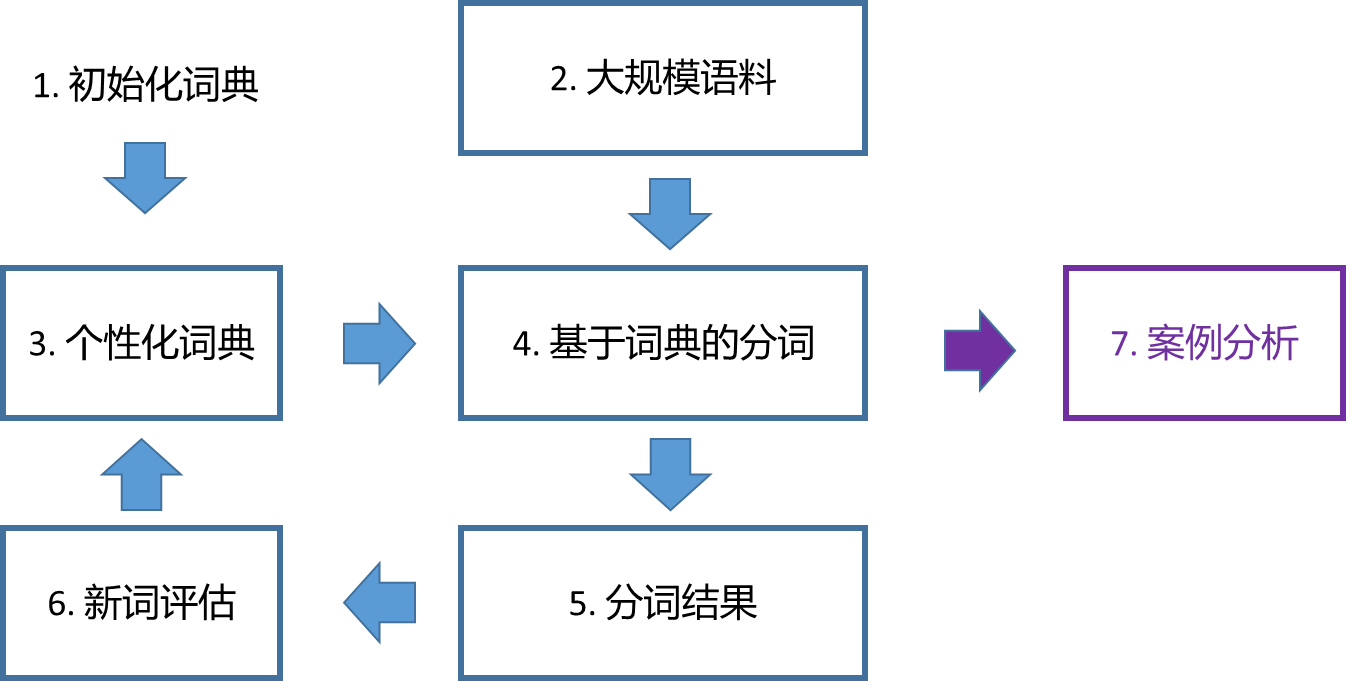
在做文本挖掘的时候，首先要做的预处理就是分词，现代分词都是基于字典或统计的分词，而字典分词依赖于词典，即按某种算法构造词然后去匹配已建好的词典集合，如果匹配到就切分出来成为词语。通常词典分词被认为是最理想的中文分词算法。不论什么样的分词方法, 优秀的词典必不可少。

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。

### 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：

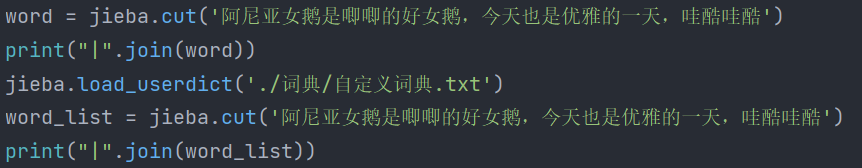


图7 比较句子分词

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：

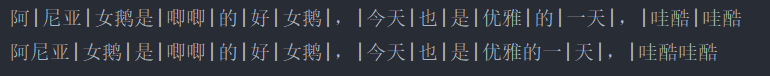


图8 句子分词比较结果

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。

### 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。

### 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：
  + ......

### 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）：

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。

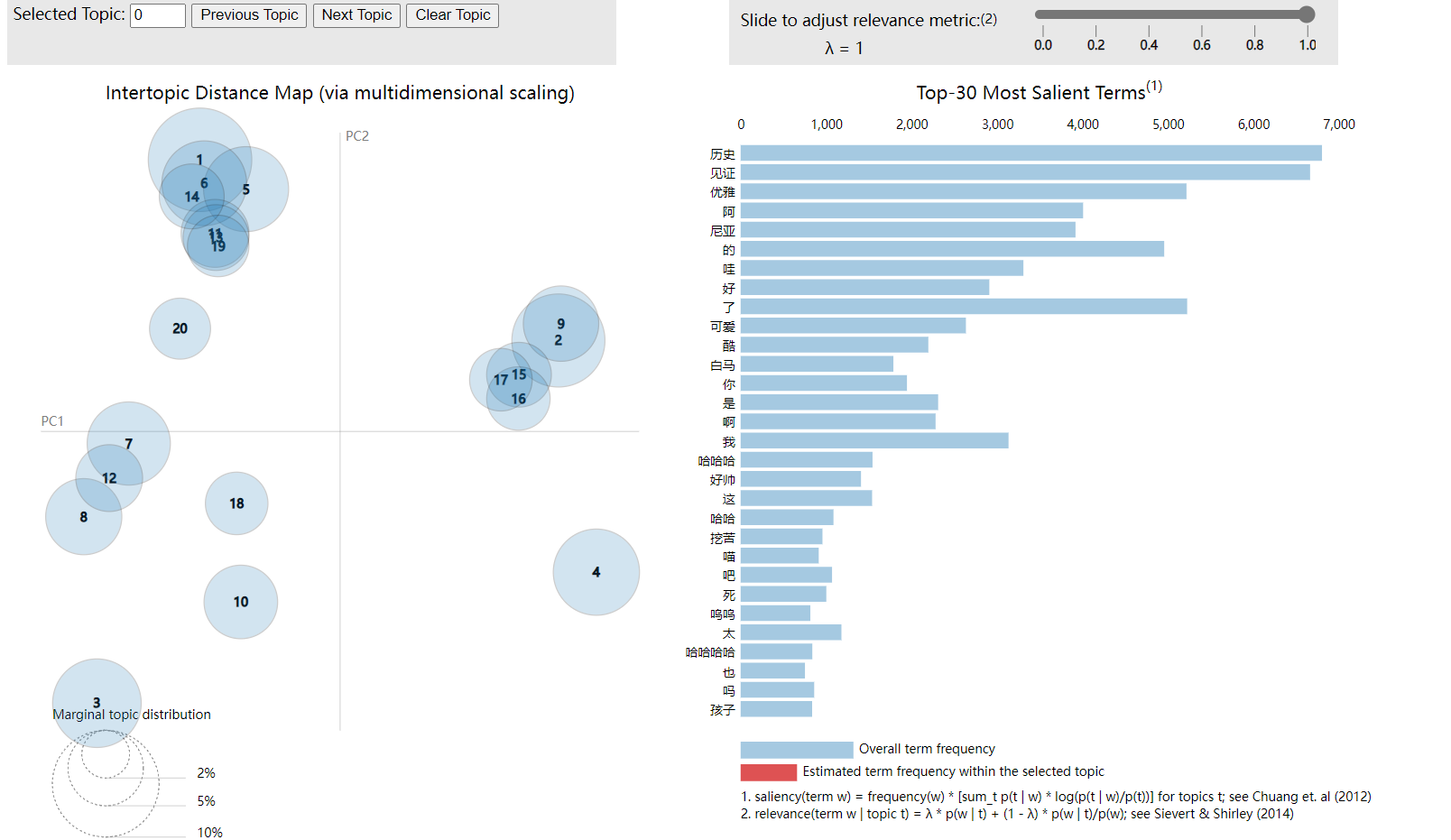
### 作业展示：

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

### 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。

### 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

### 摘要：

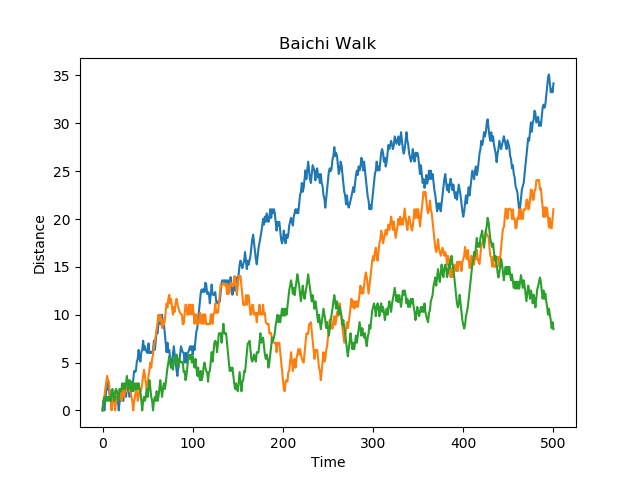
随机游走（random walk）也称随机漫步，随机行走等是指基于过去的表现，无法预测将来的发展步骤和方向。核心概念是指任何无规则行走者所带的守恒量都各自对应着一个扩散运输定律，接近于布朗运动，是布朗运动理想的数学状态，现阶段主要应用于互联网链接分析及金融股票市场中[4]。

随机游走理论:

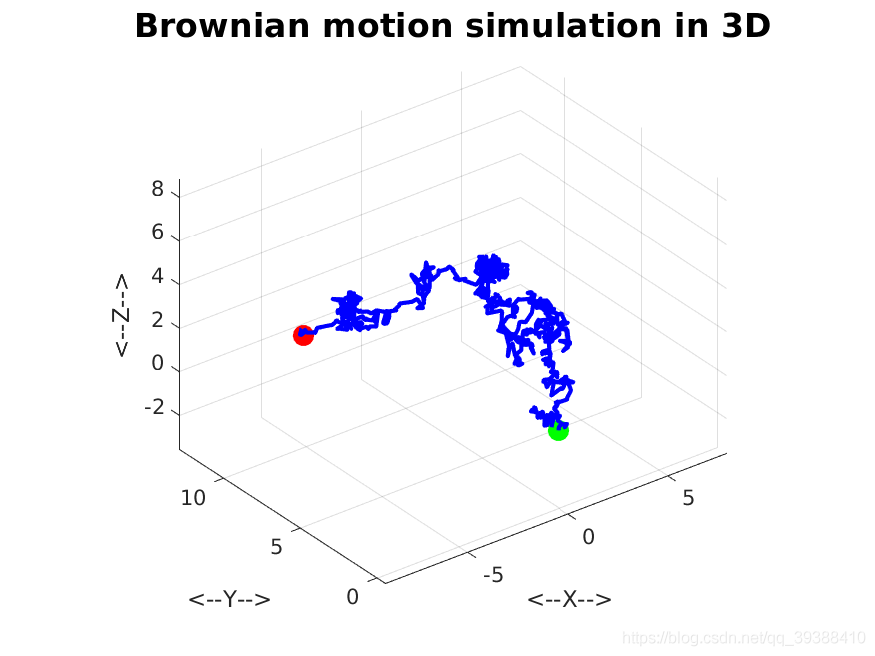
1905年，英国统计学家Pearson在《自然》杂志上公开求解随机游走问题（Random Walk Problem）：如果一个醉汉走路时每步的方向和大小完全随机，经过一段时间之后，在什么地方找到他的可能性最大？

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。

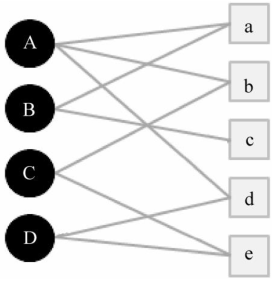
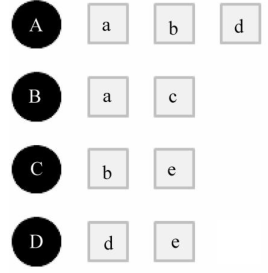


图表 1：醉汉（小白）的游走路线



图表 2：3D游走

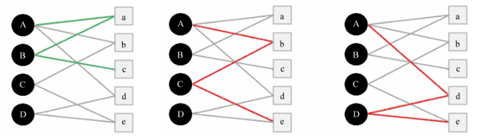
随机游走算法：



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

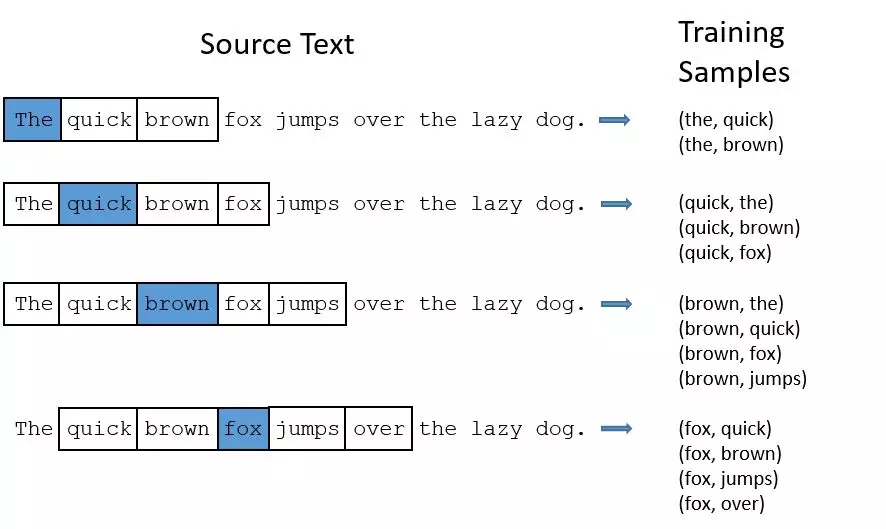
发现A到达c和e的最短路径都是3，于是进一步判断。

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)

### 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

### 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：

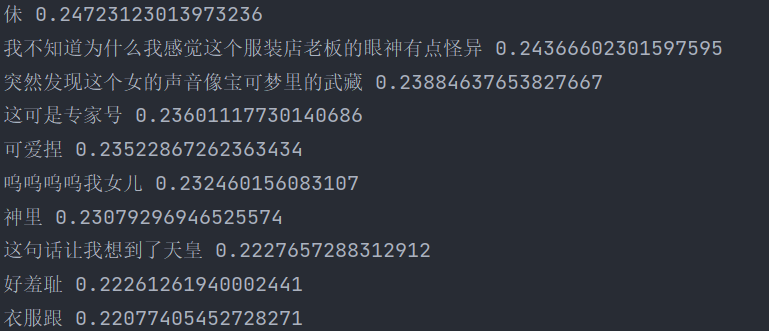


图10 输出’哇库哇库’相近的词语和概率图

输出’哇酷哇酷’的词向量如图11，12所示

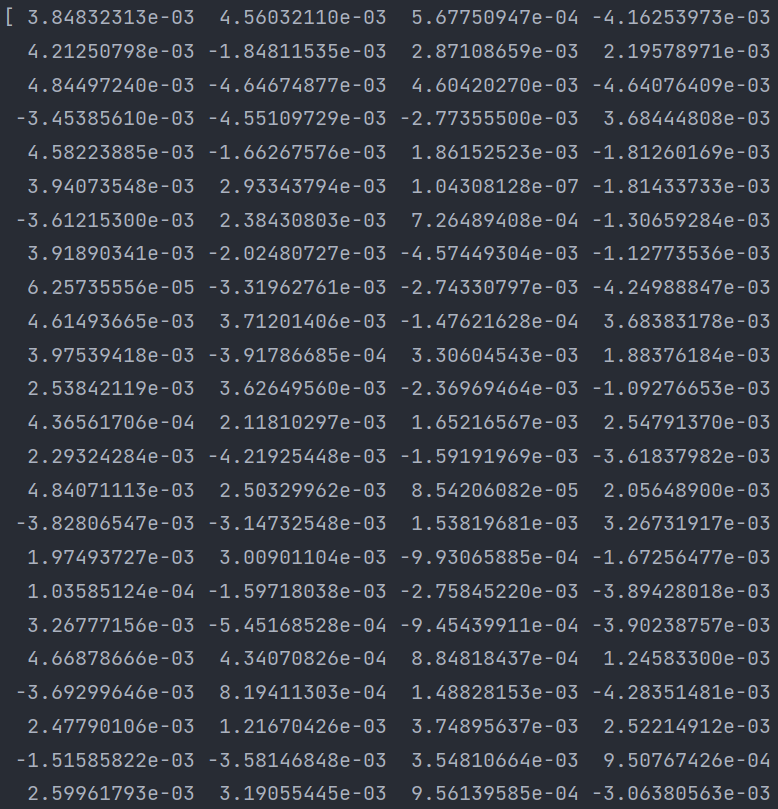
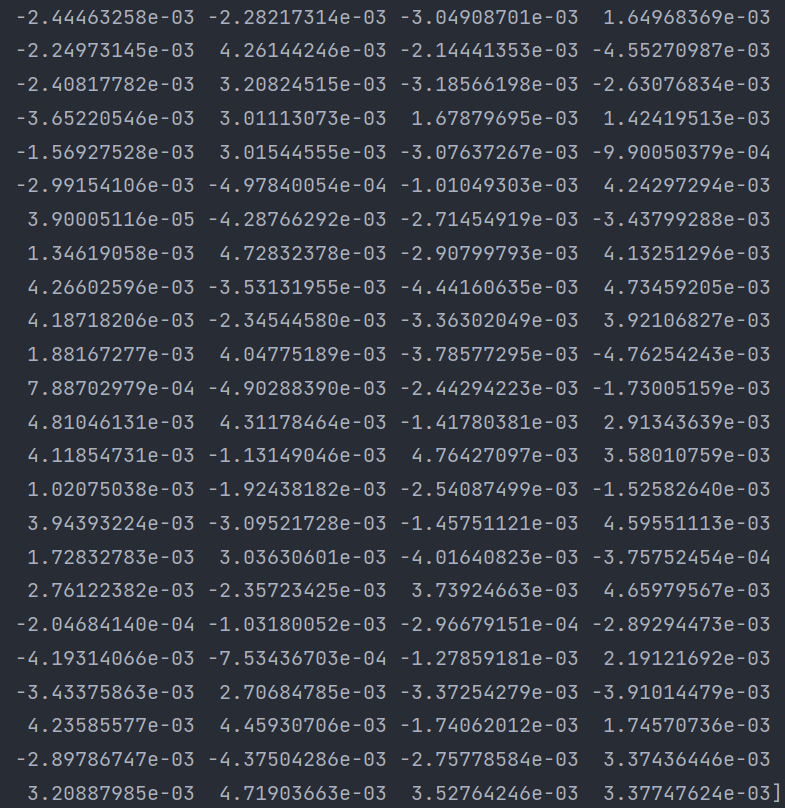
 

图11 词向量1 图12 词向量2

### 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

### 引用：

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