



**[作业4]游走在wiki游戏推荐链**

**课程名称 文本信息挖掘概论**

**学生学院 计算机学院**

**专业班 18级软件工程2班**

**学 号 3118005002**

**学生姓名 胡梓泽**

**指导老师 杨易杨**

**日期：2021年6月20日**

目录

[1. 数据集说明 4](#_Toc75818303)

[1.1数据集Wikipedia图数据集 4](#_Toc75818304)

[1.2 数据爬取 4](#_Toc75818305)

[1.3 数据集展示 4](#_Toc75818306)

[2. 模型介绍 5](#_Toc75818307)

[2.1 两个涉及的算法及其伪代码 5](#_Toc75818308)

[2.2 DeepWalk模型简介 6](#_Toc75818309)

[3. 执行过程与展示 8](#_Toc75818310)

[3.1 模型训练与保存 8](#_Toc75818311)

[3.1.1 导入工具包 8](#_Toc75818312)

[3.1.2 展示程序处理过程 8](#_Toc75818313)

[3.2 节点联系与选择 10](#_Toc75818314)

[3.2.2 程序处理过程 10](#_Toc75818315)

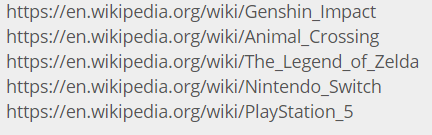
[3.3 5个初始节点的10个余弦相似度最大节点 10](#_Toc75818316)

# 1. 数据集说明

## 1.1数据集Wikipedia图数据集

Wiki关键词：原神，动物之森，塞尔达，Switch, PS5

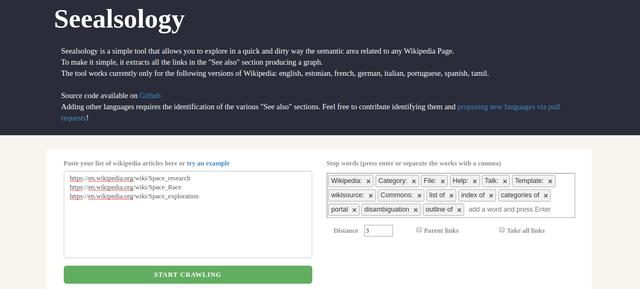
爬取5个页面中所有的链接，深度设定为3.共计为366405条数据。



## 1.2 数据爬取

抓取数据采用开源的Seealsology

这有助于我们从任何Wikipedia页面创建图。你甚至可以提供多个Wikipedia页面作为输入。这是该工具的屏幕截图：



如果一个页面链接到另一个页面,就会有一个图表示两个页面之间的联系。

## 1.3 数据集展示

在爬取完成后，将数据下载为TSV格式，保存于项目根目录的seealsology\_data.tsv。

如下表，共有三列（父节点-子节点-深度）。有了这个数据，我们便可**在Python中实施DeepWalk以查找相似的Wikipedia页面。**

|  |  |  |
| --- | --- | --- |
| 父节点 | 子节点 | 深度 |
| …… | …… | …… |
| animal crossing | nookazon 1 | 1 |
| animal crossing | online game | 1 |
| …… | …… | …… |

**我们将使用Wikipedia文章图，并使用DeepWalk从中提取节点嵌入**。

# 2. 模型介绍

## 2.1 两个涉及的算法及其伪代码

我们在本作业中共用到**DeepWalk和**RandomWalk。

以下为两个算法的伪代码：

算法1：DeepWalk

输入：图，游走次数r，游走长度l，时间窗口大小w，维度d，负采样个数n

输出：图节点的向量矩阵

1. 初始化图节点向量矩阵

2. 

3. 

4. 

5. 

6. 

7. 

算法2 RandomWalk

输入：图，起始节点，游走长度

输出：游走序列

1. 

2. 

3. 

4. 为空集，

5. 随机选取中的一个节点加入

6. 

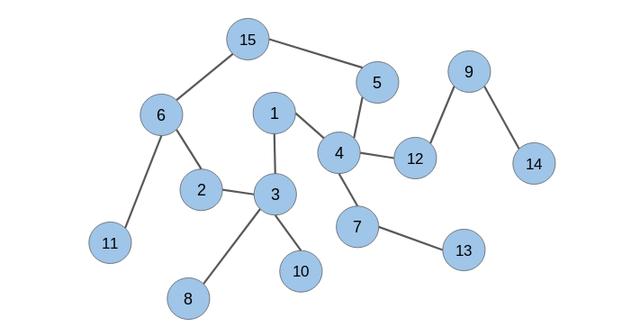
## 2.2 DeepWalk模型简介

**引用自[**<https://www.analyticsvidhya.com/blog/2019/11/graph-feature-extraction-deepwalk/>**]**

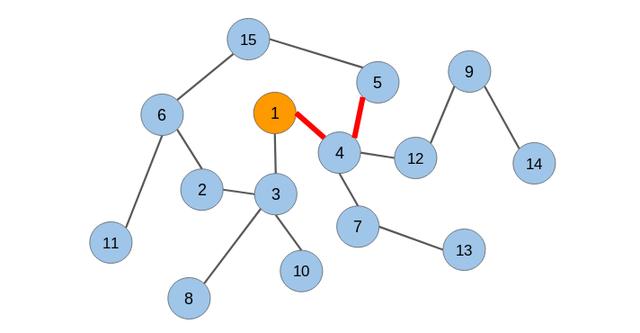
**什么是随机游走？**

随机游走是一种从图中提取序列的技术。我们可以使用这些序列来训练一个skip-gram模型来学习节点嵌入。

让我说明一下随机游走的工作原理。让我们考虑下面的无向图：

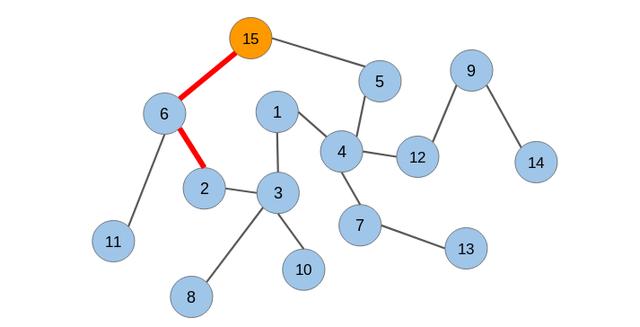


我们将在该图上应用随机游走并从中提取节点序列。我们将从节点1开始，并覆盖任意方向的两条边：



从节点1，我们可以转到任何连接的节点（节点3或节点4）。我们随机选择了节点4。现在再次从节点4开始，我们不得不随机选择前进的方向。我们将转到节点5。现在我们有3个节点的序列：[节点1 –节点4 –节点5]。

让我们生成另一个序列，但是这次是从另一个节点生成的：



让我们选择节点15作为原始节点。从节点5和6，我们将随机选择节点6。然后从节点11和2，我们选择节点2。新序列为[节点15 –节点6 –节点2]。

我们将对图中的每个节点重复此过程。这就是随机游走技术的工作原理。

在生成节点序列之后，我们必须将它们提供给一个skip-gram模型以获得节点嵌入。整个过程被称为Deepwalk。

# 3. 执行过程与展示

## 3.1 模型训练与保存

### 3.1.1 导入工具包

|  |
| --- |
| **import** networkx **as** nx **import** pandas **as** pd **import** numpy **as** np **import** random **from** tqdm **import** tqdm **from** sklearn.decomposition **import** PCA **import** matplotlib.pyplot **as** plt %matplotlib inline |

### 3.1.2 展示程序处理过程

①如下，读入数据集，显示节点及深度信息。共有366405条边信息。

|  |
| --- |
| df = pd.read\_csv(**"seealsology\_data.tsv"**, sep = **"\t"**)  df.head()  df.shape |
|  |

②构造无向图

|  |
| --- |
| G=nx.from\_pandas\_edgelist(df, **"source"**, **"target"**, edge\_attr=**True**, create\_using=nx.Graph()) |

③显示可用的节点数，共140806个

|  |
| --- |
| len(G) *# 节点数 ？个Wikipedia页面* |

④构造RandomWalk函数

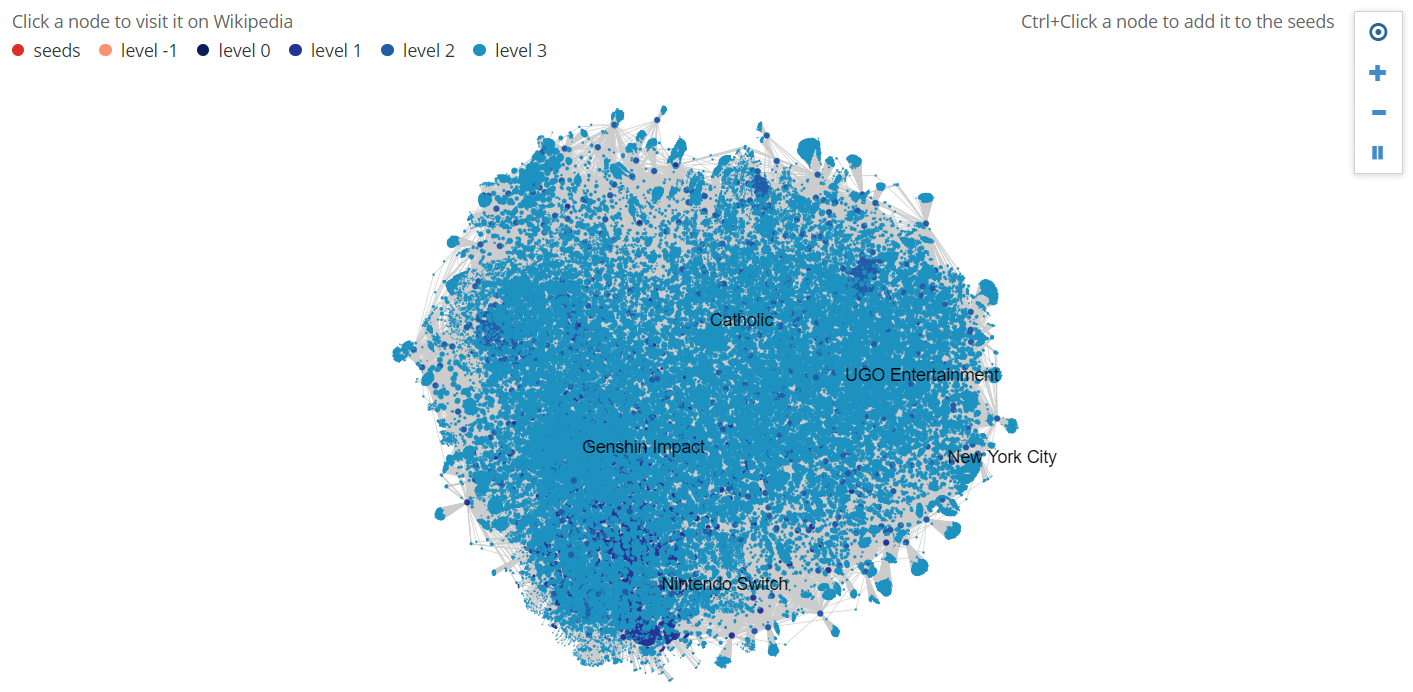
|  |
| --- |
| *# function 生成节点的随机行走序列 #它将从指定的输入节点以随机的方式穿过连接节点。最后,它将返回遍历节点的顺序* **def** get\_randomwalk(node, path\_length):    random\_walk = [node]    **for** i **in** range(path\_length-1):  temp = list(G.neighbors(node))  temp = list(set(temp) - set(random\_walk))   **if** len(temp) == 0:  **break** random\_node = random.choice(temp)  random\_walk.append(random\_node)  node = random\_node    **return** random\_walk |

⑤训练skip-gram (word2vec)模型，并保存model

|  |
| --- |
| model = Word2Vec(window = 4, sg = 1, hs = 0,  negative = 10, *# 负采样* alpha=0.03, min\_alpha=0.0007,  seed = 14)  model.build\_vocab(random\_walks, progress\_per=2)  model.train(random\_walks, total\_examples = model.corpus\_count, epochs=20, report\_delay=1)  *#%% 保存模型* model.save(**"gen/model"**) |

## 3.2 节点联系与选择

以下为节点间联系的无向图：



选择5个初始的wiki节点值

[**'genshin impact'**,**'animal crossing'**,**'the legend of zelda'**,**'nintendo switch'**,**'playstation 5'**]

### 3.2.2 PCA降维程序处理过程

|  |
| --- |
| **def** plot\_nodes(word\_list):  X = model[word\_list]    *# reduce dimensions to 2* pca = PCA(n\_components=2)  result = pca.fit\_transform(X)    plt.figure(figsize=(10,12))  *# create a scatter plot of the projection* plt.scatter(result[:, 0], result[:, 1])  **for** i, word **in** enumerate(word\_list):  plt.annotate(word, xy=(result[i, 0], result[i, 1]))  plt.gca().set\_aspect(**'equal'**, **'datalim'**)   plt.title(**'cosine measure'**)  plt.savefig(**'./cosine measure.jpg'**)  plt.show() |

## 3.3 5个初始节点的10个余弦相似度最大节点

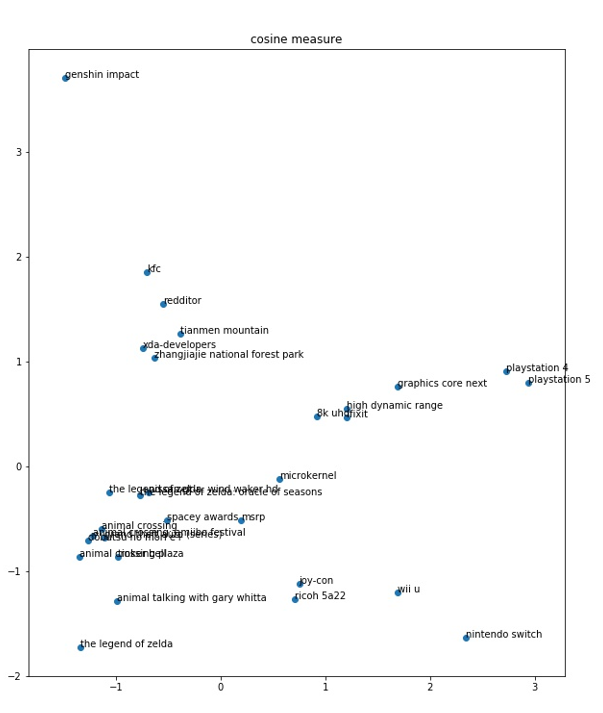
|  |
| --- |
| *# find top n similar nodes* all\_wordList=[**'genshin impact'**,**'animal crossing'**,**'the legend of zelda'**,**'nintendo switch'**,**'playstation 5'**]  similar\_wordList=[**'genshin impact'**,**'animal crossing'**,**'the legend of zelda'**,**'nintendo switch'**,**'playstation 5'**]  **for** word **in** all\_wordList:  **for** tmp **in** model.similar\_by\_word(word,topn=10): *#选最近的10个* similar\_wordList.append(tmp[0])  similar\_wordList = model.similar\_by\_word(**'genshin impact'**) |

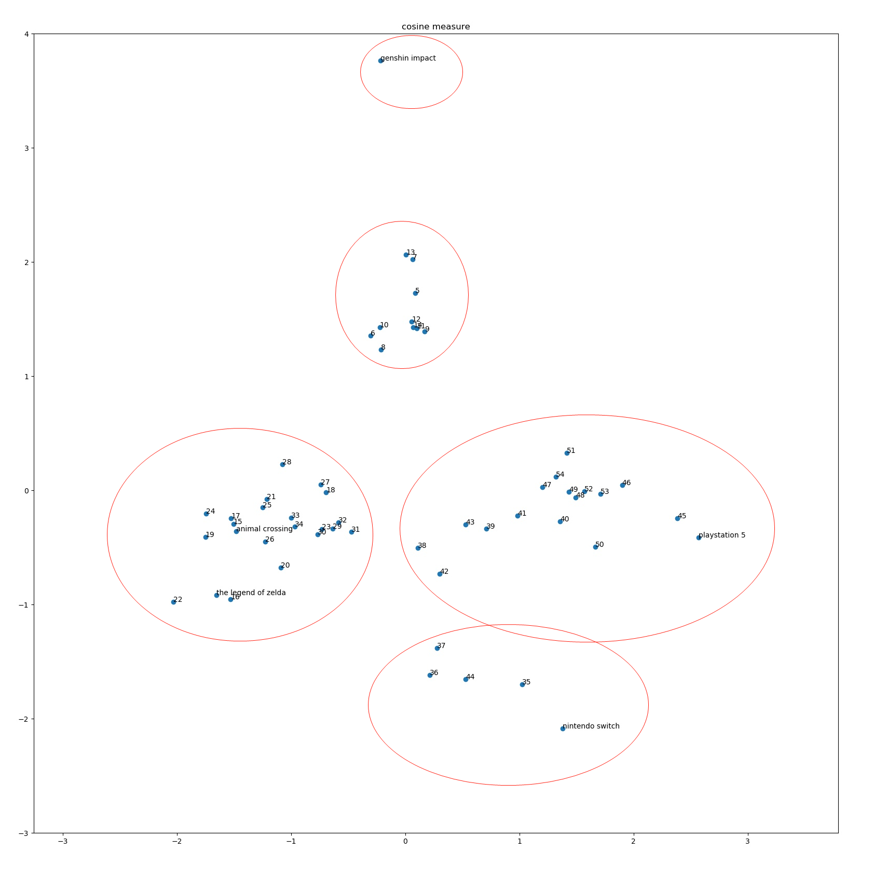
然后将离他们最近的10个节点拼接起来，共55个节点。得到如下结果

|  |  |  |
| --- | --- | --- |
| 初始节点 | 余弦相似度 | 子节点 |
| **'genshin impact'**  **（原神）** | ('redditor', 0.7745962142944336)  ('xda-developers', 0.7738863229751587)  ('kfc', 0.765745997428894)  ('zhangjiajie national forest park', 0.7566518783569336)  ('tianmen mountain', 0.7325072288513184)  ………………………………………………………… | |
| **'animal crossing'**  （动物之森） | ('dōbutsu no mori e+', 0.7387773990631104),  ('animal talking with gary whitta', 0.7251527309417725), ('animal crossing: amiibo festival', 0.7068615555763245), ('nissan gt-r', 0.6991083025932312),  ('animal crossing plaza', 0.6922386884689331)  ………………………………………………………… | |
| **'the legend of zelda'**  （塞尔达传说） | ('grand theft auto (series)', 0.7027507424354553),  ('tinker bell', 0.6862916946411133),  ('the legend of zelda: oracle of seasons', 0.6827098727226257),  ('the legend of zelda: wind waker hd', 0.6781565546989441),  ('spacey awards', 0.6713072657585144)  ………………………………………………………… | |
| **'nintendo switch'**  （Switch游戏机） | ('wii u', 0.6520593166351318),  ('ricoh 5a22', 0.6477434635162354),  ('joy-con', 0.6264757513999939),  ('msrp', 0.6156337261199951),  ('microkernel', 0.6108360886573792)  ………………………………………………………… | |
| **'playstation 5'**  **（PS5 游戏机）** | ('playstation 4', 0.7667080760002136),  ('graphics core next', 0.7506585121154785),  ('8k uhd', 0.7293373346328735),  ('ifixit', 0.7264735698699951),  ('high dynamic range', 0.7239927053451538)  ………………………………………………………… | |

## 3.4 展示结果

以下分别为带文本以及只带序号的余弦相似度PCA降维图：





以上55个节点，出了原神外，共可分为4个分类区域。序号对应名称请见附录。

|  |  |
| --- | --- |
| 22、2、16、20、19、24、1、15、17、26、20、25、21、28、33、34、30、23、29、32、31、27、18  （23个） | 13、7、5、6、8、10、12、14、11、9  （10个） |
| 42、38、43、39、41、40、50、4、45、47、54、51、49、48、52、53、46  （17个） | 3、35、36、37、44  （5个） |

①分类一中，相似的页面为任天堂开发的相关游戏。

②分类二中，为中国国内的相关资讯

③分类三中，相似的页面为PS5上公布的游戏。

④分类四中，相似的页面为Switch掌机上的游戏。

# 附录

55个节点名字

|  |
| --- |
| 0 genshin impact  1 animal crossing  2 the legend of zelda  3 nintendo switch  4 playstation 5  5 redditor  6 xda-developers  7 kfc  8 zhangjiajie national forest park  9 tianmen mountain  10 floating cities and islands in fiction  11 quest (video games)  12 shanghai symphony orchestra  13 chinese culture  14 vladimir putin  15 dōbutsu no mori e+  16 animal talking with gary whitta  17 animal crossing: amiibo festival  18 nissan gt-r  19 animal crossing plaza  20 wii music  21 cubello  22 animal crossing: city folk  23 boxboy! (video game)  24 tom nook  25 grand theft auto (series)  26 tinker bell  27 the legend of zelda: oracle of seasons  28 the legend of zelda: wind waker hd  29 spacey awards  30 epona (character)  31 heuristic  32 loz  33 origin systems  34 the legend of zelda: twilight princess  35 wii u  36 ricoh 5a22  37 joy-con  38 msrp  39 microkernel  40 nand flash memory  41 texture mapping unit  42 near-field communications  43 802.11b  44 u-force  45 playstation 4  46 graphics core next  47 8k uhd  48 ifixit  49 high dynamic range  50 casio loopy  51 horizon forbidden west  52 gpu  53 7.1 surround sound  54 solid-state storage |