使用DeepWalk从图中提取特征

OpenCV学堂 2019-12-03

以下文章来源于磐创AI ，作者VK

磐创AI

AI行业最新动态，机器学习干货文章，深度学习原创博客，深度学习实战项目，Tenso...

点击上方↑↑↑“OpenCV学堂”关注我

来源：公众号 磐创AI 授权转

概述

- 从表格或图像数据中提取特征的方法已经众所周知了，但是图(数据结构的图)数据呢?
- 学习如何使用DeepWalk从图中提取特征
- 我们还将用Python实现DeepWalk来查找相似的Wikipedia页面

介绍

我被谷歌搜索的工作方式迷住了。每次我搜索一个主题都会有很多小问题出现。以“人们也在搜索?”为例。当我搜索一个特定的人或一本书，从谷歌我总是得到与搜索内容类似的建议。

例如，当我搜索“Lewis Hamilton”时，我得到了其他著名f1车手的名单:

这些丰富而相关的内容是由高度复杂的图处理数据处理算法提供的。正是这种图和网的力量让我(以及许多其他数据科学家)着迷!自从我开始使用图以来，出现了许多新的技术。

在本文中，我将介绍任何机器学习项目中最重要的步骤之一—**特征提取**。不过，这里有一个小小的转折。我们将从图数据集中提取特征，并使用这些特征来查找相似的节点(实体)。

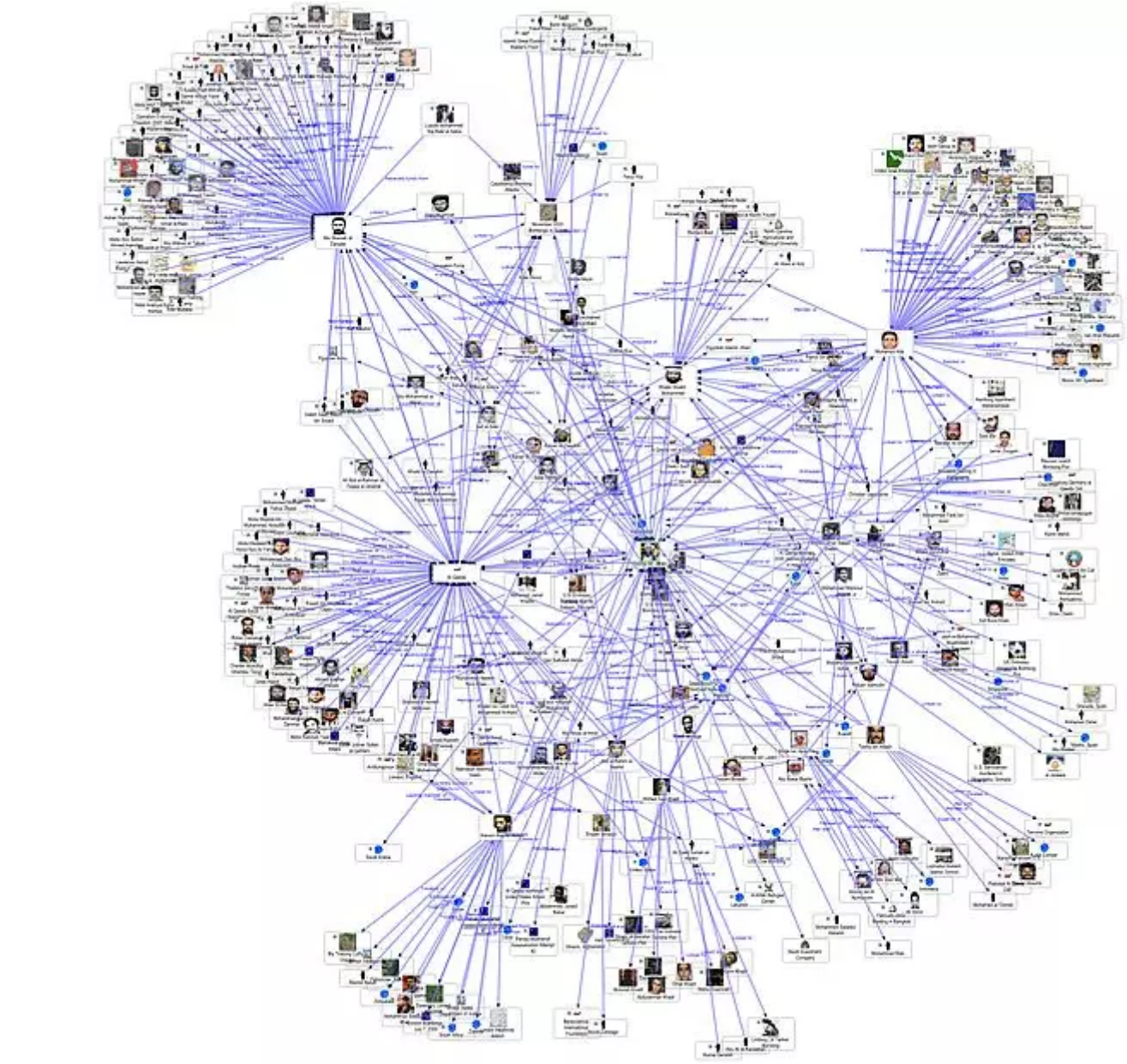
目录

数据的图示

当你想到“网络”时，会想到什么？通常是诸如社交网络，互联网，已连接的IoT设备，铁路网络或电信网络之类的事物。**在图论中，这些网络称为图。**

> 网络是互连节点的集合。节点表示实体，它们之间的连接是某种关系。

例如，我们可以用图的形式表示一组社交媒体帐户：



节点是用户的数字档案，连接表示他们之间的关系，例如谁跟随谁或谁与谁是朋友。

图的用例不仅限于社交媒体！我们还可以使用图和网络表示其他类型的数据（并且在本文中我们将介绍一个独特的行业用例）。

为什么我们将数据表示为图?

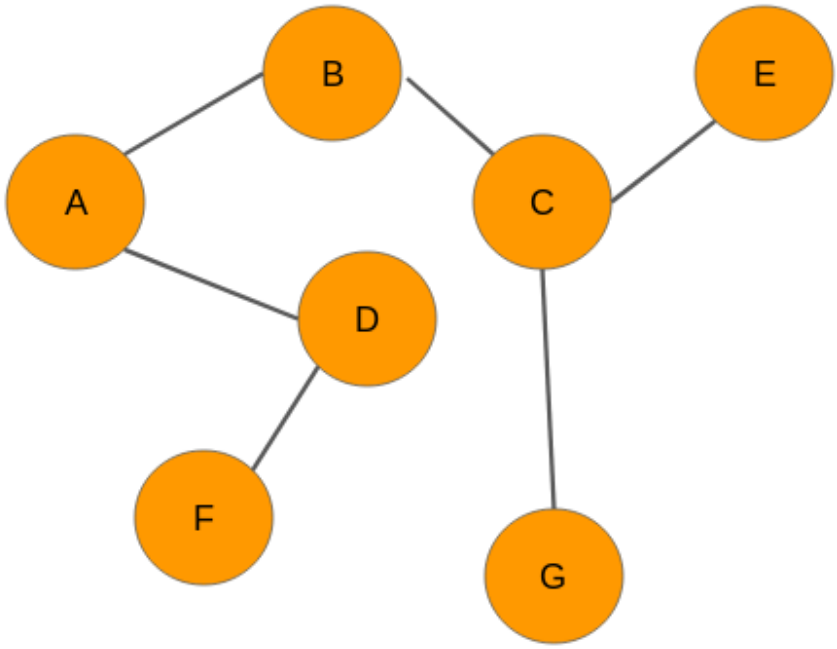
为什么不仅仅使用典型的数据可视化技术来可视化数据？为什么要更复杂并学习新概念？下面我们会给出答案。

图数据集和数据库可帮助我们应对在处理结构化数据时面临的若干挑战。这就是为什么当今的主要科技公司，例如Google，Uber，Amazon和Facebook使用某种形式的图的原因。

让我们以一个例子来理解为什么图是数据的重要表示形式。看下图：

User	Friend
A	B
A	D
B	A
B	C
C	B
C	E
C	G
D	A
D	F
E	C
F	D
G	C

VS



这是一小部分Facebook用户(a, B, C, D, E, F, G)的数据集。图像的左半边包含这个数据的表格形式。每一行代表一个用户和他/她的一个朋友。

右半部分包含代表同一组用户的图。该图的边缘告诉我们，连接的节点是Facebook上的朋友。现在，让我们解决一个简单的查询：

“找到用户A的朋友和用户A朋友的朋友。”

查看表格数据和上面的图。哪种数据形式更适合回答此类查询?

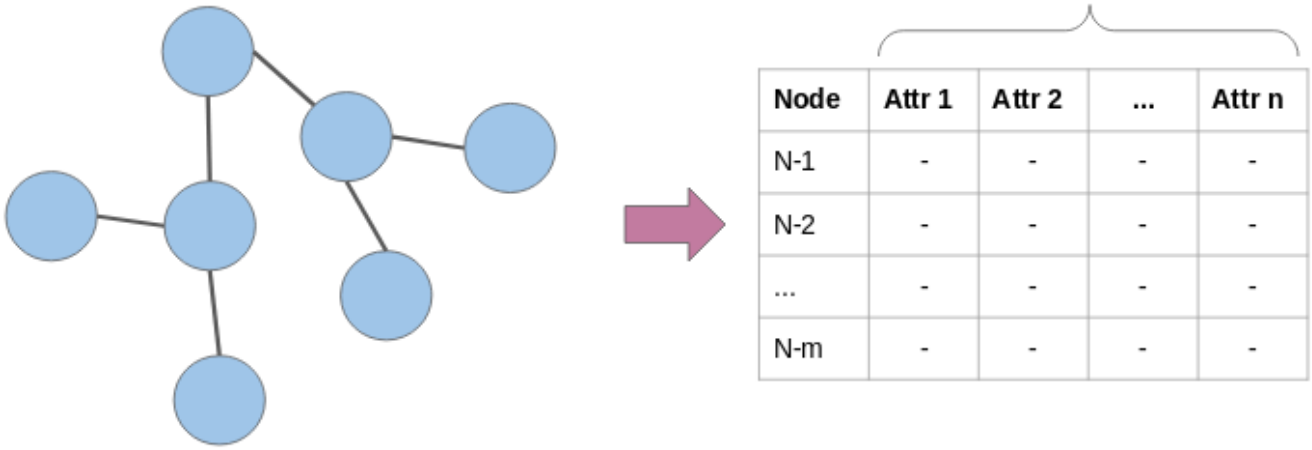
使用图来解决该问题要容易得多，因为我们只需要遍历从节点A长度为2的路径（ABC和ADF），即可找到朋友和朋友的朋友。

因此，**图可以轻松捕获节点之间的关系，这在常规数据结构中是一项艰巨的任务**。现在，让我们看看使用图可以解决什么样的问题。

基于图的特征的不同类型

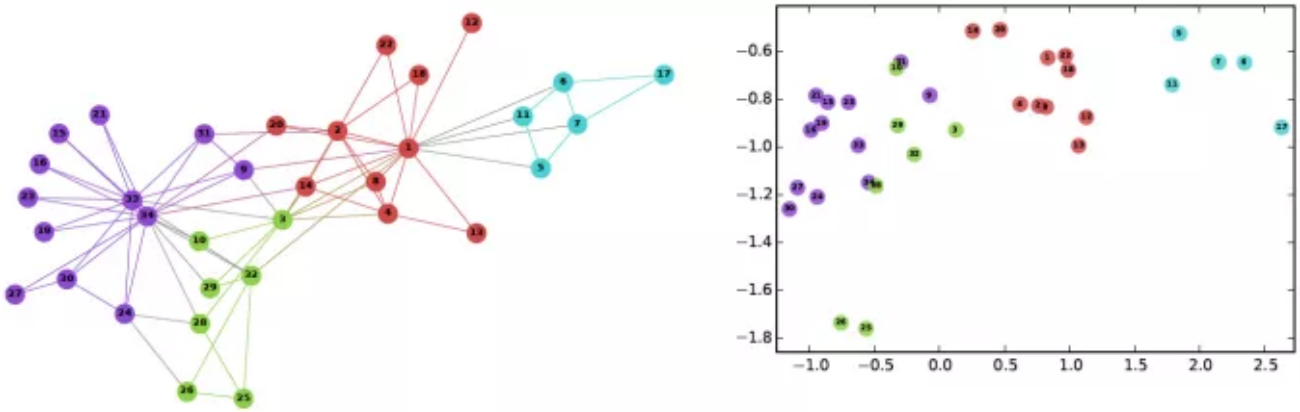
为了解决上述问题，我们无法将图直接提供给机器学习模型。我们必须首先从中创建特征，然后模型将使用这些特征。

此过程类似于我们在自然语言处理（NLP）或计算机视觉中所做的过程。我们首先从文本或图像中提取数字特征，然后将这些特征作为输入提供给机器学习模型：



从图中提取的特征可以大致分为三类：

1. **节点属性**：我们知道图中的节点代表实体，并且这些实体具有自己的特征属性。我们可以将这些属性用作每个节点的特征。例如，在航空公司航线网络中，节点将代表机场。这些节点将具有飞机容量，航站楼数量，着陆区等特征。
   2.**局部结构特点**：节点的*度*（相邻节点的数量），相邻节点的平均度，一个节点与其他节点形成的三角形数，等等。
2. **节点嵌入**：上面讨论的特征仅包含与节点有关的信息。它们不捕获有关节点上下文的信息。在上下文中，我指的是周围的节点。节点嵌入通过用固定长度向量表示每个节点，在一定程度上解决了这个问题。这些向量能够捕获有关周围节点的信息（上下文信息）



用于学习节点嵌入的两个重要的现代算法是**DeepWalk**和**Node2Vec**。在本文中，我们将介绍并实现DeepWalk算法。

DeepWalk简介

要了解DeepWalk，重要的是要正确理解词嵌入及其在NLP中的使用方式。我建议在下面的文章中仔细阅读Word2Vec的解释：

https://www.analyticsvidhya.com/blog/2019/07/how-to-build-recommendation-system-word2vec-python/?utm_source=blog&utm_medium=graph-feature-extraction-deepwalk

为了将事物置于上下文中，词嵌入是文本的向量表示形式，它们捕获上下文信息。让我们看看下面的句子：

- 我乘**巴士**孟买
- 我乘**火车**去孟买

粗体字（公共汽车和火车）的向量将非常相似，因为它们出现在相同的上下文中，即粗体文本之前和之后的词。该信息对于许多NLP任务非常有用，例如文本分类，命名实体识别，语言建模，机器翻译等等。

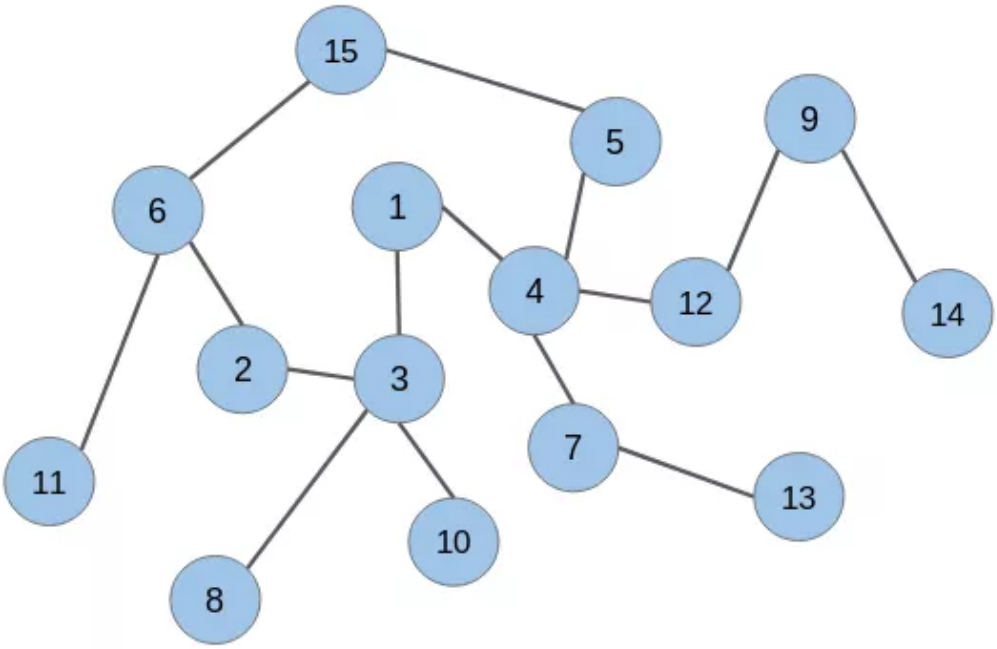
我们还可以在每个节点的图中捕获此类上下文信息。但是，为了学习NLP空间中的词嵌入，我们将句子提供给Skip-gram模型（浅层神经网络）。句子是按一定顺序排列的单词序列。

因此，要获得**节点嵌入**，我们首先需要安排图中的节点序列。我们如何从图中获得这些序列？有一项针对该任务的技术称为随机游走。

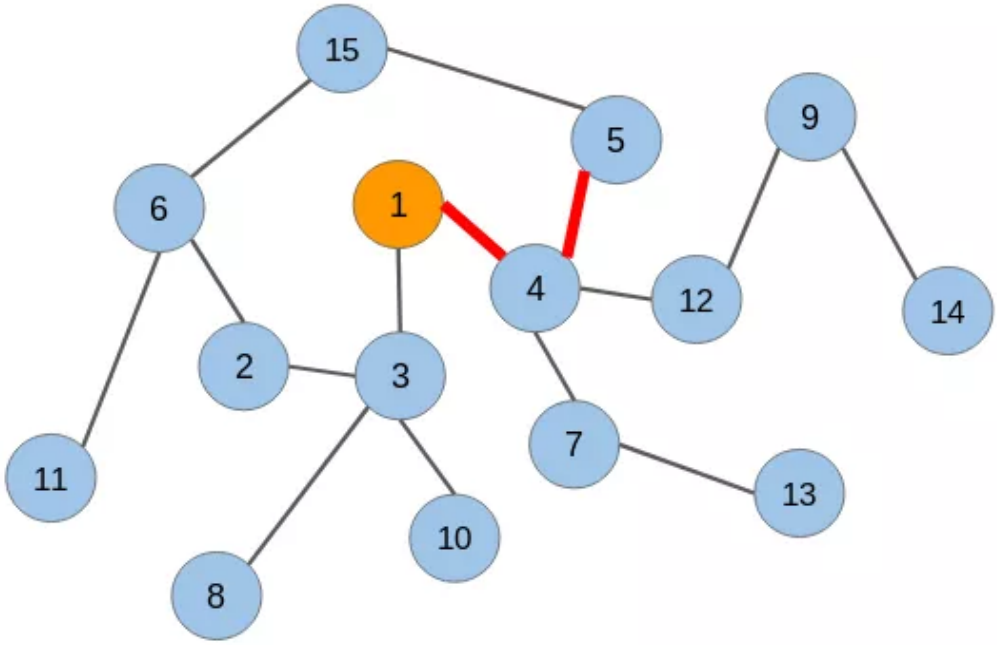
什么是随机游走?

> 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：

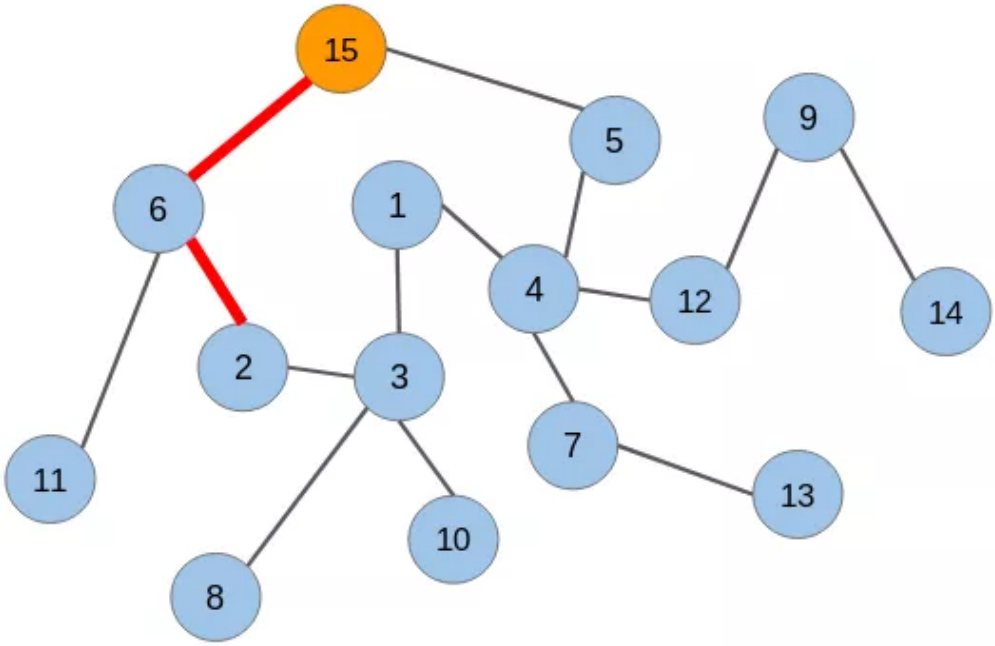


我们将在该图上应用随机游走并从中提取节点序列。我们将从节点1开始，并覆盖任意方向的两条边：



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。

在下一节中，我们将在Wikipedia文章网络上从头开始实施DeepWalk。

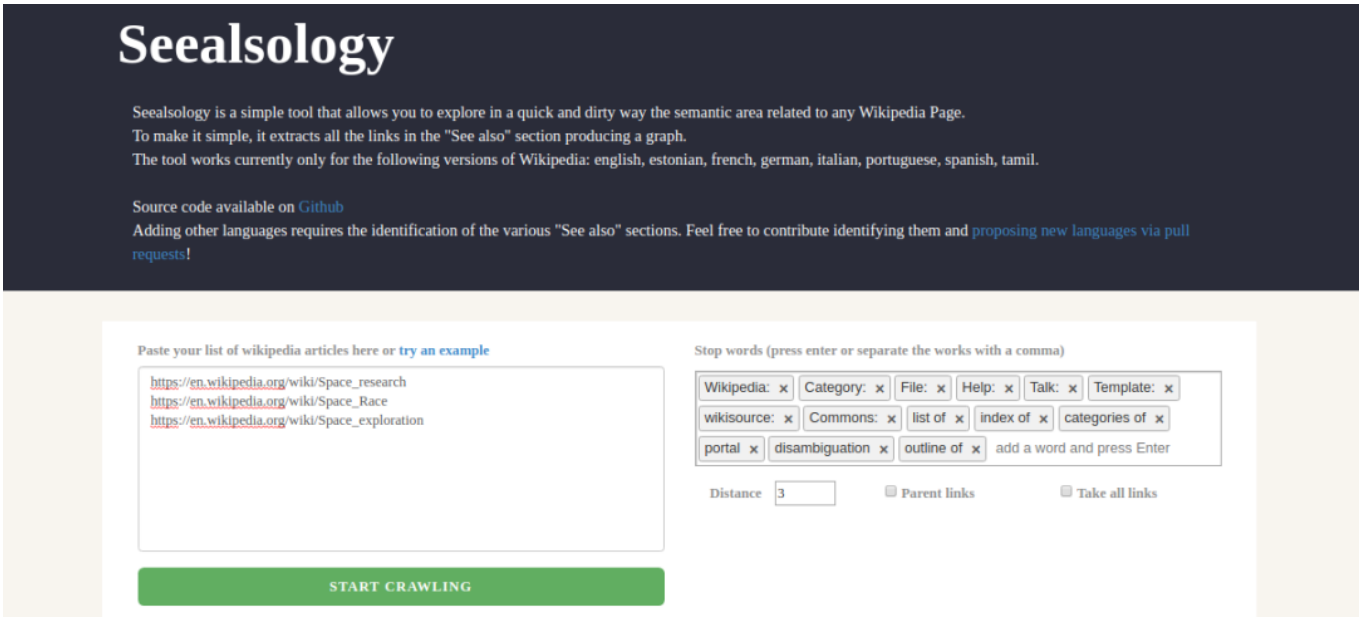
在Python中实施DeepWalk以查找相似的Wikipedia页面

这将是本文中最令人兴奋的部分，尤其是如果你喜欢代码。因此，请启动这些Jupyter notebook！

我们将使用Wikipedia文章图，并使用DeepWalk从中提取节点嵌入。然后，我们将使用这些嵌入来查找相似的Wikipedia页面。

我们不会触及这些文章中的任何文本。我们的目标是纯粹基于图的结构来计算页面之间的相似度。

但是，等等。我们如何以及在何处获得Wikipedia图数据集？Seealsology这个出色的工具将为我们提供帮助。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。

看看在Seealsology中该图的形成方式。值得一看：

https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2019/11/graph_buildup.mp4?_=1

图中的节点非常接近，并不一定意味着它们在语义上相似。因此，需要在向量空间中表示这些节点，我们可以在其中识别相似的节点。

当然，我们可以使用其他方法来完成此任务。例如，我们可以解析这些节点（Wikipedia页面）中的所有文本，并在词嵌入的帮助下用向量表示每个页面。然后，我们可以计算这些向量之间的相似度以找到相似的页面。但是，这种基于NLP的方法存在一些缺点：

- 如果有数百万个节点，那么我们需要大量的计算能力来解析文本并从所有这些节点或页面中学习词嵌入
- 这种方法不会捕获这些页面之间连接的信息。例如，一对直接连接的页面可能比一对间接连接的页面具有更强的关系

这些缺点可以通过图和节点嵌入轻松解决。因此，一旦你的图准备就绪，就可以从Seealsology下载TSV文件。在此文件中，每一行都是一对节点。我们将使用此数据来重构图，并在其上应用DeepWalk算法以获得节点嵌入。

让我们开始吧！你可以为此使用Jupyter Notebook或Colab。

导入所需的Python库

```
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
```

加载数据集

你可以从这里下载.tsv文件：

https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2019/11/space_data.zip

```
df = pd.read_csv("space_data.tsv", sep = "\t")
df.head()
```

Output:

	source	target	depth
0	space exploration	discovery and exploration of the solar system	1
1	space exploration	in-space propulsion technologies	1
2	space exploration	robotic spacecraft	1
3	space exploration	timeline of planetary exploration	1
4	space exploration	landings on other planets	1

源和**目标**都包含Wikipedia实体。对于所有行，目标实体在源实体的Wikipedia页面有其超链接。

构造图

```
G = nx.from_pandas_edgelist(df, "source", "target", edge_attr=True, create_using=nx
```

让我们检查图中的节点数：

```
len(G)
```

Output: 2088

我们将处理2,088个Wikipedia页面。

随机游走

在这里,我定义了一个函数,将节点和被遍历的路径的长度作为输入。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
```

让我们来试试节点“space exploration”这个函数:

```
get_randomwalk('space exploration', 10)
```

输出:

```
['space exploration',
 'atmospheric reentry',
 'mars pathfinder',
 'scientific information from the mars exploration rover mission',
 'opportunity rover',
 'spirit rover',
 'aeolis quadrangle',
 'mars science laboratory',
 'mars 2020',
 'exomars']
```

在这里，我已指定要遍历的长度为10。你可以更改此数字并进行操作。接下来，我们将捕获数据集中所有节点的随机游走序列：

```
# 从图获取所有节点的列表
all_nodes = list(G.nodes())

random_walks = []
for n in tqdm(all_nodes):
    for i in range(5):
        random_walks.append(get_randomwalk(n,10))

# 序列个数
len(random_walks)
```

输出: 10,440

因此，将遍历长度设置为10，我们得到了10,440个节点的随机游动序列。我们可以将这些序列用作skip-gram模型的输入，并提取该模型学习到的权重。

```
# importing required libraries

import pandas as pd
import networkx as nx
import numpy as np
import random
from tqdm import tqdm
from sklearn.decomposition import PCA
import pprint
from gensim.models import Word2Vec
import warnings
warnings.filterwarnings('ignore')
```

```
13
14 # read the dataset
15 df = pd.read_csv("space_data.tsv", sep = "\t")
16 print(df.head())
17
18 G = nx.from_pandas_edgelist(df, "source", "target", edge_attr=True, create_using=n
19
20 G = nx.from_pandas_edgelist(df, "source", "target", edge_attr=True, create_using=n
21
22 print('The number of nodes in pur graph: ',len(G))
23
24 def get_randomwalk(node, path_length):
25
26     random_walk = [node]
27
28     for i in range(path_length-1):
29         temp = list(G.neighbors(node))
30         temp = list(set(temp) - set(random_walk))
31         if len(temp) == 0:
32             break
33
34         random_node = random.choice(temp)
35         random_walk.append(random_node)
36         node = random_node
37
38     return random_walk
39
40 print('\n\nRandom sequence of nodes generated from Random Walk\n\n')
41 while True:
42   first_node = input("Enter name of first node (for example 'space exploration') :
43   if len(first_node) > 0:
44     break
45 pprint.pprint(get_randomwalk(first_node, 10))
46
47
48
49 # 从图中获取所有节点的列表
50 all_nodes = list(G.nodes())
51
52 random_walks = []
```

```
for n in tqdm(all_nodes):
    for i in range(5):
        random_walks.append(get_randomwalk(n,10))

# 序列长度
len(random_walks)


# 训练skip-gram (word2vec)模型
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_d
print('\n\n Get similar nodes\n\n')
while True:
  any_node = input("Enter name of any node (for example 'space toursim') : ")
  if len(any_node) > 0:
    break
pprint.pprint(model.similar_by_word(any_node))
```

结尾

我真的很喜欢在本文中探索DeepWalk中的图形数据，我迫不及待地想尝试其他图形算法。在接下来的几周里，继续关注这个系列吧!

我鼓励你实施此代码，试用它，并建立自己的图模型。这是学习任何概念的最佳方法。完整的代码在这里：https://github.com/prateekjoshi565/DeepWalk。