Intro to Data Science in Python

subtitle

Spencer Grewe

Invalid Date

Table of Contents

References

Roughly following the lessons published in Data Science from Scratch 2ed - (O'Reilly 2019).

01. Questions and Answers

Data science has many complicated workflows and practices, but at the end of the day, it is all about getting answers to expansive questions. Suppose a company is curious about the most 'important' person in a group of friends. They provide a list of 10 people and the connections between them.

```
users = [
    { "id": 0, "name": "Hero" },
    { "id": 1, "name": "Dunn" },
    { "id": 2, "name": "Sue" },
    { "id": 3, "name": "Chi" },
    { "id": 4, "name": "Thor" },
    { "id": 5, "name": "Clive" },
    { "id": 6, "name": "Hicks" },
    { "id": 7, "name": "Devin" },
    { "id": 8, "name": "Kate" },
    { "id": 9, "name": "Klein" }
]
```

```
friendship_pairs = [(0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (3, 4),
                    (4, 5), (5, 6), (5, 7), (6, 8), (7, 8), (8, 9)]
```

For each user in the users, there is an id and a name, eg. `user[2] -> 2, "Sue"`. For each tuple in friendship_pairs, there is a named 2 way connection, eg. `(2, 3)` connects users 2, Sue, and user 3, Chi. We can keep track of the number of

```
friendships = {user["id"]: [] for user in users }
friendships # a dict with 10 ids, all of which have an empty list

for i, j in friendship_pairs: # pull out the (i,j) in every tuple...
      friendships[i].append(j) # for the tuple (0,1), add "1" to 0's friends
      friendships[j].append(i) # for the tuple (0,1), now add "0" to 1's friends

friendships # everyone has either 1, 2, or 3 friends
```

```
{0: [1, 2],
 1: [0, 2, 3],
 2: [0, 1, 3],
 3: [1, 2, 4],
 4: [3, 5],
 5: [4, 6, 7],
 6: [5, 8],
 7: [5, 8],
 8: [6, 7, 9],
 9: [8]}
```

```
def number_of_friends(user):
    """How many friends does _user_ have?"""
    user_id = user["id"]
    friend_ids = friendships[user_id]
    return len(friend_ids)

total_connections = sum(number_of_friends(user)
                        for user in users)        # 24 total

n_users = len(users)
total_connections/n_users # average is 2.4 friends

# Create a list (user_id, number_of_friends).
```

```
num_friends_by_id = [(user["id"], number_of_friends(user))
                     for user in users]

num_friends_by_id.sort(                                # Sort the list
       key=lambda id_and_friends: id_and_friends[1],   # by num_friends
       reverse=True)                                   # largest to smallest

num_friends_by_id
```

```
[(1, 3),
 (2, 3),
 (3, 3),
 (5, 3),
 (8, 3),
 (0, 2),
 (4, 2),
 (6, 2),
 (7, 2),
 (9, 1)]
```

We now have a sorted list of everyone's IDs by their number of friends. We can see that IDs 1, 2, 3, and 5 all have 3 friends, making them some of the most important members of the network of friends by number only. However, 3 and 5 are near the center of the network, something we can only see when graphing out the friends and their connections as nodes and edges.

```
flowchart LR
    id0((0)) & id1((1)) & id2((2)) & id3((3)) & id4((4))
    id5((5)) & id6((6)) & id7((7)) & id8((8)) & id9((9))
    id0 --- id1 & id2 --- id3
    id1 --- id2
    id3 --- id4 --- id5 --- id6 & id7
    id6 & id7 --- id8 --- id9
```