

FRIEND-RECOMMENDATION SCHEME USING Ontology Representation

Team Members:

Kavi Abhishek Venkat (15BCE0081)
Adityan Elangovan (15BCE0079)

Report submitted for the
Final Project Review of

Course Code: CSE3021– Social and Information Networks

Slot: A1 + TA1

Professor: Meenakshi S.P



1. Introduction

An online social network (OSN) facilitates users to stay in touch with other users (such as distant family members or friends), easily share information, look for old acquaintances and establish friendship with new users based on shared interests. The wide availability of the Internet has resulted in the fast growth of OSNs, such as Google+, Facebook, MySpace, Twitter and LinkedIn, which resulted in a vast amount of social data containing personal and sensitive information about individual users.

Our project is a friend recommendation system which is an important aspect in the modern social network systems including Facebook, Twitter, Instagram and countless others. Using our mutual likes with other people the social network system adds people in our “suggested list” of friends. So our python project stores a dictionary that contains names of people along with the things they like. For implementation purposes we have created a Data Set of 50 different people. At runtime after we enter a desired name, our program calculates strength of “possible friendship” by taking in consideration of mutual likes with all other names in the dataset. This strength is between 0 and 1. After that, it displays all the suggested friends along with their corresponding strengths in the increasing order of the strengths.

Online social networks, such as Facebook and Google+, have been emerging as a new communication service for users to stay in touch and share information with family members and friends over the Internet. Since the users are generating huge amounts of data on social network sites, an interesting question is how to mine this enormous amount of data to retrieve useful information. Along this direction, social network analysis has emerged as an important tool for many business intelligence applications such as identifying potential customers and promoting items based on their interests. In particular, since users are often interested to make new friends, a friend recommendation application provides the medium for users to expand his/her social connections and share information of interest with more friends. Besides this, it also helps to enhance the development of the entire network structure.

Social network analysis involve mining the social data to understand user activities and to identify the relationships among various users. Especially, in applications such as business intelligence, social network analyses have boosted the research in developing various recommendation algorithms. For example, an algorithm may recommend a new application to a Facebook user based on either the applications he/she used in the past or the usage pattern of various applications used by his/her friends.

In general, a recommendation can be a friend, a product, an ad or even a content potentially relevant to the user. This work focuses on recommending new friends to a given user in an OSN.

OntologyRepresentations are ways of representing the relationships between various concepts. For example, "Cats are a type of mammal," or "An arm is a part of the body," or "If A is the father of B, then B is the son of A", or "Every animal

has exactly one father, and the father of an animal is always himself an animal". Ontology representations can talk about types of concepts (i.e. cats, mammal) but they can also talk about properties (the "is father of" property, for example).

The line between "ontological" statements and simple logical "statements" (e.g. "John has exactly one car, and his car is a Buick") can be fuzzy. In short, ontological statements are kind of "meta". In more detail, ontological statements have to do with the properties of the vocabulary that we use to define things, rather than contingent properties of this particular world.

2. Literature Review Summary Table

<i>Authors and Year (Reference)</i>	<i>Title (Study)</i>	<i>Concept / Theoretical model/ Framework</i>	<i>Methodology used/ Implementation</i>	<i>Dataset details/ Analysis</i>	<i>Relevant Finding</i>	<i>Limitations/ Future Research/ Gaps identified</i>
Dr. Wei Jiang, Advisor Dr. Sanja Y Madria Dr. Bruce M. McMillin Dr. Dan Lin Dr. Akim Adekunle	PRIVACY-PRESERVING FRIEND RECOMMENDATION BASED ON DIFFERENT EXISTING SIMILARITY METRICS IN THE LITERATURE. Briefly, depending on the underlying similarity metric used, the	This work introduces a set of privacy-preserving friend recommendation protocols based on different existing similarity metrics in the literature. Briefly, depending on the underlying similarity metric used, the	This work considers the case of outsourced social networks, where users' profile data are encrypted and outsourced to third-party cloud providers who provide social	Since it is hard to control parameters in a real-world dataset, this work simulated the environment and computed the computation costs. In all the experiments, the minimum number of messages exchanged between any two users U and V is assumed to	The proposed protocol computes the recommendation scores of all users within a radius of h from the target user A by using the similarity metric proposed as a baseline. More specifically, the	Proposed protocols in Sections 3 to 5 assume that either users' friend lists, social tags, or messages exchanged with other users of an online social network (OSN) as private informati

YEA R- 2013		proposed protocols guarantee the privacy of a user's personal information such as friend lists. These protocols are the first to make the friend recommendation process possible in privacy-enhanced social networking environments	networking services to the users. Under such an environment, this work proposes novel protocols for the cloud to do friend recommendations in a privacy preserving manner.	be uniformly distributed in [1, 1000]. Also, this work assumes that there is no communication delay between participating users (which further implies that users are online). In addition, the number of child friends for each user (including the target user A) is varied from 50 to 250.	proposed protocol generates the (scaled) recommendation scores along with the corresponding user IDs in such a way that the relative ordering among the users in the list of recommended users is preserved.	on. As a future work, it is also desirable to develop protocols that can recommend friends based on other details such as education and employment in a privacy-preserving manner.
-------------------	--	---	--	---	--	--

3. Objective of the project

The objective of our project is to accurately recommend friends to a particular user in a dataset in python based on mutual likes of that user with all the other users in the dataset as well as to quantify the relationship of suggested friends with that particular user by assigning a relationship score which lies between 0 to 1 with 0 meaning no mutual likes and 1 meaning either the friend suggested is the same user or the friend suggested has exactly same likes as that person. We plan to achieve this system by using pre-defined library RECSYS in python. Currently we plan to take 50 plus users in our dataset with at least 10 likes of each users so that we can properly display functioning of our friend recommendation system as well as accurately show the relationship score of the recommended friends.

4. Innovation component in the project

Our project is innovative and it is different from the reference project used. Like all other recommended systems it displays recommended friends based on mutual likes using ontology . But the innovation component on which it differs from all other projects is that it accurately displays relationship score which all other projects fail to do. Thus it is very convenient in our system for a user to check and assess all the relationships and properly choose their friends circle.;

5. Work done and implementation

a. Methodology adapted:

In our case, we have utilized “svd” and “pyrecsys” for providing friend recommendations based on their likes. python-recsys supports two Recommender Algorithms: Singular Value Decomposition (SVD) and Neighborhood SVD. We have used Neighborhood SVD.

“pyrecsys” provides, out of the box, some basic algorithms based on matrix factorization.

python-recsys is a fast recommender engine for Python. It uses matrix factorization to provide recommendations and similarities among items or users. The library is built on top of divisi2, which already implements SVD-like matrix factorization using numpy.

Under this library the main sub-tool we have utilized is “SVD”. pyrecsys makes use of SVD in order to decompose the input data (a matrix). Once the matrix is reduced into a lower dimensional space, pyrecsys can provide predictions, recommendations and similarity among the “elements”.

“from recsys.algorithm.factorize import SVD” : We have used this import statement to load dataset containing 50 different people along with the things they like.

“from recsys.datamodel.data import Data” : We have used this import statement in order to split the data into a lower dimensional space to be used by pyrecsys.

“svd.compute()” : is written to utilize classic Neighborhood SVD. Classic Neighbourhood algorithm uses the ratings of the similar users (or items) to predict the values of the input matrix.

We calculated svd for all the sub categories and then for the final representation we calculates an average of them which gave us a clear picture of the relationship

b. Dataset used:

The dataset of 50 + users that we created randomly for this project with at least 10 different “likes” of each user.

*Small example -
comic*

katy	Spider-Man (Peter Parker)
paul	Captain America (Steven Rogers)
paul	Wolverine (James \"Logan\" Howlett)
paul	Iron Man (Anthony \"Tony\" Stark)
paul	Thor (Thor Odinson)

music

rajat	Take On Me
steve	Baby, Please Don't Go
steve	Back In Black
steve	Big Gun
steve	CAN'T STOP ROCK'N'ROLL
steve	Dirty Deeds Done Dirt Cheap

Movies

rajat	Richard Grayson (New Earth)
rajat	Wonder Woman (Diana Prince)
rajat	Aquaman (Arthur Curry)
rajat	Timothy Drake (New Earth)
rajat	Dinah Laurel Lance (New Earth)
rajat	Flash (Barry Allen)

c. Tools used:

The main tool used in our python program is “python-recsys” also referred to as pyrecsys which is a Python Library for implementing a Recommender System.

python-recsys is a fast recommender engine for Python. It uses matrix factorization to provide recommendations and similarities among items or users.

Under this library the main sub-tool we have utilized is “SVD”. pyrecsys makes use of SVD in order to decompose the input data (a matrix). Once the matrix is reduced into a lower dimensional space, pyrecsys can provide predictions, recommendations and similarity among the “elements”.

Python-recsys supports two Recommender Algorithms: Singular Value Decomposition (SVD) and Neighborhood SVD.

We have used Neighborhood SVD.

d. Screenshot and Demo

The following two screen-shots include our python code utilizing pyrecsys and neighborhood SVD.

Code-

```
from recsys.algorithm.factorize import SVD
from recsys.datamodel.data import Data
import csv
import numpy as np
import matplotlib.pyplot as plt
print ("similarity in marvel comics")
data_file=open('test1.csv','rU')
reader=csv.DictReader(data_file)
likes={}
for row in reader:
    if row['username'] in likes:
        likes[row['username']].append(row['user_likes'])
    else:
        likes[row['username']] = [row['user_likes']]
data_file.close()
data = Data()
VALUE = 1.0
for username in likes:
    for user_likes in likes[username]:
        data.add_tuple((VALUE, username, user_likes)) # Tuple format is: <value,
row, column>

svd = SVD()
svd.set_data(data)
k = 5 # Usually, in a real dataset, you should set a higher number, e.g. 100
svd.compute(k=k, min_values=3, pre_normalize=None, mean_center=False,
post_normalize=True)
```

```

l1=svd.similar('toby')
keylist3=sorted(l1,key=lambda x: x[0])
print(l1)
labels, ys = zip(*keylist3)
xs = np.arange(len(labels))
width = 1
plt.bar(xs, ys, width, align='center')
plt.xticks(xs, labels) #Replace default x-ticks with xs, then replace xs with labels
plt.yticks(ys)

```

```

plt.savefig('netscore.png')
plt.cla() # Clear axis
plt.clf() # Clear figure
#close()
print ("similarity in music")
data_file=open('test2.csv','rU')
reader=csv.DictReader(data_file)
likes={}
for row in reader:
    if row['username'] in likes:
        likes[row['username']].append(row['user_likes'])
    else:
        likes[row['username']] = [row['user_likes']]
data_file.close()
data = Data()
VALUE = 1.0
for username in likes:
    for user_likes in likes[username]:
        data.add_tuple((VALUE, username, user_likes)) # Tuple format is: <value,
row, column>

```

```

svd = SVD()
svd.set_data(data)
k = 5 # Usually, in a real dataset, you should set a higher number, e.g. 100
svd.compute(k=k, min_values=3, pre_normalize=None, mean_center=False,
post_normalize=True)
l2=svd.similar('toby')
keylist2=sorted(l2,key=lambda x: x[0])
print(l2)
labels, ys = zip(*keylist2)

```

```

xs = np.arange(len(labels))
width = 1

plt.bar(xs, ys, width, align='center')

plt.xticks(xs, labels) #Replace default x-ticks with xs, then replace xs with labels
plt.yticks(ys)
plt.savefig('netscore2.png')
plt.cla() # Clear axis
plt.clf() # Clear figure
#close()
print ("similarity in Dc comics")
data_file=open('test3.csv','rU')
reader=csv.DictReader(data_file)
likes={}
for row in reader:
    if row['username'] in likes:
        likes[row['username']].append(row['user_likes'])
    else:
        likes[row['username']] = [row['user_likes']]
data_file.close()
data = Data()
VALUE = 1.0
for username in likes:
    for user_likes in likes[username]:
        data.add_tuple((VALUE, username, user_likes)) # Tuple format is: <value,
row, column>

svd = SVD()
svd.set_data(data)
k = 5 # Usually, in a real dataset, you should set a higher number, e.g. 100
svd.compute(k=k, min_values=3, pre_normalize=None, mean_center=False,
post_normalize=True)
l3=svd.similar('toby')
keylist = sorted(l3,key=lambda x: x[0])
labels, ys = zip(*keylist)
xs = np.arange(len(labels))
width = 1

plt.bar(xs, ys, width, align='center')

```

```
plt.xticks(xs, labels) #Replace default x-ticks with xs, then replace xs with labels
plt.yticks(ys)
plt.savefig('netscore3.png')
print(l3)
plt.cla() # Clear axis
plt.clf() # Clear figure
#close()
```

```
#keylist = sorted(l3,key=lambda x: x[0])
#keylist2=sorted(l2,key=lambda x: x[0])
#keylist3=sorted(l1,key=lambda x: x[0])
#keylist = keylist+keylist2+keylist3
final=[]
final2=[]
```

```
#for a,b,c in zip(keylist,keylist2,keylist3):
#    fin.append(a[1]+b[1]+c[1])
#print (final)
list(keylist)
list(keylist2)
list(keylist3)
```

```
#t = [[0 for x in range(len(keylist))] for y in range(len(keylist))]
```

```
final=[]
for a in range(0,len(keylist)):
    temp=[]
    temp.append(keylist[a][0])
    temp.append(keylist[a][1]+keylist2[a][1]+keylist3[a][1])
    final.append(temp)
```

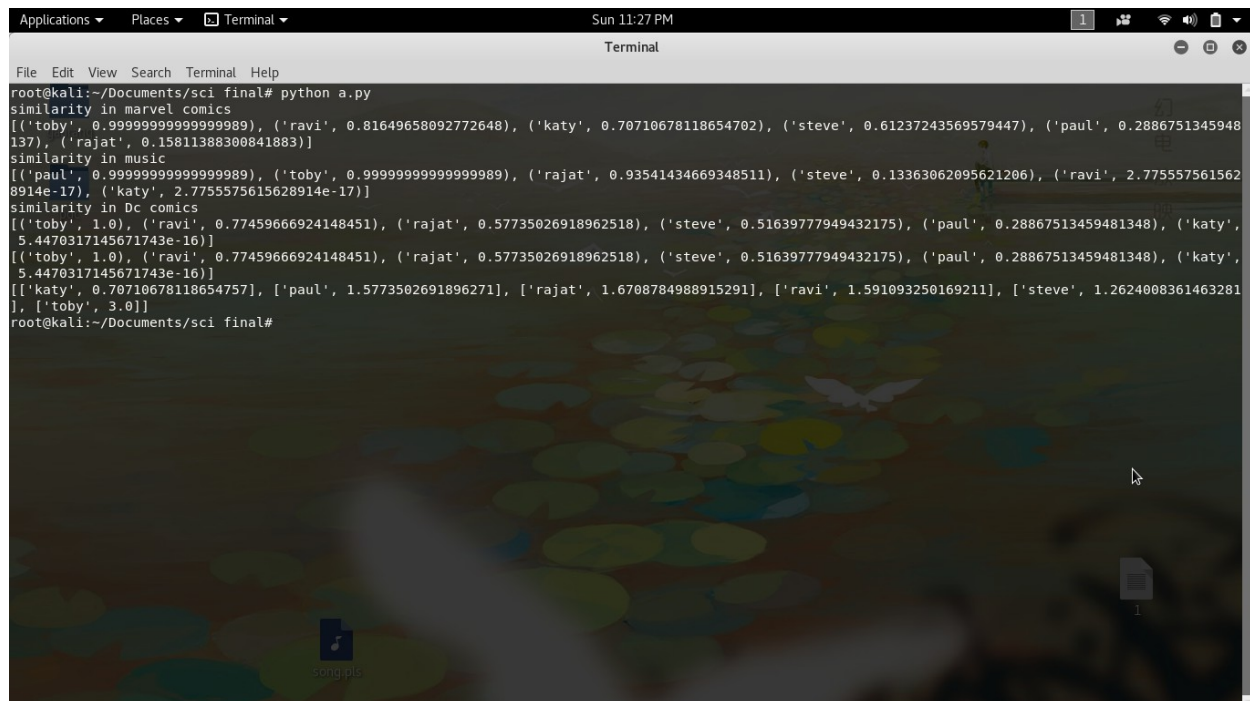
```
labels, ys = zip(*final)
xs = np.arange(len(labels))
width = 1
```

```
plt.bar(xs, ys, width, align='center')
```

```
plt.xticks(xs, labels) #Replace default x-ticks with xs, then replace xs with labels
plt.yticks(ys)
```

```
plt.savefig('final.png')
print(l3)
plt.cla() # Clear axis
plt.clf() # Clear figure
print(final)
```

To show the implementation we have taken two inputs from the user. One input is given as the user “Toby”



```
Applications ▾ Places ▾ Terminal ▾ Sun 11:27 PM
Terminal
File Edit View Search Terminal Help
root@kali:~/Documents/sci final# python a.py
similarity in marvel comics
[('toby', 0.9999999999999999), ('ravi', 0.81649658092772648), ('katy', 0.70710678118654702), ('steve', 0.61237243569579447), ('paul', 0.2886751345948137), ('rajat', 0.15811388300841883)]
similarity in music
[('paul', 0.9999999999999999), ('toby', 0.9999999999999999), ('rajat', 0.93541434669348511), ('steve', 0.13363062095621206), ('ravi', 2.7755575615628914e-17), ('katy', 2.7755575615628914e-17)]
similarity in Dc comics
[('toby', 1.0), ('ravi', 0.77459666924148451), ('rajat', 0.57735026918962518), ('steve', 0.51639777949432175), ('paul', 0.28867513459481348), ('katy', 5.4470317145671743e-16)]
[('toby', 1.0), ('ravi', 0.77459666924148451), ('rajat', 0.57735026918962518), ('steve', 0.51639777949432175), ('paul', 0.28867513459481348), ('katy', 5.4470317145671743e-16)]
[('katy', 0.70710678118654757), ('paul', 1.5773502691896271), ('rajat', 1.6708784988915291), ('ravi', 1.591093250169211), ('steve', 1.2624008361463281), ('toby', 3.0)]
root@kali:~/Documents/sci final#
```

6. Results and discussion

The results obtained in our project are as we claimed in our objective.

We have successfully created a friend recommender system in python using pre-defined pyrecsys library. We included 50+ users in our dataset. Along with this we included at least 10 “likes” of each user for proper implementation of our system.

We are accurately able to suggest friends for any given user based on mutual likes that are searched from pool of different likes in our dataset.

Apart from this for proper assessing and creation of friend circle for any user the relationship with the suggested friends are quantified accurately by using a relationship score of each user with that particular user and displaying only those friends in the suggested friends list whose relationship score is greater than 0.

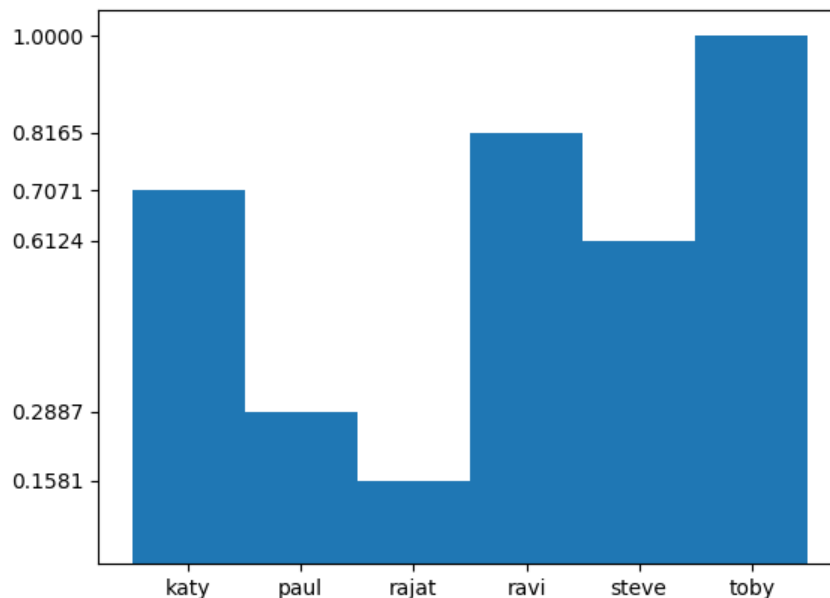


Illustration 1: comic

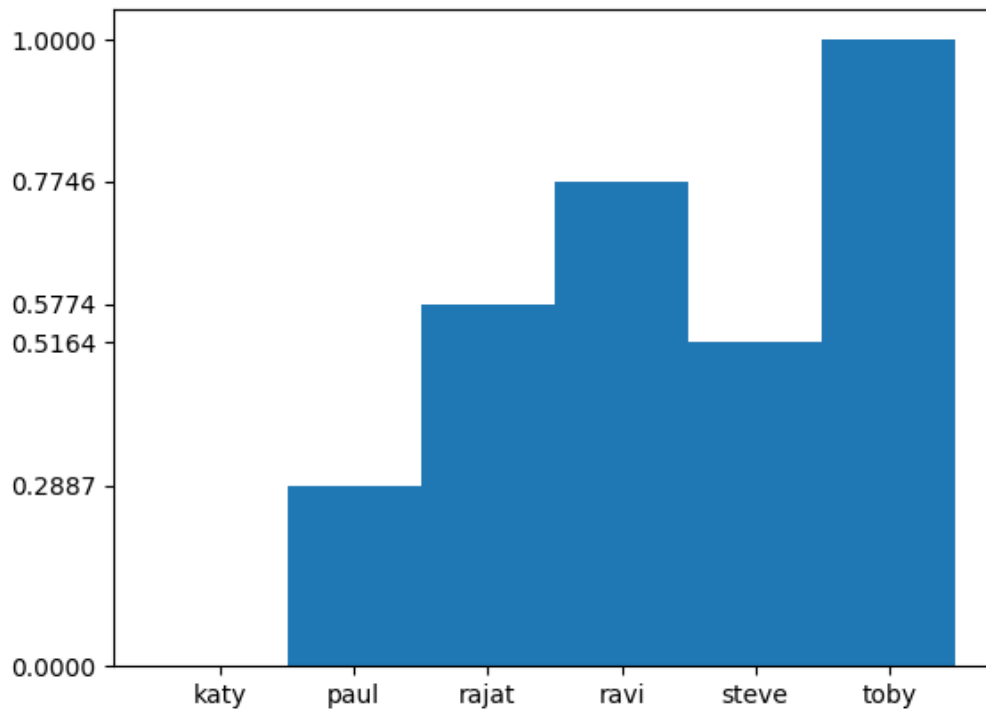


Illustration 2: films

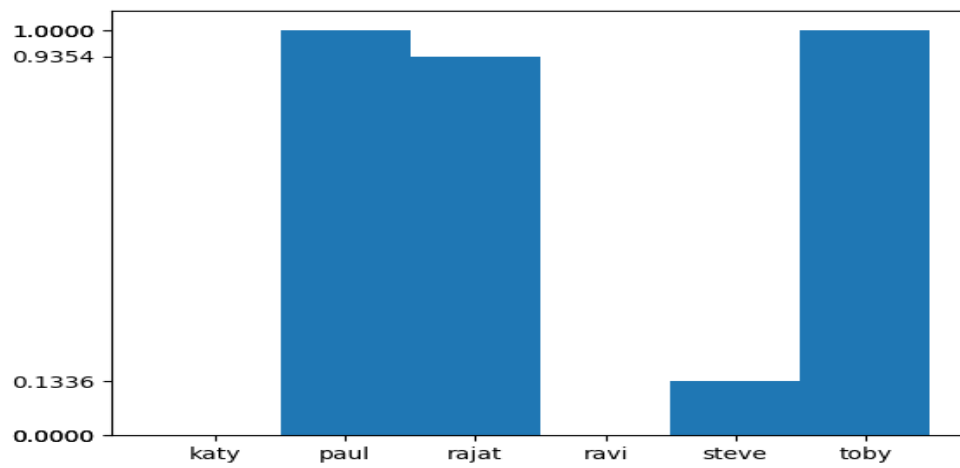


Illustration 3: music

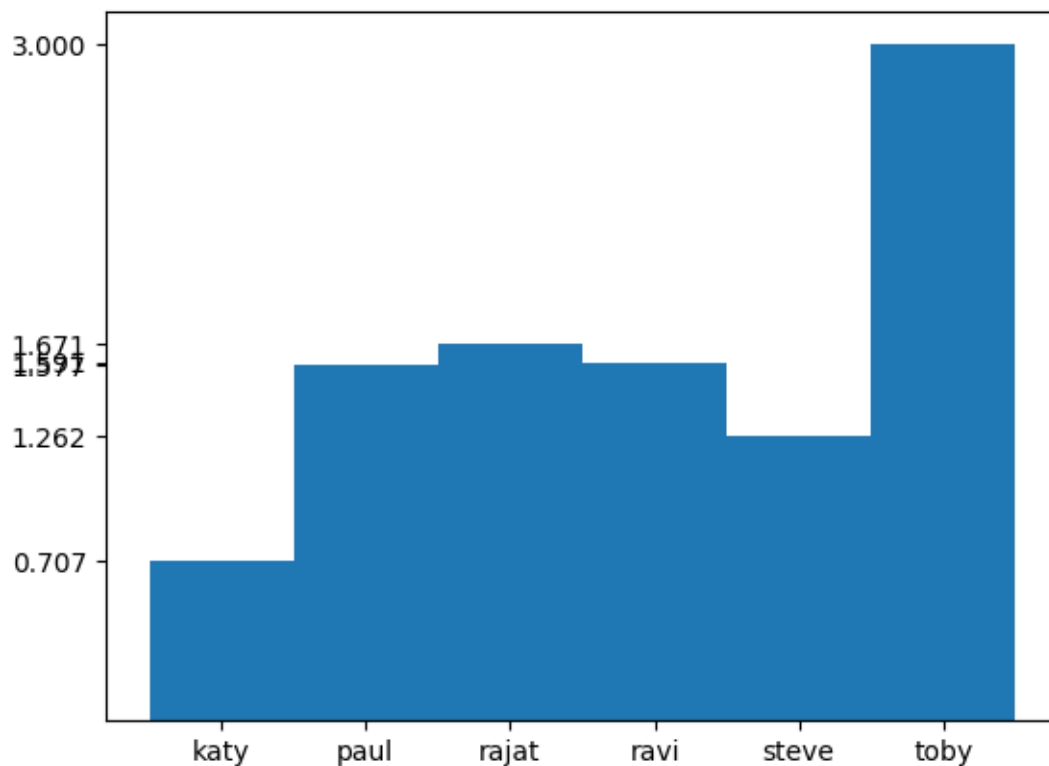


Illustration 4: final

Thus we were successfully able to create a friend recommender system just like a recommender system which is an essential part of large social networking sites including Facebook, Twitter, Instagram and many more which uses advanced and modified version of our system which is a very basic yet effective system on python.

We calculated svd for all the sub categories and then for the final representation we calculates an average of them which gave us a clear picture of the relationship

7. References

J. Chen, W. Geyer, C. Dugan, M. Muller, and I. Guy. Make new friends, but keep the old: recommending people on social networking sites. In Proceedings of the 27th international conference on Human factors in computing systems, pages 201–210, 2009.

J. Naruchitparames, M. H. Giine, and S. J. Louis. Friend recommendations in social networks using genetic algorithms and network topology. In IEEE Congress on Evolutionary Computation (CEC), pages 2207 –2214, 2011.

X. Xie. Potential friend recommendation in online social network. In IEEE/ACM Int’l Conference on Cyber, Physical and Social Computing and Int’l Conference on Green Computing and Communications, pages 831 –835, 2010.

S. Asur and B. A. Huberman. Predicting the future with social media. In Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pages 492–499, 2010

W. Dong, V. Dave, L. Qiu, and Y. Zhang. Secure friend discovery in mobile social networks. In Proceedings IEEE INFOCOM, pages 1647 –1655, April 2011