Fall

2014

SMM - Term Project: Facebook Event Guest Recommendation

Instructor
Dr. Edmund Yu

14st December 2014

Nikhil M Chandrappa

Contents

1. Introduction

1.1. Motivation

Syracuse University is ranked No.1 school in social activities for a reason, there are many social events like volunteering meets, frat meetings, parties etc. organized every week and hundreds of those events will be organized in the future. In this era where social media is rampantly used, easiest way to organize an event is to create a Facebook event and invite friends to them. However it is somewhat tedious and time consuming work to invite select group of people from friends list which runs into 1000's, it requires endless scrolling of the pages.

In a specific use case, if you want to organize a sendoff party of your roommate, it is very likely that close friends of yours and all your roommate will be invited to party. Also its likely that people other than mutual friends will be invited. Facebook currently doesn't provide suggestions for inviting people and I believe there should be a handy tool which suggests invitees to an event hence I would like to create an Facebook Event Guest Suggestion tool.

1.2. Features

For the purpose of recommending guests for events, I will mine the Facebook profile of the person creating the event and recommend guests based on the social interaction he has with his friends. Social interaction can be as simple as liking someone post or photo. In order to thoroughly analyze the interactions, I will consider only the Facebook activities which are considered hotspots for interactions between people. I have categorized the data collection into three groups as specified below,

- Place
  - Hometown
  - Current Location
  - Work places
- Photo
  - Tags
  - Comments
  - Likes
- Posts
  - Comments
  - Likes

Data Collection or mining the profile will not be sufficient to provide accurate recommendations, we have to provide weights for each of the parameter considered above to measure the interaction, followed by normalizing the data and clustering the data to provide recommendations. Result of the recommendation system will includes set of 'Most Likely' and 'Likely' guests.

2. Development Approach

Recommendations system consists of three important steps, 1) Data Collection 2) Normalization 3) Clustering. In the following sections, I will discuss the design considerations of the system and the algorithms which are used in the system.

2.1. Data Collections

First step in recommendation system was to identify set of activities in Facebook which can accurately correlate to social interaction between people, the majority of interaction in Facebook is through updating status, uploading photos, liking and commenting on them. Also, based on the experiment conducted by Dr. Martha Garcia-Murillo we know that people are more likely interacting with others having same origin or coming from same place hence I believe it is essential to consider place in social interaction. I categorized the data collection into,

- Place
- Photo
- post

After Indentifying the items for data collections, I was of the opinion that I can make use of latest Facebook Graph API since I will be mining only the profile who is using the recommendation system. After running couple of queries, I came to know that it is not possible to mine necessary information and had to resort to using Facebook Graph API 1.0

Initial step in the Facebook Graph API is to select the necessary access permissions and generate the access token from the Graph API Explorer, following permissions are required for the recommendation system,

- user_about_me, user_activities, user_photos, user_work_history, user_status, user_friends, user_hometown, user_location
- friends_about_me, friends_work_history, friends_hometown, friends_location
- read_stream, read_requests, export_stream

Following are the list of steps performed for collecting necessary information,

- Create a dictionary of friends to hold the count of 'place', 'tags', 'comments' and 'likes'
- Retrieve hometown, current location and work locations for each friend. calculate match and make entry in dictionary with key 'place'
- Retrieve list of photos. Get tags, comments and likes from them and make corresponding entries in dictionary
- Retrieve List of status updates of types 'status_update', ' mobile_status_update' and ' wall_post'. Get comments and likes make corresponding entries in dictionary

At the end of this process, we will have a dictionary of friends having count of 'place', 'tags', 'comments' and 'likes' which forms the basis of interaction between person using the recommendation system and his friends. Currently I have used a threshold for 365 days, only photos and posts in past 365 days will be considered

<u>WEIGHTAGE</u>

In order to evaluate the social interaction it is essential to provide weights to different data collected from mining the profile. A trivial approach can be to give a certain value to each of the different attributes however to obtain more accurate result there has to be different approach to select weights. After going through literature, I finalized on using Rank order Centroid (RoC) to given weights for 'place', 'tags', 'comments' and 'likes'.

$$W_i = (1/M)\sum_{n=i}^{M}\frac{1}{n}$$

where M is the number of items and W_i is the weight for i^{th} item.

RoC is based on providing ranks to different attributes under consideration. Following ranks are assigned in recommendation system,

Attributes	**Rank**
Tags	1
Comments	2
place	3
likes	4

2.2. Z-score Normalization

In order to effectively analyze the data gathered we need to normalize it in orders of magnitude. Based on the characteristics of the data set where there are some friends with high interaction resulting high weighted score and also some other friends with minimal interactions with low scores. In order to effectively cluster the data, I have used z-score normalization where normalized data tends to be more towards the standard deviation.

$$Normalized\,(e_i) = \frac{e_i - \bar{E}}{std(E)}$$

where

$$std(E) = \sqrt{\frac{1}{(n-1)}\sum_{i=1}^{n}(e_i - \bar{E})^2}$$

$$\bar{E} = \frac{1}{n}\sum_{i=1}^{n} e_i$$

stats package of sciPy library provides the implementation of z-score normalization. An array of weighted scores will passed to zscore() which will be return the normalized data.

2.3. k-means clustering

k-means clustering is a widely used unsupervised clustering algorithm which clusters set of observations into subsets such that all the elements within the group are more similar among them than they are to others.

I make use of the k-means algorithm to cluster friends data set into three groups - 'most likely', 'likely' and 'unlikely'.

The procedure alternates between two operations. (1) Once a set of centroids μ_k is available, the clusters are updated to contain the points closest in distance to each centroid. (2) Given a set of clusters, the centroids are recalculated as the means of all points belonging to a cluster.[4]

$$C_k = \{\mathrm{x}_n : ||\mathrm{x}_n - \mu_k|| \le \text{ all } ||\mathrm{x}_n - \mu_l||\} \qquad (1)$$

$$\mu_k = \frac{1}{C_k}\sum_{\mathrm{x}_n \in C_k} \mathrm{x}_n \qquad (2)$$

I have used the sciPy implementation of kscore and I have configured it to work with random initialization of centroid and also with pre-determined centroid

```
res, idx = kmeans2(zscore,k)
```

In the above code, k specifies the number of clusters or ndarray of pre determined centroids

With **random initialization of centroid**, I noticed that sometimes it categorize the friends in just two clusters based on the initial centroid selected hence its preferable to use the second approach, to specify the pre determined centroid which results in more effective clustering of nodes.

k = np.array((centroid[0], centroid[1], centroid[2]))

centroid[0]: provide facebook id who is "most likely" to be invited
centroid[1]: provide facebook id who is "likely" to be invited
centroid[2]: provide facebook id with least interaction(Unlikely)

Note: Initially I plan to user semi supervised version of kmeans algorithm, however there is no library implementation of it and because of time constraints I couldn't implement the algorithm. Hence resorted to using kmeans algorithm with pre determined centroids.

3. Architecture

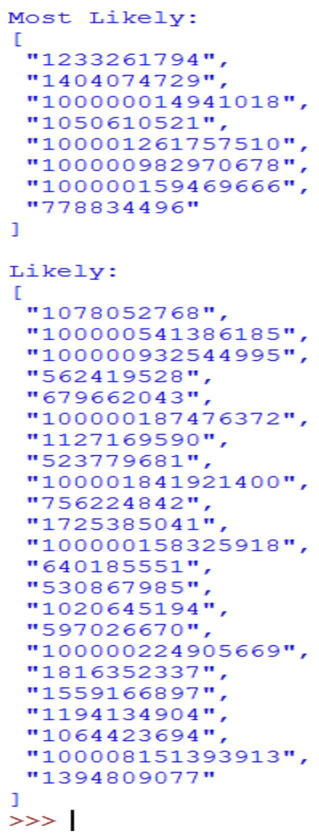
Recommendation system is designed in a modular manner where code is separated among different mining activities into four different modules,

- Guest_Recommendation.py - This is main module of the system which is responsible for invoking the modules harvesting profile information. Also, this module performs the normalization and clustering of the weighted score
- PlaceInfo.py - This module mines the place information of friends and calculates the place score
- PhotoInfo.py - This module mines the photos. It retrieves tags, comments and likes of each photo.
- PostInfo.py - This modules mines the posts. It retrieves comments and likes of each post.

4. output

Output will return set of 'Most Likely' and 'Likely' Facebook id's,

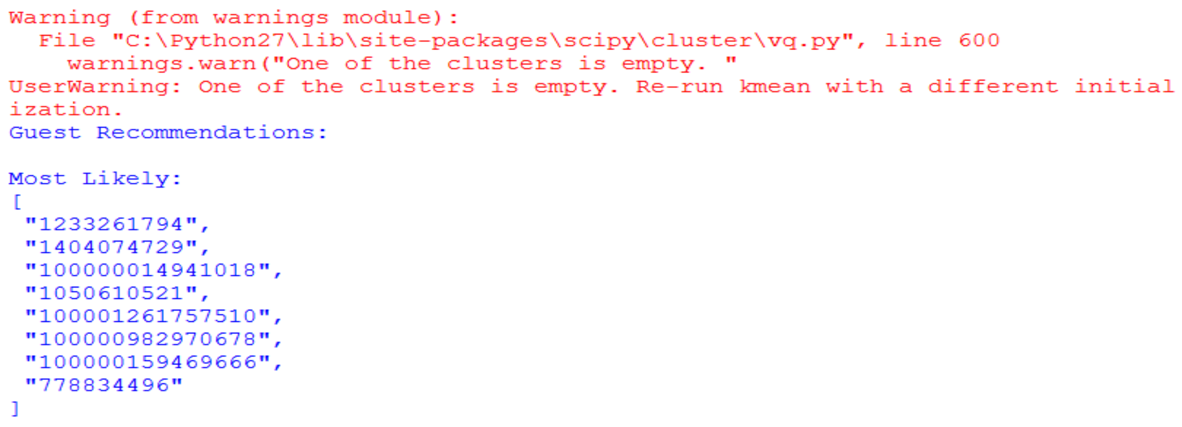
Option 1: Random Initialization of Centroid



```
Most Likely:
[
 "1233261794",
 "1404074729",
 "100000014941018",
 "1050610521",
 "100001261757510",
 "100000982970678",
 "100000159469666",
 "778834496"
]

Likely:
[
 "1078052768",
 "100000541386185",
 "100000932544995",
 "562419528",
 "679662043",
 "100000187476372",
 "1127169590",
 "523779681",
 "100001841921400",
 "756224842",
 "1725385041",
 "100000158325918",
 "640185551",
 "530867985",
 "1020645194",
 "597026670",
 "100000224905669",
 "1816352337",
 "1559166897",
 "1194134904",
 "1064423694",
 "100008151393913",
 "1394809077"
]
>>> |
```

Fig (a)



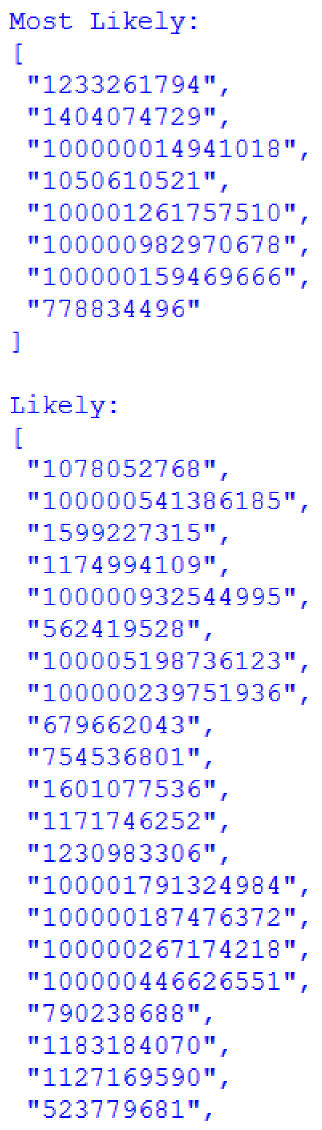
```
Warning (from warnings module):
  File "C:\Python27\lib\site-packages\scipy\cluster\vq.py", line 600
    warnings.warn("One of the clusters is empty. "
UserWarning: One of the clusters is empty. Re-run kmean with a different initial
ization.
Guest Recommendations:

Most Likely:
[
 "1233261794",
 "1404074729",
 "100000014941018",
 "1050610521",
 "100001261757510",
 "100000982970678",
 "100000159469666",
 "778834496"
]
```

Fig(b)

As show in the fig(b) in random initialization of centroid, sometimes cluster with moderate interaction will be empty and hence clustering is dependent on centroid. From the above results, we can infer that more accurate clustering will be obtained by selecting predefined clusters.

Pre-defined Clusters:



```
Most Likely:
[
 "1233261794",
 "1404074729",
 "100000014941018",
 "1050610521",
 "100001261757510",
 "100000982970678",
 "100000159469666",
 "778834496"
]

Likely:
[
 "1078052768",
 "100000541386185",
 "1599227315",
 "1174994109",
 "100000932544995",
 "562419528",
 "100005198736123",
 "100000239751936",
 "679662043",
 "754536801",
 "1601077536",
 "1171746252",
 "1230983306",
 "100001791324984",
 "100000187476372",
 "100000267174218",
 "100000446626551",
 "790238688",
 "1183184070",
 "1127169590",
 "523779681",
```

Fig(c)

5. Conclusion

After analyzing the above results, I conclude that 'Most Likely' suggestions are 100% accurate as I would invite all them to any event I host. From 'Likely' suggestions, 69% of suggested friends would be invited but still recommendation system considerably reduced the number of friends considered while forming the guest list of an event. There are quite a few false-negative, friends ids listed in unlikely cluster because of inadequate social interaction on Facebook.

From the results following observations can be made, accuracy of the results depends on the how active are your inner circle of friends on Facebook, greater the social interaction better are the results of Guest Recommendation System.

6. References

- http://www.tcrponline.org/PDFDocuments/tcrp_rpt_131AppF.pdf
- http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/K-Means
- http://www.enggjournals.com/ijet/docs/IJET13-05-03-273.pdf
- http://cs229.stanford.edu/DaniyalzadeLipus-FacebookFriendSuggestion.pdf