Abstract

We live in the age of technology and at the very center of it is communication. The world has gladly accepted communication over the internet making social networking platforms extremely popular. The number of users in the social networks is increasing day by day. Besides the traditional desktop PCs and laptops, new emerging mobile devices makes accessing the social networking platforms much easier. Social networking platforms thrives on its users connecting and communicating with each other. With so much activity on the social networking platforms, the users are constantly engaged and do not explicitly search for people they know, on the platform unless they really need to. Therefore, an algorithm to suggest other users to a user based on some criteria is fundamental to any social networking platform.

Introduction

Social networking platforms have become an important part of our lives and online friendships have now become similarly appealing as offline friendships if not more. People tend to enjoy the companionship of their real-life friends in the virtual world and at the same time, are ready to form new friendships online. But, with increasing online crimes it is hard to guess whom to trust online. The recent surge of research in the friend recommendation algorithms is not surprising. Historically there have been two main recommendation algorithms, content based and collaborative based. Content based algorithm requires textual information as its name suggests and recommends websites, news paper articles, blogs and other contents. Collaborative based algorithms recommends products to a user which it believes has been liked by similar users. Both of these algorithms have yielded unsatisfactory results in friend recommendation systems. In real-life scenarios, people tend to trust strangers who are friends of their friends as they trust in the judgement of their friends. Higher the number of mutual friends between them, higher can be the trust between strangers meeting for the first time. This behavior can be mimicked by the friend suggestion system where strangers can be introduced to each other where they have some mutual friends, making them trustworthy to each other.

Related Work

There has been a lot of related work in the field of recommendation algorithm based on different aspects of the graphs and user behavior as node in the graph. Some of them are mentioned below:

- Context-based friend suggestion in online photo-sharing community

  “We propose a new approach, named context-based friend suggestion, to leverage the diverse form of contextual cues for more effective friend suggestion in the social media community. Different from existing approaches, we consider both visual and geographical cues, and develop two user-based similarity measurements, i.e., visual similarity and geo similarity for characterizing user relationship. The problem of friend suggestion is casted as a contextual graph modeling problem, where users are nodes and the edges between them are weighted by geo similarity. Meanwhile, the graph is initialized in a way that users with higher visual similarity to a given query have better chance to be recommended”

  DOI - 10.1145/2072298.2071909

- Ego-net community mining applied to friend suggestion

  “We, then, develop new features for friend suggestion based on co-occurrences of two nodes in different ego-nets' communities. Our new features can be computed efficiently on very large scale graphs by just analyzing the neighborhood of each node. Furthermore, we prove formally on a stylized model, and by experimental analysis that this new similarity measure outperforms the classic local features employed for friend suggestions”

  DOI - 10.14778/2856318.2856327

- Suggesting friends using the implicit social graph

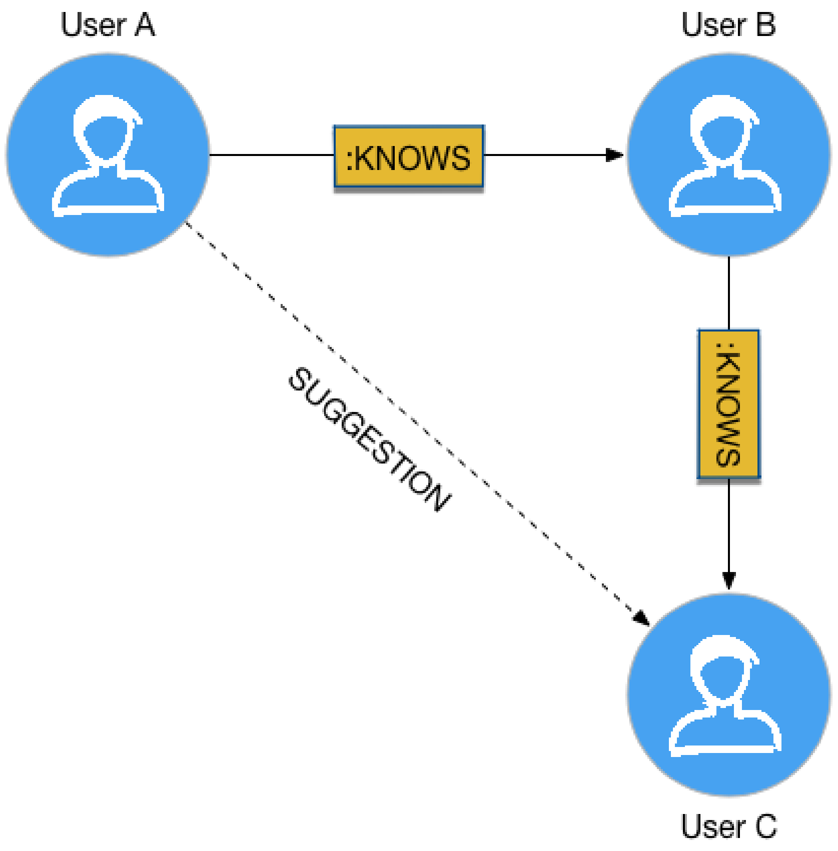
  “We introduce an interaction-based metric for estimating a user's affinity to his contacts and groups. We then describe a novel friend suggestion algorithm that uses a user's implicit social graph to generate a friend group, given a small seed set of contacts which the user has already labeled as friends”

  DOI - 10.1145/1835804.1835836

Methodology

The algorithm suggests users to other users based on the number of mutual friends between them. The algorithm uses an undirected graph, where each node is a user and the edges of the graph indicates the relation between them i.e. is there is an edge between two nodes then those two users are friends.

This algorithm has been developed for social networks which prefer bidirectional friendships i.e. if user A is a friend of user B then user B is also a friend of user A unlike some social networks which implement directional graphs, having the follow relationship rather friendship where user A follows user B but user B may or may not follow user A. The graph used by the algorithm is unweighted which means there are not stronger or weaker friendships in the network and all the friendships are equivalent. The algorithm also doesn't allow self-loops where there can be an edge from and to the same node.



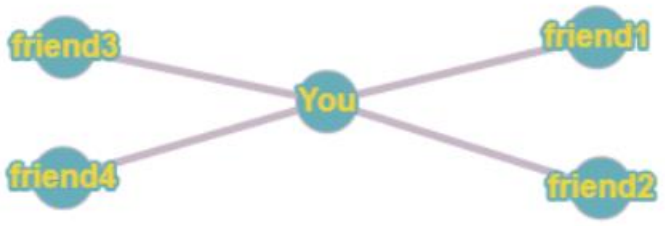
The algorithm works on the property of triadic closure. Triadic closure is the property of social networks where if node A is connected with node B and node B is connected with node C, then the nodes A and C will be attracted to each other to form an edge between them. Or we can say that if two nodes have one or more common nodes between them, then they tend to be attracted towards each other to form a bond.

The algorithm closely follows this principle and tries to close triangles in the network. For any user, the algorithm first computes potential friends. To do so, it first finds the current friends on the user, then one by one for each friend of the user it finds the friends of the friend and adds to the acquaintance list. This acquaintance list has a counter for the number of times an acquaintance is appearing in the lists of friends of friends of the user. Every time the algorithm encounters a user in the list of friends of any friend of the user, it first checks if this new person is already in the acquaintance list, if it is then it would increment the counter for that user else if it is not in the acquaintance list, it will add it to the list and set its counter value as one. Once all the friends of the friends of the user have been traversed, then the algorithm starts suggesting the user other users in the decreasing order of the counter values, implying any user in the network with the highest number of mutual friends with the host user will be suggested first, the user with the second highest number of mutual friends would be suggested second and so on.

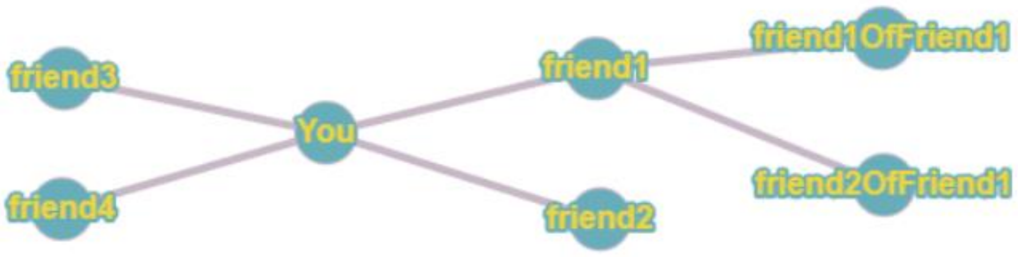
The user which the algorithm is suggesting as the first suggestion for the host user, will close the most number of open triangles in the network. The algorithm is essentially trying to close all possible triangles and will only stop suggesting friends when the user is connected with each and every user in the network. After one computation cycle, if the user connects with all the suggested users then it is not the end of possible suggestions for the user. Every new edge in the network, every new connection mane by any user opens new doors for multiple other users in the network to connect to and discover other new users in the network who are now or were always closer to them in the network.

An example of how the algorithm works in a network has been given below:

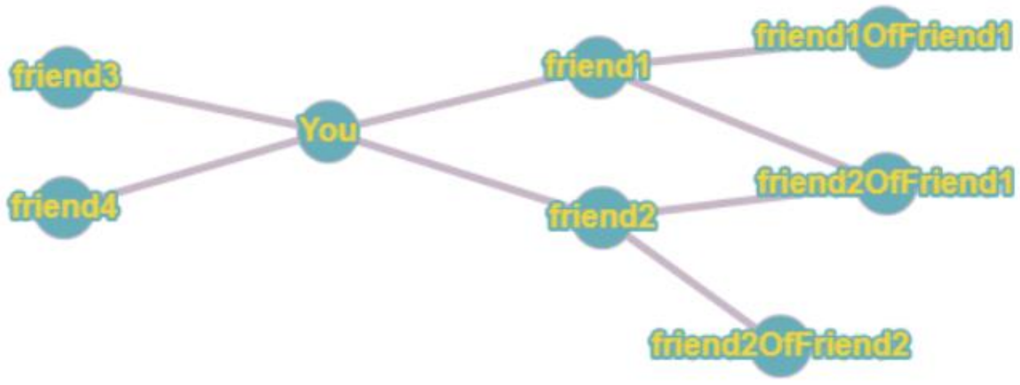
The algorithm starts with you in the network and tries to discover your own friends first.



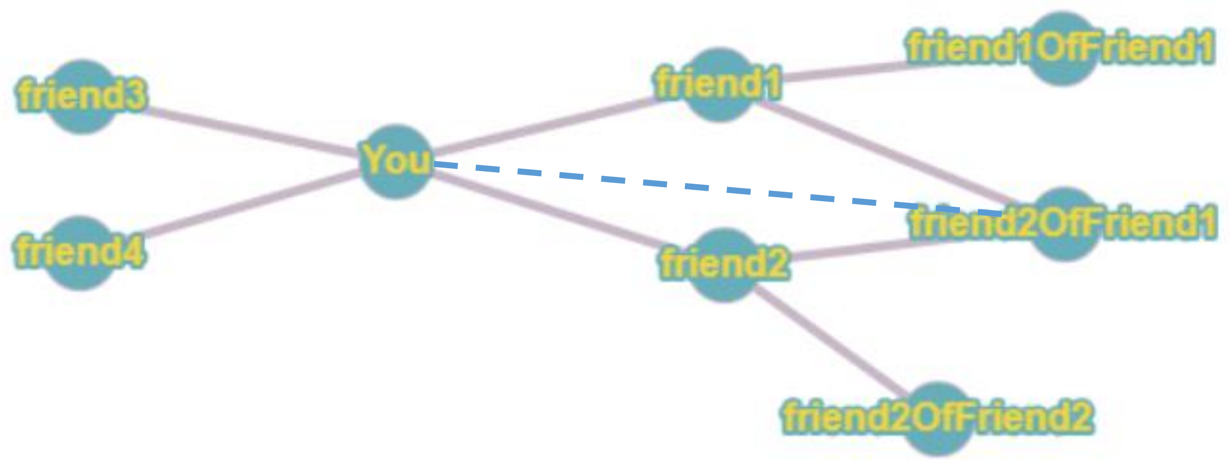
The algorithm will then start traversing among your friends one by one and will try to discover the friends of your friends.



Now as the algorithm as discovered the friends of the first friend of yours, it will backtrack and move on to the next friend of yours.



Now as one of the user in the network is a friend of two of your friends, or we can say while traversing the list of friends of your friends, as the algorithm was adding users to the acquaintance list, if the user already exists in the list, the algorithm will increment the counter for this user.



Assuming there were no other friends of your friends in the network, the algorithm will, in this case, suggest the user with counter value 2 to the user. Notice out of all the three possible acquaintances, the user suggested by the algorithm completes the most number of triangles which is 2 in this case. If the user connects with any other node than the one suggested by the algorithm, then only one triangle can be completed at a time.

The algorithm is trying to increase the clustering coefficient of each node in the network and ultimately increase the average clustering coefficient of the algorithm. Clustering coefficient is the measure of the degree to which nodes in a graph tend to cluster together. We can say it the measure of how connected the immediate neighbors of any nodes are. When all the neighbors of the node are connected with each other, then the clustering coefficient of the node is equal to one. The average clustering coefficient of the graph is the sum of the clustering coefficient of all the nodes in the network divided by the number of nodes in the network. Only when all the nodes have clustering coefficient value equal to one, which means each and every person in the network knows each and every other person in the network, only then the average clustering coefficient of the graph will be one. If such a state of the graph is reached, then the algorithm will be left with no suggestions to be made. As with every suggestion made by the algorithm, provided the user connects with the suggested user, the clustering coefficient should move closer to one.

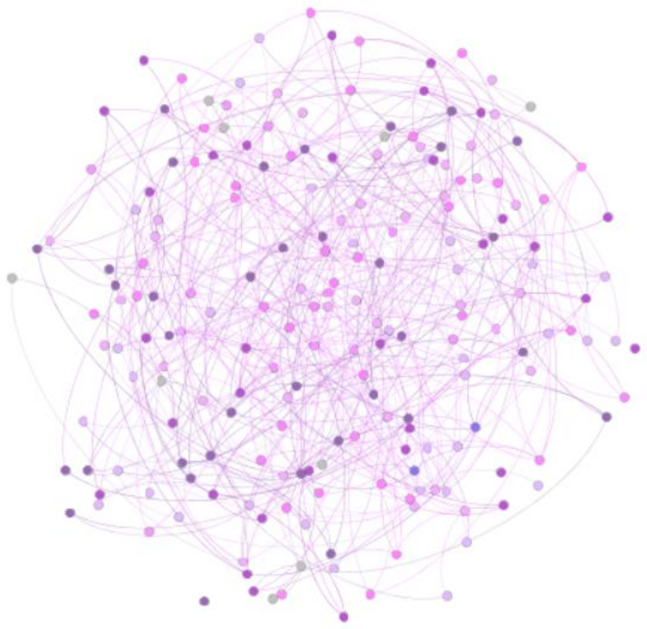
Results

The algorithm was tested on an un-directed and un-weighted graph with 2000 nodes and 40,000 edges. The average degree of nodes was 39 at beginning. To test the algorithm, it was made to select any random node in the graph and compute the acquaintance list. Then just add the user with the highest value of the counter in the list as a friend by registering an edge between the two nodes. The algorithm was made to do so for 100 random nodes in one run. Below are the results after 500 runs of the algorithm.

	Before	After
Average Degree	39	87
Average Clustering Coefficient	0.019	0.425
Triangles	10,065	937,476
Nodes	2000	2000
Edges	40,000	90,000
Diameter	3	2
Path Length	2.45	1.95

The worst case complexity of the suggestion algorithm = O(n^2)

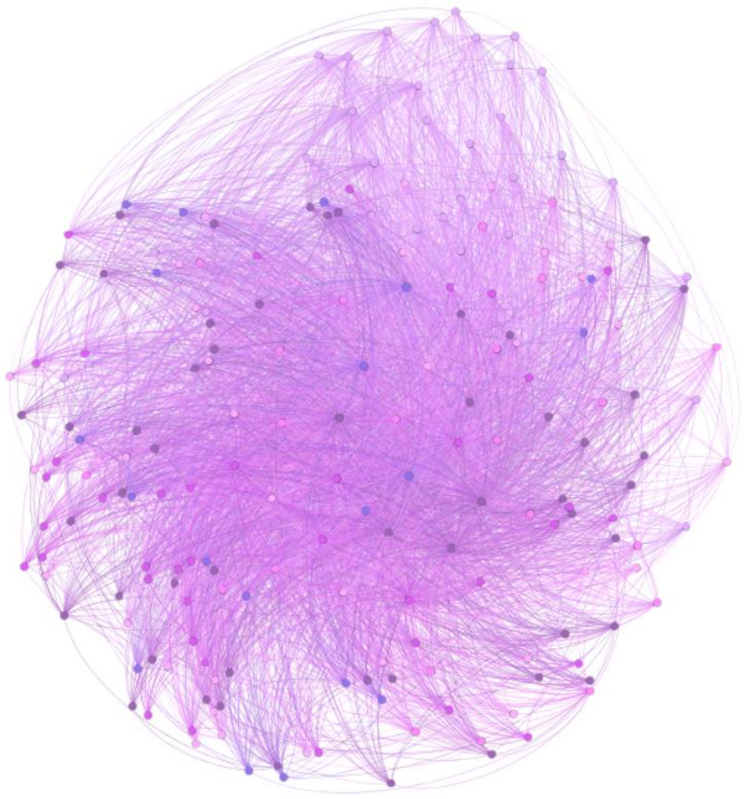
Where n is the number of users in the network but practically there is a very slim chance of the algorithm reaching it's worst case as after 500 runs of the algorithm which adds 100 new friends each making it total of 50,000 new friends made in the network, the average degree of each node is 87. In the real world, it is not very likely that even one user of the network is friends with all other users in the network.



Initial graph with

2000 Nodes

40,000 Edges

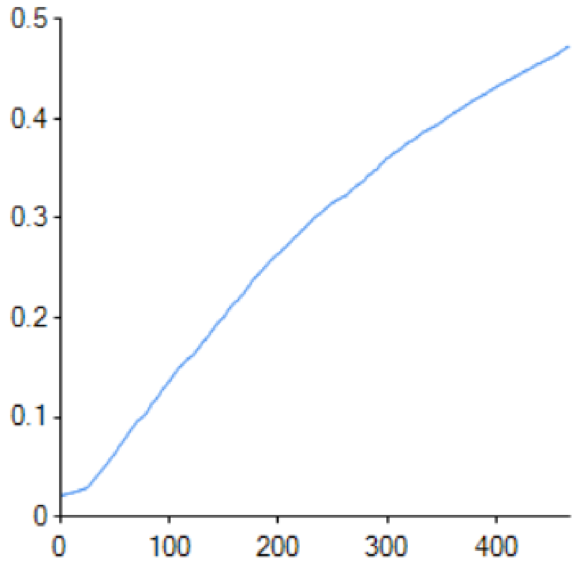


Graph after 500 runs

2000 Nodes

90,000 Edges

<u>Proof of correctness of the algorithm</u>



The algorithm tries to complete triangles and increase the clustering coefficient of each node thereby increasing the average clustering coefficient of the graph.

The graph shows the increase in the average clustering coefficient of the graph computed after each run of the 500 runs of the algorithm.

Discussion

Limitations: This version of the algorithm is based solely on the current friends of the user and the friends of their friends so if there is a new user who has created a new account and is not connected to any other user in the network, the algorithm wont be able to suggest any users to this user. Although the algorithm can be slightly modified to follow power law in such cases where any new user can be pointed to the most popular node in the network if it doesn't have any connection with any other node of the network.

Future Scope: The algorithm traverses the graph and visits every node which is a friend of friend. This implementation just counts the number of appearance of the node in acquaintance list however, the algorithm can be altered as needed to count not the number of occurrences of the node itself but some property of the node like the interests of the node eg – Music, so the algorithm will then maintain the counter of acquaintances who have most number of matching interests (like music).

Acknowledgement

I would like to thank Dr. Zoran Obradovic, CIS Dept. Temple University for the expert guidance and informative discussions, pointing me to the right direction when required and helping me complete this project.

References

1) Networks, Crowds, and Markets: Reasoning about a Highly Connected World - David Easley & Jon Kleinberg
    - Chapter 3 – Strong and Weak Ties – Triadic Closure

2) Graphs in the database: SQL meets social network
    - https://inviqa.com/blog/graphs-database-sql-meets-social-network

3) https://en.wikipedia.org/wiki/Clustering_coefficient