ANONYMIZATION OF SOCIAL NETWORKS

Team: int_elligence (dc3383,sc4401, ik2442)

I. INTRODUCTION:

Social Network Analysis can be defined as the process of exploring and analyzing social structures through the use of networks and graph theory.
Networks are characterized in terms of nodes (actors or objects) and links (relationships or interactions) between the nodes. With the rapid growth of social networks, such as Facebook and Linkedin, more and more searchers found that it is a great opportunity to obtain useful information from these social network data, such as the user behavior, community growth, disease spreading, etc. However, it is paramount that published social network data should not reveal private information of individuals.

Such kind of cases in extremely prevalent during current time where lots of companies give their data to different companies for analysis. The nodes in a social network can have several attributes associated with them. Some of these attributes contain very private information. These nodes are termed as *'sensitive attributes'*. Though it may appear that not publishing the sensitive attribute would ensure the privacy of the sensitive attribute, it is not true. Similarity attacks utilize other published information to identify the individuals and their sensitive attribute. Background knowledge attacks utilize pre-existing knowledge to attack the privacy.

This is where anonymization of social networks comes in. Companies need to make sure that patterns are safe and all the sensitive attributes and relationship are not recognized.

A process of anonymization has generally the following flow.

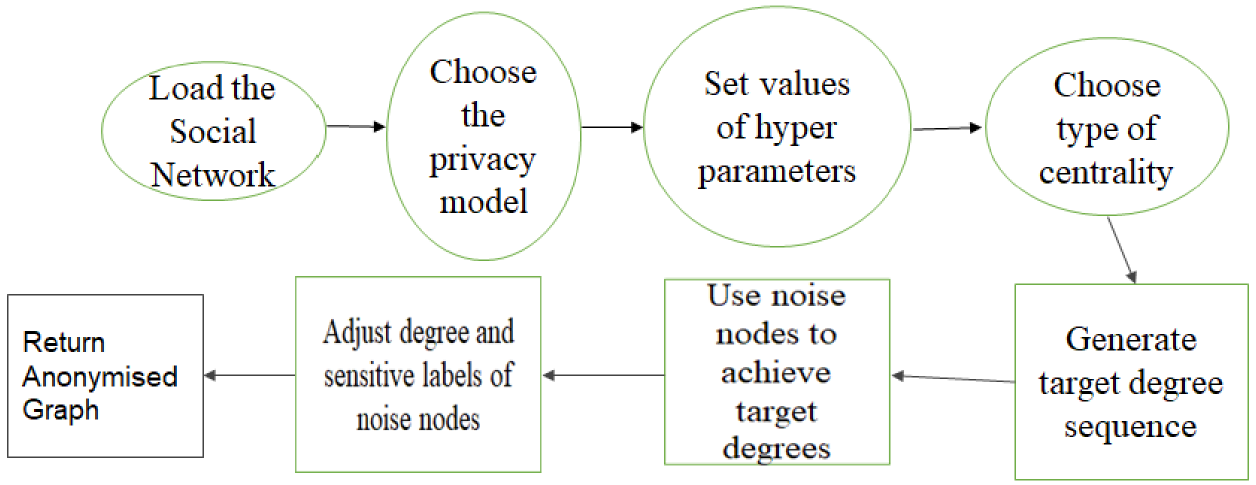

Load the Social Network
Choose the privacy model
Set values of hyper parameters
Choose type of centrality
Generate target degree sequence
Use noise nodes to achieve target degrees
Adjust degree and sensitive labels of noise nodes
Return Anonymised Graph

Fig 1. Flow-Chart for anonymization using noise node addition algorithm.

II. <u>AIM:</u>

We wanted to compare two anonymization methods namely edge editing and noise node edition (Corresponding to label 1 in *Fig 2*). While doing that we also wanted to compare sub parameters for anonymization namely privacy measures (Corresponding to label 4 in *Fig 2*). and centrality values (Corresponding to label 4in *Fig 2*). that could be used. While making these comparisons, we wanted to also analyze these methods from software engineering perspective i.e. analyzing time, memory, effort taken to develop the code.

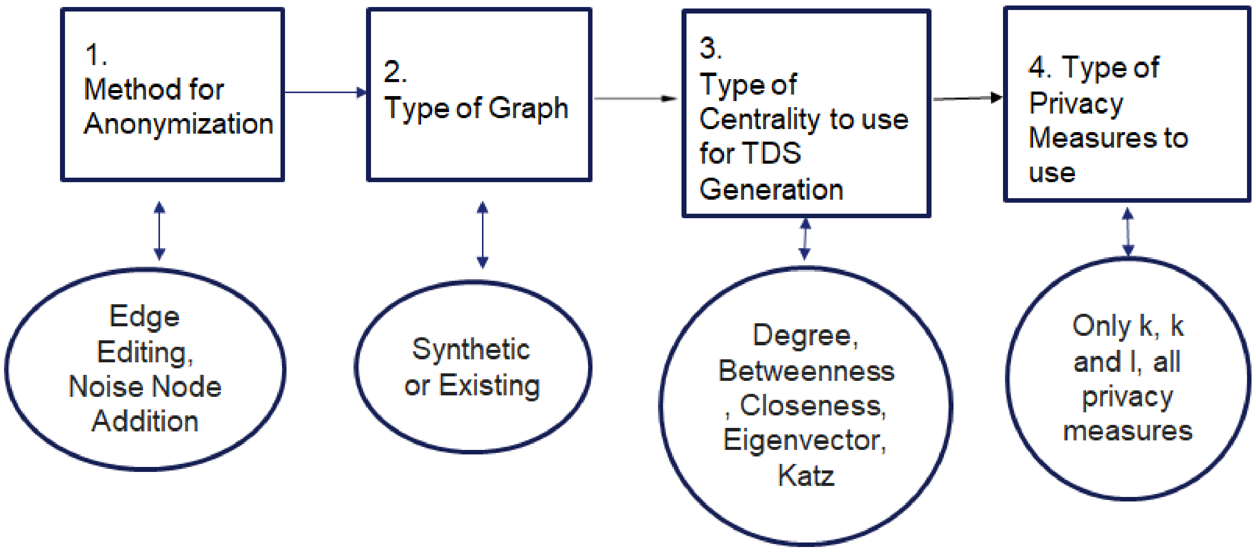

1. Method for Anonymization
2. Type of Graph
3. Type of Centrality to use for TDS Generation
4. Type of Privacy Measures to use
Edge Editing, Noise Node Addition
Synthetic or Existing
Degree, Betweenness , Closeness, Eigenvector, Katz
Only k, k and l, all privacy measures

Fig 2. Parameters that we wanted to test on

But there was no sample code base available that could be used to properly analyze different parameters and subfunctions for anonymization. A code base where all the different parameters and method seen in the graph above are all properly included.

This is where the novelty of our project comes in. We developed our own code base, which can be used as a base for creating anonymized social networks.

- This code base will be useful for people who need to create anonymized social networks for different data and want to choose their sub parameters like centrality or privacy measures. Any organization which aims to release graphical data which includes sensitive information will benefit by using our anonymization model.
- This code base could act as a base for different kinds of research endeavors where people want to compare anonymization of social graphs with different methods. Researchers and developers can easily build upon our code base because of it being highly modularized.

- We have also included different tests facilities in our code like memory, time and also average path length, centrality etc. These are helpful tools that could be used by researchers.

Apart from the code base we have developed, the questions below are also research questions that we were able to answer as a virtue of extensive testing:

1. Which method among noise node addition and edge editing performs better in terms of both algorithmic and ease of software development perspective?
2. Apart from the two methods above, amongst the rest of the parameters namely centrality values, types of privacy measures and values of privacy measures.
3. Finally, all these methods are for single sensitive labels. So, can we extend the definition of the privacy measures to account for multiple sensitive attributes without compromising the structural integrity of the original social network while obtaining comparable results to single sensitive labels?

III. CODEBASE:

https://github.com/dhruvchamania/TSE_Final_Project

These contain in detail things like the dataset and tests that we conducted.

Dataset:

We during our testing used the data primarily from the following sources:

1) http://www-personal.umich.edu/~mejn/netdata/ (This data was compiled by Prof Mark Newman)

2) http://snap.stanford.edu/data/#amazon (This data was compiled by Stanford Labs)

3) https://github.com/gephi/gephi/wiki/Datasets

The 4 datasets that our tests based on are:

1. Karate.gml : A very small dataset
2. Netscience2.gml: Dataset of order 10^3
3. Netscience.gml: Dataset of order 10^3
4. as-july22july06.gml: Dataset of order 10^4 (Restricted testing)

TESTING:

1. For each, a single method and a single dataset, there were 15 permutation that we tested own. Namely 5 centralities and 3 different privacy measures.
2. We tested all of these methods for 4 different values of k & l.
3. There were 3 datasets that we tested on.
4. So, for any one method, we have tested 180 different times (15x4x3)

And there were 4 different tests that we conducted for each of these permutations:

1. Amount of memory the process consumed.
2. Time taken for the anonymization
3. *Error in path length ->Very important property of graph, tells us how much has the graph changed*
4. Top 20 centrality. Centrality value for top two-degree nodes.

1&2 are more interesting to a software engineer and 3&4 are more interesting to a researcher. Apart from this, we compare the development difficulty between noise and edge editing and also node/edge difference for noise node/edge editing respectively.

IV. RESULTS:

In this section, we will go over what the results that we have obtained research question wise. (All the data for this can be found in the "docs/Compete_Summary" of our codebase. Below are our test results across different

1. Which is better edge editing or noise node?

In terms of software development implementing the noise node was trickier. But our good modularized code helped us not feel much difference between the two methods. Below are the graphs for time (Fig 3), memory (Fig 4), error in path length (Fig 5) and top 20 centrality values (Fig 6).

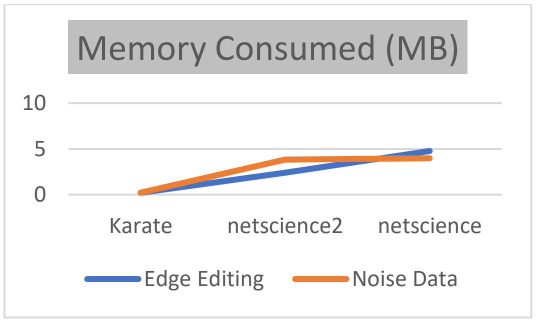


Fig 3

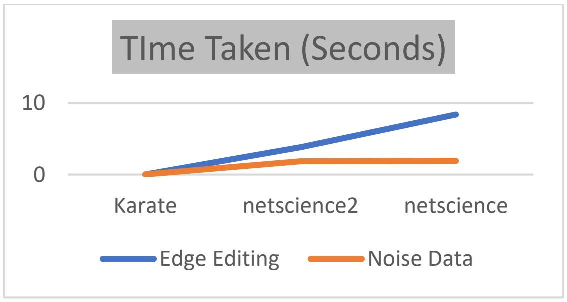


Fig 4

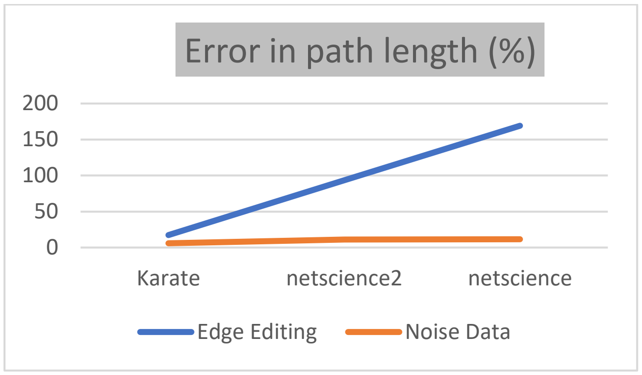


Fig 5

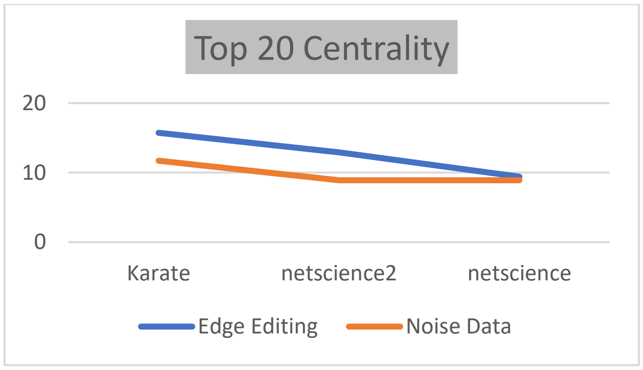
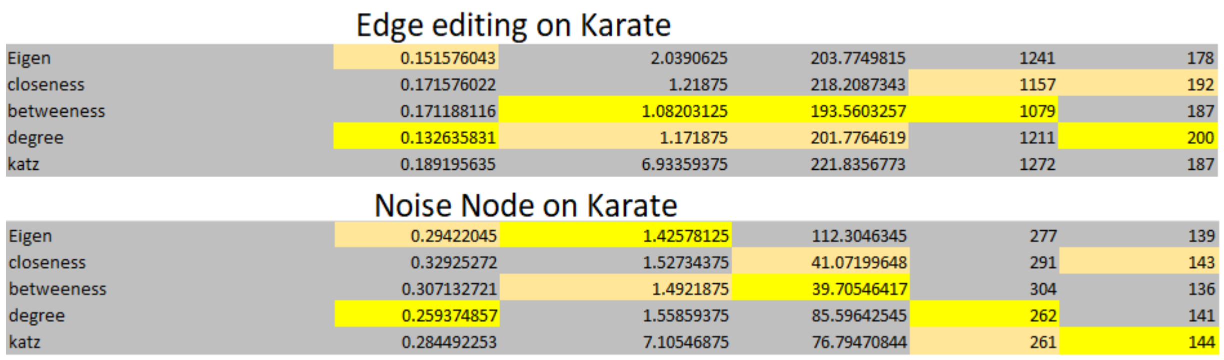


Fig 6.

As seen from the graph above, the values for all the values gives slight edge to either one or the other but error in path length is very high for edge editing. Error in path length is a very important characteristic and edge editing is performing very poor.

Thus, we can conclude that Noise Node addition is better of the two methods.

2. Can we say something regarding the rest of the parameters?
   a) Centrality values
   On an average degree centrality was performing the best followed by eigenvector centrality and betweenness centrality. But these were across the first three datasets of the range 10^3. But when we ran these tests over the internet database which was of a much higher order dimension wise, eigenvector centrality was the only one that was able to work properly where for the rest of centrality especially closeness and betweenness the time complexity went haywire.



Edge editing on Karate

Eigen	0.151576043	2.0390625	203.7749815	1241	178
closeness	0.171576022	1.21875	218.2087343	1157	192
betweeness	0.171188116	1.08203125	193.5603257	1079	187
degree	0.132635831	1.171875	201.7764619	1211	200
katz	0.189195635	6.93359375	221.8356773	1272	187

Noise Node on Karate

Eigen	0.29422045	1.42578125	112.3046345	277	139
closeness	0.32925272	1.52734375	41.07199648	291	143
betweeness	0.307132721	1.4921875	39.70546417	304	136
degree	0.259374857	1.55859375	85.59642545	262	141
katz	0.284492253	7.10546875	76.79470844	261	144

Fig 7. Examples of different centrality values across different test parameters. Yellow depicts the best and orange second best

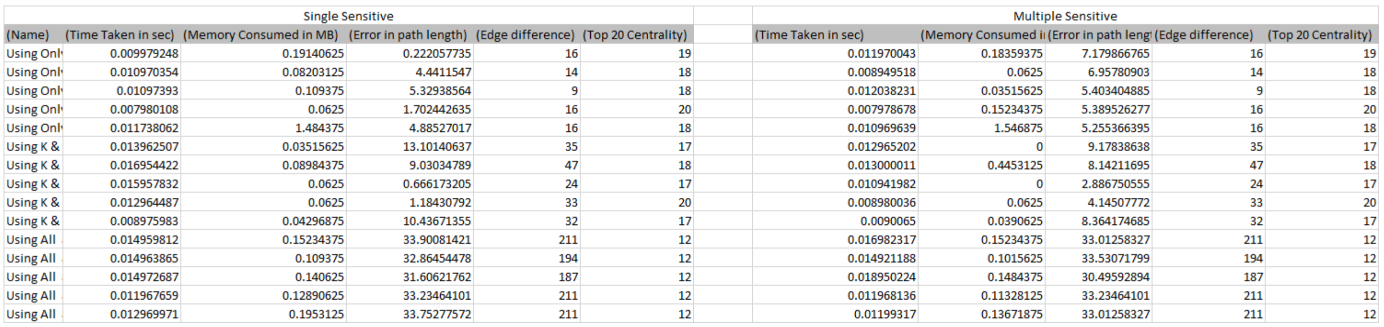
We can conclude that eigenvector centrality is the most ideal although for smaller social networks degree centrality is the best.

b) <u>Privacy Measures and different values of k&l</u>

Analyzing data for different privacy measures and different values of k & l did not lead us any results. The different test parameters did not yield any discernable patterns. Also, with respect to number of privacy measures we introduce, i.e. from k, k to l and then all the 4. There are no discernable patterns that we can make sense of.

3. <u>Whether it is possible to extend the existing theory to two multiple sensitive attributes?</u>

We were able to extend and create our own multiple sensitive attribute. It has worked well as seen in the figure *Fig 8* where we can see that the error in path length is not a lot different. And the rest of the important attributes remain constant.



	Single Sensitive						Multiple Sensitive				
(Name)	(Time Taken in sec)	(Memory Consumed in MB)	(Error in path length)	(Edge difference)	(Top 20 Centrality)		(Time Taken in sec)	(Memory Consumed ir	(Error in path lengt	(Edge difference)	(Top 20 Centrality)
Using Onl	0.009979248	0.19140625	0.222057735	16	19		0.011970043	0.18359375	7.179866765	16	19
Using Onl	0.010970354	0.08203125	4.4411547	14	18		0.008949518	0.0625	6.95780903	14	18
Using Onl	0.01097393	0.109375	5.32938564	9	18		0.012038231	0.03515625	5.403404885	9	18
Using Onl	0.007980108	0.0625	1.702442635	16	20		0.007978678	0.15234375	5.389526277	16	20
Using Onl	0.011738062	1.484375	4.88527017	16	18		0.010969639	1.546875	5.255366395	16	18
Using K &	0.013962507	0.03515625	13.10140637	35	17		0.012965202	0	9.17838638	35	17
Using K &	0.016954422	0.08984375	9.03034789	47	18		0.013000011	0.4453125	8.14211695	47	18
Using K &	0.015957832	0.0625	0.666173205	24	17		0.010941982	0	2.886750555	24	17
Using K &	0.012964487	0.0625	1.18430792	33	20		0.008980036	0.0625	4.14507772	33	20
Using K &	0.008975983	0.04296875	10.43671355	32	17		0.0090065	0.0390625	8.364174685	32	17
Using All	0.014959812	0.15234375	33.90081421	211	12		0.016982317	0.15234375	33.01258327	211	12
Using All	0.014963865	0.109375	32.86454478	194	12		0.014921188	0.1015625	33.53071799	194	12
Using All	0.014972687	0.140625	31.60621762	187	12		0.018950224	0.1484375	30.49592894	187	12
Using All	0.011967659	0.12890625	33.23464101	211	12		0.011968136	0.11328125	33.23464101	211	12
Using All	0.012969971	0.1953125	33.75277572	211	12		0.01199317	0.13671875	33.01258327	211	12

Fig 8. Comparison between single sensitive attribute and multiple sensitive attribute.

<u>Hence, we can say yes, it is possible for the existing methods to extend for multiple sensitive attributes.</u>

V. Contributions:

Dhruv

- Analysis and testing of the methods.
- Writing test methods on the basis of which we test our code.
- Pipelining and modularization of code (Including Bug Finding).

Srujan

- Implemented k-degree anonymity, l-diversity and recursive(c,l) diversity privacy measures.
- Wrote code for the edge editing method and the noise node addition method.
- Implemented the various centrality measures and created the target degree sequence generation module.

Ishan

- Writing code for the implementation of privacy models on synthetic datasets.
- Writing code for visualization of graphs.
- Debugging of code.

VI. What we have learned

Dhruv

Till now I had only worked on research projects. Working on this project is my first experience in software engineering field and looking through the perspective of a software engineer, what is considered important there and I learned a lot. I particularly liked writing the test cases for the project which helped us evaluate all the techniques by providing an in-depth analysis consisting of time-taken, memory consumed, error in path length as well as the edge difference between the original and the anonymized graph. While trying to find different ways I can test my code, I came across quite interesting testing methods literature like unit testing, integration. I also learned about good software engineering practices to make our code modularized and also establish proper pipeline. Apart from that, I had no prior knowledge of anonymization so I learned a lot about that. Also, I was misinformed, believing that network security equates to cryptography and nothing but now have learned even through relations people can make decode data.

Ishan

This project was a huge learning experience for me. Having previously done research on social networks (focused mainly on data mining from social networks data), this project gave me the opportunity to explore a previously unfamiliar topic in this field: anonymization of social networks. Prior to this project, I had little knowledge about the complexities involved in maintaining this privacy through anonymization of graphs. I had a false idea that just the hiding of nodes was adequate to protect the privacy of a graph. However, I was surprised to learn about the vast number of ways to exploit the privacy of a graph by leveraging on the properties of a graph structure. I thoroughly enjoyed learning about the 2 kinds of privacy models- Edge-editing, and Noise node addition. I found all 2 of them quite intuitive to understand.

Srujan

I was able to empirically estimate the effects of privacy measures on the structural integrity of social networks. This analysis gave me a great insight into the trade-off between privacy and the usefulness of anonymised information. This project also gave me the opportunity to explore various properties of social networks such as various centrality measures (and their unique interpretations), average path length, clustering co-efficient etc. I can now better appreciate the nuances associated with anonymising sensitive information. Furthermore, these properties that I gained knowledge about also hold good in tabular data (with certain changes). Overall, this project introduced me to the world of social network analysis and the challenges therein, and certainly enhanced my interest in this field.

VII. Future Scope

We used the python networkx open-source library extensively. Networkx provides a multitude of features for the manipulation and study of graphs. However, as of now, it doesn't provide any method to related to the privacy and anonymization graphs.
We have started communication and we hope to commit our code to the networkx open-source library in Github to provide method for the privacy measures and noise node and edge editing method.
Moreover, new privacy measures can be incorporated. We can also take a look at the state-of-the-art privacy measures in the field of databases and try to extend them to social network analysis.