Fall 2019

CMPE 256

Large-Scale Analytics

Project Report

Community detection in Twitter ego-networks

https://github.com/arkil/Twitter_Graph_Mining

Team Spartans

Presented to :
Dr. Magdalini Eirinaki

Group Members :

Arkil Thakkar (ID: 013825292)
Shravani Pande (ID: 01384924)
Sneha (ID: 013850174)

Section 1. Introduction

1.1 Motivation

Groups of vertices which share common interests or characteristics within a graph together form communities. A variety of communities are formed in organizations like family, friends, co-workers, politics, biology, etc. It becomes essential to identify communities of people or elements with similar interests to improve the services and make them efficient to set up recommendation systems, analyze social and political trends and other services that need to have knowledge about groups of elements with similar characteristics.

1.2 Objective

Twitter ego-network dataset consists of node features (profiles) and circles. We aim to detect communities from this directed graph data and to evaluate the results with reference to the ground truth data using various community detection algorithms.

Section 2. System Design & Implementation Details

2.1 Algorithm(s) considered/selected (and why)

1. **Louvain:**
   Louvain is one of the fastest among modularity based algorithms. This algorithm works best for large graphs and hence was our first choice.
   This algorithm works in two phases, each executed iteratively. Initially, every node in the graph is considered to be a single community. Modularity gain between the current node is calculated with all neighbor nodes. Modularity is defined as the density of links between nodes within the community as compared to the density of the nodes from different communities.
   For each node X, the algorithm takes Y neighbors into consideration. We then calculate the gain in modularity occurring when X is removed from its current community and placed in the community of Y. If the modularity gain is positive, the node X is placed in the community of Y. This occurs in iteration and node X is finally placed in the community with maximum gain. In the case of negative gain, the community of the node remains unchanged.

2. **k-Cliques:**
   Cliques are subsets of vertices in a graph that are all adjacent to each other; also called complete subgraphs. In a real-world scenario, a clique represents a subset of people who all know each other. The algorithm is used to find such groups of mutual friends. Finding such large near-cliques is an NP-Hard problem. However, relaxed versions of the algorithm aid in understanding the sub-communities formed.
   The k-clique algorithm takes as an input the graph G that is formed by edges in an ego network and the number of desired cliques within every ego network k.
   The input graph is considered as an undirected graph. It was observed that the result

obtained with this algorithm was the most accurate as compared to all the other algorithms tried by us, considering the “edit calculations (loss)”.

3. **Spectral Clustering:**
   Spectral clustering is used to identify communities of nodes in a graph based on the edges connecting between them. Spectral clustering uses information from the eigenvalues of the matrices built from the dataset/graph.
   Our implementation of the algorithm takes the graph G of the network as input and computes Cosine similarity between the nodes for clustering.

4. **Girvan-Newman:**
   Girvan-Newman algorithm prunes the original network based on the centrality of the edges. Instead of creating communities from scratch, this algorithm finds out the edges that are most central to communities. Thus the algorithm considers the “edge betweenness” for community detection. The algorithm steps are as follows:
   a) Calculate the betweenness centrality of all the edges in the network.
   b) Remove the edges with the highest betweenness.
   c) Recompute the betweenness for all the edges affected by the removal.
   d) Repeat steps b and c until no such edges remain.

   The algorithm takes graph G - the original network as the input and returns the list of communities.

5. **InfoMap:**
   Infomap is a network clustering algorithm based on the Map equation. Map equation analyzes how network structure influences system behavior by recognizing that links in a network induce movement across the network resulting in system-wide interdependence. The map equation highlights and simplifies the network structure with respect to this flow.

2.2 Technologies & Tools Used

Tools/ Programming Language/Platform	**Libraries**
1) Conda 2) Jupyter Notebook 3) Python 4) HPC	1) SciPy 2) NumPy 3) NetworkX 4) Scikit-learn 5) Matplotlib 6) Community 7) Munkres 8) Igraph 9) infomap

Section 3. Experiments

3.1 Dataset

The Twitter ego-network dataset is present at https://snap.stanford.edu/data/ego-Twitter.html.

For each network, the dataset has 5 files – edges, circles, feat, egofeat and featnames. The edges in this ego network are directed.

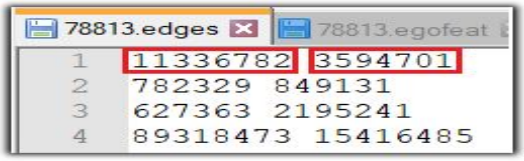
Dataset Statistics:

Total Ego-networks	973
Nodes	81306
Edges	1768149

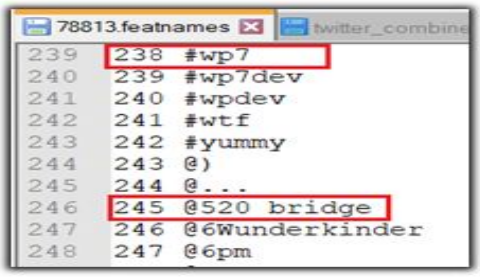
Files in the dataset:

There are 973 ego-networks in the dataset. Each ego-network has the following 5 files:

1. **nodeId.edges:**
   This file contains all the edges that join the nodes (alters) in the ego-network of 'nodeId'. The ego node does not appear in this file, but it is assumed that the ego node follows every node that appears in the *.edges file.
   In the figure below, '78813.edges' contains the edges in the ego network of nodeId 78813. The highlighted nodes joined together form an edge.

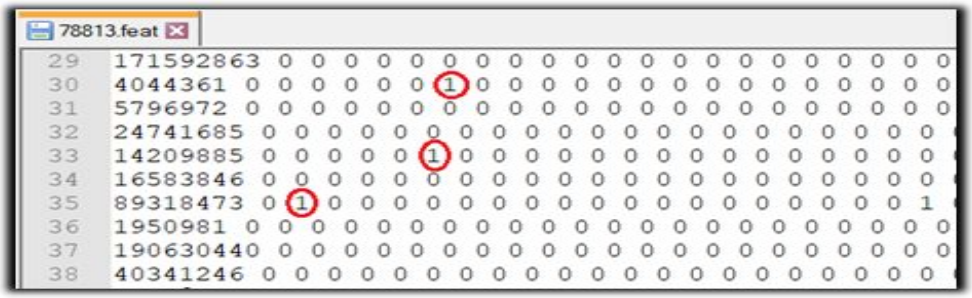


2. **nodeId.featnames:**
   This file contains the list of all the features that are being followed by the members of a particular ego-network. An ego node and its alters can follow any number of '#' hashtags or other profiles.

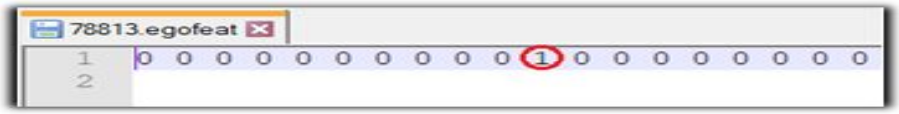


:

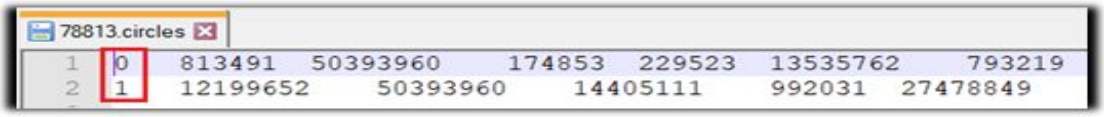
3. **nodeId.feat:**
   This file lists the profiles of all the nodes in an ego-network by indicating which features (from the featnames file) are being followed. These values are indicating in boolean. In the file below, all the 1s indicate the features followed by the respective node. For instance, node 4044361 follows the 7th feature from its ego-network's featurename file. All the 0s indicate the features not being followed by the node.



4. **nodeId.egofeat:**
   This file lists all the features followed by the ego-node.



5. **nodeId.circles:**
   This file lists all the communities within all the ego-networks. An ego-network may have multiple communities/ circles. Each circle is made up of nodes (within that ego-network) based on factors like closeness, similarities based on features, etc. This file is being used by us as ground truth to evaluate the correctness of our community detection algorithms. The highlighted values indicate the names of the circles within the ego-network.



3.2 Data Preprocessing

Initial analysis of the edges files yielded the following results.

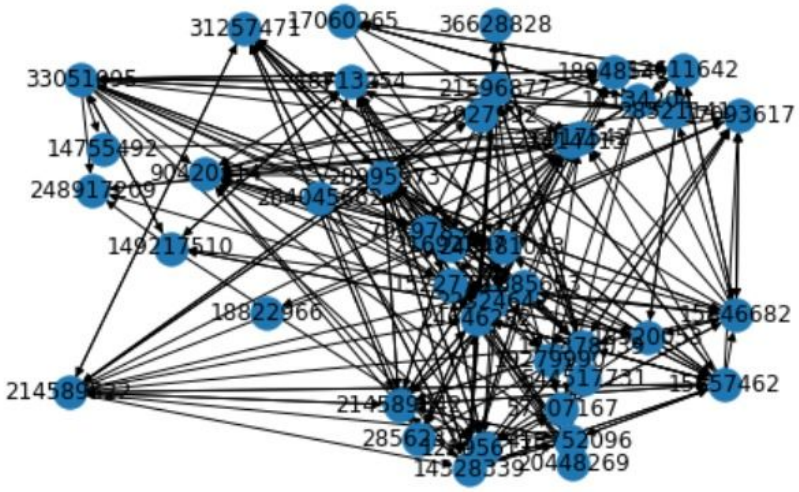
Min no. of edges: 5
Max number of edges: 17938

Mean of the edges: 2300.

Min no. of circles: 0
Max no of circles: 100

Mean of circles: 5

- Using edges from edges file constructed graph for each ego node.
- Directed graph visualization of ego node.



3.3 Methodology

Spectral Clustering:

- Filter Egonets with more number of edges as it was computationally expensive.
- Calculated Similarity matrix of features between users present in edge file
- Applied Spectral clustering with affinity parameter as calculated matrix
- Calculated loss, normalized F1, modularity with ground truth files

Louvain:

- Assign a different community to each node of the network. Evaluated gain modularity of node with neighbors and placed in the community with maximum gain.
- Calculated loss and Normalized F1 with ground truth circles.

K-Clique:

- Filter Egonets with more number of edges as it was getting stuck.
- Applied K cliques algorithm with different values of k.
- Calculated loss with ground truth files.

Girvan Newman:

- Applied algorithm and generated different tuples in communities using communities iterator.
- Calculated loss by comparing predicted circles with ground truth files.

3.4 Results

1. *Louvain:*

   Total nodes: 973
   Total loss: 147356
   Average Loss: 151.44

2. *Spectral Clustering:*

   Total nodes: 583
   Total Loss: 63830
   Average Loss: 109

3. *Girvan-Newman:*

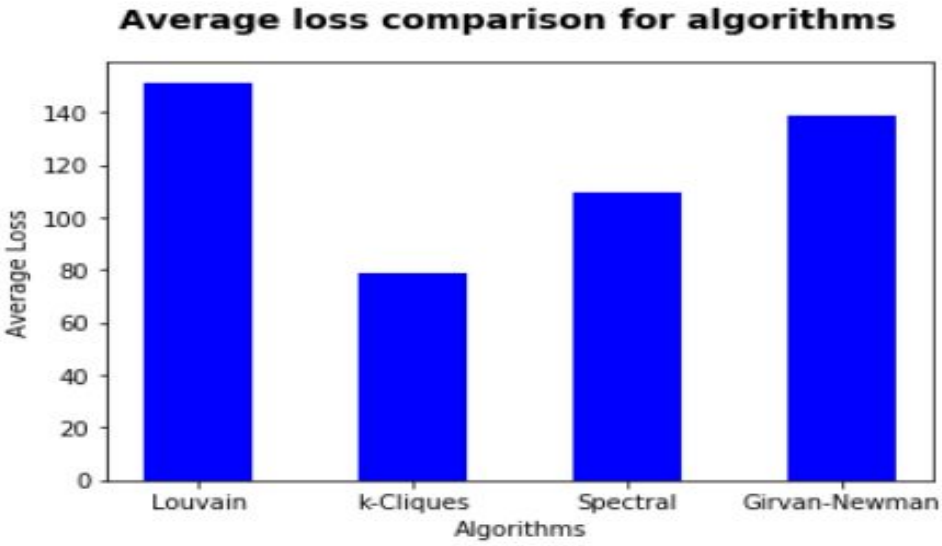
   Total Nodes: 973
   Total Loss: 135372
   Average Loss: 139

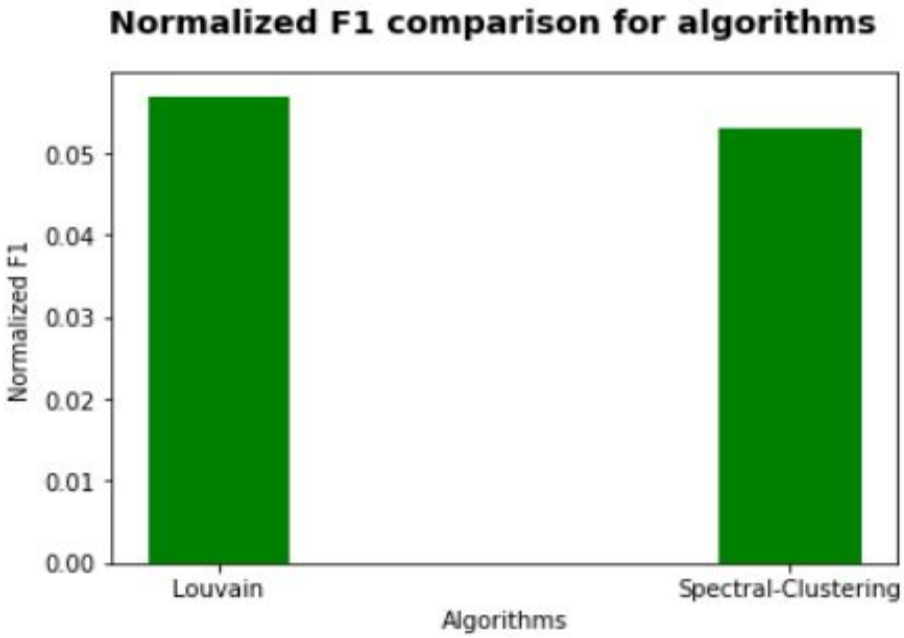
4. *K-Clique:*

   Total nodes: 583
   Total Loss: 46127
   Average Loss: 79

3.5 Analysis of Results

- K Clique failed to work when ego files had many edges.
- Spectral Clustering took long time to compute as it was calculating the similarity between nodes.
- Louvain algorithm was fast to compute but did not provide good results. There is a general problem with modularity optimization algorithms, that they have trouble detecting small communities in large networks. Hence the resultant communities differed from the ground truth.
- Girvan Newman requires k tuples to return after generating Iterator





Section 4. Discussion & Conclusion

4.1 Decisions Made

- Filtered out ego networks that contained empty circles in the ground truth files.
- Selecting algorithms on which evaluation parameter can be applied easily.
- Most algorithms/libraries work well for undirected graphs. We considered the data to be undirected to make use of the available libraries.

4.2 Difficulties Faced

- Dataset was difficult to understand.
- Ground Truth 'Circles' files contain many empty entries which caused issues while calculating the evaluation metrics.
- I-graph can not be installed in windows
- Most libraries can calculate modularity only on undirected graphs.
- Choosing an evaluation parameter that works on all algorithms was challenging.

4.3 Things That Worked

- Used loss function(provided in Learning Social Circles in Networks Kaggle Competition) as the evaluation metric for the algorithms.
- Filtering out the graphs(egonets) that have edges less than 2300(the mean of all the edges) worked for the k-Clique algorithm.

4.4 Things That Didn't Work

- Could not calculate normalized F1 due to empty ground truth files and empty predictions.
- Hierarchical clustering was computationally expensive. Also, it required predefined cluster value which did not suit our problem statement.
- Tried plotting the entire network (all edges, and all ego nodes) using Gephi but the visualization was not clear and understandable. The plot was difficult to decipher and we had to discard it.

4.5 Conclusion

1) Dataset exploration and preprocessing play a vital role
2) Hyper tuning the parameters contributes to improving the results.
3) Data cleaning is also a very important step before beginning the data mining process.
4) Identifying subgraphs from the network is a challenging task.

Section 5. Project Plan & Task Distribution

Task	Assignee
Project Topic	All
Data Analysis	All
Data Preprocessing	All
Algorithm - Louvain	Shravani
Algorithm - Spectral Clustering	Arkil
Algorithm - k-Cliques	Sneha
Algorithm - Girvan-Newman	Sneha
Project Report	All
Project Presentation	All

Section 6. References

Learning Social Circles in Networks-Kaggle Competition

Learning-Social-Circles-in-Networks - Kaggle Solution

Social Network Analysis with NetworkX

Kaggle Social Networks Competition

Learning to Discover Social Circles in Networks

Generating A Twitter Ego-Network & Detecting Communities

Thinking About Ego

Community Detection in Social Networks

Social-Circles-in-Ego-Network

Ego-centric Networks

Link to source code:

https://github.com/arkil/Twitter_Graph_Mining