Graph Clustering Experiments Report

This is a report/summary of a group of graph clustering algorithms (written in Python), profiled over on the Facebook Large Page-Page Network. The complete source code, as well as visualization, is available here.

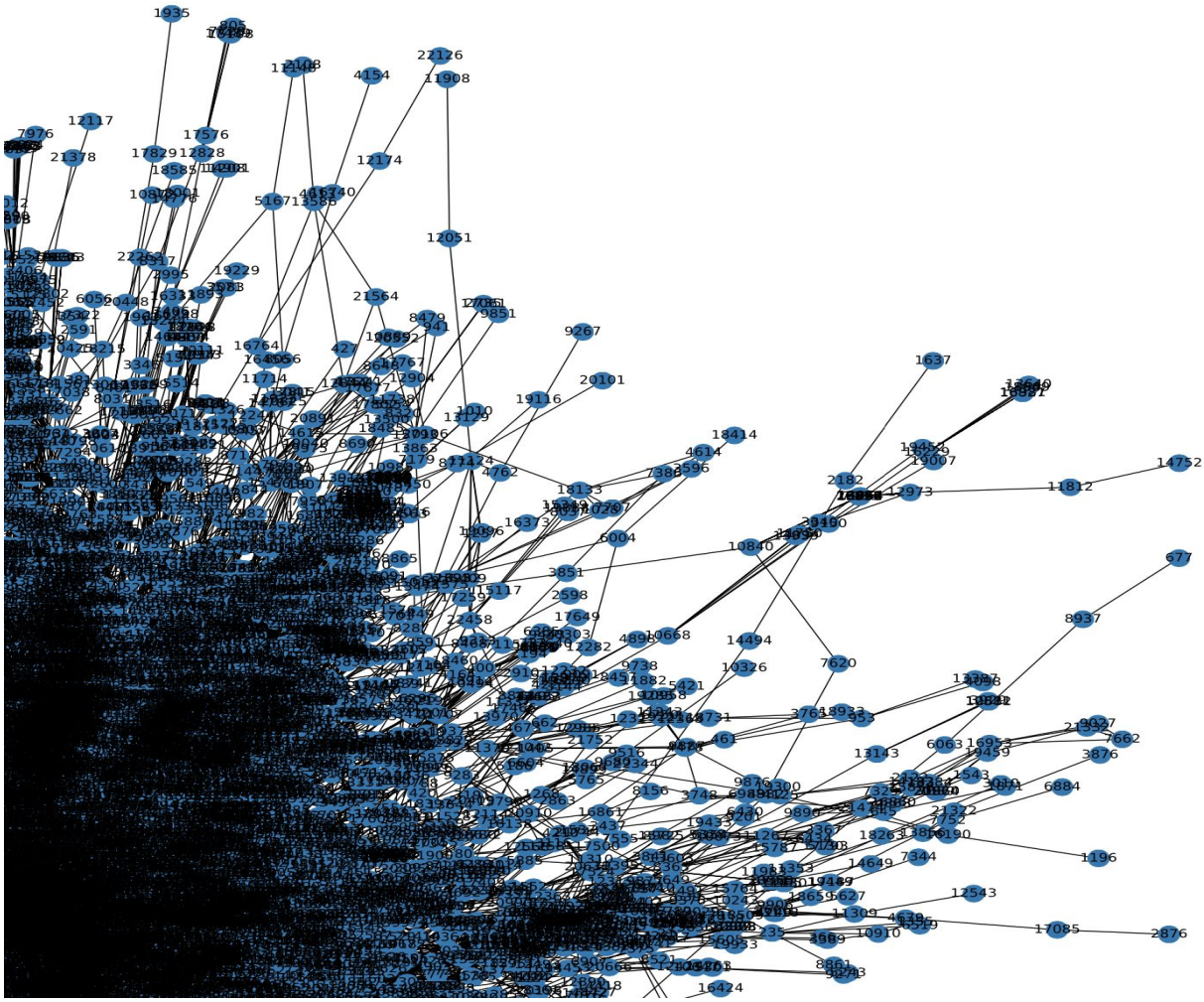
Problem Statement

Check statement.pdf

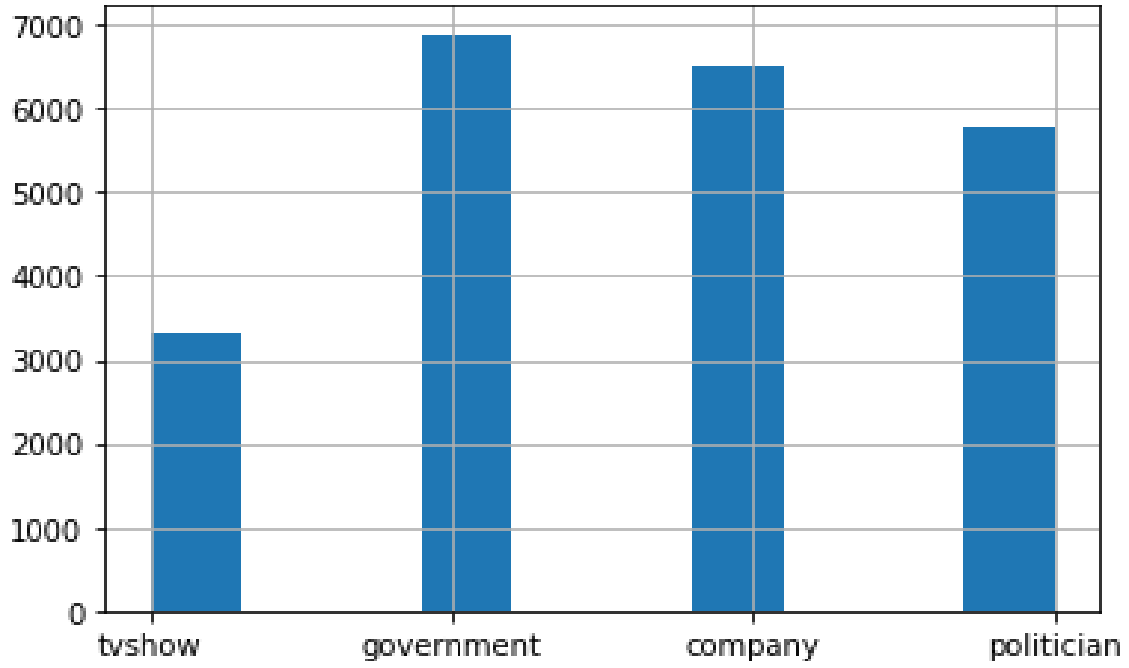
Dataset Description

Check SNAP Facebook Large Page-Page Dataset.

This is a visualization of the sparse dataset:



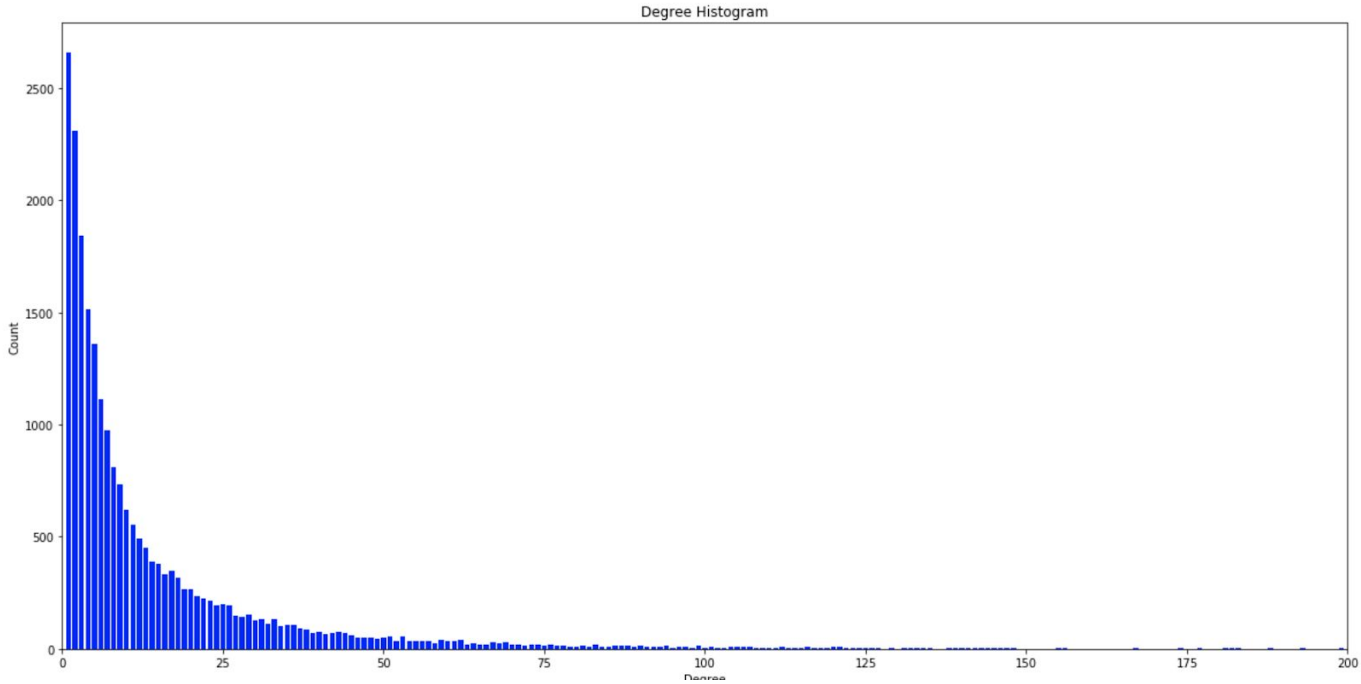
And this is a histogram of the class distribution:



Also, the nodes carry information about the page, which is stored tabularly like this:

	id	facebook_id	page_name	page_type
0	0	145647315578475	The Voice of China 中国好声音	tvshow
1	1	191483281412	U.S. Consulate General Mumbai	government
2	2	144761358898518	ESET	company
3	3	568700043198473	Consulate General of Switzerland in Montreal	government
4	4	1408935539376139	Mark Bailey MP - Labor for Miller	politician
5	5	134464673284112	Victor Dominello MP	politician
6	6	282657255260177	Jean-Claude Poissant	politician
7	7	239338246176789	Deputado Ademir Camilo	politician
8	8	544818128942324	T.C. Mezar-ı Şerif Başkonsolosluğu	government
9	9	285155655705	Army ROTC Fighting Saints Battalion	government

This is what the degree histogram of this graph looks like:



Summary of Experiments and Results

For network data visualization and basic graph operations, we can use a number of libraries. I have used networkX. All visualizations with analysis are available at visual.ipynb.

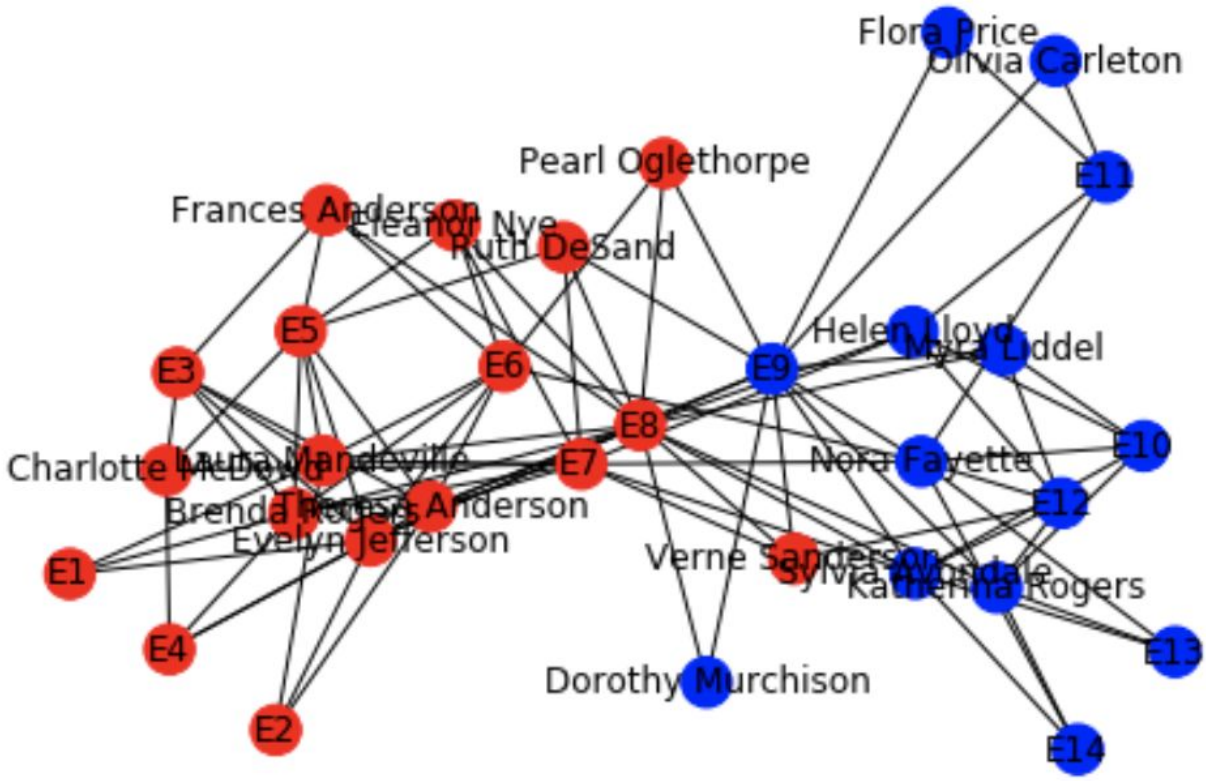
The problem of graph clustering for social network graphs like page-page naturally molds into one of community detection. There are a number of community detection algorithms. I tested the Girvan Newman's Algorithm. More algorithmic details are there in the research paper (1).

The algorithm's steps for community detection are summarized below:

- The betweenness of all existing edges in the network is calculated first.
- The edge(s) with the highest betweenness are removed.
- The betweenness of all edges affected by the removal is recalculated.
- The above two steps are repeated until no edges remain.

This algorithm worked really well on a sample graph, but on the large page-page dataset, I could not get it to run with my processing power. The code is available in girvan-newman.ipynb

This is how this algorithm clustered a sample graph into 2 clusters:



Next, I tried the most common clustering algorithm: KMeans Clustering. K-Means Clustering: Start with all vertices in different clusters and merge pairs of clusters that minimize a given linkage distance (Euclidean by default). We'll test the sklearn implementation of this seminal research paper (2) on the page-page dataset.

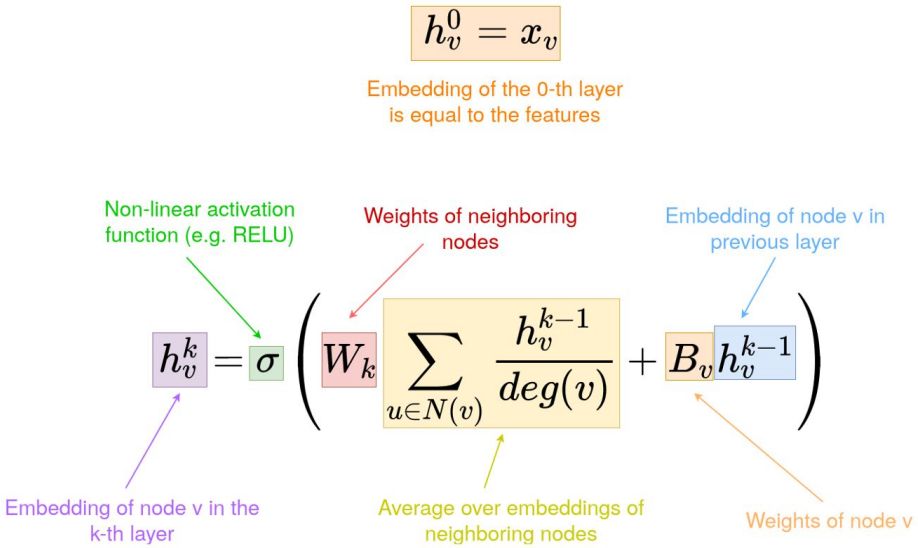
Note: We will pass the adjacency matrix rows as features to KMeans, with the untested hypothesis that samples from the same cluster would be closer to each other on this feature space. The code is available at kmeans.ipynb.

The results were unsuccessful (trivially), and thus the hypothesis was incorrect.

After visualizing the graph data and trying two classical approaches to clustering, we'll move on to try some ML-based approaches.

I've never done graph ML before. This article was a really comfortable starting point into the field. One difference we can see in the page-page data vs usual graph NN data is that here, the nodes don't have features. Just a page_type from 1-4 that acts as the target in the usual node classification problem.

To model graphs as neural nets, we need embeddings for each layer of the graph. This image helps understand how weights are used to define these embeddings. More details in this blog.



GCNs (Graph Convolutional Networks) almost always prove to perform better than vanilla Graph Networks. I'm not reading about the theoretical details of how they work (except the main embedding equation) for now, and will be trying out PyTorch Geometric's implementation to see how they perform later.

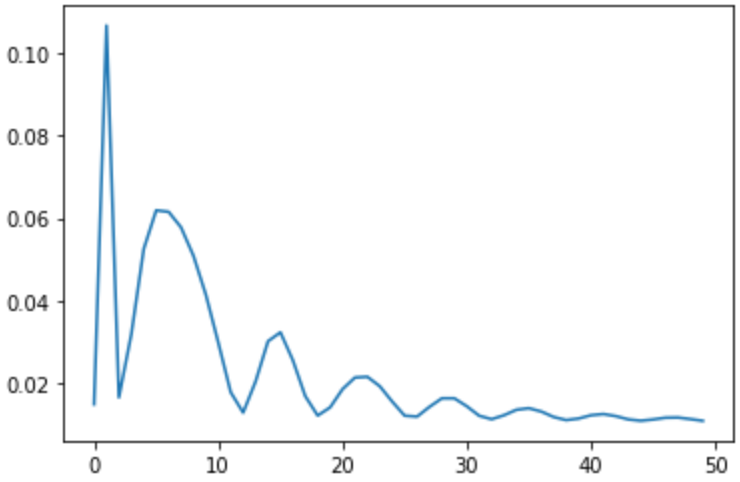
Note to self: Read up more about GCNs here, and the papers linked with this.

- (3) is the seminal paper on Graph Convolutional Nets.

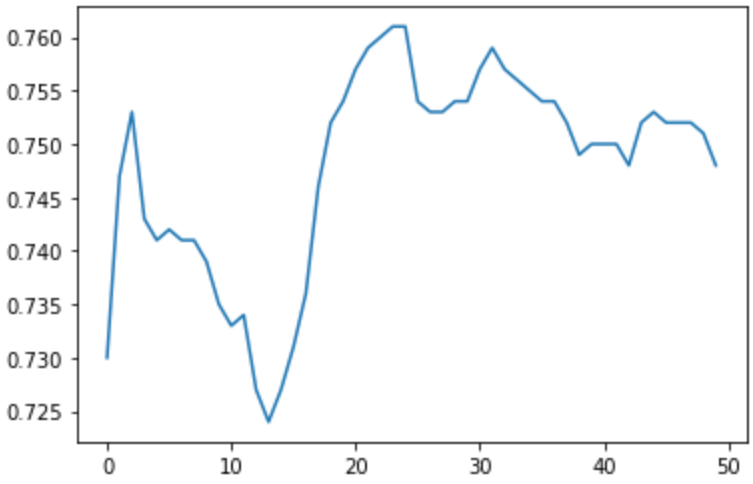
More theory and code for graph convnets is available at graphnets.ipynb and graphconv.ipynb.

We try to use neural networks on graphs using the DGL library. It will follow the implementation of a GCN using PyTorch backend. First I'm trying it out on an inbuilt dataset from the DGL Library and later will apply it to the page-page dataset.

This is the loss curve over 50 epochs of training:



And this is the accuracy over 50 epochs:



The algorithm works well on the inbuilt dataset, and it will require some tweaking to feed the page-page dataset to it.

Dataset Citation:

B. Rozemberczki, C. Allen and R. Sarkar. Multi-scale Attributed Node Embedding. 2019.

```
      @misc{rozemberczki2019multiscale,
           title={Multi-scale Attributed Node Embedding},
           author={Benedek Rozemberczki and Carl Allen and Rik Sarkar},
           year={2019},
           eprint={1909.13021},
           archivePrefix={arXiv},
           primaryClass={cs.LG}
        }
```

Reference to Research Papers:

1. Girvan-Newman Community Detection Algorithm
2. K-Means Clustering
3. Graph Convolutional Networks