Community Detection, Link Prediction and Node Classification on Ego-Facebook and Citeseer Datasets

Chen Liguo 17307110182
Shao Yanjun 19307110036
Shao Yi 19307130113

July 4, 2021

1 Introduction

1.1 Problem and Background

The final project of Social Network Mining consists of four different objectives, covering topics from basic analysis on graph structure to advanced ones such as machine learning and graph neural network (GNN). To be more specific, first of all, our team is going to make use of the topological structure of the graph to display the centrality measures, triangle count and the degree distribution in Ego-Facebook dataset. Secondly, an algorithm on **community detection** will separate the whole graph into several parties where nodes and edges in a certain party displays a high tendency of similarity. And for the most challenging part of the project, that is, to **classify nodes** and **predict links** on a graph with both topological measures and individual features of all nodes and relationship, we will implement several state-of-the-art models and discuss the pros and cons for each of them by means of comparison. Understanding how and why complicated algorithms work comes as top priority, while implementing with code and visualizing the graphs in an elegant way deserves same attention.

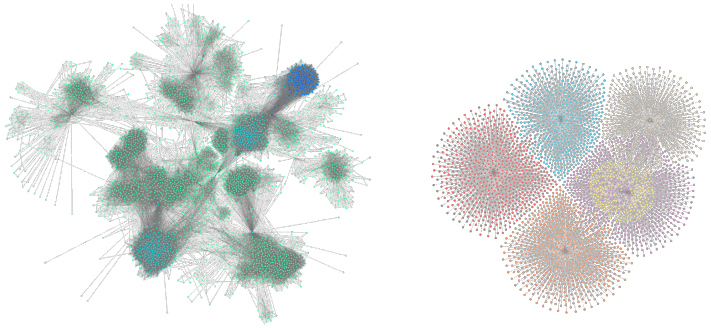
The whole project is launched with the help of Pytorch[1] and Neo4j[2]. Some models may require a CUDA-only version while the rest exhibits extra compatibility to run on CPU.

Neo4j is a graph database management system based on Java and accessible from software written in other languages using the Cypher query language. In Neo4j, everything is stored in the form of an edge, node, or attribute. Each node and edge can have any number of attributes. The declarative graph query language offers possibility for users to conduct expressive and efficient data querying in a property graph with high concurrency.

[1] https://pytorch.org/
[2] https://neo4j.com/

1.2 Datasets Overview

The first two parts of our final project mainly used the ego-Facebook dataset[3] and the rest used the Citeseer dataset[4], which has been regarded as the benchmark dataset for several topics related to graph algorithms.



(a) Ego-Facebook (b) Citeseer

Figure 1: Dataset Overview (powered by Neo4j)

The former dataset consists of 'circles' (or 'friends lists') from Facebook. It was collected from survey participants using this Facebook app. And the latter dataset consists of 3312 scientific publications classified into one of six classes. It includes publications described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary, which consists of 3703 unique words. The citation network consists of 4732 links. Here are the basic statistics and visualizations over the two datasets.

Ego-Facebook		Citeseer	
Nodes	4039	Nodes	3312
Edges	88234	Edges	4732
Number of triangles	1612010	Number of triangles	0
Fraction of closed triangles	0.2647	Fraction of closed triangles	0
Diameter (longest shortest path)	8	Diameter (longest shortest path)	6
Average cost	3.6925	Average cost	3.6541

Figure 2: Network Data Statistics

2 Degree, Centrality and Graph Analytics

In this section, we will generally use ego-Facebook dataset for analytics.

2.1 Zipf's Law

Zipf's law is an empirical law that for many types of data studied in the physical and social sciences, the rank-frequency distribution exhibits an inverse relation, which belongs to a family of

[3]http://snap.stanford.edu/data/ego-Facebook.html
[4]http://networkrepository.com/citeseer.php

related discrete power law probability distributions. In most network data, the degree of each node also follows a certain power law, that is, $f(x) = ax^{-k}$. This is also true if we apply this natural rule to ego-Facebook network, because the following log-log has shown the property.

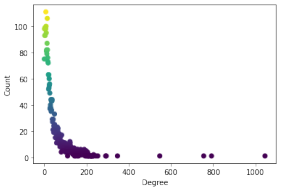


Figure 3: Zipf's Law

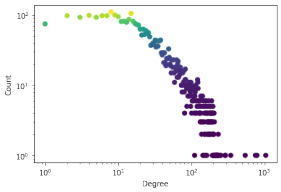


Figure 4: Zipf's Law (Under log-log)

2.2 Centrality Measures

In graph theory and network analysis, indicators of centrality assign numbers or rankings to nodes within a graph corresponding to their network position. There are several different types of measures on node centrality described in the textbook[5] listed as follows,

Centrality	
Eigenvector	$c_v = \frac{1}{\lambda}\sum_{u \in N} c_u$
PageRank	$C_p(v) = \alpha \sum_{j=1} A_{j,i} \frac{C_p(v_j)}{d_j^{out}} + \beta$
Betweenness	$c_v = \sum_{s \neq v \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$
Closeness	$c_v = \frac{n-1}{\sum_{i \neq j} l_{i,j}}$

Figure 5: Table

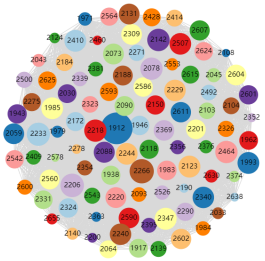


Figure 6: Eigenvector Centrality

We will only list top 5 nodes with respect to each centrality properties in order to get a clear overview of the difference between measures.

ID	Eigenvector	ID	PageRank	ID	Betweenness	ID	Closeness
1912	296.5844	3437	29.680	107	3916560	107	0.4597
2266	254.0166	107	27.084	1684	2753587	58	0.3974
2206	250.4103	1684	24.785	3437	1924506	428	0.3948
2233	248.8579	0	24.498	1912	1868918	563	0.3939
2142	245.7742	1912	15.061	1085	1214578	1684	0.3936

Table 1: Difference

As we can see from Table 1, Eigenvector Centrality cannot capture the essense of the topological structure of the network because all the nodes with high Eigenvector rankings gather in small clique. With iteration going on, all the centrality weights stream into several closely-related neighbour nodes, making them outrageously too powerful.

[5]Zafarani, R. (2014). Social Media Mining: An Introduction (1st ed.). Cambridge University Press.