COMMUNITY FINDING

CASE STUDY 2 : ALGORITHIMS (LOUVAIN, K-MEANS, HIERARCHICAL CLUSTERING & GIRVAN-NEWMAN)

Author : *Paula Dwan*
Email : *paula.dwan@gmail.com*

Student ID : *13208660*

Course : *MSc Advanced Software Engineering*
Module : *COMP-47270 Computational Network Analysis and Modelling*

Lecturer : *Dr. Neil Hurley*
Email : *neil.hurley@ucd.ie*

Due Date : *20 April 2015*

TABLE OF CONTENTS

1 Case Study Requirements

1.1 Prerequisites

- Write some functions to evaluate the quality of a community:

  1. An edge-cut function counts the total number of edges that are cut by a partitioning.
  2. The normalised mutual information function compares two community partitioning and determines how alike they are.
  3. Newman's modularity function computes how cohesive the communities in a community partitioning are.

- Using some of the SNAP graphs that you downloaded in Lab 1, apply the k-means clustering methods in ***laplacian.py*** to partition the graph into $k \geq 2$ communities, where the k-means vectors are generated from eigenvectors and from rows of the adjacency matrix.
- Compute how similar the partitionings are by using NMI.
- Compute how good the partitionings are by using modularity and edge-cut.

1.2 Case Study Requirements

Evaluate some community-finding methods on SNAP data and on simulated networks with embedded communities:

1. Implement your own community-finding algorithm.
2. Compare with at least two other algorithms.
3. Compute their relative performance in terms of quality (NMI, modularity) and run-time.

1.3 Understanding of Requirements from Class Discussions

1. Write a new community finding algorithm or use laplacian as covered in class.
2. Compare to at least two others.
3. Evaluate one against the other two by performance / scalability:

   NMI score (0 totally different → 1 = very similar)
   Newman's Modularity
   Time taken to calculate NMI
   Size of dataset used

2 Introduction

2.1 Community Finding

To start, what is a community?

> A community of a graph G is a sub-graph C of G, such that the number of edges joining two vertices in C is greater than the number of edges joining vertices in C to vertices in G\C. [3]

So basically, a community is a sub-set of a graph where groups of nodes are very similar to each other, all of which are close neighbours of each other.

2.2 Datasets Used

Each of the directed graphs chosen deal with Social media and vary in the number of nodes present. For more information on each, please see the section : References – Datasets

- Small → ≤ 10k Nodes http://snap.stanford.edu/data/wiki-Elec.html

- Medium → 10k to 100k Nodes http://snap.stanford.edu/data/egonets-Twitter.html
- Large → > 100k Nodes http://snap.stanford.edu/data/egonets-Gplus.html

3 Evaluations Used

Once each community is created using the community-finding algorithms, it is evaluated using :

- NMI score (0 = totally different → 1 = very similar)
- Newman's Modularity
- Time taken to calculate NMI
- Impact of dataset size

3.1 Normalised Mutual Information (NMI) Function

3.1.1 Overview

3.1.2 How Calculated in Python?

3.2 Newman's Modularity Function

3.2.1 Overview

3.2.2 How Calculated in Python?

3.3 Calculate Run-time

3.3.1 Overview

Basically, how long does it take algorithm to produce the clustered graphs for the communities?

3.3.2 How Calculated in Python?

I used an existing python module called ***timeit*** [2]. This calculates the start and end time and subtraction produces the difference. Time taken may then be compared across all algorithms and all datasets.

```
import timeit                    # import inbuilt python method

def getStartTime():              # function to return the start time
   return timeit.default_timer()

def showTimeTaken(start):        # function to obtain the end time and calculate total time taken
   logging.info(cs_ref, 'calculate Time Taken to run partitioning')
   time_end = timeit.default_timer()
   print 'Time taken to run Community Partitioning = {0} - {1} = {2}'. \
      format(time_end, start, time_end-start)
```

3.4 Impact of Dataset Size

3.4.1 Overview

Does a larger dataset produce better communities? Does a larger dataset significantly increase the time taken? Is the a ratio between the dataset size and the time taken? In short, does the size of the dataset rather than the algorithm used impact the time taken?

3.4.2 How Calculated in Python?

It is not but is based on the results obtained for each algorithm, NMI, modularity and time taken.

4 K-Means Algorithm

4.1 Overview

4.1.1 What is the K-Means Algorithm?

4.1.2 Components

Component	Explanation

4.1.3 How Calculated Using Python?

Component	Explanation

4.2 Results obtained

4.2.1 Results Obtained : Small Dataset→ ≤ 10k Nodes

Performance	Discussion
NMI	
Newman's Modularity	
Time taken	
Dataset comparison	

4.2.2 Results Obtained : Medium Dataset → 10k to 100k Nodes

Performance	Discussion
NMI	
Newman's Modularity	
Time taken	
Dataset comparison	

4.2.3 Results Obtained : Large Dataset→ > 100k Nodes

Performance	Discussion
NMI	
Newman's Modularity	
Time taken	
Dataset comparison	

5 Louvain Method (Community Detection in Large Networks)

5.1 Overview

5.1.1 What is the Louvain Method?

The Louvain Method was developed by Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre, when all were based in Louvain. It computes the best partitions of large scale networks and works as follows :

1. First, it looks for "small" communities by optimizing modularity in a local way.

2. Second, it aggregates nodes of the same community and builds a new network whose nodes are the communities.

3. Repeat until a maximum of modularity is attained. [1]

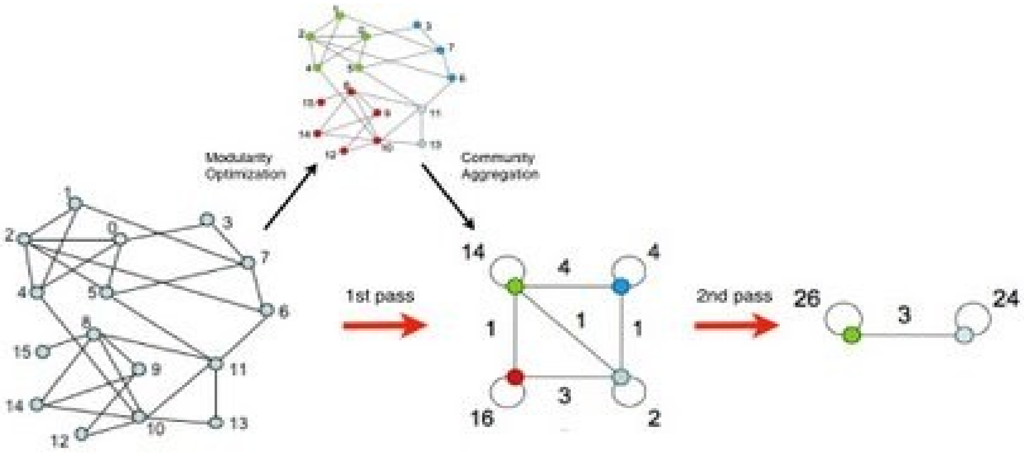


Figure 1 : Louvain Process

Initially, many communities are created, each of which is small in size. After each iteration, the communities become larger and larger until no more partitions can be created.

5.1.2 Components

Component	Explanation
Modularity	
Best partition	

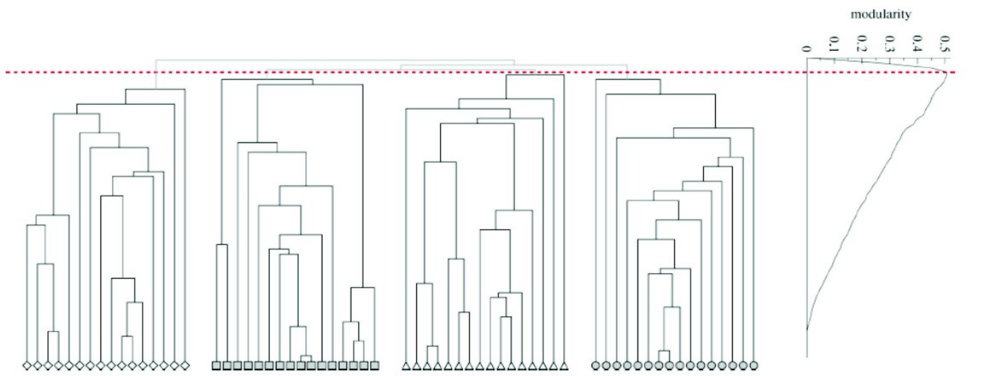
<table>
<tr><th>Component</th><th>Explanation</th></tr>
<tr><td>dendrogram</td><td>Hierarchical tree structure representing the level / partition of each node in the graph. Level 0 is at the bottom (smallest communities) and len(dendrogram)-1 is the highest (biggest communities).


FIG. 6. Plot of the modularity and dendrogram for a 64-vertex random community-structured graph generated as described in the text with, in this case, $z_{\text{in}}=6$ and $z_{\text{out}}=2$. The shapes at the bottom denote the four communities in the graph and, as we can see, the peak in the modularity (dotted line) corresponds to a perfect identification of the communities.
Figure 2 : Sample Dendrogram from lecture slides [3]</td></tr>
</table>

5.1.3 How Calculated Using Python?

<table>
<tr><th>Component</th><th>Usage & Explanation</th></tr>
<tr><td>community.modularity
(partition, graph)</td><td>Compute the modularity of a partition of a graph
<pre><code>G = graph()
partition = community.best_partition(G)
modularity(part, G)
</code></pre></td></tr>
<tr><td>community.best_partition
(graph, partition=None)</td><td>Compute the partition of the graph nodes which maximises the modularity (or try..) using the Louvain heuristics
<pre><code>G = graph()
partition = community.best_partition(G)

size = float(len(set(partition.values())))
pos = nx.spring_layout(G)

count = 0.
for com in set(partition.values()) :
   count = count + 1.
   list_nodes = [nodes for nodes in partition.keys()
      if partition[nodes] == com]
   nx.draw_networkx_nodes(G, pos, list_nodes, node_size = 20,
      node_color = str(count / size))
</code></pre></td></tr>
<tr><td>community.generate_dendrogram
(graph, part_init=None)</td><td>Find communities in the graph and return the associated dendrogram
<pre><code>dendo = generate_dendrogram(G)

for level in range(len(dendo) - 1) :
   print "partition at level", level, "is",\
      partition_at_level(dendo, level)
</code></pre></td></tr>
<tr><td>Plot graph</td><td>Use standard networkx methods.
<pre><code>nx.draw_networkx_edges(G,pos, alpha=0.5)
plt.show()
</code></pre></td></tr>
</table>

5.2 *Results Obtained*

5.2.1 Results Obtained : Small Dataset→ ≤ 10k Nodes

Performance	Discussion
NMI	

Performance	Discussion
Newman's Modularity	
Time taken	
Dataset comparison	

5.2.2 Results Obtained : Medium Dataset → 10k to 100k Nodes

Performance	Discussion
NMI	
Newman's Modularity	
Time taken	
Dataset comparison	

5.2.3 Results Obtained : Large Dataset→ > 100k Nodes

Performance	Discussion
NMI	
Newman's Modularity	
Time taken	
Dataset comparison	

6 Hierarchical Clustering Algorithm

6.1 Overview

6.1.1 What is Hierarchical Clustering Algorithm?

6.1.2 Components

Component	Explanation

6.1.3 How Calculated Using Python?

Component	Usage & Explanation

6.2 RESULTS OBTAINED

6.2.1 RESULTS OBTAINED : SMALL DATASET→ ≤ 10K NODES

Performance	Discussion
NMI	
Newman's Modularity	
Time taken	
Dataset comparison	

6.2.2 RESULTS OBTAINED : MEDIUM DATASET → 10K TO 100K NODES

Performance	Discussion
NMI	
Newman's Modularity	
Time taken	
Dataset comparison	

6.2.3 RESULTS OBTAINED : LARGE DATASET→ > 100K NODES

Performance	Discussion
NMI	
Newman's Modularity	
Time taken	
Dataset comparison	

7 GIRVAN & NEWMAN ALGORITHM

7.1 OVERVIEW

7.1.1 WHAT IS GIRVAN & NEWMAN ALGORITHM?

The process is summarised as follows :

1. First, calculate the betweenness of all existing edges in the network.
2. Remove the edge with the highest betweenness
3. Recalculate the betweenness of any edge impacted by step 2.
4. Repeat until no edges remain in the network.

7.1.2 COMPONENTS

Component	Explanation

7.1.3 How Calculated Using Python?

Component	Usage & Explanation

7.2 Results obtained

7.2.1 Results Obtained : Small Dataset→ ≤ 10k Nodes

Performance	Discussion
NMI	
Newman's Modularity	
Time taken	
Dataset comparison	

7.2.2 Results Obtained : Medium Dataset → 10k to 100k Nodes

Performance	Discussion
NMI	
Newman's Modularity	
Time taken	
Dataset comparison	

7.2.3 Results Obtained : Large Dataset→ > 100k Nodes

Performance	Discussion
NMI	
Newman's Modularity	
Time taken	
Dataset comparison	

8 Conclusions

9 REFERENCES - DATASETS

9.1 NETWORKS DATASETS

Datasets : http://snap.stanford.edu/data/index.html
The information in this section is taken directly from http://snap.stanford.edu/ directly and is included for informational purposes only.

9.1.1 NETWORK DATASET STATISTICS

Dataset statistics	
Nodes	Number of nodes in the network
Edges	Number of edges in the network
Nodes in largest WCC	Number of nodes in the largest weakly connected component
Edges in largest WCC	Number of edges in the largest weakly connected component
Nodes in largest SCC	Number of nodes in the largest strongly connected component
Edges in largest SCC	Number of edges in the largest strongly connected component
Average clustering coefficient	In graph theory, a clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together. Evidence suggests that in most real-world networks, and in particular social networks, nodes tend to create tightly knit groups characterised by a relatively high density of ties; this likelihood tends to be greater than the average probability of a tie randomly established between two nodes (Holland and Leinhardt, 1971; Watts and Strogatz, 1998).
Number of triangles	Number of triples of connected nodes (considering the network as undirected)
Fraction of closed triangles	Number of connected triples of nodes / number of (undirected) length 2 paths
Diameter (longest shortest path)	Maximum undirected shortest path length (sampled over 1,000 random nodes)
90-percentile effective diameter	90^{th} percentile of undirected shortest path length distribution (sampled over 1,000 random nodes)

9.2 SOCIAL CIRCLES : WIKIPEDIA VOTE NETWORK

9.2.1 DATASET INFORMATION

http://snap.stanford.edu/data/wiki-Vote.html
Wikipedia is a free encyclopedia written collaboratively by volunteers around the world. A small part of Wikipedia contributors are administrators, who are users with access to additional technical features that aid in maintenance. In order for a user to become an administrator a Request for adminship (RfA) is issued and the Wikipedia community via a public discussion or a vote decides who to promote to adminship. Using the latest complete dump of Wikipedia page edit history (from January 3 2008) we extracted all administrator elections and vote history data. This gave us 2,794 elections with 103,663 total votes and 7,066 users participating in the elections (either casting a vote or being voted on). Out of these 1,235 elections resulted in a successful promotion, while 1,559 elections did not result in the promotion. About half of the votes in the dataset are by existing admins, while the other half comes from ordinary Wikipedia users.

The network contains all the Wikipedia voting data from the inception of Wikipedia till January 2008. Nodes in the network represent wikipedia users and a directed edge from node i to node j represents that user i voted on user j.

Dataset statistics	
Nodes	7115
Edges	103689
Nodes in largest WCC	7066 (0.993)
Edges in largest WCC	103663 (1.000)

Dataset statistics	
Nodes in largest SCC	1300 (0.183)
Edges in largest SCC	39456 (0.381)
Average clustering coefficient	0.1409
Number of triangles	608389
Fraction of closed triangles	0.04564
Diameter (longest shortest path)	7
90-percentile effective diameter	3.8

Raw Wikipedia adminship election data that was used to create the Wiki-vote network can be accessed at http://snap.stanford.edu/data/wiki-Elec.html.

9.2.2 Source (citation)

- J. Leskovec, D. Huttenlocher, J. Kleinberg. Signed Networks in Social Media. CHI 2010.
- J. Leskovec, D. Huttenlocher, J. Kleinberg. Predicting Positive and Negative Links in Online Social Networks. WWW 2010.

9.2.3 Files

File	Description
Wiki-Vote.txt.gz	Wikipedia adminship vote network till January 2008

9.3 Social circles: Twitter (Social / Directed)

9.3.1 Dataset information

http://snap.stanford.edu/data/egonets-Twitter.html
This dataset consists of 'circles' (or 'lists') from Twitter. Twitter data was crawled from public sources. The dataset includes node features (profiles), circles, and ego networks. Data is also available from Facebook and Google+.

Dataset statistics	
Nodes	81306
Edges	1768149
Nodes in largest WCC	81306 (1.000)
Edges in largest WCC	1768149 (1.000)
Nodes in largest SCC	68413 (0.841)
Edges in largest SCC	1685163 (0.953)
Average clustering coefficient	0.5653
Number of triangles	13082506
Fraction of closed triangles	0.06415
Diameter (longest shortest path)	7
90-percentile effective diameter	4.5

9.3.2 Source (citation)

J. McAuley and J. Leskovec. Learning to Discover Social Circles in Ego Networks. NIPS, 2012.

9.3.3 FILES

File	Description
twitter.tar.gz	Twitter data (973 networks)
twitter_combined.txt.gz	Edges from all egonets combined
readme-Ego.txt	Description of files

9.4 SOCIAL CIRCLES: GOOGLE+ (SOCIAL / DIRECTED)

9.4.1 DATASET INFORMATION

http://snap.stanford.edu/data/egonets-Gplus.html
This dataset consists of 'circles' from Google+. Google+ data was collected from users who had manually shared their circles using the 'share circle' feature. The dataset includes node features (profiles), circles, and ego networks. Data is also available from Facebook and Twitter.

Dataset statistics	
Nodes	107614
Edges	13673453
Nodes in largest WCC	107614 (1.000)
Edges in largest WCC	13673453 (1.000)
Nodes in largest SCC	69501 (0.646)
Edges in largest SCC	9168660 (0.671)
Average clustering coefficient	0.4901
Number of triangles	1073677742
Fraction of closed triangles	0.6552
Diameter (longest shortest path)	6
90-percentile effective diameter	3

9.4.2 SOURCE (CITATION)

J. McAuley and J. Leskovec. Learning to Discover Social Circles in Ego Networks. NIPS, 2012.

9.4.3 FILES

File	Description
gplus.tar.gz	Google+ (132 networks)
gplus_combined.txt.gz	Edges from all egonets combined
readme-Ego.txt	Description of files

9.5 CITING SNAP

We encourage you to cite our datasets if you have used them in your work.
You can use the following BibTeX citation:

```
@misc{snapnets,
  author       = {Jure Leskovec and Andrej Krevl},
  title        = {{SNAP Datasets}: {Stanford} Large Network Dataset Collection},
  howpublished = {\url{http://snap.stanford.edu/data}},
  month        = jun,
  year         = 2014
}
```

10 References

[1] Best partition using Louvain :
https://bitbucket.org/taynaud/python-louvain & http://perso.crans.org/aynaud/communities/
https://sites.google.com/site/findcommunities/

[2] Python timeit module
http://pymotw.com/2/timeit/
https://docs.python.org/2/library/timeit.html

[3] Comp-47270 lectures → Section 4 Graph Clustering and Community Finding
Dr. Neil Hurley

[4] Hierarchical Clustering :
http://en.wikipedia.org/wiki/Hierarchical_clustering

[5] Girvan & Newman's Algorithm

[6]