

# การวิเคราะห์การตรวจจับชุมชน: กรณีศึกษาความเกี่ยวข้องกับเพจเฟซบุ๊กขนาดใหญ่ในช่วงเดือนพฤศจิกายน 2017

## Community Detection: The Case Study of the Relevance of Large Facebook Page in November 2017

นายกริช ปรัชญาวาทิน (63199130106)

สาขาวิชาวิทยาการข้อมูล คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ

กรุงเทพฯ ประเทศไทย

E-mail: krich.prach@g.swu.ac.th

### บทคัดย่อ (Abstract)

ในบทความนี้ เป็นการทดลองจากสิ่งที่ได้เรียนในวิชา Social Network Analysis โดยชุดข้อมูลที่ใช้จะเป็นข้อมูลเกี่ยวกับ Facebook page ขนาดใหญ่ที่มีความเกี่ยวข้องกันโดยมีการเก็บในช่วงเดือน พ.ย.2017 [1] โดยศึกษาจากสมาชิกเครือข่ายในชุมชน ความเกี่ยวข้องของเครือข่าย และความเข้ากันของประเภทของเพจ เพื่อวิเคราะห์ถึงความเป็นไปได้ที่แต่ละเพจของ Facebook

จากการสำรวจและทดสอบ พบว่า เพจ Facebook ส่วนใหญ่จะมีความเกี่ยวข้องกันในส่วนประเภท Category ของเพจ เช่น เพจบริษัทเกี่ยวกับเทคโนโลยี ก็จะมี ความเกี่ยวข้อง กับเพจของบริษัทที่มีความเกี่ยวข้องกับเทคโนโลยีเช่นกัน หรือเพจของนักการเมือง ก็จะมี ความเชื่อมโยงกับเพจของนักการเมืองในประเทศนั้น ๆ ทำให้เห็นว่า ความเชื่อมโยงส่วนใหญ่ของเพจใน Facebook เกี่ยวข้องกับ Category ของแต่ละเพจ

**คำสำคัญ (keywords)** – community detection, Facebook Page, ความสัมพันธ์ระหว่างเพจ, เพจเฟซบุ๊ก

### บทนำ (Introduction)

เนื่องจากในปัจจุบัน เรามีการใช้เครือข่ายสังคมออนไลน์มากมาย ทั้ง Facebook, Instagram, Twitter และอื่น ๆ เครือข่ายเหล่านี้ ไม่ได้อยู่แต่ในด้านเทคโนโลยี แต่มีความเกี่ยวข้องกับผู้คนในสังคม ในทุกกลุ่ม เนื่องจากหลายคนสามารถเข้าถึงอินเทอร์เน็ตได้ และมีการ

ใช้งานกันเป็นชีวิตประจำวัน ชุมชนเป็นส่วนหนึ่งของเครือข่ายที่มีอยู่เป็นจำนวนมาก ซึ่งเครือข่ายหนึ่งสามารถมีหลายชุมชน ที่ไหนตภายในชุมชนมีการเชื่อมโยงกันอย่างแน่นหนา จึงทำให้หลายกลุ่มต้องการที่จะศึกษาในการตรวจจับชุมชน (Community Detection) เพื่อทำความเข้าใจ ศึกษาในความสัมพันธ์ของแต่ละเครือข่ายชุมชน เพื่อให้เราได้เข้าใจพฤติกรรมของแต่ละชุมชนและปรับตัวเข้ากับแต่ละเครือข่ายในชุมชนต่อไปได้

การตรวจจับชุมชน (Community Detection) คือ การทำความเข้าใจในโครงสร้างของเครือข่ายที่มีความซับซ้อน สามารถนำเอาข้อมูลส่วนที่มีประโยชน์เพื่อนำมาปรับใช้ต่อไป [2]

แนวคิดในการตรวจจับชุมชนคือ การค้นหากลุ่มที่ซับซ้อนที่ออกมาในรูปแบบกราฟ สามารถเห็นเครือข่ายย่อย ๆ ที่มีความเชื่อมโยงกันในระหว่างโหนดในกลุ่มเดียวกันมากกว่าโหนดในกลุ่มต่าง ๆ ทางสถิติ โดยมีศูนย์กลางของการตรวจจับชุมชนคือ แนวคิดในเชิงโมดูลาร์ซึ่งมีเมตริกที่สามารถตรวจจับความแตกต่างของชุมชนได้ [3]

### งานที่เกี่ยวข้อง (Related Works)

Zhige Xin, Chun-ming Lai, Jon William Chapman (2019) ได้ทำการศึกษา Facebook Page ของ CBS News และ The New York Times พบว่า เครือข่ายมีความกระจุกกระจาย และโครงสร้างของชุมชนที่หนาแน่น สามารถพบได้ในช่วงระหว่างการเข้าใช้ในหน้าของ New York Times [4]

Pranita Jain และ Deepak Singh Tomar (2019) ได้ทำการปรับขนาดและคุณภาพของเครือข่ายชุมชนขนาดใหญ่ตามพารามิเตอร์ เมื่อทำการประเมิน พบว่า การตรวจจับชุมชน (Community Detection) ให้ผลดีกับการปรับขนาดของเครือข่ายชุมชน [5]

Victor Stany Rozario, AZM Ehtesham Chowdhury, Muhammad Sarwar Jahan Morshed (2019) ได้นำเสนอ Interlinked Spatial Clustering Model (ILSCM) ซึ่งสามารถทำการตรวจจับชุมชน (Community Detection) ตามเนื้อหาและ edges มีการเชื่อมโยงลิงก์ระหว่างผู้ใช้กับโหนดโดยรอบ ซึ่งมีการเชื่อมโยงที่ดี มีการจัดเรียงหัวข้อ และกราฟแสดงผลที่เหมาะสม [6]

### เบื้องหลังทางเทคนิค (Technical Background)

เนื่องจากในช่วงหลัง ได้มีการเรียน Social Network Analysis โดยใช้ Neo4j ซึ่งสามารถวิเคราะห์เครือข่ายสังคมได้ จึงได้ทำการทดลองการเรียกดูข้อมูลจากชุดข้อมูลที่มีอยู่ตามขั้นตอน จนได้มาพบว่า ไม่สามารถทำการเชื่อมข้อมูลเข้าหากันได้ เนื่องจากชุดข้อมูลเป็นแบบ Undirected Graph ที่เป็นกราฟไม่มีทิศทางที่ชัดเจน ซึ่งในส่วนของ Neo4j สามารถทำได้เพียงข้อมูลแบบ Directed Graph ที่มีทิศทางของกราฟที่ชัดเจนเท่านั้น จึงได้ลองเปลี่ยนวิธีการโดยการไปทดลองทำโดยใช้ Python ในส่วนของ NetworkX แทน เพื่อให้สามารถทำการสำรวจข้อมูลต่อไปได้

โดยในการทำครั้งนี้เป็นการสำรวจตามความเข้าใจ ไม่เน้นไปทางทฤษฎีที่จะต้องมียุทธศาสตร์แน่นอน แต่จะเน้นไปในภาพรวมที่สามารถเข้าใจได้ง่าย

### ขั้นตอนการทำงาน (Methods)

ขั้นตอนในการทำงาน มีการอ้างอิงแนวทาง ขั้นตอนการทำงานมาจาก [7] มีขั้นตอนการทำงาน ดังนี้

1. ทำการค้นหาคัดข้อมูลที่เหมาะสมในการทำการทดลอง จนได้ค้นพบกับชุดข้อมูล Facebook Large Page-Page Network จึงได้นำข้อมูลชุดนี้มาศึกษา
2. ทำการทดลองกับชุดข้อมูลในครั้งแรกกับ Neo4j ผลออกมาคือ มีชุดข้อมูลมากเกินไป ทำให้เสียเวลาในการประมวลผลของข้อมูล
3. ทำการลดจำนวนของข้อมูลลง เหลือในส่วนชุดข้อมูลของเพจจำนวน 10,000 เพจ ลดการเชื่อมโยงของข้อมูลที่เกี่ยวข้องกับข้อมูลที่อยู่ในลำดับที่ 10,001 เป็นต้นไป ทั้งหมดออกในทุกไฟล์ของชุดข้อมูล และตัดสินใจใช้เพียง

แค่ไฟล์ fb\_target\_edge.csv สำหรับไฟล์ node และไฟล์ fb\_edges\_edit.csv สำหรับไฟล์ edges

4. ทำการทดลองกับชุดข้อมูลที่ทำการลดจำนวนของข้อมูลแล้ว สามารถทำงานกับชุดข้อมูลได้ดี แต่ก็พบปัญหาการไม่เชื่อมของโหนด จนมาพบว่า Neo4j ไม่เหมาะกับการทำงานกับชุดข้อมูลนี้ จนได้ปรึกษากับอาจารย์จนได้ข้อสรุปว่า ให้ลองกลับไปใช้กับ Python ที่สามารถทำงานร่วมกันได้กับ Undirected Graph
5. นำเข้าข้อมูลลงใน Python โดยในการทดลองนี้ ใช้วิธีนำเข้าลงใน Jupyter notebook

```
In [1]: import csv
import networkx
import csv
from operator import itemgetter
import networkx as nx
from networkx.algorithms import community

In [2]: #นำเข้าข้อมูล
with open('fb_target_edit.csv', 'r') as nodescsv: # Open the file
    nodereader = csv.reader(nodescsv) # Read the csv
    # Retrieve the data
    nodes = [n for n in nodereader][1:]

    node_names = [n[0] for n in nodes] # Get a list of only the node names

    with open('fb_edges_edit.csv', 'r') as edgescsv: # Open the file
        edgereader = csv.reader(edgescsv) # Read the csv
        edges = [(e[0] for e in edgereader)[1:]] # Retrieve the data
```

รูปที่ 1 import CSV และ NetworkX และนำเข้าข้อมูล CSV ลงใน Jupyter

6. ทำการสำรวจข้อมูล ดูว่ามีกี่โหนด กี่จุดเชื่อม

```
In [3]: #สำรวจโหนด
print(len(node_names))

10000

In [4]: #สำรวจจุดเชื่อม (edges)
print(len(edges))

34818
```

รูปที่ 2 สำรวจโหนดและจุดเชื่อมของกราฟ

7. ทำการสร้างกราฟ ดูข้อมูลโดยรวมของกราฟ และสร้าง attribute ของข้อมูล ในฟอร์มของ dictionary

```
In [5]: #การสร้างกราฟ
G = nx.Graph()
G.add_nodes_from(node_names)
G.add_edges_from(edges)

In [6]: #ข้อมูลของกราฟ
print(nx.info(G))

Name:
Type: Graph
Number of nodes: 10000
Number of edges: 34818
Average degree: 6.9636

In [7]: #สร้าง attributes ให้ข้อมูล dictionary
facebook_id_dict = {}
page_name_dict = {}
page_type_dict = {}
```

รูปที่ 3 สร้างกราฟ ดูข้อมูลของกราฟ สร้าง attribute ข้อมูล

8. ทำการตั้งค่า Loop ในลิสต์ของโหนด จากนั้นก็ทำการตั้งค่า node attributes

```
In [8]: #ตั้งค่า Loop ใน Node list
for node in nodes:
    facebook_id_dict[node[0]] = node[1]
    page_name_dict[node[0]] = node[2]
    page_type_dict[node[0]] = node[3]

In [9]: #ตั้งค่า node attributes
nx.set_node_attributes(G, facebook_id_dict, 'facebook_id')
nx.set_node_attributes(G, page_name_dict, 'page_name')
nx.set_node_attributes(G, page_type_dict, 'page_type')
```

รูปที่ 4 ตั้งค่า loop ในลิสต์โหนด และตั้งค่า node attributes

## 9. สํารวจควา สามารถดูข้อมูลของโหนดไดหรือไม

```
In [11]: #สํารวจ/สามารถดูข้อมูลของโหนดไดหรือไม
for n in G.nodes():
    print(n, G.nodes[n]['page_name'])

2528 WUOLAH "WUOLAH"
2511 WOODS Nutritional Drink
2512 Instytut Polski w Petersburgu
2517 Tame has medicines on berla
2516 Margford doonan
2515 SORINA "SARIT" DE SAUTAMARA
2516 THAS NAOMT
2517 Councilwoman Jessica P. Abbott
2518 New Zealand Embassy - Washington, D.C.
2519 Kuznetsov Niles
2520 Microsoft
2521 The Home Depot Careers
2520 Itrial.NE
2521 City of Highland Park, Illinois - Government
2521 Laurette Onelzine
2520 genat
2520 Dm Express Egypt
2527 PUDr. Gabriela Peckova - TOP 80
2528 KUN Turkey
2529 McDonald's Bolivia
```

## รูปที่ 5 สํารวจดูข้อมูลโหนดในทีนี้ดูในส่วนชื่อของเพ

## 10. ดูและตั้งคํา Network density และตั้งคํา degree

```
In [1]: #ดู network density
density = nx.density(G)
print('Network density:', density)

Network density: 0.00008428042042085

In [15]: degree_dict = dict(nx.degree(G.nodes()))
nx.set_node_attributes(G, degree_dict, 'degree')
```

## รูปที่ 6 ดูคํา Network density และตั้งคํา degree

## 11. สํารวจข้อมูลแบบเป็นรายโหนด เพื่อดูข้อมูลในโหนดว่ามีอะไร เป็นอย่างไรบ้าง จากนั้นก็ดู sort ของ degree

```
In [19]: #สํารวจข้อมูลรายโหนด ว่ามีคุณสมบัติอะไรบ้าง
print(G.nodes[2528])
{'facebook_ID': '28124688812855', 'page_name': 'Microsoft', 'page_type': 'company', 'degree': 2}

In [20]: print(G.nodes[1])
{'facebook_ID': '191483281412', 'page_name': 'U.S. Consulate General Mumbai', 'page_type': 'government', 'degree': 11}

In [21]: print(G.nodes[7812])
{'facebook_ID': '69778172881', 'page_name': 'The Couch-to-5K Running Plan', 'page_type': 'company', 'degree': 8}

In [22]: print(G.nodes[229])
{'facebook_ID': '148134965281436', 'page_name': 'Cuarto Milenio', 'page_type': 'tvshow', 'degree': 5}

In [23]: print(G.nodes[342])
{'facebook_ID': '148098938515423', 'page_name': 'Sofie Carsten Nielson', 'page_type': 'politician', 'degree': 18}
```

## รูปที่ 7 สํารวจดูข้อมูลเป็นรายโหนด เพื่อดูข้อมูลว่ามีอะไรบ้าง

```
In [24]: sorted_degree = sorted(degree_dict.items(), key=itemgetter(1), reverse=True)
```

## รูปที่ 8 สํารวจดูข้อมูลเป็นรายโหนด เพื่อดูข้อมูลว่ามีอะไรบ้าง

## 12. ทำการสํารวจว่าโหนดไหนมีการเชื่อมของ degree มากที่สุด 20 อันดับ จากนั้นก็สํารวจดูโหนดนั้นว่าเป็นอย่างไร

```
In [25]: #สํารวจโหนดที่มีการเชื่อมของ degree มากที่สุด 20 อันดับ
print('Top 20 nodes by degree:')
for d in sorted_degree[:20]:
    print(d)

Top 20 nodes by degree:
('1387', 221)
('2442', 192)
('9139', 162)
('781', 158)
('9319', 153)
('9294', 151)
('5458', 150)
('8883', 143)
('9228', 136)
('1654', 129)
('4982', 125)
('961', 120)
('5212', 112)
('6441', 108)
('1123', 105)
('3876', 102)
('7467', 100)
('7357', 99)
('496', 98)
('8482', 98)

In [26]: #สํารวจโหนดที่มีการเชื่อมมากที่สุด
print(G.nodes[1387])
{'facebook_ID': '15583727772692', 'page_name': 'Honolulu District, U.S. Army Corps of Engineers', 'page_type': 'government', 'degree': 221}
```

## รูปที่ 9 สํารวจดูโหนดที่มีการเชื่อม degree มากที่สุด 20 อันดับ พร้อมดูโหนดนั้นว่าเป็นอย่างไรบ้าง

## 13. สร้างการเชื่อมต่อ betweenness และ eigenvector

```
In [28]: #สร้างการเชื่อมต่อ betweenness และ eigenvector
betweenness_dict = nx.betweenness_centrality(G) # Run Betweenness centrality
eigenvector_dict = nx.eigenvector_centrality(G) # Run eigenvector centrality

In [29]: nx.set_node_attributes(G, betweenness_dict, 'betweenness')
nx.set_node_attributes(G, eigenvector_dict, 'eigenvector')
```

## รูปที่ 10 เชื่อม betweenness และ eigenvector

## 14. สํารวจดูว่า 20 อันดับของโหนดที่มีการเชื่อม betweenness เป็นอย่างไรบ้าง

```
In [28]: #สํารวจโหนดที่มีการเชื่อม betweenness centrality มากที่สุด 20 อันดับ
sorted_betweenness = sorted(betweenness_dict.items(), key=itemgetter(1), reverse=True)

print('Top 20 nodes by betweenness centrality:')
for b in sorted_betweenness[:20]:
    print(b)

Top 20 nodes by betweenness centrality:
('781', 0.1506531783280452) Degree: 158
('8482', 0.03637958383373884) Degree: 98
('989', 0.02622388579526174) Degree: 69
('7863', 0.02574817834023046) Degree: 79
('9359', 0.02329921665572351) Degree: 58
('7362', 0.02166827886381872) Degree: 83
('1387', 0.02092897138863996) Degree: 221
('9319', 0.01953812585317126) Degree: 153
('4296', 0.018126255775361137) Degree: 150
('5448', 0.01723283841899929) Degree: 38
('7745', 0.0170805151494263) Degree: 39
('984', 0.01635871857887788) Degree: 73
('1964', 0.014757371568467959) Degree: 44
('364', 0.013759575766288688) Degree: 72
('5458', 0.013168778214915564) Degree: 150
('8883', 0.01158821388865152) Degree: 143
('3735', 0.011088778488767617) Degree: 85
('7225', 0.01076884128473278) Degree: 8
('6932', 0.010418828823838729) Degree: 35
('7966', 0.009988893661931492) Degree: 68
```

## รูปที่ 11 สํารวจ 20 อันดับของโหนดที่เชื่อมแบบ betweenness มากที่สุด

```
In [31]: #first get the top 20 nodes by betweenness as a list
top_betweenness = sorted(betweenness[:20])

#then find and print their degree
for tb in top_betweenness: # loop through top_betweenness
    degree = degree_dict[tb[0]] # use degree_dict to access a node's degree, see footnote 2
    print('Name: ', tb[0], ' | Betweenness Centrality: ', tb[1], ' | Degree: ', degree)

Name: 781 | Betweenness Centrality: 0.1506531783280452 | Degree: 158
Name: 8482 | Betweenness Centrality: 0.03637958383373884 | Degree: 98
Name: 989 | Betweenness Centrality: 0.02622388579526174 | Degree: 69
Name: 7863 | Betweenness Centrality: 0.02574817834023046 | Degree: 79
Name: 9359 | Betweenness Centrality: 0.02329921665572351 | Degree: 58
Name: 7362 | Betweenness Centrality: 0.02166827886381872 | Degree: 83
Name: 1387 | Betweenness Centrality: 0.02092897138863996 | Degree: 221
Name: 9319 | Betweenness Centrality: 0.01953812585317126 | Degree: 153
Name: 4296 | Betweenness Centrality: 0.018126255775361137 | Degree: 150
Name: 5448 | Betweenness Centrality: 0.01723283841899929 | Degree: 38
Name: 7745 | Betweenness Centrality: 0.0170805151494263 | Degree: 39
Name: 984 | Betweenness Centrality: 0.01635871857887788 | Degree: 73
Name: 1964 | Betweenness Centrality: 0.014757371568467959 | Degree: 44
Name: 364 | Betweenness Centrality: 0.013759575766288688 | Degree: 72
Name: 5458 | Betweenness Centrality: 0.013168778214915564 | Degree: 150
Name: 8883 | Betweenness Centrality: 0.01158821388865152 | Degree: 143
Name: 3735 | Betweenness Centrality: 0.011088778488767617 | Degree: 85
Name: 7225 | Betweenness Centrality: 0.01076884128473278 | Degree: 8
Name: 6932 | Betweenness Centrality: 0.010418828823838729 | Degree: 35
Name: 7966 | Betweenness Centrality: 0.009988893661931492 | Degree: 68
```

## รูปที่ 12 สํารวจ 20 อันดับของโหนดที่เชื่อมแบบ betweenness มากที่สุด โดยดูเพิ่มในส่วนข้อมูล

## 15. สํารวจในส่วน community detection และ modularity และส่งออกไฟล์เป็น .gexf

```
In [32]: communities = community.greedy_modularity_communities(G)

In [33]: modularity_dict = {} # Create a blank dictionary
for i,c in enumerate(communities): # Loop through the list of communities, keeping track of the number for the community
    for name in c: # Loop through each person in a community
        modularity_dict[name] = i # Create an entry in the dictionary for the person, where the value is which group they belong

# Now you can add modularity information (like we did the other metrics
nx.set_node_attributes(G, modularity_dict, 'modularity')

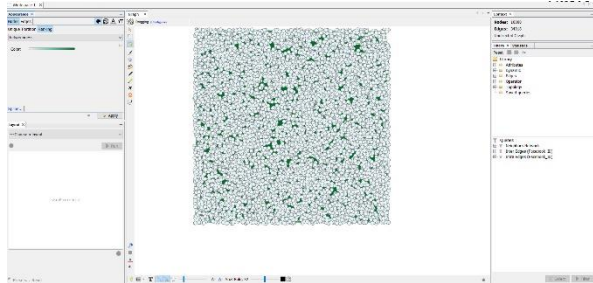
In [34]: for i,c in enumerate(communities): # Loop through the list of communities
    if len(c) > 2: # Filter out modularity classes with 2 or fewer nodes
        print('Class: ',str(i),', ',list(c)) # Print out the classes and their members

Class 149: ['2944', '2927', '2926']
Class 151: ['5896', '1448', '9361', '2796']
Class 158: ['4875', '9243', '9432', '1584']
Class 157: ['4685', '5521', '4713', '9578']
Class 158: ['6785', '9622', '4318', '6497']
Class 159: ['651', '6699', '908', '7684']
Class 128: ['1333', '5689', '1757', '9821']
Class 121: ['7344', '9986', '7752', '6884']
Class 122: ['5007', '8027', '1378', '9977']
Class 123: ['2383', '1638', '5915']
Class 124: ['1181', '3237', '735']
Class 125: ['1480', '2663', '47']
Class 126: ['4449', '3825', '1891']
Class 127: ['1858', '2191', '5145']
Class 128: ['3890', '324', '5328']
Class 129: ['4847', '1586', '5399']
Class 130: ['137', '5519', '3344']
Class 131: ['4514', '5536', '4484']
Class 132: ['1561', '5971', '2487']
Class 133: ['34', '4681', '6808']

In [35]: #ส่งออกไปไฟล์ gexf
nx.write_gexf(G, 'facebookpage_network.gexf')
```

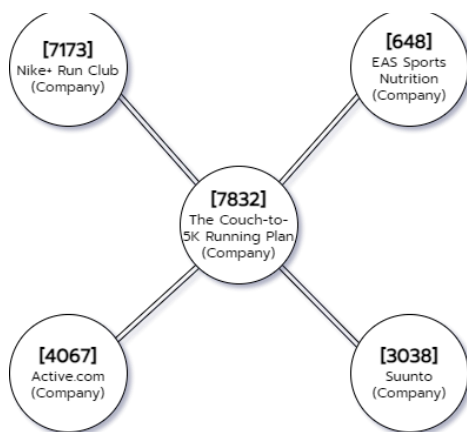
## รูปที่ 13 สํารวจข้อมูลส่วน community detection และ modularity จากนั้นก็ส่งออกไฟล์เป็น gexf

16. ลองเปิดข้อมูลสำรวจใน Gephi โดยใช้ไฟล์ gexf ที่ได้จาก การส่งออกข้อมูลใน Jupyter มาเปิดเพื่อให้ได้เห็นภาพ ของ network ได้มากขึ้น แต่ก็ไม่สามารถดูข้อมูลในแต่ละ โหนดได้เนื่องจากมีข้อมูลที่มากเกินไป แต่สามารถดูข้อมูล ที่มีการประมวลผลได้ ทำได้เพียงเห็นภาพว่า ส่วนที่มีการ เชื่อมโยงกันมาก มีจำนวนน้อยกว่าจำนวนปกติ



รูปที่ 14 ภาพข้อมูลจากการเปิดใน Gephi

17. ทำการสร้างเป็น diagram เพื่อให้เห็นรูปแบบของการ เชื่อมต่อที่น่าสนใจ



รูปที่ 15 ตัวอย่าง diagram การเชื่อมต่อของโหนด 7832 หมวด Company

### ผลการทดลอง (Experimental Results)

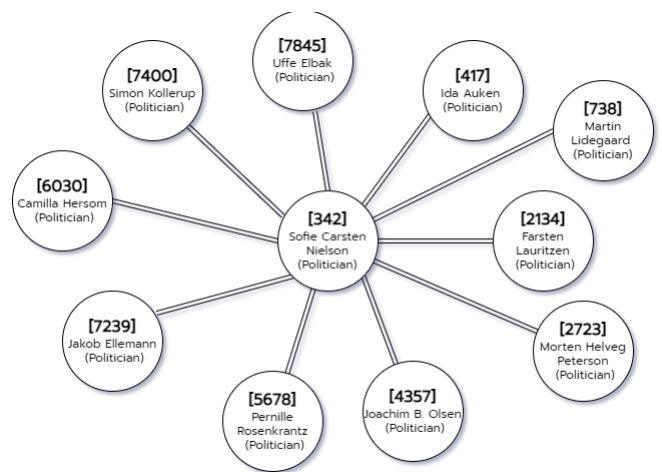
จากการทดลอง สํารวจข้อมูลทั้งหมด พบได้ว่า จำนวนของข้อมูล มีผลกับการสํารวจเป็นอย่างมาก เนื่องจากในการประมวลผลจะต้อง มีการใช้เวลาของเครื่องคอมพิวเตอร์ในการประมวลผลอยู่พอสมควร

เมื่อได้ทำการสํารวจในหลายส่วนของข้อมูลชุดนี้ ได้ทำการ ปรับปรุงข้อมูลแล้วแล้ว พบว่า

1. โหนดที่มีการเชื่อมต่อกันมากที่สุดคือโหนด 1387 เพจ “Honolulu District, U.S. Army Corps of Engineers”

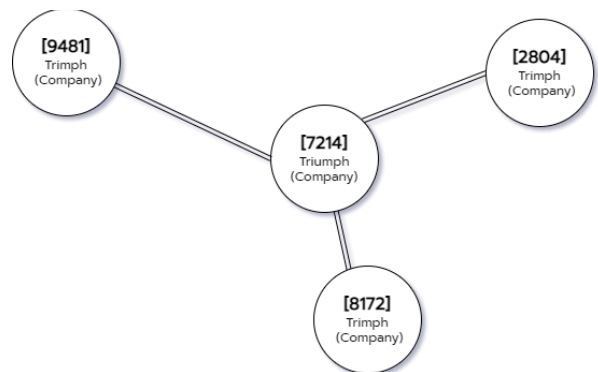
หมวด Government มีการเชื่อม degree ถึง 221 โหนด เลยทีเดียว ทำให้เห็นว่า ในหมวด Government มีการ พบปะ รู้จักกันและกันได้มากที่สุด

2. หมวดที่มีการเชื่อม Betweenness Centrality มากที่สุด คือ โหนด 701 “Facebook” ซึ่งอยู่ในหมวด Company มี ค่าในส่วนนี้สูงถึง 0.15065317838209452
3. จากการสํารวจในหมวดอื่น พบว่า ส่วนใหญ่การเชื่อมโยงถึง กัน มักจะอยู่ในหมวดเดียวกัน เช่น หมวดนักการเมือง (Politician) ดังภาพจะเห็นว่า Sofie Carsten Neilson นักการเมืองจากเดนมาร์ก มีการเชื่อมโยงกับนักการเมืองใน ชาติเดียวกัน แสดงให้เห็นว่านักการเมืองแต่ละท่านมีการ ติดต่อสื่อสาร ทำความรู้จักกันในประเทศของตนมาก



รูปที่ 16 diagram การเชื่อมโยงของ Sofie Carsten Neilson นักการเมืองชาวเดนมาร์ก

4. ในแต่ละบริษัท ก็จะมีเพจของตัวเอง ทำให้มีการเชื่อมโยง ถึงกันของในเพจในต่างพื้นที่หรือต่างสาขากันขึ้น ดังภาพจะ เห็นได้จาก Triumph มีมากกว่า 1 เพจ



รูปที่ 17 diagram การเชื่อมโยงของบริษัท Triumph ที่มีเพจชื่อ เดียวกัน 3 เพจ

5. โหนดที่น่าสนใจโหนดหนึ่งที่ค้นพบคือ โหนด 986 รายการ series ทางโทรทัศน์ เรื่อง Marvel's DareDevil มีการเชื่อมโยงเพียงโหนดเดียว แต่ต่างหมวดกัน ซึ่งอีกโหนดอยู่ในหมวด Company คิดว่าน่าจะเพราะเป็น Sponsor ให้หรืออาจจะมีการรู้จักกันในระหว่าง 2 เพจ ดังจะเห็นได้จากภาพ



รูปที่ 18 diagram การเชื่อมโยงรายการ Marvel's DareDevil ที่มีการเชื่อมโยงกับเพจที่อยู่คนละหมวดกัน

### บทสรุป (Conclusions)

การสำรวจข้อมูล ศึกษาชุดข้อมูลนี้ ทำให้เราได้เรียนรู้ถึงแนวทางและวิธีการในการสำรวจข้อมูลการเชื่อมโยงของเครือข่ายในหลายรูปแบบ ทั้งจาก Neo4j ทั้งจาก Python ทั้งจาก Gephi หรือจากการสำรวจข้อมูลภายในไฟล์ของชุดข้อมูลว่ามีการเชื่อมต่ออะไรน่าสนใจ โดยเลือกจากการสุ่มมาหลาย ๆ category

เมื่อเราได้ลองสำรวจ ลองทำดู จึงทำให้เห็นว่า

1. Facebook page ส่วนใหญ่ที่มีความสัมพันธ์กัน มักจะเป็นเพจที่มีลักษณะที่ใกล้เคียงกัน โดยเฉพาะในส่วนของความสนใจ หากเพจหนึ่งเป็นเพจเกี่ยวกับนักการเมือง ก็จะมีความสัมพันธ์กับเพจของนักการเมืองเพจอื่น ๆ หรือถ้าหากเป็นเพจบริษัทเทคโนโลยี ก็จะมีความสัมพันธ์กับเพจที่มีลักษณะไปในทางเทคโนโลยีเหมือนกัน
2. Facebook page ที่เป็นบริษัทที่มีหลายสาขา มักจะมีเพจย่อยของแต่ละสาขา ตัวอย่างที่เห็นคือ บริษัท Triumph ที่มีชื่อเพจที่เหมือนกันหลายเพจ ทำให้เห็นว่า ถ้าเป็นพวกบริษัทที่มีเพจ แล้วเป็นบริษัทใหญ่ ก็จะมีเพจย่อย ๆ กระจายไปตามสาขางานของบริษัทหรือส่วนนั้น ๆ
3. Facebook page อาจจะไม่จำเป็นต้องมี category ที่เหมือนกัน แต่อาจเกี่ยวข้องกันในการทำงาน การเป็น

ผู้ร่วมงานกันได้ จะเห็นได้จากตัวอย่างของเพจ series เรื่อง DareDevil ที่อยู่ในหมวด TV Show แต่มีความเกี่ยวข้องกับเพจ Wizard World ที่เกี่ยวกับวัฒนธรรม Pop Culture ที่เกี่ยวข้องกับ Comics ภาพยนตร์ Sci-fi และเกม ที่สามารถเกี่ยวข้องกับรายการโทรทัศน์เรื่องนี้ได้

เมื่อเป็นดังนี้แล้ว การจัดตั้งเพจของ Facebook เราจึงควรมีปฏิสัมพันธ์กับเพจอื่นที่มีความใกล้เคียง หรือเกี่ยวข้อง เพื่อให้เราจะได้สามารถร่วมงานกัน เพื่อให้งานออกมามีความน่าสนใจ และทำให้มียอดผู้คนเข้ามากันได้ อย่างแน่นอน

### เอกสารอ้างอิง (References)

- [1] B. Rozemberczki, C. Allen, and R. Sarkar. "Facebook Large Page-Page Network." Accessed April 4, 2021. <https://snap.stanford.edu/data/facebook-large-page-page-network.html>.
- [2] Sobolevsky, Stanislav, and Riccardo Campari. "General Optimization Technique for High-Quality Community Detection in Complex Networks," January 1, 2014. [http://senseable.mit.edu/papers/pdf/20141001\\_Sobolevsky\\_etal\\_GeneralOptimization\\_PhysicalReview.pdf](http://senseable.mit.edu/papers/pdf/20141001_Sobolevsky_etal_GeneralOptimization_PhysicalReview.pdf). K. Elissa, "Title of paper if known," unpublished.
- [3] Allman, Andrew. "Community Detection." Accessed April 28, 2021. <https://www.sciencedirect.com/topics/computer-science/community-detection>. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetism Japan, p. 301, 1982].
- [4] Xin, Zhige, Chun-ming Lai, and Jon William Chapman. "Multi-View Community Detection in Facebook Public Pages," January 7, 2019. [https://www.researchgate.net/publication/334382861\\_Multi-View\\_Community\\_Detection\\_in\\_Facebook\\_Public\\_Pages](https://www.researchgate.net/publication/334382861_Multi-View_Community_Detection_in_Facebook_Public_Pages).
- [5] Jain, Pranita, and Deepak Singh Tomar. "Review of Community Detection over Social Media: Graph Prospective," n.d., 600–601.
- [6] Rozario, Victor Stany, AZM Ehtesham Chowdhury, and Muhammad Sarwar Jahan Morshed. "Community Detection in Social Network Using Temporal Data," January 4, 2019. <https://arxiv.org/ftp/arxiv/papers/1904/1904.05291.pdf>.
- [7] Ladd, John R., Jessica Otis, Christopher N. Warren, and Scott Weingart. "Exploring and Analyzing Network Data with Python." Accessed April 29, 2021. <https://programminghistorian.org/en/lessons/exploring-and-analyzing-network-data-with-python>.