GROUP PROJECT

TOPIC - NETWORK ANALYSIS ON AMAZON PRODUCT CO-PURCHASING DATA

BY:

PRANAV VARDHAN G A	**19Z232**
RAKESH M	**19Z235**
SIVASUBRAMANIAM J	**19Z244**
SOUMEN SAHA	**19Z245**
VIVEKANANDHAN S	**19Z261**

19ZO02 – SOCIAL AND ECONOMIC NETWORK ANALYSIS

BACHELOR OF ENGINEERING

BRANCH: COMPUTER SCIENCE AND ENGINEERING [2019-2023]

FACULTY: MS. VANI K

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PSG COLLEGE OF TECHNOLOGY
(Autonomous Institution)
COIMBATORE - 641004

TABLE OF CONTENTS

1. PROBLEM STATEMENT

Social networks in the real world frequently have a time element. To determine the significance of nodes with high in- and out-degrees, network data must be analysed. Analysis of the co-purchase data on Amazon is required to construct a social network among specific products (books) and identify the critical measures for sales.
The main aim of this project is to :

- Predict new links to discover relationships between existing nodes which were previously unconnected.
- Detect communities by grouping the nodes into various clusters to identify which similar books are co-purchased and fall under the sub-categories.
- Recommend the top 5 books for a particular book purchase based on categories.

2. DATASET DESCRIPTION

The data was collected from the website of Stanford Network Analysis Project used as general purpose graph mining library and includes 548,552 different product reviews and product metadata.
Dataset Link - https://snap.stanford.edu/data/index.html#amazon
For each product the following features are available:

- Title,
- Sales rank,
- Lists of the similar products (that get co-purchased with the present product),
- Product categorization details,
- Product reviews details like time, customer rating, number of votes, number of people that found the review useful.

Dataset statistics	
Products	548,552
Product-Project Edges	1,788,725
Reviews	7,781,990
Product category memberships	2,509,699
Products by product group	
Books	393561
DVDs	19828
Music CDs	103144
Videos	26132

fig. Amazon product metadata stats.

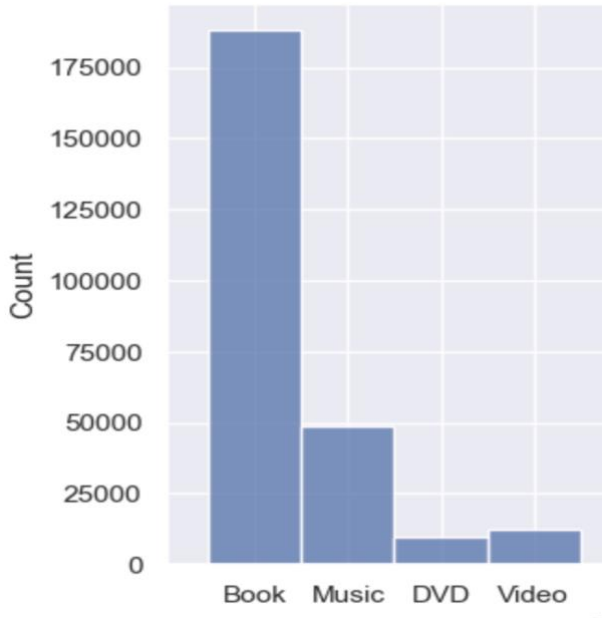

fig. Category counts in Co-purchasing network data

3. TOOLS USED

- **Python**: The Python programming language has been used for the coding part because of its user-friendly data structures and vast support for network and visualisation-related libraries.

- **Gephi**: The free and open-source software, Gephi has been used for network visualisation and analysis of various network properties.

- **NetworkX:** NetworkX, the most popular Python package for manipulating and analysing graphs, is used for graph object creation and community detection.

- **Jupyter notebook:** The Jupyter Notebook, an open-source web application, is used for data cleaning and transformation, numerical simulation, statistical modelling, data visualization, and link prediction using machine learning.

- **Igraph**: Igraph, a library collection, is used for creating and manipulating graphs and analysing networks and was found to execute operations faster compared to NetworkX.

- **Other Python Modules/Libraries Used**: Matplotlib, Pandas, Random, NLTK, re, Seaborn, Scikit-Learn.

4. CHALLENGES FACED

- The size of the dataset provided by Stanford University is very large, so it was time-consuming to load and visualise the graph initially using Gephi and NetworkX.
- Since the team members were new to the network libraries, NetworkX and Igraph, it was tiring to understand the syntax and visualise the graphs using these tools.
- Executing the community formation algorithms on the dataset took a long time to form clusters and generate the corresponding graph structure.
- High training time for model building due to limited hardware configuration.

5. CONTRIBUTION OF TEAM MEMBERS

Registered Id.	Contributor Name	Contribution (Module/Task wise)
19z232	Pranav Vardhan G A	Data Pre-processing
19z235	Rakesh M	Link Prediction
19z244	Sivasubramaniam J	Book Recommendation
19z245	Soumen Saha	Community Detection
19z261	Vivekanandhan S	Collecting the Datasets and its Analysis

Project Tasks or Modules

- **Collecting the Datasets and doing its Analysis**: Gathering and exploring co-purchased datasets from various websites and doing analysis on them based on the statistics and distribution.
- **Data Pre-processing**: Filtering the books out of other products and co-purchases, truncating the first 5000 edges out of about 6 lakh edges to smoothly load data and apply network algorithms.
- **Link Prediction**: Randomly two edges or connections from each node are removed and a ML model based on SVM Classifier would be trained by using both the connected and unconnected edges to detect new possible edges.
- **Community Detection**: The communities are formed in the form of clusters based on the Girvan-Newman algorithm and edge-betweenness techniques. Different clusters are represented using unique colors, and the visualizations are compared using NetworkX and igraph.
- **Book Recommendation**: The top 5 books are given as output for a particular book whose similarity measure is greater than a threshold value of 0.5. This is done to predict customer preference and increase sales based on the output recommendation.

ANNEXURE – I : CODE

Code to detect new link from the existing non-connecting nodes:

```
Source = set(Dataset['Source'].unique())
Target = set(Dataset['Target'].unique())
Nodes = list(Source.union(Target))
Number_of_Nodes = len(Nodes)
Adjacency_Matrix = nx.to_numpy_array(Graph, nodelist = Nodes)
```

```
Missing_Nodes = []
Number = 0
for i in tqdm(range(Number_of_Nodes)):
    for j in range(Number, Number_of_Nodes):
        if(i != j):
            try:
                if nx.shortest_path_length(Graph, Nodes[i], Nodes[j]) <= 2:
                    if Adjacency_Matrix[i,j] == 0:
                        Missing_Nodes.append([Nodes[i], Nodes[j]])
            except:
                continue
    Number += 1
```

```
100%|██████████| 3364/3364 [07:42<00:00,  7.28it/s]
```

Sampling a Part of Connected Edges for Training

```
Removed_Edges = []
DF = Dataset.copy()
N = 0
for i in tqdm(Dataset.index.values):
    Current = nx.from_pandas_edgelist(DF.drop(index = i), 'Source', 'Target', create_using = nx.Graph())
    if(nx.number_connected_components(Current) == 1 and len(Current.nodes) == Number_of_Nodes):
        N += 1
        if(N >= 2):
            N = 0
            continue
        Removed_Edges.append(i)
        DF = DF.drop(index = i)
```

```
100%|██████████| 10000/10000 [03:07<00:00, 53.20it/s]
```

```
kernels = ['rbf', 'linear']
c_values = [0.001, 0.01, 0.1, 1]
param_grid = {'kernel':kernels, 'C':c_values}

SVM_Model = GridSearchCV(cv = 10, estimator = svm.SVC(), param_grid = param_grid)
```

```
SVM_Model.fit(X_Train, Y_Train)
```

```
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': [0.001, 0.01, 0.1, 1],
                         'kernel': ['rbf', 'linear']})
```

```
print(classification_report(SVM_Model.best_estimator_.predict(X_Test), Y_Test))
```

```
              precision    recall  f1-score   support

           0       1.00      0.94      0.97      1374
           1       0.94      1.00      0.97      1224

    accuracy                           0.97      2598
   macro avg       0.97      0.97      0.97      2598
weighted avg       0.97      0.97      0.97      2598
```

Code to detect communities in co-purchases using NetworkX based on Girvan-Neuman Algorithm

```
G=nx.Graph()
G.add_edges_from(edge_df.values.tolist())
```

```
# Finding the communities using the Girvan Newman algorithm
communities = girvan_newman(G)

node_groups = []
for com in next(communities):
    node_groups.append(list(com))
```

```
color_map = []

# Taking only 7 groups out of 51
for node in G:
    if node in node_groups[0]:
        color_map.append('red')
    elif node in node_groups[1]:
        color_map.append('blue')
    elif node in node_groups[2]:
        color_map.append('green')
    elif node in node_groups[3]:
        color_map.append('yellow')
    elif node in node_groups[4]:
        color_map.append('brown')
    elif node in node_groups[5]:
        color_map.append('pink')
    elif node in node_groups[6]:
        color_map.append('cyan')
    else:
        color_map.append('#FF000000')

# Visualizing Communities represented by different colors
nx.draw(G, node_color=color_map,)
plt.show()
```

Code to detect communities in co-purchases using igraph based on edge betweenness:

```
# created commmunities based on edge betweeness
v = g2.community_edge_betweenness(directed=True)
print(v)
```

```
Dendrogram, 3476 elements, 3426 merges
```

```
# groupping the nodes into clusters with cluster indices
cluster1 = v.as_clustering()
print(cluster1)
```

```
Clustering with 3476 elements and 62 clusters
[ 0] 0, 3, 4, 5, 7, 8, 13, 14, 15, 25, 26, 28, 29, 39, 41, 42, 43, 47,
```

```
# Assigning differnt colors to the nodes belonging to same cluster
color_palletes = igraph.drawing.colors.ClusterColoringPalette(len(cluster1))
g2.vs['color'] = color_palletes.get_many(cluster1.membership)
```

```
# Visualizing the communities present in the graph
visual_style["layout"] = g2.layout_fruchterman_reingold()
visual_style["bbox"] = (300, 300)
visual_style["margin"] = 10

fig, ax = plt.subplots()
fig.set_figwidth(400)
fig.set_figheight(400)
igraph.plot(g2,**visual_style, target=ax)
```

Code associated with book recommendation:

```
print("Looking for Recommendations for Customer Purchasing this Book: ")
print("---------------------------------------------------------------")
Purchased_ASIN = '0805047905'
print("ASIN = ", Purchased_ASIN)
print("Title = ", Books[Purchased_ASIN]['Title'])
print("SalesRank = ", Books[Purchased_ASIN]['SalesRank'])
print("TotalReviews = ", Books[Purchased_ASIN]['TotalReviews'])
print("AvgRating = ", Books[Purchased_ASIN]['AvgRating'])
print("DegreeCentrality = ", Books[Purchased_ASIN]['DegreeCentrality'])
print("ClusteringCoeff = ", Books[Purchased_ASIN]['ClusteringCoeff'])
```

```
Looking for Recommendations for Customer Purchasing this Book:
---------------------------------------------------------------
ASIN =  0805047905
Title =  Brown Bear, Brown Bear, What Do You See?
SalesRank =  171
TotalReviews =  172
AvgRating =  5.0
DegreeCentrality =  213
ClusteringCoeff =  0.66
```

```
n = Purchased_ASIN
ego = networkx.ego_graph(Copurchase_Graph, n, radius = 1)
Purchased_ASIN_Ego_Graph = networkx.Graph(ego)
```

```
threshold = 0.5 # finding the nodes having similarity measure based on category above the threshold value
Purchased_ASIN_Ego_Trim_Graph = networkx.Graph()
for f, t, e in Purchased_ASIN_Ego_Graph.edges(data = True):
    if e['weight'] >= threshold:
        Purchased_ASIN_Ego_Trim_Graph.add_edge(f, t)
```

```
Purchased_ASIN_Neighbors = Purchased_ASIN_Ego_Trim_Graph.neighbors(Purchased_ASIN)
```

Showing Top 5 Recommendations

```
Top5_ByAbgRating_ThenByTotalReviews = sorted(ASIN_Meta, key = lambda x: (x[4], x[3]), reverse = True)[:5]
```

```
print()
print("Top 5 Recommendations By AvgRating Then By TotalReviews for Users Purchased The Book: ")
print("-------------------------------------------------------------------------------------")
print('ASIN\t', 'Title\t', 'SalesRank\t', 'TotalReviews\t', 'AvgRating\t', 'DegreeCentrality\t', 'ClusteringCoeff')
for asin in Top5_ByAbgRating_ThenByTotalReviews:
    print(asin)

print()
```

```
Top 5 Recommendations By AvgRating Then By TotalReviews for Users Purchased The Book:
-------------------------------------------------------------------------------------
ASIN     Title   SalesRank       TotalReviews    AvgRating       DegreeCentrality        ClusteringCoeff
('0152010661', 'Time for Bed', 3122, 87, 5.0, 60, 0.0)
('0694006246', 'Big Red Barn Board Book', 4457, 40, 5.0, 27, 0.0)
('1581170769', 'What Makes a Rainbow?: Pop-Up', 40821, 29, 5.0, 7, 0.0)
('0064435962', 'From Head to Toe', 187777, 22, 5.0, 5, 0.0)
('0694013013', 'From Head to Toe Board Book', 6026, 22, 5.0, 47, 0.0)
```

ANNEXURE – II : SNAPSHOTS OF OUTPUT

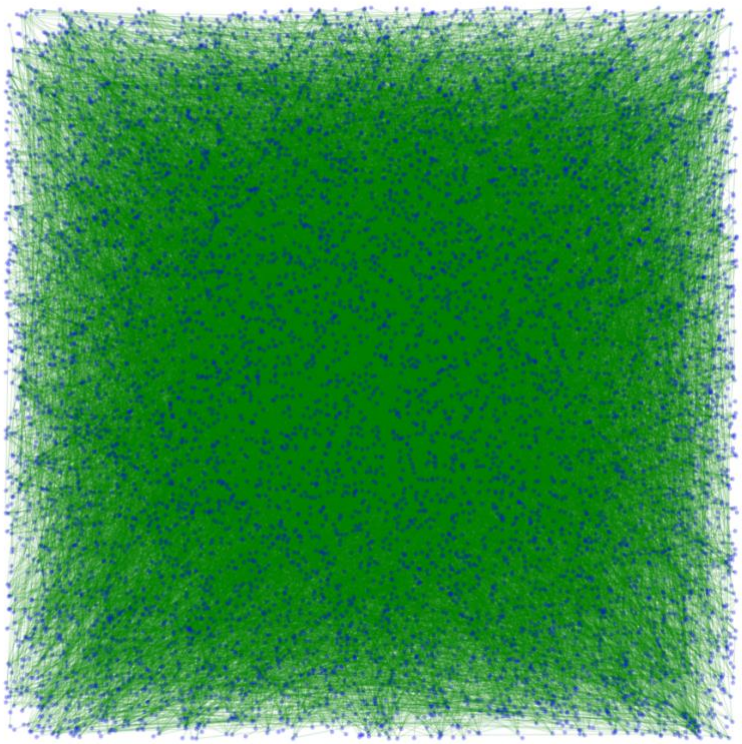

Fig 1 : Visualizing Graph using NetworkX

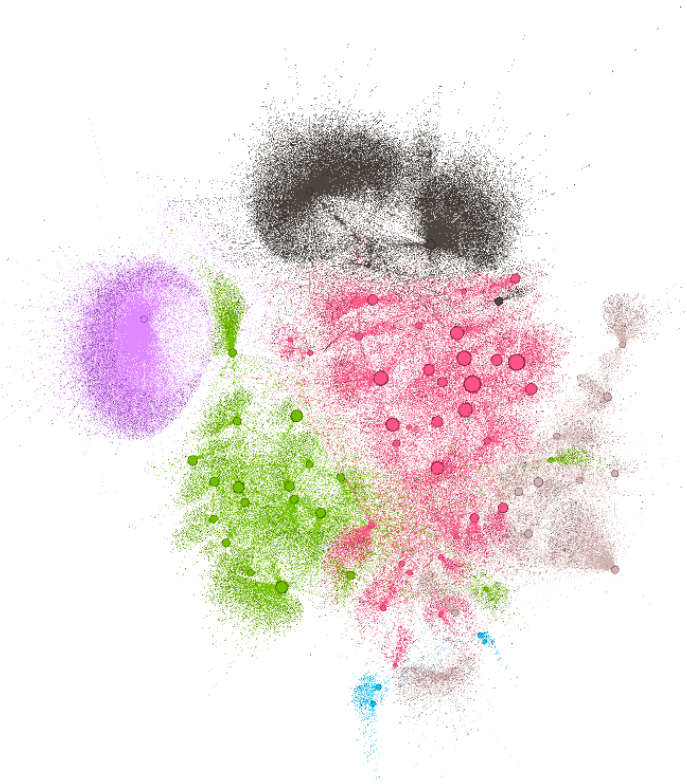

Fig 2 : Detecting Communities using Gephi

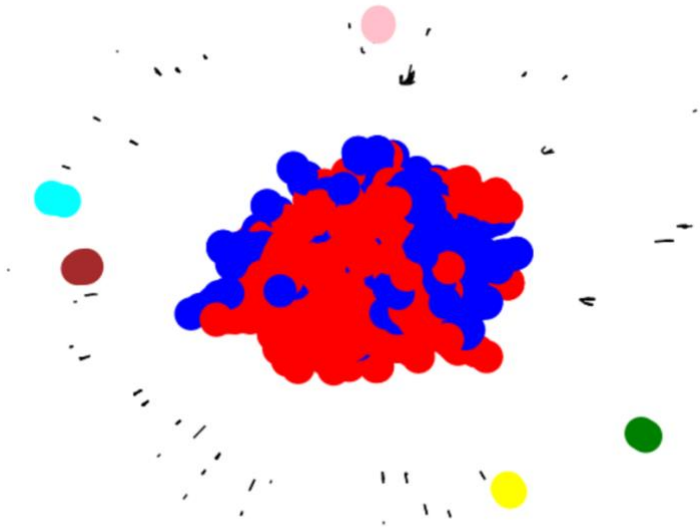

Fig 3 : Detecting Communities using NetworkX

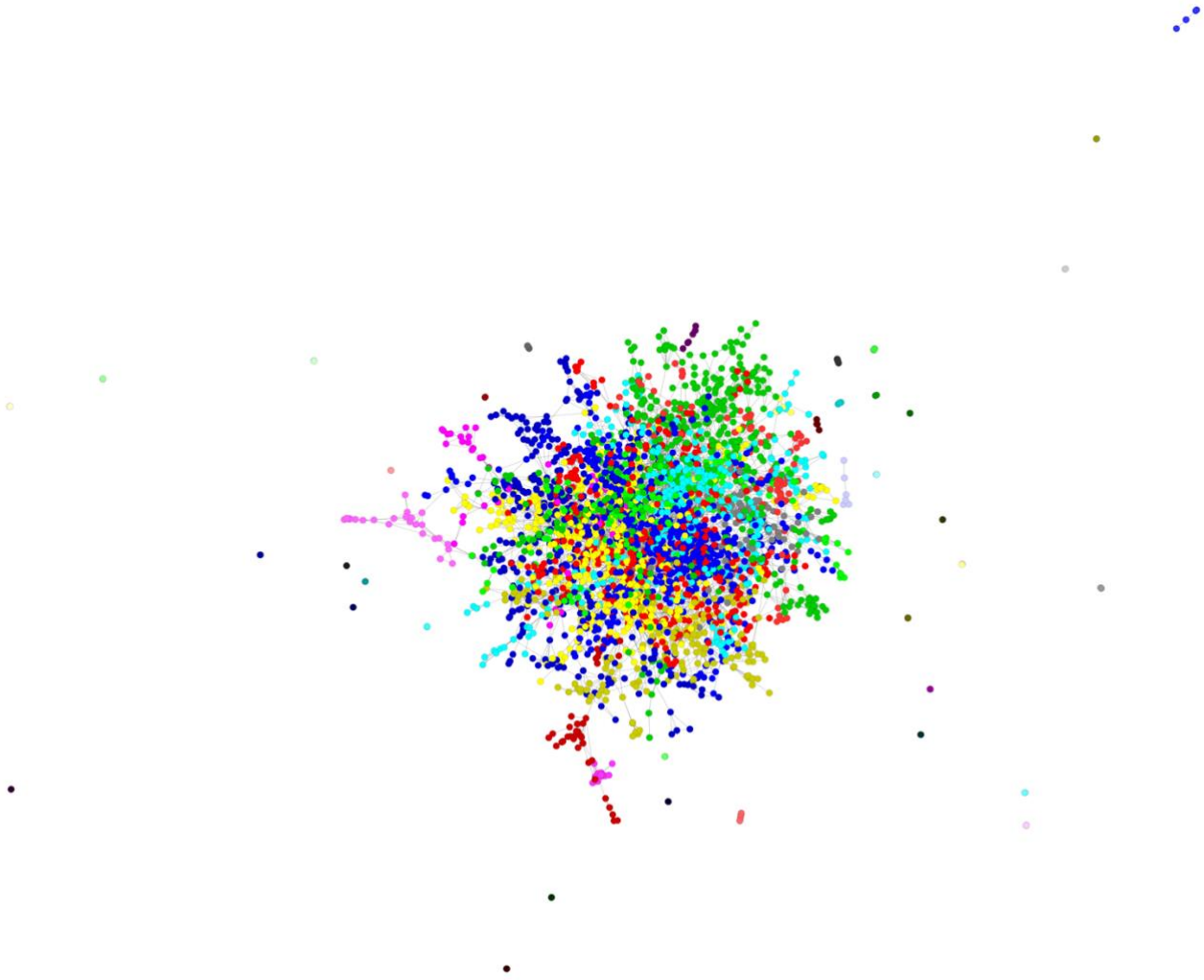

Fig 2 : Detecting Communities using igraph

REFERENCES

[1] **Dataset Link** - https://snap.stanford.edu/data/index.html#amazon
[2] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[3] https://link.springer.com/chapter/10.1007/978-3-031-12097-8_8
[4] https://igraph.org/python/tutorial/0.9.6/
[5] https://networkx.org/documentation/networkx-1.9/
[6] https://www.youtube.com/watch?v=mua3LJFTj7s
[7] http://web.stanford.edu/class/cs224w/index.html#content
[8] https://igraph.org/c/html/latest/igraph-Community.html
[9] https://www.statisticshowto.com/jaccard-index/
[10] https://www.sciencedirect.com/topics/computer-science/community-detection

VeriGuide - Originality Report
Individual Report

Background Information

File Name:	SENA_GROUP_PROJECT.pdf
Report Generated On:	12/11/2022, 11:01:24 AM

Similarity Statistics Overview

Similar Sentence(s) Found By VeriGuide:	1 out of 48 sentences = 2.08%
Similar Sentence(s) Filtered by User:	1 out of 48 sentences = 2.08%
Sentence(s) Selected By User To Export:	0

Similarity Statistics for Each Source

Entry	Source	From	Similarity
1	http://snap.stanford.edu/class/cs224w-2018/data.html	Internet	1 / 48 = 2.08%
2	https://snap.stanford.edu/data/	Internet	1 / 48 = 2.08%
3	https://snap.stanford.edu/data/amazon-meta.html	Internet	1 / 48 = 2.08%