**Influence on Sales**

A picture containing colorfulness, text, map

Description automatically generated

Figure 1 - Graph Visualization, author image

FS2023: 63091 Social Media Analytics

**Sophie Caroni & Emmanuel Cazzato & Quentin Nater** [sophie.caroni@unifr.ch](mailto:sophie.caroni@unifr.ch) & [emmanuel.cazzato@unifr.ch](mailto:emmanuel.cazzato@unifr.ch) & [quentin.nater@unifr.ch](mailto:quentin.nater@unifr.ch)

**Table of Contents**

[1 Introduction 1](#_Toc135840470)

[2 Load data and pre-processing 1](#_Toc135840471)

[3 Graph Exploration and Visualization 2](#_Toc135840472)

[4 Data Persistence 3](#_Toc135840473)

[5 Data Enrichment 3](#_Toc135840474)

[6 Community Detection 3](#_Toc135840475)

[7 Quality of the Community Detection 4](#_Toc135840476)

[8 Results 4](#_Toc135840477)

[9 Conclusion 4](#_Toc135840478)

**Table of Figures**

[Figure 1 - Graph Visualization, author image 1](#_Toc135840479)

[Table 1 - Quality of the graph samplings 2](#_Toc135840480)

[Table 2 - Comparison of community detection algorithms 3](#_Toc135840481)

[Table 3 - Results of the community detections 4](#_Toc135840482)

# Introduction

In recent years, social media has become a vast source of valuable data for businesses and researchers alike. The ability to extract insights and make informed decisions based on this data has led to the emergence of social media analysis as a powerful tool. In our project, we focus on leveraging social media data to gain insights into the influence of the structure of different nodes, neighbors, and edges within a graph represented by an Amazon dataset (<http://snap.stanford.edu/data/amazon-meta.html>).

The Amazon dataset provides us with a rich collection of information about products, including categories, sales rank, review, or similar products. However, we focused on the Amazon Standard Identification Number (ASIN) and its similar products. By harnessing the power of this dataset and analyzing the social media interactions around these products, we aim to uncover patterns and relationships that demonstrate if central nodes of communities are in the top rank sales.

The first step in our analysis involves constructing a sampled graph representation of the dataset in order to do some graph exploration and visualization. Each product is represented by its ASIN, while their similar products list is captured as edges. This constructed graph allows us to detect communities and analyze the relation between products.

We aim to answer technical questions such as: What is the better way to sample our graph? Is our design graph fully connected? Is noise corrupting the graph? What is the better algorithm to produce the most efficient community detection? How to evaluate this algorithm?

These technical answers will finally lead to business questions such as: Which product has the highest degree of influence within their community? Are the popular nodes of each community the most sold?

Through our project, we hope to provide a comprehensive analysis of the popular nodes and their community detection. This research has the potential to inform publishers, authors, and marketers about effective strategies for promoting products, identifying influential nodes in the network, and understanding the impact of social media recommendation on sales.

# Load data and pre-processing

For our project, we used the Amazon metadata dataset. This dataset contains comprehensive information for 548,552 products, which leads to 719,150 nodes (with all similar not registered products) and 1’788’712 edges. We collected social media data spanning the time period from summer 2006 onwards.

Considering the large size of the dataset, we worked on a sampling algorithm to manage the computational requirements efficiently. To preprocess and clean the social media data, we focused on extracting essential information while discarding unnecessary details.

First, we only keep the data necessary from the original dataset. We loaded the large original text file and exported the necessary data into a clean and smaller text file. During this process, we constructed the original graph by using all ASIN as nodes and their similarity list as edges.

Then, we have created a loop calling 4 algorithms to clean the noise and reduce the size of the dataset, keeping in mind that the goal is to keep the popular nodes and remove the unnecessary ones. The loop is the following: while it is impacting the graph, we removed all isolated, not in-edged and not out-edged nodes, because these nodes would have less impact on the community detection and the popular nodes, then if needed, we deleted nodes with the smallest degree possible and looped again.

With this loop, we created sampled graphs of desired size that keep the popular nodes and main communities, managing the less change possible in the cluster coefficient and the average degree.

Finally, we have created features that will export and quickly reconstruct the sampled graph. The idea here is to run only one time the sampling long algorithm and then recall with a small amount of time the exported graph for any future utilization.

In the table below, we can find the conclusion of the pre-processing quality check. We can see that the “*dataset\_off\_amazon\_enrichment.txt*” has enough nodes and edges with an affordable reconstruction runtime. The clustering coefficient difference can be explained by the fact that we remove unnecessary nodes that lower the cluster qualities which also explained the increment of the average degrees.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **The quality of the graph sampling** | | | | | |
| **Datasets** | **Nodes** | **Edges** | **Runtime** | **Clustering Coefficient** | **Average Degree** |
| *amazon-meta.txt* | 719'150 | 1'788'712 | 0:20.946405 | 0.174 | 4.974 |
| *dataset\_off\_amazon\_enrichment.txt* | 189'825 | 613'464 | 0:03.691040 | 0.333 | 6.463 |
| *dataset\_off\_amazon\_big.txt* | 126'981 | 527'243 | 0:02.314533 | 0.495 | 8.304 |
| *dataset\_off\_amazon\_small.txt* | 60'707 | 260'472 | 0:01.220927 | 0.562 | 8.581 |

Table 1 - Quality of the graph samplings

# Graph Exploration and Visualization

Many features were implemented for the exploration and the visualization of the graph. The focus of these tools is to see easily and clearly the properties of the graph and communities.

In the exploration of the graph, we checked if the graph was homogeneous and connected. These values change deeply depending on the pre-processing choice and the size of the sampling. We compared the exploration algorithm with our graph and used the Depth-first search (DFS) which was the most efficient one to check if all nodes were reachable on graph. Then, we run A\* search to compute the smallest path between two nodes and check the minimum size of their leap.

In visualization, we aimed to display the whole graph with each community highlighted in different color and the central node in gold. With the zoom tool of the plot, it is easy to see cluster of nodes and their influence. It is also possible to print a single community to see their inner nodes and inner relationships. Finally, we can plot the degree distribution of the whole graph.

# Data Persistence

We have implemented an online persistence graph on neo4j. The goal of this database is to store the whole nodes and edges of the graph and mainly to launch queries for visualization purposes faster than python plots. 5 queries have been implemented to display:

1) A sampled graph (1,000 nodes),

2) The whole graph,

3) The desired community (for example the number 1),

4) Communities as hypernodes (convert all communities in hypernodes to see their interconnections).

The database is accessible in <https://workspace-preview.neo4j.io/workspace/query> with the google account “[neographunifr@gmail.com](mailto:neographunifr@gmail.com)” password “*neo\_Pa$$w0rd*”.

# Data Enrichment

A newer and more complete dataset of amazon products has been found online and analyzed to extract the common ASIN for merging. We check for every sample of this new dataset the effect on the current graph community detection with the score of the silhouette index. The merged graph created with the current graph and the better sample of the new dataset has been exported as the enriched dataset that will be used for the future tests.

# Community Detection

To determine which algorithm of community detection will be used, we simply checked all group-based algorithms with our graphs. We use the comparison of the runtime, the number of communities and the silhouette index to determine which algorithm is most efficient for our graphs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Comparison of algorithms** | | | | |
| **dataset** | **Algorithm** | **Runtime** | **Silhouette Index** | **Communities** |
| dataset\_off\_amazon\_big.txt | Louvain |  |  |  |
| dataset\_off\_amazon\_big.txt | Girvan Newman |  |  |  |
| dataset\_off\_amazon\_big.txt | Modularity |  |  |  |

Table 2 - Comparison of community detection algorithms

Table 3 shows that the Louvain algorithm was the better one for our graphs and computation limitations. Then, we decided to code it by hand. We implemented a simple homemade algorithm based on the Louvain idea. The code ran and the results were not that bad, but the runtime was too long.

To correct this problem, we searched for new techniques on online papers and found a weight algorithm implemented on top of the Louvain main idea to limit the time needed for computation. With this in mind, we get inspired from these ideas to develop the homemade amazon community detection, thought for a graph with only directed edges logic and none defined weight adaptation.

# Quality of the Community Detection

The homemade amazon community detection was running well, but we needed to evaluate its quality. To do so without ground truth, we implemented metrics to compare the quality of our community detection with the library of NetworkX Louvain community detection.

We first compare the silhouette index and the number of communities of the library with homemade implementations. Then, we compute the Accuracy, Precision, Recall and the Jaccard similarity between the homemade implementation communities and the library communities transformed as labels. In order to have a throughout view, we also compare the results with different datasets with runtimes.

# Results

The bellowed table summarizes the test of the community detection and its quality.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **The quality of the community detection** | | | | | | | | |
| **Algorithms** | **Datasets** | **Runtime** | **Silhouette Idx** | **Accuracy** | **Precision** | **Recall** | **Jaccard** | **Communities** |
| **Simple Detection**  **(homemade)** | amazon\_big |  |  |  |  |  |  |  |
| **Simple Detection**  **(homemade)** | amazon\_enrichment |  |  |  |  |  |  |  |
| **Amazon Detection**  **(homemade)** | amazon\_big | 0:00:12.8 | 0.05008 | 97.01% | 95.36% | 97.01% | 0.925 | 21157 |
| **Amazon Detection**  **(homemade)** | amazon\_ enrichment | 0:00:17.5 | 0.19851 | 97.54% | 89.76% | 97.54% | 0.875 | 17818 |
| **NetworkX Louvain** | amazon\_big | 0:00:21.4 | 0.02363 | - | - | - | - | 23521 |
| **NetworkX Louvain** | amazon\_enrichment | 0:00:17.7 | 0.08338 | - | - | - | - | 26887 |

Table 3 - Results of the community detections

First, we can see that the simple homemade detection and the amazon complex detection, both implemented by hand, demonstrate similar results, but the runtime is excessively high on the first one compared to the second one (+ 0.000) proving the second implementation is more efficient.

We can highlight that the homemade Amazon detection made for our graph has a better silhouette index than the NetworkX Louvain algorithm (+0.02645 for the dataset\_off\_amazon\_big.txt and +0.11513 for the dataset\_off\_amazon\_enrichment.txt) with a better runtime. This result also proves that the enriched dataset is more effective than the original sampled one. Furthermore, the Accuracy, Precision and Recall demonstrates a close relationship between the community detected on the NetworkX and the homemade implementation, proving that the logic of Louvain is maintained despite our own implementation. Finally, the number of communities is lower with our own implementation which is an advantage for the evaluation of our popular nodes.

# Conclusion

We were able to implement an effective community detection homemade algorithm even if the original amazon dataset in not well distributed for community detection with a silhouette index score that demonstrates a small distance between communities and not a too high distance between nodes of the community. However, the analysis of the popular nodes proves a correlation between the rank sales and the center of community (popular nodes).