Project Final Report

comparison of Community Detection Algorithms

on Amazon co-product Network

Instructor: Assefaw Gebremedhin

Team Members: Ping-Wen Chen, Hsueh-Jen Lih, Zhifu Yang

Abstract

When it comes to recommending products, detecting sub-networks in the co-purchasing network is demanded. While involving this idea, our team decided to bring this project up with the analysis of five community detection algorithms. However, to better make this experiment closer to a real-world case, we then tested the five community detection algorithms on the Amazon co-purchasing network datasets. Our team also want to see a future work in combining machine learning methods with community detection algorithms. Which in this case, a run time and modularity score means more. Therefore, this paper will be focusing on analyzing the run time and modularity score of each algorithm.

Introduction

Online retailers become more and more for the past decade. Where recommendation for certain products become a huge concern to online retailers, such as Amazon. We can also consider the products relationship as a significant network as people intend to purchase similar products following one another choice. From that aspect, community detection occurs more than just an algorithm but a way to recommend product efficiently. Detecting co-purchasing products network, therefore, brings great help to online retailer. As community detection algorithm plays more than just a key role in finding potential communities in a network, our team comes up with an idea of combining machine learning method which could make the whole process automatically. Before we move into the step of arising machine learning methods, we intended to do comparison on five iconic community detection algorithms and analyze on the resulting value of modularity score and run time. Such result reasoning from the restriction of how machine learning algorithms works. To be more precise, if one algorithm has time complexity $O(n^2)$ than the total time complexity will brings up to more than $O(n^3)$ or more especially with the effect of machine learning algorithm. Therefore, finding a relatively efficient community detection algorithm become a crucial point to our entire idea. Furthermore, that leads to the reason we want to analyze on the run time and modularity score of the five algorithms.

Problem Definition

The background problem arouses with the exploration to specific datasets, Amazon co-purchasing network, followed with the interesting article “The Long Tail” written by C. Anderson. The article points out the key difference that causes particular

product from been forgotten to be been demanded. Basic idea for the article is that, back in the era where we have the Blockbuster selling CDs and DVDs entity particular product might be forgotten while it is not the "famous" product. On the contrary, in the era of online shopping, every products are in need based on each persons' interest. Then a new article comes in front of our eyes, the "Qualitative Comparison of Community Detection Algorithms". We, thus, comes up with the idea; instead of having one search wanted produce, why could not one recommend product just like how advertisement showed on a random Facebook page. Combining these background study, we therefore intended to have a full analysis on the five community detection algorithms.

Proposed Algorithms

Our proposal proposed 3 algorithms. The Louvain, the Infoamp, and the fastgreedy. However, we find fastgreedy to be similar to Louvain. Thus, we move on and account the label propagation instead. However, we later joined the expeirment with two more iconic algorithms – the Neman's and the Leiden. Which will be revealed in the next section.

Info map:

In our case, the project proposal proposed 3 different algorithms, the first is Info map, it is also called map equation. Info map algorithm tries to minimize a cost function. Partitioning is based on the flow induced (by the pattern of connections in a given network, it uses Huffman code, random walk. Info map can run on unweighted or weighted standards, multi-layer, memory and sparse memory networks and undirected or directed links and identify two or more levels without overlapping or overlapping clusters. [1]. The disadvantage is accuracy drop on Large (>1000 nodes) networks.

Louvain:

The Louvain method for community detection is a method to extract communities from large networks. which is popular due to its good efficiency and stability, the method is a greedy optimization method that appears to run in. The algorithm has three disadvantages, first is the accuracy of community division has limitations. The second is the size of the cell distribution density within the group will affect the identification of subgroups. The last is within the cell population identified as the same subpopulation, there are two small subpopulations that are not connected.

Label Propagation:

The label propagation is a semi-supervised machine learning algorithm, the algorithm

is a local community partition algorithm based on label propagation. The Label Propagation Algorithm (LPA) is an iterative algorithm in which we assign labels to unmarked points by propagating labels in a data set. The core idea of the label propagation algorithm is similar data should have the same label. In our project, there are many different labels, we can assume if customer interested in Cola, they can be connected, based on the suppose. If two people are connected, it means that the two people are likely to have the same interests.

Measures

Modularity

Modularity is a system property to measures the degree of division of a network into modules, also called clusters. A network with a high degree of modularity has dense connections between nodes in a module, but sparse connections between nodes in different modules. Modularity is often used in optimization methods to detect the community structure in the network.

Modularity Score

The modularity score of the graph is the sum of the number of edges in the cluster in all clusters minus the number of edges encountered by chance in the cluster. In the project, for a given different graph Amazon_0302, Amazon_0312, Amazon_0505, Amazon _0601, each partition of the vertices has modularity score, the higher score indicating that the partition has better connection with the community structure in the given graph. Which according to our original idea of bringing in machine learning method, this should take good care and observation.

Run time

As our original idea of bringing out the machine learning method, the algorithms efficiency should definitely take into account. Run time can be more intuitively express the computational efficiency of different algorithm, the smaller run time meaning the higher efficiency. Same rule apply to the opposite performance.

Implementation and Analysis

The project using the dataset collected by crawling Amazon website. We choose four different time data as sample data for this project. The first dataset contains the data collected on March 2nd, 2003, including 262,111nodes and 1234,877 edges, Some statistics of this dataset is presented in Table I. The Second dataset contains the data

collected on March 12th, 2003, including 400,727 nodes and 3200,440 edges, some statistics of this dataset is presented in Table II. The Third dataset contains the data collected on May 5th, 2003, including 410,236 nodes and 3356,824 edges, some statistics of this dataset is presented in Table III. The last dataset contains the data collected on June 1st , 2003, including 403,394 nodes and 3387,388 edges, some statics of this dataset is presented in Table IV.

Dataset statistics	
Nodes	262111
Edges	1234877
Nodes in largest WCC	262111 (1.000)
Edges in largest WCC	1234877 (1.000)
Nodes in largest SCC	241761 (0.922)
Edges in largest SCC	1131217 (0.916)
Average clustering coefficient	0.4198
Number of triangles	717719
Fraction of closed triangles	0.09339
Diameter (longest shortest path)	32
90-percentile effective diameter	11

TABLE I. Dataset Statistics

Dataset statistics	
Nodes	400727
Edges	3200440
Nodes in largest WCC	400727 (1.000)
Edges in largest WCC	3200440 (1.000)
Nodes in largest SCC	380167 (0.949)
Edges in largest SCC	3069889 (0.959)
Average clustering coefficient	0.4022
Number of triangles	3686467
Fraction of closed triangles	0.05991
Diameter (longest shortest path)	18
90-percentile effective diameter	7.6

TABLE II. Dataset Statistics

Dataset statistics	
Nodes	410236
Edges	3356824
Nodes in largest WCC	410236 (1.000)
Edges in largest WCC	3356824 (1.000)
Nodes in largest SCC	390304 (0.951)
Edges in largest SCC	3255816 (0.970)
Average clustering coefficient	0.4064
Number of triangles	3951063
Fraction of closed triangles	0.06068
Diameter (longest shortest path)	20
90-percentile effective diameter	7.6

TABLE III. Dataset Statistics

Dataset statistics	
Nodes	403394
Edges	3387388
Nodes in largest WCC	403364 (1.000)
Edges in largest WCC	3387224 (1.000)
Nodes in largest SCC	395234 (0.980)
Edges in largest SCC	3301092 (0.975)
Average clustering coefficient	0.4177
Number of triangles	3986507
Fraction of closed triangles	0.06206
Diameter (longest shortest path)	21
90-percentile effective diameter	7.6

TABLE IV. Dataset Statistics

During implementation of this project, our group came up with two hypotheses. First is that the size of a graph influence run time and score. We utilize four different datasets with different size. The idea of this is to observe different data size affect running time and modularity score. Second is that the score of “Label Propagation” may increase after several epochs. In order to verify our conjecture, we indeed try increasing epoch while running this algorithm. However, as the studies goes on, we found that the characteristic of the label propagation make this hypotheses ending up fault. Since once an epoch start, the label propagation initiate starting value. Thus, results will be different to one another.

```
Algorithm 1 Infomap
1       Infomap_start = timeit.default_timer()
2       network_cluster = network.community_infomap()
3       network_infomap = network.modularity(network_cluster)
4       Infomap_stop = timeit.default_timer()
5       runtime_Info = Infomap_stop - Infomap_start

Algorithm 2 louvain
1       Louvain_start = timeit.default_timer()
2       network_cluster = network.community_multilevel()
3       network_multilevel = network.modularity(network_cluster)
4       Louvain_stop = timeit.default_timer()
5       runtime_Louvain = Louvain_stop - Louvain_start

Algorithm 3 Propotation
1       Prop_start = timeit.default_timer()
2       network_cluster = network.community_label_propagation()
3         network_propagation = network.modularity(network_cluster)
4         Prop_stop = timeit.default_timer()
5         runtime_Prop = Prop_stop - Prop_start
```

After test, we found the first hypotheses is right. Almost every method was influenced by the size of the network. By contrast, the second hypothesis resulting negatively. The conclusion is that each time label propagation starts with a new random walk and initiate value. Therefore, the score will never be affected by the previous result. Considering the disadvantage of the Louvain method, the cell population identified as the same subpopulation, the two subpopulations cannot connect. Therefore, we brought up new algorithm – The Leiden algorithm.

Contrary to Louvain, Leiden's algorithm can split clusters instead of just merging clusters like Louvain's algorithm. By splitting the clusters in a specific way, the Leiden algorithm can ensure a good connection between the clusters. Moreover, the algorithm guarantees more than this. That is if we run the algorithm repeatedly, we eventually obtain clusters that are subset optimal. This means that it is impossible to improve the quality of the clusters by moving one or more nodes from one cluster to another.[2] Also, we compare the Newman’s Eigenvector algorithm to generalize the test. The core idea of the Newman’s Eigenvector algorithm is community structure detecting based on the leading eigenvector of the community matrix. This function tries to find densely connected subgraphs in a graph by calculating the leading non-negative eigenvector of

the modularity matrix. In the project, an amazon network G with n nodes can be expressed as an adjacency matrix A of n x n. On this matrix, if there is an edge between nodes i and j, then Aij=1, otherwise it is 0. We calculate the run time and score of two algorithms.

Algorithm 4 Leiden

```
1        Leiden_start = timeit.default_timer()
2        network_cluster = network.community_leiden()
3        network_leiden = network.modularity(network_cluster)
4        Leiden_stop = timeit.default_timer()
5        runtime_Leiden = Leiden_stop - Leiden_star
```

Algorithm 5 Newman's Eigenvector

```
1        Newman_start = timeit.default_timer()
2        network_cluster = network.community_leading_eigenvector()
3        network_newman = network.modularity(network_cluster
4        Newman_stop = timeit.default_timer()
5        runtime_Newman = Newman_stop - Newman_start
```

Results and Discussion

We will analysis the result of Amazon datasets from five community detection algorithms in this section. For the results, we focus on running time and modularity score of five algorithms with four Amazon datasets, and finding which algorithm is the best choice to apply in product recommendation with machine learning method. Since machine learning method is not main topic of this project, I will not explain too much. These results all calculated by the packages networkx, in Python.

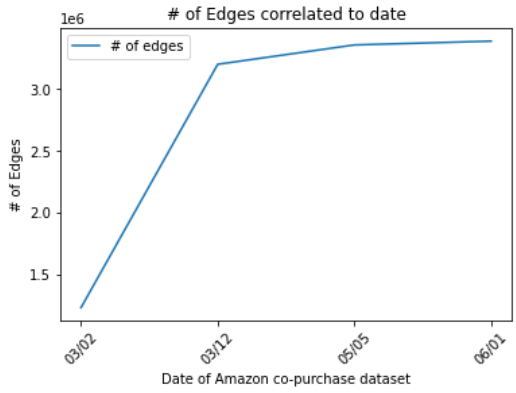


Fig. Edge numbers

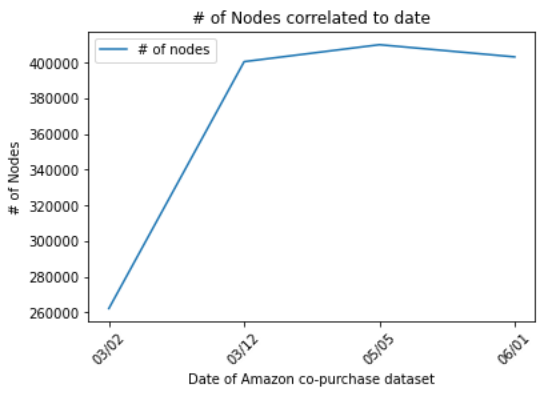


Fig. Node numbers

The above two figures show the numbers of edge and node of four Amazon datasets (ordering with date). From the two figures, we can observe that 03/02 dataset has the lowest numbers of node and edge in four datasets; 05/05 dataset has the largest numbers of node, and 06/01 has the largest numbers of edge. Then we can know 03/02 dataset has huge different of numbers of node and edge with other datasets, other three datasets has slight different.

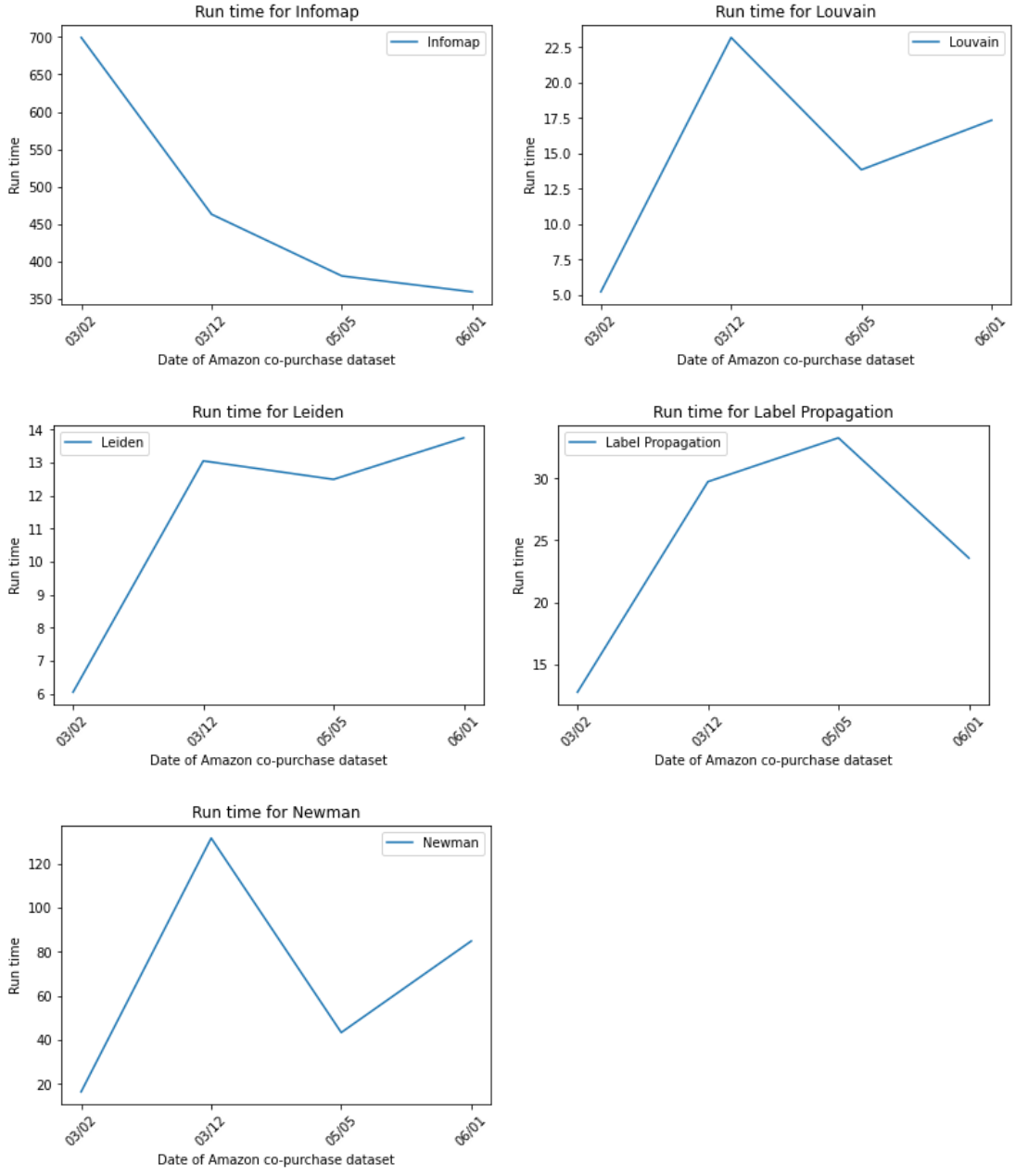


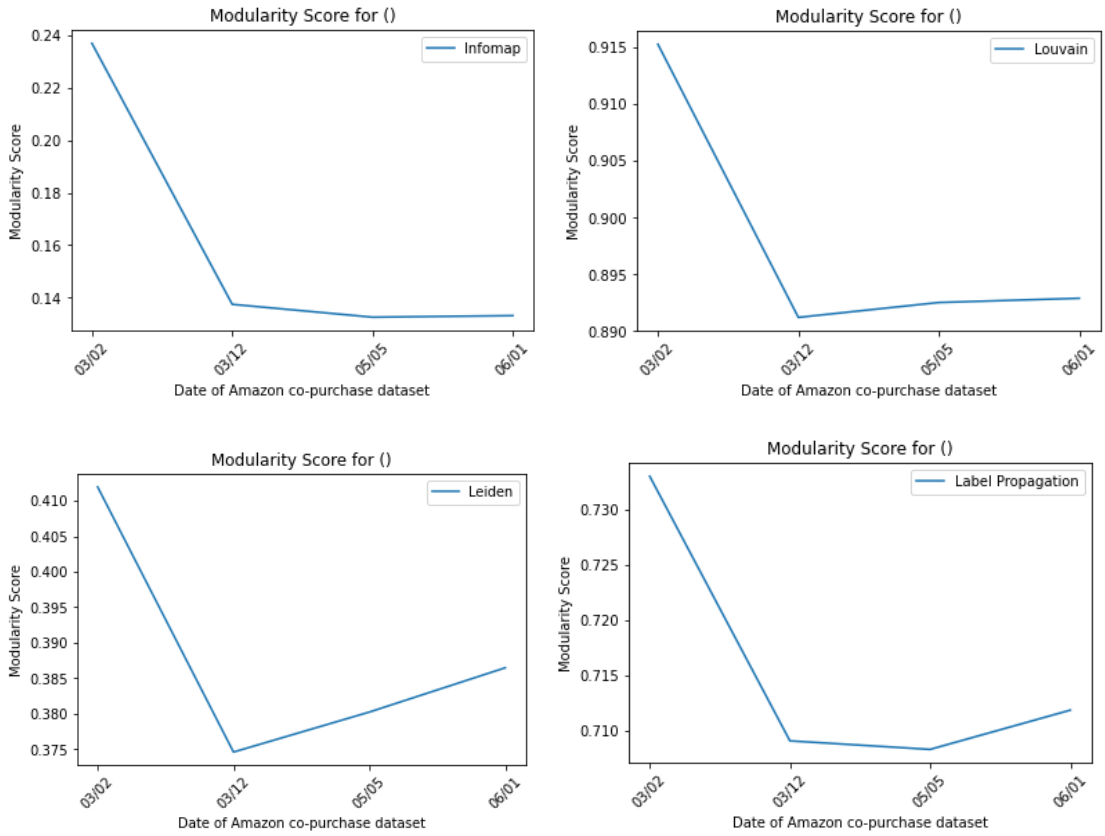
Fig. Running time of five community detection algorithms

The five figures above show the running time result curve of five algorithms in each Amazon datasets. Based on running time, we can know which algorithm has better efficiency. We can observe that except Infomap algorithm, other four algorithms have

similar running time result curve, and all 03/02 dataset has the lowest running time, Louvain is 5 seconds, Leiden is around 6 seconds, Label Propagation is around 12 seconds and Newman is 18 seconds.

In Infomap figure, shows 03/02 dataset has the lowest edge and node numbers but has the highest running time, up to 700 seconds; and running time gradually decrease with higher edge and node numbers. 06/01 dataset has the lowest running time, but still has 360 seconds. In Louvain and Newman figure, 03/12 dataset has the highest running time, Louvain is 23 seconds and Newman is 130 seconds. In Label Propagation figure, 05/05 dataset has highest running time 35 seconds. In Leiden figure, 06/01 dataset has highest running time 14 seconds.

Based on these five algorithms running time result. Louvain, Leiden and Label propagation have relatively low running time. Leiden has best efficiency, average running time only has 11.34 seconds. Infomap and Newman have quiet high running time. Average running time of Infomap is 457.64 seconds, Newman is 69.11 seconds. It is clearly shows that Infomap has too low efficiency and not suit for analyzing these Amazon datasets.



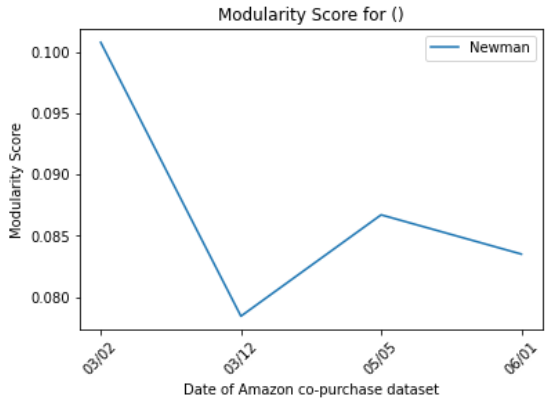


Fig. Modularity Score of five community detection algorithms

The five figures above show the modularity score result of five algorithms in each Amazon datasets. These five algorithms have similar modularity score curve, all algorithms' 03/02 dataset have the highest modularity score. Infomap highest modularity score is 0.23, the lowest is 06/01 dataset 0.12. Louvain highest modularity score is 0.916, the lowest is 03/12 dataset 0.891. Leiden highest modularity score is 0.413, the lowest is 03/12 dataset 0.375. Label Propagation highest modularity score is 0.736, the lowest is 05/05 dataset 0.695. Newman highest modularity score is 0.103, the lowest is 3/12 dataset 0.076.

These modularity score figures show Amazon datasets in Louvain and Label Propagation algorithm have strongly connection in community; Amazon datasets in Infomap, Leiden and Newman have relatively weak connection in community. Louvain and Label Propagation algorithm are suit for analysis these Amazon datasets. The further result discussion will in next section.

*average	Infomap	Louvain	Leiden	Label Propagation	Newman
Run time	457.74	14.92	11.34	24.83	69.11
Modularity Score	0.16	0.8975	0.38825	0.7165	0.087

Fig. average value of run time and modularity score with five Amazon datasets

The form above shows the average value of give algorithms' running time and modularity score. Based on the form, we can easily to observe that Infomap and Newman algorithm have very high running time and low modularity score, these factors will cause high time complexity and weak community connection. If combine machine learning method with Infomap and Newman algorithm, the analysis efficiency will become very low and hard to precisely recommend suitable products to customers, since low modularity score. Infomap and Newman algorithm are not suit for applying

in products recommendation.

Louvain algorithm has very low running time and high modularity score, it means low time complexity and high community connection. Louvain algorithm combine with machine learning method will have very high efficiency and easily to precisely recommend suitable products to customers, user can quickly receive precisely analysis result. Online retailers would apply Louvain algorithm to improve their business.

About Leiden and Label Propagation algorithm, even Leiden has the shortest running time, Leiden's modularity score is not good enough to easily to get precisely product recommendation result. Label Propagation algorithm's running time and modularity both worse than Louvain algorithm. So Label Propagation algorithm is not the first priority.

In the result of algorithms' comparison, Louvain algorithm has best efficiency that can receive high precisely analysis of product recommendation in shortest time with machine learning method. The result is achieve our goal that fully analysis five algorithms and find which algorithm is the most suitable applied in product recommendation with Amazon datasets. However, our running time and modularity score results cannot prove other algorithms are not good, the results just show other algorithms not suit for analyzing Amazon datasets, like Infomap algorithm would has best performance to analyze other kinds of dataset.

Related Work

We review several similar works that either compare the community detection algorithms or focus on more deeper analyzation of the algorithms. One article arise our curiosity and stimulate the idea that force this research. **"A Comparative Analysis of Community Detection Algorithms on Artificial Networks"[3]** written by Z. Yang et al. did a comprehensive research and comparison on multiple known community detection algorithms by randomly creating artificial networks. To be more precise, they establish networks with a mixture of different parameters and tested out with the algorithms. Further, analyze whether an algorithm should better fit what kind of network. Z.Yang et al. claim that multilevel Louvain fit networks that includes more nodes (nodes>1000), but when it comes to over 6000 nodes both multilevel Louvain and the Infomap is not suitable. By contrast, we still stick to the plan of testing out the Louvain and Infomap on the Amazon co-purchasing networks, which includes nodes over 400 thousand. The result came out a bit different as the article expected. The Infomap algorithm did perform as their research found out to be, however, the Louvain did not. Louvain, in this case, outperform every other algorithm in both run time and modularity score which is pretty surprising. Our team dig into another same title article

"A comparison of Community Detection Algorithms on Artificial Networks"[4] written earlier by G.K. Orman et al. Clearly that coefficient counts more than we expect in the mixing parameter network. This leads to another assumption of our datasets, that is, the whole network contains relatively low coefficient. The assumption would be account into future work tasks for better combination to machine learning methods. Besides those articles, the article **"From Louvain to Leiden: guaranteeing well-connected communities"[5]** brings our attention with the Leiden algorithm which known as improved version of Louvain. According to the article, the Leiden improve by overcoming the problem of randomly abandon node in the Louvain algorithm and resulting in better modularity score. However, our result performed completely different since we implement multilevel refinement of Louvain. That brings more significant modularity score and outperform Leiden. Nevertheless, Leiden's run time indeed outperform multilevel Louvain which is the same complexity to original Louvain. Other than these algorithms, we also brings Newman's and Label propagation algorithm to account. Both performed as Z. Yang at el. expected to be. In short, most of the algorithms resulting similar to existing research. Although some might seem different, reason can be found. We can, therefore, further this result into our future work and involve machine learning methods with confident.

Conclusion and future work

The five community detection algorithms show different running time (efficiency) and modularity score (connection) range in four Amazon datasets. After compare running time and modularity score results, Louvain algorithm is the best efficiency algorithm to analyze Amazon datasets. Infomap and Newman algorithm have too low efficiency to analysis Amazon datasets.

Based on the finding, our further work will extend to analyze more Amazon datasets with combining Louvain algorithm and machine learning method and even AI technology to achieve higher accuracy of product recommendation. Also apply this method to analysis other recently and larger datasets in different areas to do various applications with five community detection algorithms, such as education, weather forecast or others.

To be more precise of how we plan to combine machine learning method along with the Louvian algorithm, one can look into the K-mean or K-cluster. Such algorithm leads with cluster similar item, which in the case of recommending co-purchasing product would be a better fit. Main idea will comes in the two major steps iteratively along the date: 1) detecting community, 2) machine learning how community changed. By only looking at assumption, this idea could greatly provide help to online retailer. This idea, apparently, will be counted into our future work along the result we conclude

from this project; the Louvain algorithm's performance better enchance the whole process.

References

[1] D. Edler, L. Bohlin, R. Martin, "Mapping Higher-Order Network Flows in Memory and Multilayer Networks with Infomap," *Algorithms*, 10(4), 112, 2007.

[2] V. Traag, L. Waltman, and N.J. van Eck, "Using the Leiden algorithm to find well-connected clusters in networks," CWTS. [Online]. Available: https://www.cwts.nl/blog?article=n-r2u2a4. [Accessed: 03-May-2021].

[3] Z. Yang, R. Algesheimer, and C.J. Tessone, "A Comparative Analysis of Community Detection Algorithms on Artificial Networks," *Science Reports*, July 2016.

[4] G.K. Orman, V. Labatut, "A Comparison of Community Detection Algorithms on Artificial Networks," *Discovery Science*, Porto, Portugal. pp.242- 256, Oct 2009.

[5] V.A. Traag, L. Waltman, And N.J van Eck, "From Louvain to Leiden: guaranteeing well-connected communities," *Science Reports*, March 2009.

[6] L.Tengfei, "Machine Learning - Introduction of Community Detection Algorithms: Infomap". [Online]: https://zhuanlan.zhihu.com/p/53085574

[7] Leading.eigenvector.community: Community structure detecting based on the leading eigenvector of the community matrix. [Online]. https://www.rdocumentation.org/packages/igraph/versions/0.4.1/topics/leading.eigenvector.community

[8] "Label propagation algorithm," Wikipedia, 11-Jul-2020. [Online]. Available: https://en.wikipedia.org/wiki/Label_propagation_algorithm. [Accessed: 03-May-2021].

[9] "Louvain method," Wikipedia, 13-Apr-2021. [Online]. Available: https://en.wikipedia.org/wiki/Louvain_method. [Accessed: 03-May-2021].

[10] V. Traag, L. Waltman, and Nees Jan van Eck, "Using the Leiden algorithm to find well-connected clusters in networks," CWTS. [Online]. Available: https://www.cwts.nl/blog?article=n-r2u2a4. [Accessed: 03-May-2021].

[11] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," Knowledge and Information Systems, vol. 42, no. 1, pp. 181-213, 2015.

[12] S. Rahiminejad, M. R. Maurya, and S. Subramaniam, "Topological and functional comparison of community detection algorithms in biological networks," *BMC* bioinformatics, vol. 20, no. 1, pp. 1-25, 2019.

[13] C. Anderson, "The Long Tail," Wired. [Online]. Available: https://www.wired.com/2004/10/tail/. [Accessed: 03-May-2021].

[14] Martin Rosvall, "Maps of networks," martinrosvall.com, 27-Feb-2020. [Online]. Available: https://www.martinrosvall.com/maps-of-networks.html. [Accessed: 03-May-2021].

[15] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.
[16] M. Rosvall, D. Axelsson, and C. T. Bergstrom, “The map equation,” *The European Physical Journal Special Topics*, vol. 178, no. 1, pp. 13-23, 2009.