Survey of Graph Clustering Algorithms Using Amazon Reviews

Shihui Song SHIHUI@STANFORD.EDU
Jason Zhao JLZHAO@STANFORD.EDU

Abstract

We built a graph clustering system to analyze the different resulting clustering from Amazon's product reviews from the dataset on SNAP. There are over 53MB of reviews, but we started from the graph constructed from the Jewelry set containing 60,000 nodes. Our focus for this graph clustering was the clustering product to ascertain whether the clusters were characterized by individual traits. We evaluated using SNAP's Clauset-Newman-Moore, Spectral Clustering, and Normalized Min-cut Clustering based on betweenness centrality. Our initial results shed light on some promising developments that support our initial hypothesis that clustered products and reviewers have similar attributes.

1. Introduction

Clustering has long been a problem that's interested many mathematicians and computer scientists as a principle of determining similarity between components. Within the recent years though, social networks have presented a new subject for graph clustering application.

1.1. Motivation

As the largest online retailer, Amazon has a unique position as the curator for the biggest set of purchase records as well. Often, buyers have tendencies to purchase products based on some sort of principle. Clustering these purchases and reviewers is one way to discover similarities. Understanding purchase patterns has significant impact on similar item suggestions, which is a crucial component to the Amazon experience.

CS224W Milestone, *Shihui Song, Jason Zhao*

2. Data Set

The data was acquired from the Amazon review data set on SNAP [1] (McAuley & Leskovec, 2013). The entire data set is comprised of over 10 million nodes divided into 26 sets by product categories. Our initial focus is on the jewelry data set with a size of 7.8M and 60,000 nodes. The review in the data set contains information on the product, such as its description and price, as well as the review's feedback like its rating and the comments.

2.1. Pre-Processing

Python was utilized for the pre-processing of the data into adjacency lists. A weighted bigraph is initially created based on the product reviews. Then another two graphs were generated, one for products and the other for reviews. Each node within the product graph represents a product and each link represents a pair of products that a single reviewer has reviewed. Each node within the reviewer graph represents a reviewer and each link represents a pair of reviewers that bought the same product.

2.2. Small Set

The small data set consists only of the `Jewelry.txt.gz` file. The graph statistics are as follows:

	Products	Reviewers
# of nodes	11,007	34,241
# of edges	59,876	915,292
file size (MB)	1	11

2.3. Large Set

The large data set consists of the following datasets:

- `Jewelry.txt.gz`
- `Arts.txt.gz`
- `Musical_Instruments.txt.gz`

[1] http://snap.stanford.edu/data/web-Amazon-links.html

- `Software.txt.gz`
- `Watches.txt.gz`

and the graph statistics are as follows:

	Products	Reviewers
# of nodes	40,180	147,909
# of edges	734,386	25,877,544
file size (MB)	4.3	152

2.4. Data Representation: Adjacency List

There are three main text file representations for a graph: an adjacency matrix, an adjacency list and an edge list. Adjacency matrices take the most space with a number representation every possible edge. We initially started utilizing both types since edge lists were always the format for homeworks. However, edge lists are more wasteful. For every node, it takes up almost twice the space as an adjacency list does since the src node has to be repeated for every edge on a separate line. Therefore, we decided to stick to adjacency lists.

3. Clustering Algorithms

For the project, we evaluated 3 different clustering algorithms and their variations. The algorithms are based on different graph clustering principles and each have their own performance characteristics. In the end, we use choose the best algorithm based on performance and modularity scores to run on the large data set. The 3 algorithms are divided into 2 main classes, hierarchical and spectral. The hierarchical algorithms can be futher divided into agglomerative or divisive. The spectral method project points into k dimensional space using the eigenvectors of the Laplacian space and clusters those points.

3.1. SNAP's Clauset-Newman-Moore

We used SNAP's Clauset-Newman-Moore community detection method. Its principle motivation is to increase modularity through a hierarchical agglomeration algorithm. The CNM algorithm uses a sparse matrix to store the current cluster labels and a max-heap of nodes that are un-clustered and their modularity improvement score. These data structures allows the algorithm to have a run time of $O(md\log(n))$ where m is the number of edges in the graph, n is the number of vertices in the graph and d is the depth of the largest dendrogram of the communities. A dendrogram is the tree generated by hierarchical agglomerative clustering. It represetns the sequence of steps that nodes were added to form the final cluster. In a typical real world network, $m \sim n$ and $d \sim \log(n)$ in which case the algorithm runs in $O(n\log^2(n))$ (Clauset et al., 2004).

3.1.1. Modularity

Modularity is refined by Newman, as the number of edges falling within groups *minus* the number of expected edges in an equivalent network with random placement. Modularity can be formally definition as:

$$Q = \frac{1}{2m}\sum_{ij}(A_{ij} - P_{ij})\delta(C_i, C_j) \tag{1}$$

Where A is the adjacency matrix, P is the expected number of edges between i and j, and $\delta(C_i, C_j) = 1$ if and only i and j are in the same cluster. The def (Newman, 2006) modularity is meant to indicate the goodness of network division. The score can range from $\frac{-1}{2}$ to 1 with a 1 implying the graph has perfect communities and 0 meaning the graph is completely random.

3.2. Spectral Clustering

Spectral clustering uses the spectrum in the eigenvectors of the adjacency matrix to cluster groups of points into different communities. The core idea behind spectral clustering is that the rows of the eigenvectors of the Laplacian matrix for two point are similar if the points are in the same cluster. Naive implementations of spectral clustering computes the all of the eigenvectors for the adjacency matrix which runs in $O(n^3)$ time. However, since we only need the top k eigenvectors, more efficient methods such as power iteration can be used which great improves the run time.

3.2.1. Non-Normalized Spectral Clustering

The basic version of spectral clutering uses the Laplacian matrix defined as follows:

$$L = D - A \tag{2}$$

Where D is the degree matrix of the graph and A is the adjacency matrix. The clustering algorithm takes the largest k vectors of L to form a new matrix U. Given U, the spectral clustering algorithm then runs k-means on the matrix treating each row as a data point and label points into one of the k clusters. We briefly describe the k means algorithm below.

3.2.2. k-Means

k-means is an iterative clustering algorithm that labels points in n dimensional space into a predetermine number of k clusters. It does so by first choosing k

centroids and assigning each cluster to its closest centroid. Closeness is defined by some similarity measure and popular choices are: Euclidean distance, Gaussian distance, and cosine distance. We decided to use the simple Euclidean L_1 norm as the distance measurement for our clustering function. Given a distance measurement, the k means algorithm iterative assigns all points in the data set into their corresponding clusters and compute a new centroid location based on the membership of the current assignment. It then tries to reassign points in the data set to the new centroids until no point changes centroid memberships. This means the algorithm has converged and all points as assigned to their corresponding clusters. The algorithm is reliant on the initial centroid assignments and we decided to use a random assignments by picking points in the data set with uniform probability.

3.2.3. Normalized Spectral Clustering

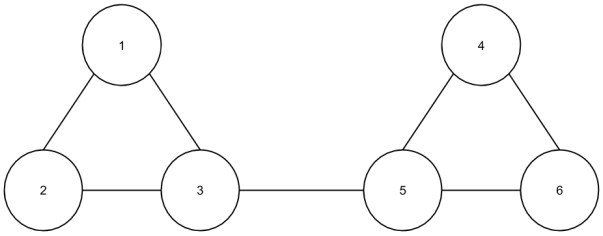
One improvement over the standard spectral clustering algorithm is to normalized the matrices. The normalized version follows a similar structure compared to the non-normalized version but uses the normalized Laplacian which is defined as:

$$L_{sym} = D^{-1/2} L D^{-1/2} \tag{3}$$

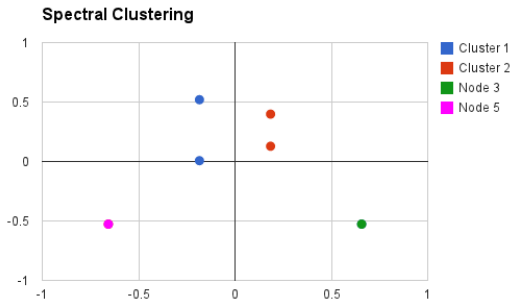
Where L is defined the same as above (von Luxburg, 2007). Furthermore, after the eigenvectors are computed and the matrix U generated, the algorithm normalizes the rows of matrix U to unit length:

$$U'_{ij} = U_{ij} / (\sum_j U_{ij}^2)^{1/2} \tag{4}$$

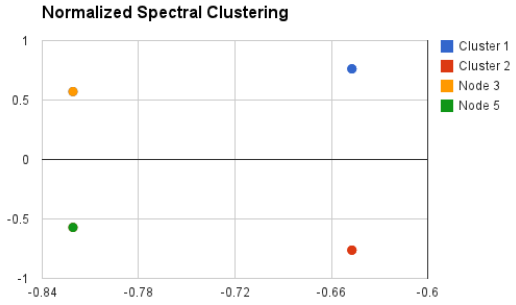
The normalized version of the algorithm performs much better when degree distribution of graph is non-uniform. This is because many clustering objectives involve the node degree since it is used to measure between cluster similarity and within cluster similarity (von Luxburg, 2007). The non-normalized Laplacian does not encode how node degrees interact and loses a lot of this information. The L_{sym} encodes this information by computing $\frac{1}{\sqrt{d_i^2 + d_j^2}}$ for each pair of nodes that share an edge. This leads to eigenvectors that are much more representative of the original clustering. Suppose we are trying to cluster the following graph into 2 clusters:



After computing the eigenvectors, the following graphs show the 2 dimensional points passed in the k-Means algorithm.



The non normalized eigenvectors place node 3 and 5 on the wrong side of the graph which leads to incorrect cluster labels.



The normalized Laplacian eigenvectors for nodes 1 and 2 were identical. This is also true for nodes 4 and 6 which is why there are only 4 nodes in the graph. Since there are 2 nodes stacked at the blue dot and 2 nodes at the red dot, the normalized algorithm computes the correct community labels for the simple graph. This illustrates that for community detection, the normalized spectral clustering algorithm generally performs better.

3.3. Max-flow/Min-Cut

Our last graph clustering techniques is max-flow/min-cut.

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v) \tag{5}$$

That is, find the min-weight cut to divide a graph into disconnected components. This type of graph clustering heavily utilizes existing max-flow/min-cut algorithms to determine the best divides. However, one common scenario to arise is the division into a singular isolated node and the rest of the graph. The solution to this problem is the normalized cut.

3.3.1. Normalized cut

Instead minimizing the weight of the cut as the optimization of each round, optimize the cut's influence on the two separated components.

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)} \tag{6}$$

$$assoc(A, V) = \sum_{u \in A, t \in V} w(u, t) \tag{7}$$

This reduces the preference to cut small isolated nodes apart because the Ncut value will always be larger since the number of edges connected to small isolated nodes will result in a cut fraction. We are using the original Matlab code shared by the authors for Normalized Cut.(Shi & Malik, 2000)

The algorithm starts by solving for $(D - W)x = \lambda Dx$ for eigenvectors with the smallest eigenvalues. The eigenvector with the second smallest eigenvalue is solution. Then recurse until the number of k clusters is reached.

The eigensolver employed by the algorithm is the Lanczos method, which reduces the runtime of the eigenvector problem. This bottleneck of the entire algorithm comes down to $O(n^{\frac{3}{2}})$(Golub & Loan, 1989).

4. Algorithm Analysis

We chose modularity defined as follows as a measure of cluster quality (Fortunato, 2010).

$$Q = \sum_{c \in C} \left(\frac{l_c}{m} - \frac{d_c}{2m}^2 \right) \tag{8}$$

Where l_c counts the number of edges within the cluster and d_c counts the degree of all the nodes in the cluster. There are several reasons in choosing modularity:

1. It is widely used in existing academic research for community detection
2. It is the objective function for one of our algorithms
3. It satisfies the additive property of a quality function:

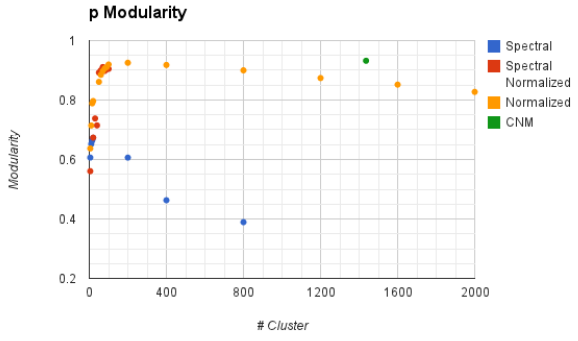
$$Q(P) = \sum_{C} q(C) \tag{9}$$

Where P is a partition of a graph and C are subparitions of P. This property is important as it allows us to compare different community labeling using the same objective function (Fortunato, 2010).

All the analysis below is done using only the small data set.

4.1. Parameter Selection

One major problem with both spectral clustering and normalized cut is that they take in the number of clusters k as an input to the algorithm. This is problematic when we do not know the actual k value that we wish to detect. Therefore we used a common tuning technique which is to maximize modularity using a variety of k values. The graph below shows the results of our experiments:



The CNM solves for modularity, therefore needs no tuning. Notice that the tuning algorithms all have decreasing modularity after reaching the max. This denotes that there are increasing number of isolated components, which does not bode well in the modularity computation. Both Spectral and Spectral Normalized had difficulty generating eigenvectors beyond 800 and 100 respectively. But we can see from the graphs, that the modularities are already starting to decrease at those max clusters after reaching their peaks earlier.

The sparse eigenvector calculation by Matlab is great for sparse matrices like the original Spectral method. However, when the matrix is condense as in the Spectral Normalized, the optimal eigensolver in Matlab degrades into the original $O(n^3)$ runtime for solving eigenvectors.

4.2. Memory Requirement

An often overlooked attribute of many clustering algorithm is the memory footprint. The theoretical bound on the memory requirement for each algorithm is the same and it is $O(n^2)$ if using an adjacency matrix or $O(nm)$ when using an adjacency list. This does not reflect the auxiliary data structures used in the algorithm which can sometimes double or triple the memory requirements. For example, CNM uses a max heap which takes and additional n space. In practice, we noticed that spectral clustering tends to run out of memory when we have around 30000 nodes. After doing some investigations, we were using a normal matrix which is n^2 and each integer takes up 4 bytes of storage, this comes out to almost exactly 4GB which is the amount of memory we have on the corn machines. We tried reducing storage by using sparse matrix representations but the performance drastically deteriorates when there are more than 100000 edges. We noticed that CNM is the most memory efficient algorithm out of all the algorithms that we tested.

4.3. Modularity Scores

	Max Modularity	k
Spectral	0.6727	20
Spectral Normalized	0.9110	70
Normalized Cut	0.9254	200
CNM	0.9103	1436

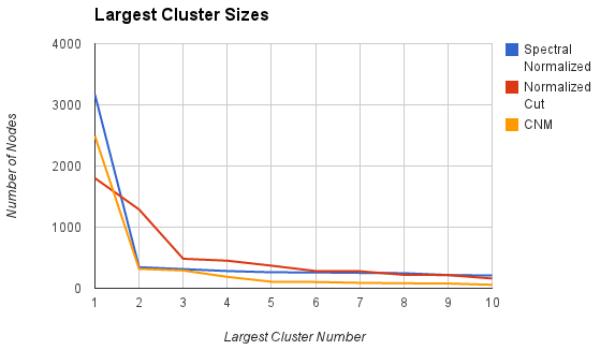
The high modularities for spectral normalized, normal cut, and CNM are good indications of the communities found in the graph. There exists inherent clusters that are easily discoverable and differentiable in the products graph for Jewelry.

5. Amazon Data Set Communities

We evaluated our algorithms on two difference sized data sets described from section 2. Each algorithm generated slightly different clusters which led to interesting insights into the shopping network for Amazon. For algorithms that require the number of clusters as input, we used the k value that resulted in optimal modularity as computed in the above section.

5.1. Small Data Set - Jewelry

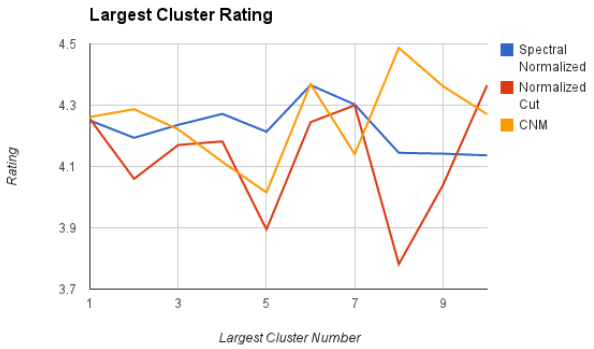
All of our algorithms were able to detect communities for the product graph for the small data set. After clusters were detected, we computed the statistics for many different features on the online review in hopes to detect what products were often bought together.



The above graph shows the size of the communities detected by our algorithms. As you can see, every algorithm detected a large cluster that is approximately $20-30\%$ of the entire network. This number rapidly drops after the first 10 clusters so we will focus most of our analysis on the largest 10 clusters in the network since they encompass most of the nodes. The sections below outline a list of statistics that we computed and examined.

5.1.1. Reviews Scores

One hypothesis that we have is that clusters will be centered around review scores. It seems natural that people will most likely buy products that are highly reviewed. The graph below show the correlation between cluster and review score:



The review score is computed by averaging all the ratings for each product in the same cluster. Although the graph did not show a clear trend between reviews and clusters, it is interesting to see that all three algorithms computed the identical rating score for the largest cluster. This seems to imply a rating of around 4.25 is optimal for products being purchased in parallel.

5.1.2. Price

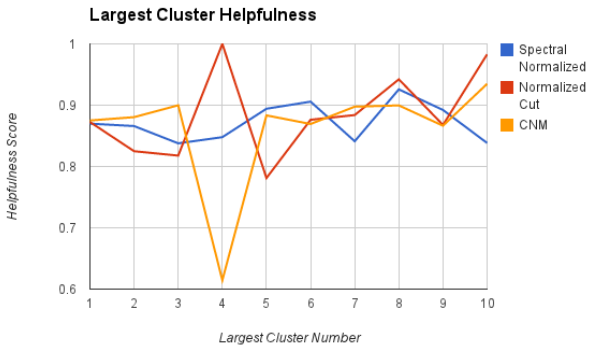
We did not have any initial hypothesis on price and were surprised to see that a relatively high price in the largest cluster. The prices for the first 4 clusters sorted by size were:

	Spectral Normalized	Normalized Cut	CNM
1	45.59	48.86	46.91
2	42.53	70.96	37.29
3	286.90	48.75	22.54
4	30.00	640	27.82

Similar to rating, the largest cluster had a very similar price point of around 46 dollars. The trend differed for each algorithm with some going up and some going down. However, a general sign is that the price tends to be pretty low compared to the average price over the entire data set which is 67 dollars.

5.1.3. Helpfulness

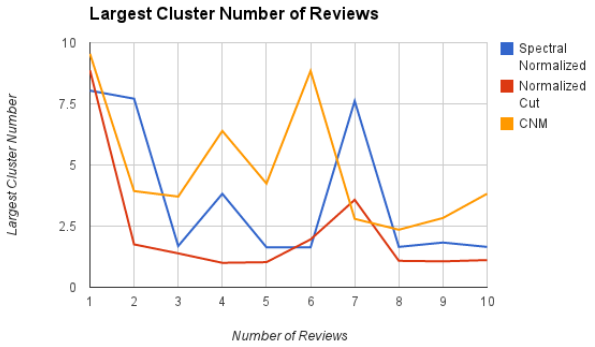
Users were also able to rate each review's helpfulness so we wanted testing to see if products with helpful reviews tend to be bought more often with other products.



Helpfulness is given as a fraction in the data set and we simply pruned out reviews with no score. It is interesting to see that aside from cluster 4 in CNM, all other algorithms picked very helpful clusters with scores generally greater than 0.8.

5.1.4. Number of Reviews

The last statistic that we tested was the number of reviews for all the products in each cluster. Although higher number of reviews mean the product will have more out going edges, the following graph shows some interesting trends.



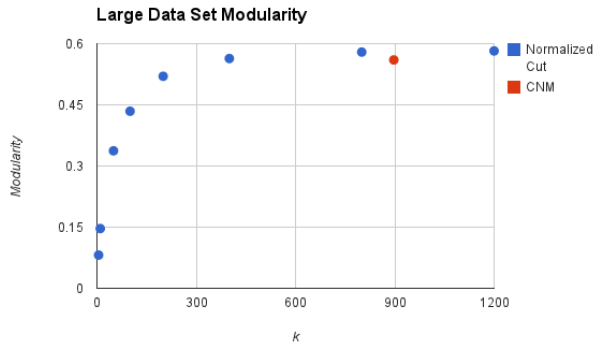
As one can see, the largest cluster has the highest number of reviews at around 10 which is not surprising on its own. This is because the number of reviews directly correlates with the number of edges the node has in our graph and the large degree nodes are more likely to be in one cluster. The interesting part is that there is a spike in cluster $6-7$ which suggests that the algorithm did in fact detect some underlying relationship between number of reviews and the other factors that causes certain nodes to be in one community that is independent of the number of reviews themselves.

5.1.5. Summary

We were able to detect interesting communities with unique characteristics in the small data set. For example, if a product has a price of 46 dollars, has around 10 reviews that are around 0.87 helpful, then it is more likely to be purchased since it is in a community with many other products. These insights can allow Amazon to generate smarter recommendation engines or for a seller to strategically promote his or her product.

5.2. Large Data Set

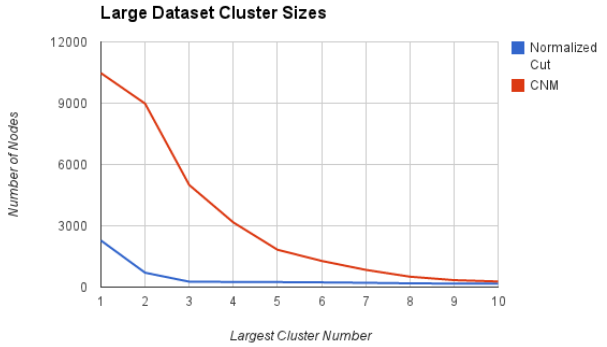
The only two algorithm that were successful at calculating the modularity were Normalized Cut and CNM. The k vs sizes for the clusters are as below:



Notice that Normalized Cut actually beats the CNM modularity as k increases. This was very unexpected since CNM actually optimizes for modularity. However, CNM utilizes a greedy algorithm, so it could have only ventured into a local maximum. Whereas because we have to tune for k in Normalized Cut, then it's easy to search for the global maximum.

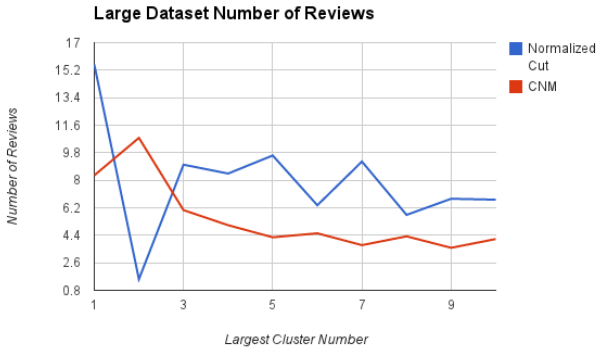
5.2.1. Cluster Sizes

One of the biggest discrepancies between this large data set and the Jewelry only small data set is the number of nodes to each of the top clusters for Normalized Cut. Instead of a big top cluster for Normalized Cut, its biggest cluster size is only 20% of the top cluster size for CNM. This more finely divided clustering is probably the reason for the higher modularity for Normalized Cut.



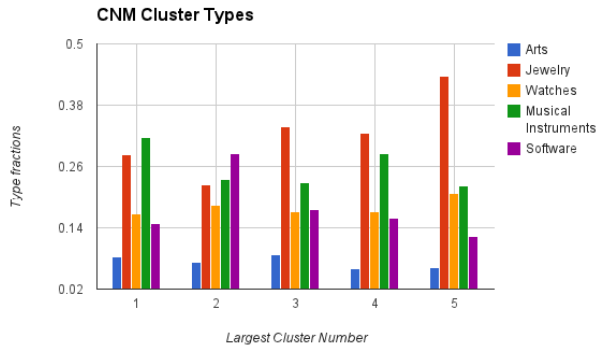
5.2.2. Number of Ratings

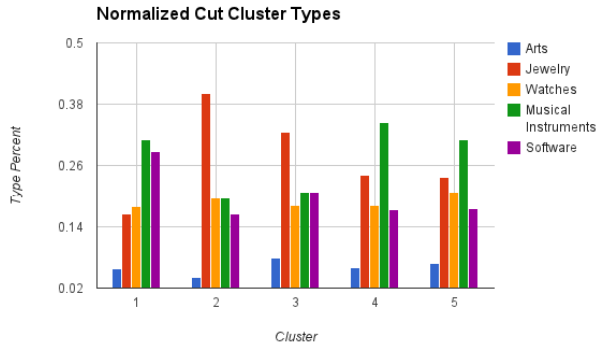
Similar to the small data set, the most distinguishable clustering characteristic is again the number of ratings for the clusters. Specifically, the largest datasets have the highest number of ratings and all the other data sets generally decrease. This again demonstrates that people are more willing to buy reputable products.



5.2.3. Types of Products

One of the reason we wanted to test the big data set was to analyze our clustering algorithms' intuitions on cross-product data sets to see how they are clustered. We were under the impression that the products in the same product group would be naturally clustered together, ie , Jewelry products would cluster with mostly other Jewelry product. The product type distributions is shown below for CNM and Normalized Cut respectively.





As we can see, instead of products clustering to the same type, the product representation is actually instead proportional to the actual sizes of the individual product datasets. One reason for this could be that a single reviewer has purchases span across multiple categories.

6. Conclusion

In this report, we tested a variety of graph community detection algorithms and evaluated many different performance characteristics. We were able to run four algorithms on two datasets and were able to detect community structures with very interesting results.

6.1. Algorithms

The four algorithm we chose were CNM, Spectral, Spectral Normalized, and Normalized Cut for their variety and representation of the spectrum of clustering techniques. Although the theoretical runtimes of the algorithms are all similar, CNM generally outperformed the other algorithms. This is especially compounded by the tunning necessary for the other two clustering techniques. In addition, many of the eigenvector-based algorithms had memory constraints due to the sparsity of the matrices whereas CNM was able to scale better. The modularity score was the highest for CNM in the small dataset, which corresponds to the optimization of CNM on modularity. But the Spectral Normalized and Normalized Cut also had very similar scores. And Normalized Cut was actually able to beat CNM on modularity for the large data set.

6.2. Amazon Reviews

To understand the various clustering techniques better, we tested on two differently sized data sets. Both datasets had a single cluster with an extremely high number of nodes while the rest trailed behind. The biggest cluster for the Jewelry only small dataset was a standout with an average price of $49 and averaged 10 reviews per product. The biggest cluster for the large dataset also distinguished itself by having a high review count as well. Understanding clustering and similar purchases through these metrics allows Amazon to create a sophisticated recommendation system and advertisers to maximize profits.

7. Individual Contributions

Coding was divided by strengths. Jason was more adept at Matlab and thus was responsible for the Spectral, Spectral Normalized and overall Matlab environment. Sweet created the graph generation and computations for cluster analysis as well as Normalized Cut.

The report was very evenly divided. The introductions, conclusions, and large data set analysis were written by Sweet whereas the algorithms and small dataset analysis were written by Jason.

References

Clauset, Aaron, Newman, M. E. J., and Moor, Cristopher. Finding community structure in very large networks. *Physical Review E.*, 2004.

Dhillon, Inderjit S., Guan, Yuqiang, and Kulis, Brian. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(11):1944–1957, November 2007. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1115. URL http://dx.doi.org/10.1109/TPAMI.2007.1115.

Fortunato, Santo. Community detection in graphs. *Physics Reports*, 486(35):75 – 174, 2010. ISSN 0370-1573. doi: http://dx.doi.org/10.1016/j.physrep.2009.11.002. URL http://www.sciencedirect.com/science/article/pii/S0370157309002841.

Girvan, M and Newman, M. E. J. Community structure in social and biological networks. *Proceedings*

of the National Academy of Sciences of the United States, 2001.

Golub, G.H. and Loan, C.F. Van. Matrix computations. *John Hopkins Press*, 1989.

Leskovec, Jure, Lang, Kevin J., Dasgupta, Anirban, and Mahoney, Michael W. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pp. 695–704, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-085-2. doi: 10.1145/1367497.1367591. URL http://doi.acm.org/10.1145/1367497.1367591.

Leskovec, Jure, Lang, Kevin J., and Mahoney, Michael. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pp. 631–640, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772755. URL http://doi.acm.org/10.1145/1772690.1772755.

McAuley, Julian and Leskovec, Jure. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pp. 165–172, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2409-0. doi: 10.1145/2507157.2507163. URL http://doi.acm.org/10.1145/2507157.2507163.

Mishra, Nina, Schreiber, Robert, Stanton, Isabelle, and Tarjan, Robert E. Clustering social networks. In *Proceedings of the 5th International Conference on Algorithms and Models for the Web-graph*, WAW'07, pp. 56–67, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-77003-8, 978-3-540-77003-9. URL http://dl.acm.org/citation.cfm?id=1777879.1777884.

Newman, M. E. J. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States*, 2006.

Shi, Jianbo and Malik, Jitendra. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, August 2000. ISSN 0162-8828. doi: 10.1109/34.868688. URL http://dx.doi.org/10.1109/34.868688.

von Luxburg, Ulrike. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007. ISSN 0960-3174. doi: 10.1007/s11222-007-9033-z. URL http://dx.doi.org/10.1007/s11222-007-9033-z.