Community Detection on Last.fm Artist Data
CS224W Project Report

Ryan Voldstad   SUNet ID: ryanv88

December 10, 2014

1 Introduction

Much like with other media, one of the most exciting results of music consumption transferring into the digital online marketplace is the explosion of user data that is generated as a result. In the tech world, the domain of personalized music recommendation has been developing and expanding rapidly over the past decade. It is most strongly associated with Pandora, which began in 2000 but didn't achieve much popularity until several years later. Since then, dozens of other similar services have been launched with the goal of bringing personalized music recommendations to a user. One such service in particular, Last.fm, provides anonymized data publicly available through a free API; datasets procured through this API are the main inspiration for and subject matter of my project.

In this project I will examine the network properties of various graphs that can be built from the available Last.fm data. Specifically, I consider graphs with artists as nodes, with the overarching question of whether significant community structures form among artists. The Last.fm data naturally takes on a graph form, as user activity involving multiple artists gives rise to an implicit edge of mutual interaction between two artists. I would like to consider the problem of community detection on graphs across a variety of edge formulations. I will evaluate the similarity of the different clusterings produced across formulations, with the hope that there is high similarity according to the 'true' artist communities.

Community detection on this graph is of interest largely for the purpose of 'categorizing' artists based on their respective listening communities. This can be used to inform recommendation systems, and we may suspect it could be more robust than collaborative filtering systems which aim to predict a user's rating of a new artist. This is intuitively an automated process of the same way that traditional radio works; users fall into certain listening communities and subscribe to a station based on their association with these communities. Community detection with respect to artists falls much along the same lines.

2 Related Work

2.1 Previous Work with Last.fm

[6] presents a similar over-arching strategy to that which I will pursue in my paper. A recommender system is built on an explicit partitioning of Last.fm users using a variety of community detection methods, compared against a basic collaborative filtering baseline. The results were very positive, evaluated from the lens of top artist prediction on a per-user basis.

There are a number of ways in which my approach will deviate from [6]. First, as explained above I am interested in community detection among artists rather than users. However, if the hypothesis of music communities holds then there should be a strong correspondence between artist clusters with user clusters. More importantly, then, I will deviate from the authors' examination of explicit friendships that are formed on Last.fm and instead form edges purely based on users' interaction with the artists. Finally, the popular spectral clustering family of techniques was not explored in solving the community detection, which I have chosen to implement.

Much of the inspiration for the graph formulation derives from [5], a student paper from last year's CS224W class which explores decentralized search over artists using Last.fm data. I will use their colistening formulation as a baseline for my graph. In this formulation, edges are added between two artists for all pairs of users who have them in their top-k listens, with edge weight equal to the total number of listens over all such user pairs. I would like to explore normalization of these weights by the total popularity of an artist, as this seems a downfall of the existing formulation; I don't want popularity to be the dominant factor in producing artist clusters. Another major difference in my paper is that they do not explore community detection.

2.2 Spectral Clustering

Spectral clustering algorithms have been commonly adapted for use in community detection in graphs, attractive for both their speed and performance. I draw largely from the surveys presented in [4] and [2] in navigating the literature on spectral clustering and arriving at the formulation used. There are two major contributions worth noting here. First, [1] presents the first use a multiway cut on the eigenvectors of the Laplacian. Second, also in [1], but further tested in [3] and [4], is the use of normalized on the Laplacian matrix prior to decomposition. In my exploration, I found that normalization was indeed necessary to avoid the pitfalls of the traditional method. I ended up implementing the normalization term recommended by [4].

3 Data Collection

A major component of preliminary work on the project has been in data collection. While I began with a substantial dataset from [7], I was inspired by the accessibility of the API and thought it would be fruitful to generate further datasets resulting in additional graph formulations. Since much of my evaluation in this project is intrinsic, my ability to form interesting results will depend largely on comparing several graphs and trying to explain discrepancies.

[7] consists of a random set of about 360K users, with listening counts for each for each of the user's top 20 artists. Due to resource limitations I truncated this to the data for the first 100K users. This resulted in a list of 22175 artists. Additionally, I decided to gather 2 additional data points provided by the API for each artist: top tags for the artist, and Last.fm's "similar artists" list for each. The former represents another user interaction-generated artifact; users may tag artists with either pre-existing tags or create their own free text. The similarity data represents a potential baseline with which to compare communities found: if meaningful communities are found within the artist graphs, then we would expect them to align well with the similarity of artists as evaluated by more sophisticated measures.

To collect tagging and similarity data, I wrote scripts to scrape the Last.fm API for data on each artist in the set of 22175. From the tagging data I extracted the name and count of the top 10 tags associated with each artist, and for the similarity lists I took the top 10 most similar artists for each. This was dumped to file in json format, and in further scripts I cleaned the data into the edge metrics used in the graph models.

I will now go into more detail on the specific models in my exploration.

3.1 Colistening Data

I explore a variant of the edge weights defined in [5]: for each pair of artists that appears in a user's top listened list, add the total listen count between the two to their adjoining edge. However, there are two biases present here that I would like to correct for. First, that of the popularity of an artist. Artists with higher total listen counts will have overly high-weighted edges that bear little significance to the proximity of their neighbors. Next, that of a user's listening habits. Some users have orders of magnitude more listens than others, so their top artists have an overly strong influence on the resulting graph.

To correct for this, I first normalize all user's listening counts such that they sum to 1.

Next, for each pair of artists I propose a metric akin to a sample covariance. For each user list in which both artists i and j appear, add

$$\left(count(i) - \frac{\sum_u count(i)}{N}\right) \cdot \left(count(j) - \frac{\sum_u count(j)}{N}\right)$$

where N is the total number of users and $\sum_u count(i)$ the total count of an artist across all users. Intuitively, this should measure how user listening counts correlate between two artists. Finally, I normalize the resulting weights so that the maximum edge weight is 1.

3.2 Cotagging Data

The raw cotagging data takes on a similar form to colistening data. Since tags are much sparser – there are much more tags used than users in the sample – it makes less sense to consider covariance between two artists' tags. Instead, I simply sum over the product of tag counts for all common tags between two artists, then normalize the resulting weights to have max 1.

3.3 Similarity Data

Here I keep the graph edges unweighted and draw an edge between two artists if either appears in the other's similarity list. This is the result of the fact that no real magnitude of interaction can be drawn from the data. However, we certainly still expect the graph to convey community information through these edges if the lists are to be trusted.

4 Graph Construction

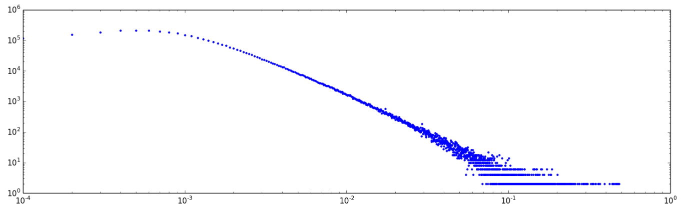
After generating the above edge weight data I proceeded to load graphs into snap and generate some preliminary statistics.

4.1 Colistening Data

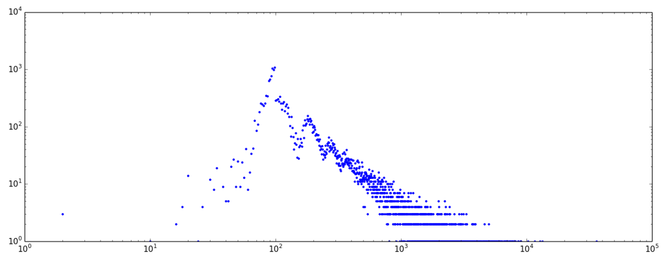
Nodes: 22175
Edges: 3636082

Interestingly, the edge weights follow a power distribution quite nicely. This suggests that some artist pairs have extremely high correlation among listeners, while a very large number have very low correlation, which is a promising result.

However, the plot of node degrees – which we expect to follow a power distribution – is much less well-behaved. The drop-off on the left may be explained by the fact that the data only covers artists in a user's top listens. Artists which have few edges must be unpopular, so any user is unlikely to listen to such an artist very frequently.



(a) Plot of edge weight distribution for Colisten graph



(b) Plot of node degrees for Colisten graph

Average clustering coefficient: .775
This may serve as a preliminary heuristic for whether a good clustering is achievable; if the CC is very low then we can't expect meaningful clusters. So this result is satisfying.

4.2 Cotagging Data

Using all of the top 10 tags for each artist yielded an intractable number of cotag edges; even cutting down to 5 per artist yielded a nearly complete graph. This is due to the fact that there are very common tag: for example, if some tag appears in all top k tags, then the graph is complete.

So, I revised the formulation to normalize against tag frequencies. First sort the user's top tags by their information content, rather than raw count. Then I tried two approaches: first, take the top k for this order; second, take all above a given threshold α. Tuning these parameters to get the desired number of edges yielded $k = 6$, $\alpha = .1$.

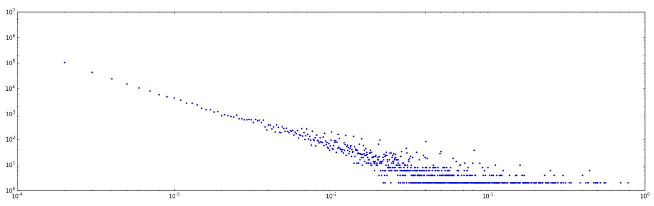
As in the colisten graph, the edge weights follow a power law quite nicely.

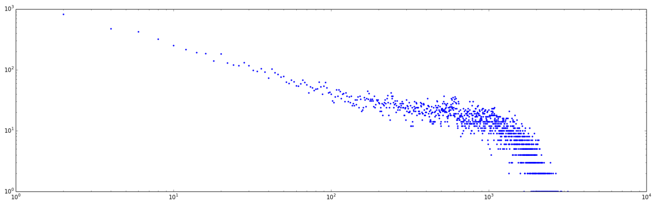
And happily, the degree distribution follows a power law relatively closely, with the exception of the upper tail, which drops off a bit quickly.

For the threshold approach, edge weights were even closer to power law, but degree distribution was much worse. So I went with top-k.

Other graph statistics:

Nodes: 21506
Edges: 5453348
Average clustering coefficient: .563
The CC is not as high as in cotagging, but is still much higher than the baseline $\frac{m}{n^2}$


(a) Edge weights for the top-k approach

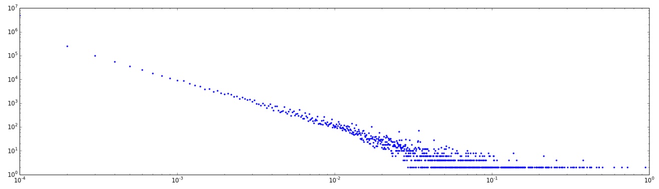

(b) Degree distribution for the top-k approach

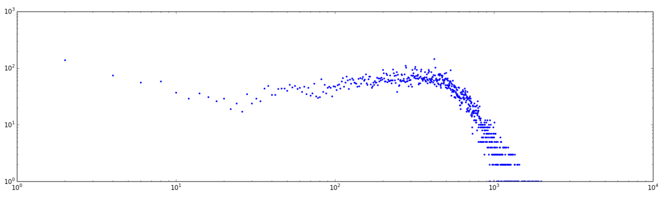
4.3 Similarity Data

Nodes: 21506
Edges: 78396
Average clustering coefficient: 0.344

The graph is deliberately constructed with fewer edges; the top-5 lists are 'gold' in a sense, and have no numerical value, so additional edges may be more likely to generate noise than information. Note that the clustering coefficient is very high relative to the G_{nm} baseline here since the number of edges is so much lower.

Since each artist had exactly 5 similar artists pulled, we don't necessarily expect the degree distribution to follow a power law, and indeed the distribution is closer to linear. Note that the similar artists retrieved are from the entire Last.fm database, not our subset of artists, so the nodes with fewer than 5 edges are valid.


(c) Edge weights for the threshold approach


(d) Degree distribution for the threshold approach

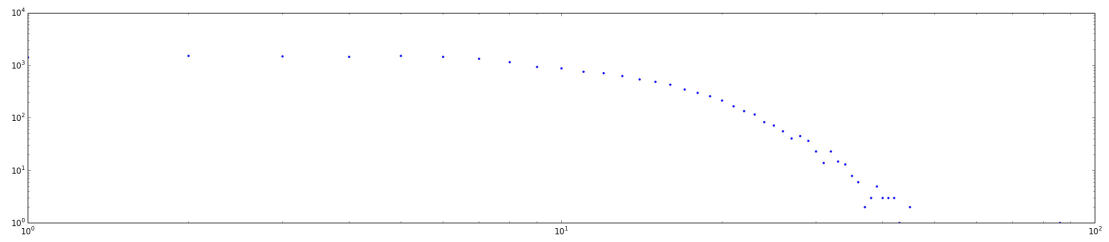


Figure 3: Degree distribution of similarity graph

4.4 Final Words

Connected components: Printing connected component sizes for each graph reveals that the vast majority are in the largest, with the remaining in small isolated components. This is encouraging, as again it follows our assumptions of naturally occurring graphs.

There about 500 missing artists in the raw tagging data for artists that had not received any tags. (The original artist list came from listening data so the colisten graph isn't missing any). As an additional cleanup measure, these were removed from the colistening graph, bringing both to 21506 nodes.

5 Clustering Results

5.1 Implementation Details and First Results

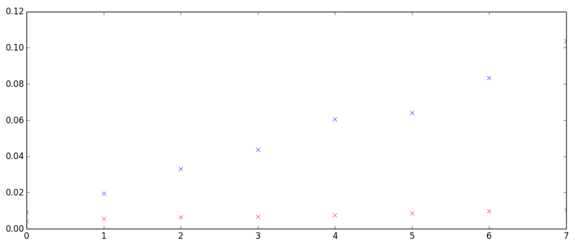
I performed community detection via the multi-cut spectral clustering algorithm described in [1], using k-means to cluster the coordinates of the n smallest non-zero eigenvectors, with $n = 5$. I was worried about performance, given the slow performance of my edge generation steps. So to improve performance, I utilized Matlab's sparse matrix representation in computing the eigenvector decomposition (eigs function) on the Laplacian.

I first tried the cotag graph, treating edges as unweighted when forming the Laplacian. For several k-values tried, as large as 50, this resulted in a single large cluster and other much smaller ones. The large cluster had, at its smallest when $k = 50$, 19676 nodes. Unsurprisingly, this included every node from the 19664 nodes of the largest WCC.

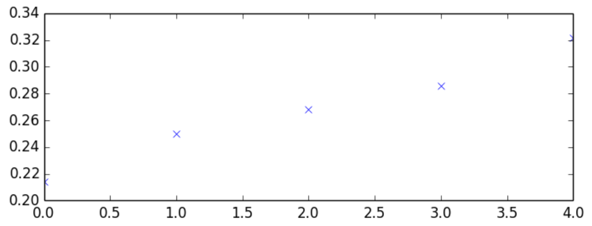
This suggested I need to filter out the small components; the variance of clustering lay entirely in the partitioning/grouping of the isolated components, which is relatively uninteresting. Therefore I removed all nodes that weren't in the first component of any of the 3 graphs. The largest component sizes in the colisten, cotag, and similarity graphs were 21493, 19664, and 17347 respectively, with an intersection of size 16175; I further truncated the graphs to only these nodes.

Unfortunately, preliminary spectral clustering on the similarity graph proved unsuccessful: I was unable to achieve clusterings with any reasonable balance in sizes, and the eigs and kmeans functions were both very unstable. This may have been to the extreme sparsity of edges in the graph; rather than spend the effort to recollect data to add additional edges, I have simply omitted the graph, as there are plenty of results to show with the others.

I then proceeded to compute many different clusterings on the data, using the variables and evaluation methods described in the next section. Plots of the n lowest eigenvalues for each eigen-decomposition are shown below. Additionally, for the remaining results, the Laplacian is amended in the natural way for the weighted graphs: $A_{ij} = \{w_{ij} \forall (i,j) \in E, 0 \text{ elsewhere}\}; D_{ii} = \sum_{j \in n(i)} w_{ij}$. Note that for all graph formulations, edges have the same weight in each direction, so that L remains symmetric as desired.



(a) λ values for $\lambda > 0$: Cotag in blue, Colisten in red



(b) λ values for Similarity graph

5.2 Variables

5.2.1 Parameters

The main parameter at play here is k. I tried k in the neighborhood of $[10, 100]$, based on an intuition for the number of major music genres that might surface from the data. k-means grew unstable (failing to converge) for $k > 16$, and the results below suggest k should not be greater, so the final set used was $\{10, 12, 16\}$.

Further, we may choose different n; the number of smallest non-zero eigenvalues of L used in k-means. As shown in the plot, the eigenvalues are quite closely packed in the lower end, suggesting that higher n may be used to improve stability while approximating minimal $NCut$. Due to resource limitations (on the complexity of k-means), I kept n at 5.

5.2.2 Normalization

Initial attempts at clustering yielded extremely unbalanced cluster sizes (even after removing disconnected components). As described in [4], this may be improved by introducing normalization. The major competing techniques are $L_{rw} = D^{-1}L$ and $L_{rw} = D^{-1/2}LD^{-1/2}$. For brevity, I consider only L_{rw}, as this optimizes for intracluster separation, which should help pull elements away from the central cluster.

5.2.3 Algorithm

As the most fundamental variable, I will compare the results of spectral clustering to standard HAC. That is, I will complement edge weights to form a distance metric across all edges. Additionally I'll need to assign some value to all distances for non-existent edges, say $\alpha \cdot d_{max}$ for some $\alpha > 1$. I chose $\alpha = 2$,arbitrarily. From this this distance matrix, I can implement average-link HAC.

5.3 Evaluation Methodology

5.3.1 Intrinsic Evaluation

I compute the modularity score (for unweighted graphs) detailed in lecture: $Q = \frac{1}{2m} \sum_{s \in S} \left[\sum_{i,j \in s} A_{ij} - \sum_{i,j \in s} \frac{k_i k_j}{2m} \right]$, which we hope to maximize.

However, due to the weighted formulation of L, the linear algebra of the spectral clustering does not minimize conductance (to hence maximize modularity). This can be seen be examining xL^Tx:

$$x^T L x = \sum_i D_{ii} x_i^2 - \sum_{(i,j) \in E} 2 w_{ij} x_i x_j = \sum_{(i,j) \in E} (w_{ij} x_i^2 + w_{ij} x_j^2 - 2 w_{ij} x_i x_j) = \sum_{(i,j) \in E} w_{ij} (x_i - x_j)^2$$

Then adjusting the Rayleigh Theorem formulation shows we approximately minimize then number of edges crossing the cut, weighted by their edge weights (as we'd expect). Then for consistency's sake, we should examine the metric $Q_w = \frac{1}{2m} \sum_{s \in S} \left[\sum_{i,j \in s} w_{ij} - \sum_{i,j \in s} \frac{k_i k_j}{2m} \right]$. I generated a graph according to G_{nm} as a baseline point of comparison for all results.

5.3.2 Extrinsic evaluation

The Adjusted Rand Index is the standard metric used to compare two clusterings. It measures the fraction of pairs which end up in either the same cluster in both, or a different cluster in both, adjusted for chance to lie in the range $(-1, 1)$.

I use this for two goals: (1) evaluating stability of clusterings achieved from a single graph by tweaking parameters. If different n, k, or edge weights drastically affect resulting clusterings then we must be relatively skeptical of the results. (2) If (1) is reasonable, then we may further hope that the different graphs produce similar clusters as well. This more directly tests the hypothesis that significant community structure underlies artist data on Last.fm. Additionally, the Rand score may compare the various spectral clusterings to HAC, as well as to all techniques on G_{mn}.

5.3.3 Anecdotal Results

Finally, I will also evaluate clusterings based on more subjective, anecdotal evidence. First, I ordered all artists by 'popularity' – their total listening counts across all users in the data set. For each clustering, I print out the top 5 artists in each cluster in the hopes that clusters have some genre coherence.

5.4 Main Results

5.4.1 Intrinsic

Graph	k	Q
Colisten	10	0.01556
	12	0.01531
	16	0.01204
Cotag	10	0.00681
	12	0.00666
	16	0.00663
GNM	10	0.0276
	12	0.0245
	16	0.0203

Table 1: Modularity scores

Graph	k	Q_w
Colisten	10	0.1225
	12	0.1253
	16	0.0843
Cotag	10	0.1953
	12	0.2025
	16	0.1986
GNM	10	0.0276
	12	0.0245
	16	0.0203

Table 2: Adjusted modularity scores

Rather dishearteningly, the null model G_{nm} outperforms both other graphs in achieved modularity through spectral clustering. However, remember that modularity is not being optimized for the weighted, but rather the weighted modularity function.

Fortunately, under this metric the real graphs outperform $G_{n,m}$ by as much as a factor of 10. Under the weighted modularity scores, we see that 12 is the highest performing value of k, as opposed to 10 for the unweighted score. Also changed is the relative ordering of Cotag and Colisten; in the weighted case, the Cotag score is about twice as good as Colisten score, while unweighted Cotag clustering performed much worse.

	Colisten$_{SC}$	Colisten$_{HAC}$	Cotag$_{SC}$	Cotag$_{HAC}$	GNM$_{SC}$	GNM$_{HAC}$
Colisten$_{SC}$	1					
Colisten$_{HAC}$	0.03087	1				
Cotag$_{SC}$	$2.346 * 10^{-4}$	$6.144 * 10^{-4}$	1			
Cotag$_{HAC}$	$7.312 * 10^{-5}$	$1.390 * 10^{-3}$	0.0220	1		
GNM$_{SC}$	$-4.775 * 10^{-5}$	$-5.301 * 10^{-4}$	$-4.194 * 10^{-5}$	$9.132 * 10^{-6}$	1	
GNM$_{HAC}$	$-1.727 * 10^{-5}$	$-2.351 * 10^{-4}$	$-6.285 * 10^{-5}$	$5.455 * 10^{-5}$	0.0107	1

Table 3: Rand similarity matrix across clusterings

5.4.2 Extrinsic

ARI for each pair of clusterings with $k = 12$ are shown. On the whole, results are very disappointing (a few details are discussed in conclusion). Informal tweaking of parameters and observation of resulting ARI (omitted) revealed a fair amount of instability, which hints at unmeaningful clusters.

5.4.3 Anecdotal

1	2	3	4	5
pink floyd	nine inch nails	linkin park	the beatles	radiohead
yuki kajiura	red hot chili peppers	system of a down	metallica	queens of the stone age
kanye west	placebo	death cab for cutie	depeche mode	queen
bad religion	bob dylan	jack johnson	garbage	the smashing pumpkins
green day	mogwai	nightwish	in flames	lady gaga

6	7	8	9	10
rammstein	sigur rós	arctic monkeys	coldplay	muse
led zeppelin	oasis	korn	rise against	the killers
nirvana	kings of leon	ac/dc	iron maiden	boards of canada
daft punk	the pillbugs	the cure	the prodigy	johnny cash
tom waits	kent	madonna	fall out boy	sufjan stevens

Table 4: Top 5 artists in each cluster for SC on Colisten graph, $k = 10$

Unfortunately, the groupings produced by one of the top-scoring clusterings appear completely nonsensical (at least to my musical sensibilities).

6 Conclusion

The modularity scores, while not great, were well above the G_{nm} baseline, which indicates the graph formulations managed to produce reasonably 'clustered' data; it was possible to partition the data such that much of the edge weight lay within clusters, with a relatively small number of clusters.

However, both the anecdotal and external evidence point in a more negative direction. While spectral clustering ended up fairly similar to HAC on the same graphs, this is almost a given based on the similarity of the underlying optimization functions. The different graphs appear to share very little structure with one another, only marginally above that with G_{nm}. Finally, the sample results don't lend much optimism to real-world applications of the clusterings; e.g. to artist recommendation. Each cluster seems equally disparate in the artists contained. However, as mentioned throughout the paper, there are many different directions one could take further work in the hopes of revealing more coherent underlying communities.

References

[1] Jianbo Shi and Jitendra Malik *Normalized Cuts and Image Segmentation*. 2000

[2] A. Rajaraman, J. Ullman, and J. Leskovec. *Mining Massive Datasets*. Chapter 10.4 2013

[3] Andrew Ng, Michael I Jordan, and Yair Weiss. *On Spectral Clustering: Analysis and an Algorithm.* 2001

[4] Urlike von Luxburg *A Tutorial on Spectral Clustering*. 2007

[5] Alexander Embiricos, Salman Quazi, and Prasanth Veerina *Hip-Hop to Deep House: Navigating the Music Graph using Decentralized Search*. 2013

[6] The Anh Dang, Emmanuel Viennet *Collaborative Filtering in Social Networks: A Community-based Approach* 2013.

[7] http://www.dtic.upf.edu/ ocelma/MusicRecommendationDataset/index.html