Clustering of Musical Genres

Henry Wallace

November 17, 2015

1 Summary

The aim of this paper is to better understand the landscape of musical genres. We use public tag data (folksonomies) to these discover genres, and make use of APIs from the music site Last.fm gather tag data for songs and artists. To be comprehensive, a uniform sampling of tag documents will be used. API calls are slow, however, while there are about 100 million artists on the database; so we sample with a Markov chain that converges to a unifrom stationary distribution.

With a corpus of tag multisets, each associated with an artist, there are several ways to define a genre. I explore simple co-occurrences, lower-rank representations of tags (Latent Semantic Indexing), and probability distributions of tags (Latent Dirichlet Allocation). For each of these we can define a distance function, respectively: relative frequency, cosine similarity, and Hellinger distance.

Once distances have been defined we can cluster the genres into either flat or nested clustering. For flat clustering, I show how Kohenen's Self Organizing Maps and correlation clustering via genetic algorithms can be used to show spatial relations between genres where K-means fails. To generate a nested clustering I employ hierarchical clustering, with complete linkage.

2 Motivation

The prevalence of online music networks has allowed for listeners to graffiti a vast amount of tag data to artists, albums, and songs. Genres are usually defined in terms by professionals who write for music journals. While there have been sites (e.g. Ishkur's Guide) that paint a picture of genres from an single experts eye, it will be interesting to see what the general listener's view of the musical landscape is, and to see how it stands up to the experts view.

While there has been some past work on generating flat clusterings of musical tag data, there has been little work that explores nested clusterings. It proves interesting to see how the data describes it to be. In addition, once clustering are generated, additional tagging data can be superimposed. For instance, Last.fm provides geographic information as well as its folksonomic tags. Thus, geographic patterns could be easily perceived from the genre clustering.

This provides a great way to show what genres originate from what countries, with little effort scraping for geographic associations.

3 Sampling Tag Data

Data from Last.fm is not publicly available for download in mass. Instead an API is offered that allows, in particular, the queries Artist.getSimilar and Artist.getTopTags. These methods are not described by the API, however, they ostenbily return a list of related artists, and artist tag frequencies, respectively. We access the API via the Python wrapper pylast to more easily manipulate the data. Each API retrieval takes about 1 second; and according to a posted comment by a Last.fm employee on Quora there are about 100 million artists in their database. Thus, downloading all of the data via API retrievals is certainly infeasible.

For the purposes of discovering non-local features for genres, a uniform sample of artists will be collected. To do so we'll employ a random walk on a graph G, whose vertices are all artists on Last.fm's database, and whose edges are enumerated by Artist.getSimilar. We'll assume that G is connected, and undirected. The problem could be trivially solved if there existed a table of merely artist names, from which we could sample at unifrom and then query; or at least a numeric pattern in the website's URLs. Unfortunately, no such table publicly exists, and instead we must recourse to random walk sampling.

The simplest random walk, selects an incident edge at every vertex with equal probability. That is, a Markov chain over the vertices of G with transition probability matrix

$$P_{ij} = \begin{cases} 1/d_i & j \in N(i) \\ 0 & otherwise \end{cases}$$

where $N(i)$ denotes the set of neighbors returned by Arist.getSimilar for an artist vertex i. Such a walk converges to it's stationary distribution after a finite number of steps, where the number of steps it takes to convergence is called its mixing time. And a long enough walk will converge no matter we start. Hence, our choice of an initial artist sample is irrelevant, so as long as we sample long enough. Despite this walk's simplicity, its stationary distirubiton is not uniform for typical networks. In fact, it can be shown that the probability of terminating the walk at a vertex is directly proportional to the degree of the vertex. A finite, irreducible, and aperiodic Markov chain (such as the one above) reaches a stationary distribution whenever $\pi = \pi P$, where π is the distribution of vertices (states). For the simple chain just described, $\pi_i = d_i/2m$ is its

stationary distribution, where m is the number of edges in the graph, as

$$\begin{aligned} \pi P &= \{ \sum_{j \in N(i)} \frac{d_j}{2m} \frac{1}{d_j} \}_{i=1}^{n} \\ &= \{ \frac{d_i}{2m} \}_{i=1}^{n} \\ &= \pi \end{aligned}$$

In our case, Artist.getSimilar yields varying neighbor sets anywhere from 1 to 250 as per the API's construction. Due to the variance in degrees, this walk will not yield a uniform sample. There exists walks such as the Metropolis-Hastings that can converge to a uniform distribution, with probability transition matrix P_{ij}^{MH}:

$$P_{ij}^{MH} = \begin{cases} 1/max(d_i, d_j) & i \neq j,\ j \in N(i) \\ 1 - \sum_{j \in N(i)} P_{ij} & i = j \\ 0 & otherwise \end{cases}$$

Unfortunately this requires knoweldge of the degree for each adjacent vertex, translating in up to 250 API calls. This would quadratically slow down our sampling, and so is infeasible. Instead we use a variation on the simple random walk, a max degree walk that converges to a uniform stationary distribution. The probability transition matrix P^{MD} of this walk is

$$P_{ij}^{MD} = \begin{cases} 1/d_{max} & i \neq j,\ j \in N(i) \\ 1 - d_i/d_{max} & i = j \\ 0 & otherwise \end{cases}$$

where d_{max} is the maximum degree over the entire graph. By construction we have that $d_{max} = 250$ so that we need only to call one request for Artist.getSimilar per vertex. It was also experimentally showed that for a power law degree distribution, which Last.fm also follows, the mixing time for the max degree walk is approximately $20log(|V|)$. Thus, as per the Last.fm employee's comment, the mixing time is approximately 160. We decide the necessary number of samples below, when defining a genre as simply a tag.

4 Defining a Genre

From the perspective of information retrieval we can treat each artist (document) as a bag-of-words or multiset of tags (terms), where order is ignored. In document processing the unorderd assumption can be constrictive, but in our case where folkosonomies are naturally unordered, the bag-of-words model works perfectly. Let a tag be denoted t, and an artist $D = \{t_1, \ldots, t_n\}$.

Given a set of documents, there are many ways we could define a genre. For each definition we define a distance, so that the genres may be clustered later

on. We cover 3 genre definitions, each more sophisticated and contemporary than the last. First, we define a genre to be simply a tag, one of any found in any document while sampling. Second, we define it as a vector of documents. And third, we define a genre as a probabilitiy distribution over tags.

4.1 Simple Co-occurences

In this view, we define each tag to be its own genre. A reasonable way to define a distance between two terms in a document, then, is as the empirical probability that they co-occur:

$$d(t_1, t_2) = \hat{p} = \frac{1}{N} \sum_{D} [t_1, t_2 \in D]$$

And given the random walk procedure described above, each $[t_1, t_2 \in D]$ can be thought of as an independent Bernoulli trials, so that we can create a confidence interval for these distances. We could use a Normal approximation, but a general rule of thumb is to have $np \geq 10$, and the relative frequencies of our tags in documents follows a power law so that it will be innappropriate to use this approximation for many tags. Instead, we use a Chernoff bound. Let p be $E[\hat{p}]$, the true probability that two tags co-occur. Using a two sided form of the bound

$$p(\mid p - \hat{p} \mid \geq \theta) \quad \leq \quad 2\exp(-\frac{\theta^2}{2p+\theta} n)$$

so that we can adjust the number of samples necessary in the walk to achieve some error$p(\mid p - \hat{p} \mid \geq \theta) \leq \alpha$ in our distances. See that $p < 1$ implies $\frac{\theta^2}{2p+\theta} \geq \frac{\theta^2}{2+\theta}$ so that we want

$$n \geq \frac{2+\theta}{\theta^2} \ln \frac{2}{\alpha}$$

samples to achieve the bound. For example, if we wanted $d(t_1, t_2)$ to be within 0.05 of p with a confidence of $\alpha = 0.1$, then we would need at least 2457 samples ($\ll$ 100 million) for t_1 and t_2. For visualization purposes, we'll use only the top 200 tags which occur in over 90% of all documents so that we need only sample $2457/0.9 \approx 2730$ artists. A symmetric distance matrix can be created from this, with number of rows and columns equal to V choose 2, where V is the number of unique terms seen on the walk. The following in Figure 4.1 is a list of the highest occuring 200 tags collected from a sample:

4.2 Latent Semantic Indexing (LSI)

Here, we define a genre to be a vector. The main idea in LSI is also to exploit term co-occurence, but by incorporating the co-occurences at a higher level. That is if term s and u never co-occured together, but each co-occured often with t, then LSI will view all three terms as co-occuring with one-another.

Seen live, Electronic, Pop, All, Rock, Male vocalists, Dance, Female vocalists, Electronica, American, Trance, Korean, Indie, Hip-hop, Japanese, K-pop, Kpop, Asian, Alternative, Spotify, Rap, Experimental, Metal, Hardcore, Psychedelic, Usa, Hip hop, Under 2000 listeners, Ambient, Techno, Punk, 00s, Jazz, Chillout, J-pop, Rnb, Soul, Korea, J-rock, House, Psytrance, 90s, Female vocalist, Electro, Instrumental, Korean pop, Progressive trance, Singer-songwriter, Visual kei, Alternative rock, Sexy, 80s, Female, Folk, Psychedelic trance, R&b, Emo, Indie rock, South korea, Love, Funk, Finnish, Punk rock, Jrock, British, Death metal, Asian pop, Girl group, Minimal, Awesome, Progressive, Asian music, Goa, Hiphop, German, Jpop, Male vocalist, Club, Boyband, Japan, Underground hip-hop, Hardcore punk, Chill, Downtempo, Favorites, Hard rock, Beautiful, K-music, <3, World, Acoustic, Indie pop, Darkpsy, Ballad, Swedish, Metalcore, Lesser known yet streamable artists, Girlband, Soundtrack, Uk, 10s, Pop rock, Christian, Female vocals, Post-hardcore, Uplifting trance, Favorite, Hot, Male, Underground rap, Screamo, Underground, Black metal, Dark psytrance, Anime, Industrial, Minimal techno, Need to rate, Vocal trance, Smooth jazz, Classic rock, Cute, Idm, Progressive house, Heavy metal, Underrated, Urban, Korean girlsband, Piano, Lounge, Electropop, New york, Easy listening, Reggae, Dark, 70s, Girl groups, Brutal death metal, J-indie, Grindcore, Idol, Dark psy, Noise, Old school, Goa trance, Disbanded, Blues, New wave, Guitar, Solo, Progressive rock, Gospel, Tech house, Post-punk, Oldies, Japanese rock, Psy, Synthpop, Amazing, Germany, World music, English, Dub, Atmospheric, Rhythm and blues, Pretty cool, Post-rock, Gangsta rap, Desi, Melodic trance, Underground hip hop, Country, Punjabi, Emocore, Bhangra, Desi artist, Mellow, Group, California, Korean music, Sweden, Groove, Cool, Smooth, 60s, Dark psychedelic trance, South korean, Crap, Boy band, Trip-hop, Indian, Suomi, Disco, Finland, Fusion, Pop punk, East coast rap, Punjabi bhangra, Lo-fi, Composer

Figure 4.1: Top 200 occuring tags from a random walk sample of 3000.

Further, LSI will also de-emphasize terms that co-occur often with most other terms.

We first convert each artist into a vector of tags. That is, a vector consisting of weights for all tags it contains, and zeros elsewhere for all other tags seen in any other artist. The Last.fm API call Artist.getTopTags returns integer weights from 1 to 100 for each tag. We naturally use these as weights in each document vector. The documents are then arranged as a rows in a document-term matrix M. Singular value decomposition is then applied, so that we can write

$$M = U\Sigma V^\top$$

where Σ contains the singular values. As $\Sigma_{1,1}, \Sigma_{2,2}, \ldots$ are ordered in significance, we can keep only the first k and continue the multiplication with U and $V^\top$ to achieve a projection into a lower rank space. We call this lower rank representation M_k. We can then use a distance function between vectors to find the similarity of terms. A cosine distance is customary for LSI. Let t_1 and t_2 denote vectors in M_k. The cosine distance is

$$d(t_1, t_2) \quad = \quad \frac{t_1 \cdot t_2}{\|t_1\| \cdot \|t_2\|}$$

In Figure 4.2, we see 8 topics from LSI, with $k = 200$, perfomed on the same sample of 3000 artists as with co-occurence genres:

4.3 Latent Dirichlet Allocation (LDA)

The algorithm LDA, modeled after LSI and probabilistic LSI, constructs a generative model for terms within documents. Let k be the number of pre-specified topics, V be the number of unique terms across the entire corpus, α be a positive k-vector, and η be a scalar. LDA uses the following generative model:

```
0.709*"k-pop" + 0.587*"korean" + 0.259*"kpop" + 0.165*"female vocalists" + 0.126*"pop" +
0.076*"girl group" + 0.075*"boyband" + 0.063*"male vocalists" + 0.055*"ballad" + 0.054*"dance"

0.519*"trance" + -0.375*"visual kei" + 0.362*"psytrance" + 0.305*"darkpsy" + 0.270*"progressive
trance" + -0.249*"j-rock" + -0.195*"japanese" + 0.158*"uplifting trance" + 0.140*"vocal trance" +
0.135*"electronic"

-0.500*"darkpsy" + 0.478*"trance" + -0.472*"psytrance" + 0.261*"progressive trance" +
0.180*"uplifting trance" + -0.173*"psychedelic" + -0.166*"psychedelic trance" + 0.162*"vocal
trance" + -0.146*"dark psytrance" + -0.136*"goa"

0.668*"hip-hop" + 0.483*"rap" + 0.294*"underground hip-hop" + 0.268*"hip hop" + 0.234*"finnish" +
0.116*"suomirap" + 0.103*"seen live" + 0.096*"new york" + 0.092*"underground rap" + 0.089*"east
coast rap"

0.736*"smooth jazz" + 0.488*"jazz" + 0.197*"saxophone" + -0.179*"j-pop" + -0.158*"anime" +
-0.129*"japanese" + 0.093*"visual kei" + 0.091*"contemporary jazz" + 0.090*"piano" + 0.090*"soul"

-0.772*"brutal death metal" + -0.492*"death metal" + -0.224*"slamming brutal death metal" +
-0.138*"technical death metal" + -0.134*"black metal" + -0.124*"grindcore" + -0.092*"goregrind" +
-0.085*"old school death metal" + -0.070*"deathcore" + -0.060*"technical brutal death metal"

0.842*"reggae" + 0.315*"dancehall" + 0.251*"roots reggae" + 0.198*"dub" + 0.121*"roots" +
0.113*"jamaica" + 0.108*"ska" + 0.099*"rocksteady" + 0.067*"ragga" + 0.066*"rasta"
```

Figure 4.2: 8 topics from LSI, $k = 200$, perfomed on a sample of 3000 artists.

1. For each topic,
   (a) Draw a distribution over words $\beta_k \sim \text{Dir}_V(\eta)$.
2. For each document,
   (a) Draw a vector of topic proportions $\theta_d \sim \text{Dir}(\alpha)$.
   (b) For each word,
      i. Draw a topic assignment $Z_{d,n} \sim Mult(\theta_d),\ Z_{d,n} \in \{1, \ldots, k\}$.
      ii. Draw a word $W_{d,n} \sim Mult(\beta_{z_{d,n}}),\ W_{d,n} \in \{1, \ldots, V\}$.

Unfortunately the posterior on β and θ is intractible. But, we can used Gibbs sampling to estimate the most likely distribution matrices $\hat{\beta}$ and $\hat{\theta}$. These are respectively the most likely term distributions per topic, and topics distributions per document.

A similar distance measure can be used for the these per topic term distributions, just as in the naive co-occurrence method. Unfortunately the familiar Kullback–Leibler distance is not symmetric, so instead we use another f-divergence, Hellinger distance:

$$H(p, q) = \frac{1}{\sqrt{2}} \sum_i (\sqrt{p_i} - \sqrt{q_i})^2 \in [0, 1]$$

In Figure 4.3 are the results on the same sample as with co-occurences of 8 topics found with LDA, with $k = 200$ topics:

```
0.588*piano + 0.145*italy + 0.077*arranger + 0.023*italia + 0.019*neo classical +
0.017*italiana + 0.015*20th century + 0.012*pianist + 0.011*italiano + 0.010*keyboards

0.550*hardcore punk + 0.125*punk + 0.059*hardcore + 0.058*oldschool hardcore +
0.029*old school hardcore + 0.017*boston hardcore + 0.006*noise punk + 0.006*80s
hardcore + 0.006*hardcore-punk + 0.006*synth-pop

0.556*hip-hop + 0.279*rap + 0.082*hip hop + 0.007*underground hip-hop + 0.004*new york
+ 0.003*alternative hip-hop + 0.003*hiphop + 0.003*experimental hip-hop + 0.003*west
coast hip-hop + 0.002*producer

0.333*metalcore + 0.241*uk hardcore + 0.065*melodic death metal + 0.051*hardcore +
0.035*metal + 0.028*melodic metalcore + 0.025*djent + 0.021*metallic hardcore +
0.020*progressive metalcore + 0.014*mathcore

0.327*darkpsy + 0.154*psytrance + 0.099*dark psytrance + 0.082*dark psy +
0.067*psychedelic + 0.041*dark psychedelic trance + 0.027*psychedelic trance +
0.018*dark + 0.013*forest music + 0.012*forestpsy

0.793*female vocalists + 0.065*female vocalist + 0.023*female vocals + 0.020*female +
0.011*female vocal + 0.007*female fronted + 0.006*00s + 0.006*10s + 0.005*hot +
0.004*cute

0.472*alternative + 0.244*post-punk + 0.110*alternative rock + 0.032*rock + 0.022*post
punk + 0.011*no wave + 0.006*seen live + 0.006*indie rock + 0.005*experimental rock +
0.005*2010
```

Figure 4.3: 8 LDA with 200 topics, showing 10 most pertinent terms, from a sample of 3000 artists.

5 Flat Clustering

I explore two methods of flat clustering that specialize in the clustering of genre topics rather than terms: Self-Organizing Map (SOM) and Correlation Clustering (CC). The more familiar K-means clustering is difficult in this situation as we can retrieve likely groups of tags but not visualizing how far apart those clusters are, i.e. how they are spatially related to each other, as we have only the distances between tags or topics. The methods SOM and CC, on the other hand will generate both spatial relations and distances simultaneously.

SOM is an unsupervised learning algorithm that projects high dimensional data onto a 2 dimensional plane, from which one can visually see groups that are close together. The algorithm is neurologically motivated, and it takes as input a set of high-dimensional training vectors $T = \{t_1, \ldots, t_n\}$. Depending on our genre definition these may be distribution of tags or subvectors resultant from LSI. We repeat the following process:

1. Initialize a 2-dimensional grid with small norm random "node vectors" of the same dimension as training vectors: $n_{i,j}$. Another initialization method is to sample vectors from the subspace spanned by the two largest eigenvectors of T.

2. Select a training vector, t.

3. Given some vector distance function d, find the closest vector n_c according to the genre's definition of distance.

4. Update the nodes of the grid as $n_{i,j} := n_{i,j} + \lambda d(t, n_c)$, where λ is the

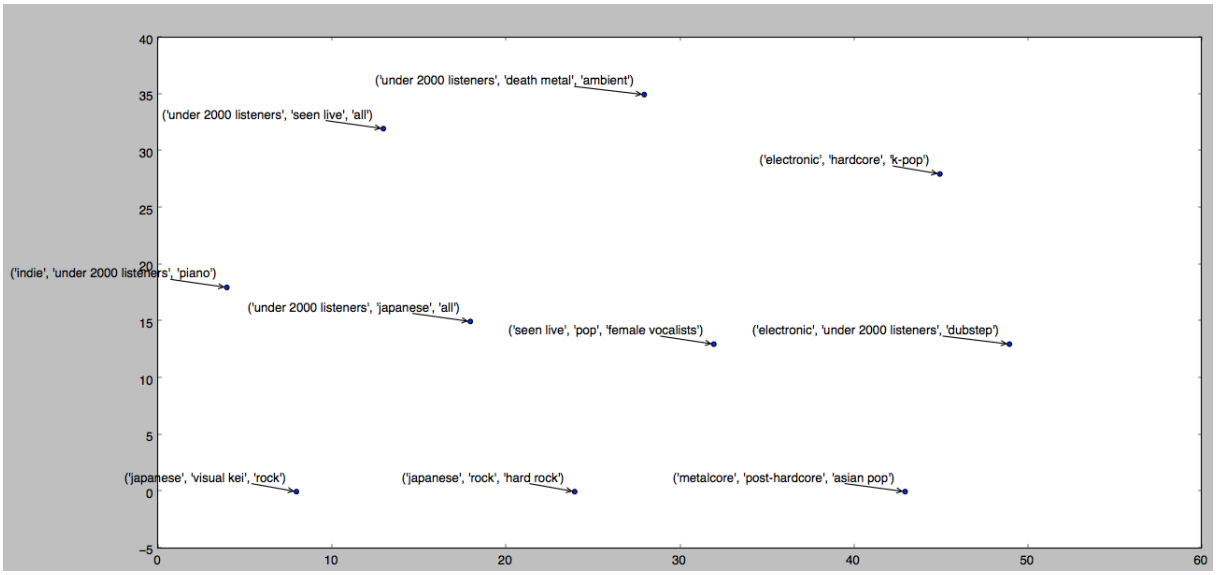

Figure 5.1: SOM performed on LDA with 10 topics, 400 iterations, and learning rate of 0.05.

learning rate. We make this learning rate a decreasing function of time and distance from the node n_c.

After several iterations, the map converges to a space that resembles the topology of the training vectors. Then, each of the training vectors can simply be places a top the coordinate of the closest node on the grid to achieve a 2-dimensional clustering of the high dimensional input, whether they be topics from LSI or LDA.

A problem still inherent to the high-dimensional plotting is how to display what each vector is. There are several thousand tags in each of the vectors, and it's infeasible to display these distributions for each point on the map. For LSI we use a simple tf-idf score fore each term, and display the top m terms according to this scoring. For LDA, we use a method from [Blei 2009] analogous to tf-idf:

$$score_{k,v} = \hat{\beta}_{k,v} \log(\frac{\hat{\beta}_{k,v}}{(\prod_{j=1}^{K} \hat{\beta}_{k,v})^{1/K}})$$

for a term v and topic k out of K topics, and a posterior topic distribution matrix $\hat{\beta}$. Result of the SOM clustering over a uniform sample of 10^5 artists, with an LDA of 10 topics, and a grid size of 10 x 10 are shown in Figure 5.1.

A problem with SOM clustering, however, is that a high resolution grid size is required to avoid overlap with a high number of topics. Further, higher grid size accommodates a finder intuitive spatial understanding of the clustering.

Unfortunately, larger grid sizes requires a quadratically larger computation size that becomes infeasible with SOM's large complexity coefficient. It would be very convenient to cluster topics according to an infinitely high resolution 2-dimensional place.

To solve this problem one can use a correlation coefficient based clustering method. In this method we aim to optimize the correlation between high-dimensional distances to lower dimensional ones. Similar to the random assignment of neural nodes on the grid for SOM, for each high dimensional vector $t_i \in \mathbb{R}^D$, $N \gg 1$ we assign a coordinate $x_i \in \mathbb{R}^2$ within an infinitely resolute grid. We then list the n choose 2 pairwise distances D_T and D_x, $D_T = \{d(t_i, t_j)\}_{i,j}$ and D_x with the same enumerate order. The correlation coefficient is then a "fitness function" for how well the 2-dimensional coordinates represent the distances in high-dimensional space:

$$\begin{aligned} \rho_{D_T, D_x} &= \frac{cov(D_T, D_x)}{\sqrt{var(D_T)}\sqrt{var(D_x)}} \\ &= \frac{\sum (t_i - \mu_t)^2 (x_i - \mu_x)^2}{\sqrt{\sum (t_i - \mu_t)^2}\sqrt{\sum (x_i - \mu_x)^2}} \end{aligned}$$

As the grid size is infinite, the sample space for $\{x_i\}$ is huge. To optimize this ρ_{D_T, D_x}, and overcome the sample space size difficulties, we can employ a genetic algorithm (GA). GAs naturally work well as parallelized algorithms, so that we can run it fairly quickly. GAs initialize a population of genomes, and the fitness (ρ in this case) of each is evaluated. Genomes are then mated to create offspring which are some combination function of their parents. A mutation function is usually also applied to the resultant children. In this case we represent each genome as an ordered list of indices $\{x_i\}$. I use a 2-point crossover mating function. To allow for changes in coordinates, I define mutation to be a Gaussian wiggle on each coordinate.

6 Nested Hierarchical Clustering

Hierarchical clustering generates a taxonomic tree that is generally more informative than an unstructured flat clustering. Further, while flat clustering usually requires a parameter for the number of clusters and is non-deterministic, hierarchical clustering does not and is deterministic.

These algorithms can take either a top-down or bottom-up approach. In each, it requires a distance function between items. In the previous section we have shown how to precompute a distance matrix between all pairs of tags, so that we these distances don't need to be calculated in the clustering step. In bottom-up clustering (or agglomerative clustering) pairs of items or clusters are merged, until there is just one remaining cluster. We choose a function that takes in a pair of items or clusters and outputs a real value, representing a merging distance. At each step, we choose the items or clusters that minimize this function.

We can view this nested clustering in the form of a dendrogram, where groups are represented as merged by a connected line. And we draw this merge on the y-axis (or radius) at a point corresponding to the distance that our merge function returns. This dendrogram can then be cut, horizontally, to generate an arbitrary number of clusters. That is, we can return to flat clustering via this hierarchical clustering. Choosing the y-axis point at which to point is essentially the same decision as choosing the number of clusters for a flat clustering.

One convenient choice for a merging function is single-linkage, where it outputs the minimum distance between members of two clusters. This method is quite local, and can allow for largely disparate clusters to be joined by a mere single strong connection. Another option is complete clustering where the merge function instead outputs the maximum distance between members of two clusters. With this, all the tags of a cluster can influence the connection, as opposed to single-linkage. For the purposes of visualizing clusters, the linkage type will strongly affect the results, each benefiting different aspects of related genres. As fickle singular connections are often in remixing in music, yet do not signify a sub-genre, we choose complete-linkage. A display of the results for co-occurence distances with complete-linkage is shown in Figure 6.1.

7 Comparison of Methods and Uses

The folksonomy of tags tends to be quite informal and at times arbitrary, sometimes referring to the manner in which an artist was listened to rather than what it sounds like. To further complicate the matter, several terms may refer to the same thing (synonymy), while the same term may have multiple unrelated meanings (polysemy). Using co-occurences as distances completely ignores the issue, and takes terms with the highest marginal probabilities so that only the most agreed upon tags (most used) will be used for distances. LSI, on the other hand, uses a lower dimensional approximation to project synonyms ontop of each other via SVD. This attempts to solve the problem of synonymy by projecting terms together. However, it still does not capture the subtly with polysemy. But it still proves to be a great improvement over co-occurences. Lastly, LDA models topics as distributions solving both synonymy and polysemy. While LSI and LDA both mitigate linguistic problems, they carry greater difficulty in displaying what each genre is about.

References

- Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA.
- Asad Awan , Ronaldo A. Ferreira , Suresh Jagannathan , Ananth Grama. Distributed uniform sampling in real-world networks. 2004.

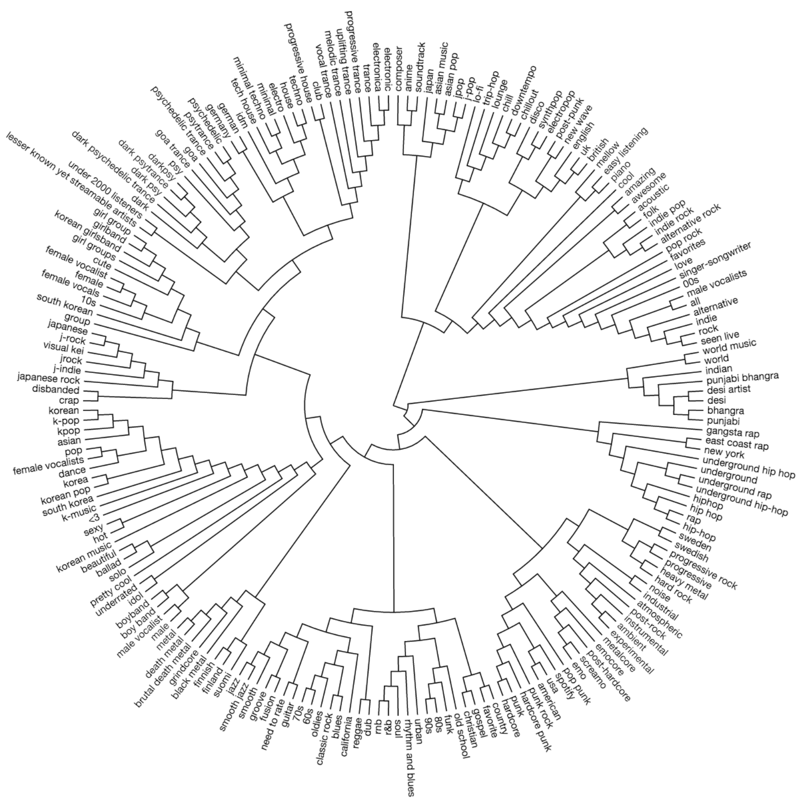

Figure 6.1: Complete-linkage hierarchical clustering on the top 200 tags on a sample of 3000 artists.

- David R. Karger and Clifford Stein. 1996. A new approach to the minimum cut problem. J. ACM 43, 4 (July 1996), 601-640.
- David M. Blei, Thomas L. Griffiths, Michael I. Jordan, Joshua B. Tenenbaum. Hierarchical Topic Models and the Nested Chinese Restaurant Process. 2004.
- Bishop, C.M., Svensén, M. and Williams, C.K.I. (1998). GTM: The Generative Topographic Mapping. Neural Computation 10(1): 215-234.
- Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. Biological Cybernetics, 43:59-69.
- Levy, M. & Sandler, M. (2007). A Semantic Space for Music Derived from Social Tags. 8th International Conference on Music Information Retrieval (ISMIR 2007) .
- David M. Blei, Andrew Ng, Michael Jordan. Latent Dirichlet allocation. JMLR (3) 2003 pp. 993-1022.
- AlSumait, Loulwah. Topic Significance Ranking of LDA Generative Models. 2010.
- Michael Mitzenmacher and Eli Upfal. 2005. Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, New York, NY, USA.
- Papadopoulos et al. A Graph-based Clustering Scheme for Identifying Related Tags in Folksonomies. 2001.
- Mark Levy & Mark Sandler (2008) Learning Latent Semantic Models for Music from Social Tags, Journal of New Music Research, 37:2, 137-150.