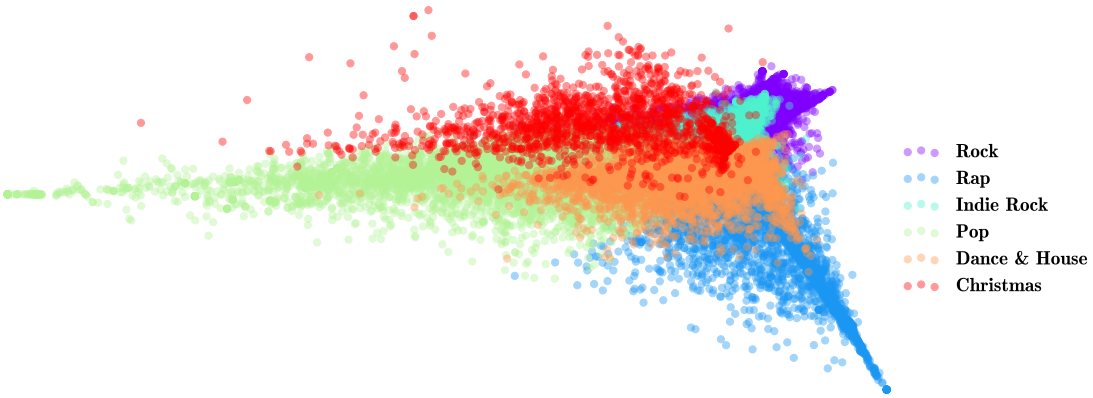
CHALMERS



Cluster User Music Sessions

Master of Science Thesis in Computer Science: Algorithms Languages and Logic

OSCAR CARLSSON

Department of Computer Science & Engineering
Chalmers University of Technology
Gothenburg, Sweden 2014

Cluster User Music Sessions

Oscar Carlsson

Examiner: DEVDATT DUBHASHI

Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover: Illustration of clustering in music domain with each color representing a genre in some space.

Department of Computer Science and Engineering

Göteborg, Sweden May 2014

Abstract

The ability to serve the "right music for every moment" is of crucial importance for streaming services like Spotify. To make that a reality, it's necessary to understand what a music moment is, which tracks that belongs to each music moment, and to understand which moments that are most popular amongst users. This thesis presents a project that attempts to cluster user session with descriptive data on tracks that has not been available before. The objective is to evaluate the usefulness of the descriptive data for this problem, and identify the most popular moments for the users.

An approach where the problem is modeled as in text document clustering is presented and showed successful. The genre and "mood" information about each track is the information from the music listening used. A comparison between different widely used techniques from document clustering is shown and the best model show results that it managed to capture clusters of what a human would interpret as similar sessions.

The results demonstrate that the method and the descriptive data is able to capture music moments of the users and it also shows a way of interpreting the clusters. Examples of identified popular moments are clusters named Pop, Indie Rock, Dance & House, Christmas, Classical, Children's music and Comedy.

Sammanfattning

Att kunna erbjuda “rätt musik för varje moment“ är viktigt för musik-streaming tjänster som Spotify. För att göra detta till verklighet är det nödvändigt att förstå vad ett moment är, vilka låtar som hör till varje moment och att förstår vilka moments som är dom mest populära. Det här exjobbet presenterar ett projekt med mål att klustra användarsessioner med ny deskriptiv data om låtar. Vi vill svara på hur användbar den deskriptiva datan är för det här problemet och identifiera det populära ögonblicken en användare kan ha.

Ett angreppssätt där problemet är modellerat som i text dokument klustring presenteras och visas vara framgångsrikt. Informationen använd från varje musiklyssnings session är genre och “mood“ informationen från varje låt. Genom att jämföra olika kända klusringstekniker för dokumentklustring visar vi resultat som indikerar att modellerna lyckas fånga en gruppering av sessioner som människor skulle tolka som kluster.

Resultatet visar att metoden och användadet av deskriptiva datan är användbart för att identifiera populära ögonblick och att det klarar av att hitta intressanta kombinationer av genrer och moods. Exempel på populära moments är inom genrerna Pop, Indie Rock, Dance & House, Julmusik, klassisk, barnmusik och komedirock.

Acknowledgements

I would first of all like to thank my supervisor at Spotify, Anders Arpteg. He has been a great supervisor encouraging me and guiding me whilst always letting me work and explore independently. The rest of the team that i have been working with at Spotify has also been great to me. A special big thank you to Boxun Zhang for helping me and professionally questioning my work, leading up to a better thesis, and a greeting to Theodoros Vasiloudis for all the late discussions at the office.

I would also like to thank the LAB group at Chalmers and especially my supervisor Devdatt Dubhashi for all the help.

Oscar Carlsson, Gothenburg 11/6/2014

Contents

1

Introduction

Music streaming services are becoming more and more popular with increasing numbers of active users. The Swedish music streaming company Spotify recently announced 40 million users [1] and other competitors such as Pandora[1], Google Play[2], Apple iTunes radio[3] is also increasing in popularity.

With the increase of online music streaming services a previous unavailable type of data on users interaction with music has become available. This type of data includes peoples interactions with music such as when and where they listen to music, how they select and categorize their music. Analysis on music behavior has long been of interest to research but has mostly based the results on surveys and questionnaires [2, 3] which is subjective to the honesty of the participants and usually contains small amounts of data. These problems can be avoided with this new type of data hence new possibilities has opened for research based on peoples actual interaction and listening behavior.

But, even though this new data from the streaming services would heavily benefit todays research it is the property of the companies as of who can be very restrictive on what information to publish for research and not. Hence there is a gap in published research based on this type of data. There are of course publications most of which are using some of the data that is publicly available. The two most famous publicly available datasets related to music listening is the Million Song Dataset [5] and the Last.fm dataset[4].

Not only academic research benefit from this data but also the streaming companies. Spotify wants to be a social service where users easily interact with music, gets personalized recommendations and accesses and finds relevant music to them, unlike only being a regular on-demand music service. Letting users easily access relevant content is a problem due to the massive size of the music catalogs. Currently Spotify's catalog is

[1] www.pandora.com
[2] play.google.com/
[3] https://www.apple.com/itunes/itunes-radio/
[4] http://ocelma.net/MusicRecommendationDataset

of over 20 million tracks with 20 000 new tracks being added each day [1].

To build personalized and complex service it is important to develop techniques and systems that can be able to study user activity and extract the usage patterns that characterizes each user. This means systems that need to handle large datasets and complicated models to process and extract information of the data. This problem and more generally, the process of finding useful patterns in datasets, has been called different names in research and literature but in this thesis it will go under the term Knowledge discovery in databases (KDD) process as of [4].

This thesis has been carried out at the streaming service Spotify with the objective to evaluate if it is possible to find what combinations of music Spotify users listens to during different uses of the client. Available to find these combinations is Spotify's unique dataset about users and a, to Spotify new, rich descriptive data about tracks. This new data categorizes tracks into genres and moods.

These combinations of genres and moods represents different moments of the users. What is refereed to as a moment is a set of selected tracks the user has chosen to accompany an environment and a situation or an emotional mood of an user. A moment could be the songs selected on a Friday night, during workout or on Valentines day etc. Spotify is trying to serve the "right music for every moment" and wants to know if this new descriptive data can help them find their users moments and later be able to support these in the client.

A user listening session, which is a sequence of subsequent played tracks from one user, is what is modeled in the data as a moment. This notion is not all new as previous user analysis has been performed internally at Spotify using user sessions, but never with this new descriptive data on tracks.

The aim is to group a subset of user sessions into groups of similar music sessions, as individual analysis on each user session is not feasible. These groups (clusters) of user sessions would represent generalized types of music moment common to the users. The goal is that the structure of the different clusters will explain underlying and hidden patterns in the data describing what combination of music genres or moods that users listened. Of course, one cannot derive the intent or full context of the music moments but one should be able to find certain common sessions representing the music choice in these moments.

1.1 Project aims

The objective of this thesis is to, using a subset of the entire Spotify user dataset, investigate if the user session-based clustering can reveal previously unknown information about combinations of music tags. Such unknown information could be previous unknown "moments".

Basic clustering techniques should be tested and experimented with on which method can best group the user sessions into natural groups of what a human would interpret as similar sessions. The interpretation and method used to understand the output of the algorithms is equally important as to find the best technique to use.

One limitation is that the thesis is solely on a subset of the data and this is of importance as a large scale analysis would, with the time constraints on a master thesis, mostly be a work focused on the distributed computation of clustering and not analysis of the results. This means that the exact results presented does not generalize to the full dataset of users.

No implementation and or time efficiency comparison of algorithms is to be done. Neither is it in the scope of this thesis to evaluate the performance of the use of the results. Such use could be recommendation evaluation, ad-targeting, radio-feature, playlist generation etc.

This thesis focus solely on the session-based clustering approach and understanding and interpretation of such results is of great importance.

1.2 Thesis outline

In the following chapter, Data mining methods, is a theoretical framework given that is needed to understand previous research and results given. The methodology chapter will map the previously mentioned KDD process to how this thesis was carried out and detailed explain how the results were obtained. In the result chapter an evaluation of clustering techniques is given at first. The result of the chosen model is later on explained and discussed in the end of the thesis.

2

Data mining methods

All models are wrong, but some models are useful.
— George Box, *[32, p. 424]*

This chapter serves as a theoretical background of the used algorithms from the field of machine learning and statistics that is needed to understand the work done. The famous quote from George Box above implies that there is no one model that works best for all applications and domains. The chosen methods are motivated by previous work and modeling of the problem. As will be clear in Section 3.3.1 the problem was chosen to be modeled as a text document clustering problem hence the choice of methods explained in this chapter.

Covering the theory of all data mining methods, including those not needed to interpret the findings of this thesis, is not intended. This chapter will present different methods that has proven to be suitable for the given problem setting and needed to understand the work of this thesis. In the end of the chapter is a summary of previous related research with short explanation on their approaches.

The naming of this chapter is from the definition of the KDD process from Fayyad in [4]. This process is explained more in detail in Related works and the next chapter Methodology.

2.1 Clustering

Clustering is the problem of grouping a dataset into different cluster with similar samples in the same group. In the field of text document clustering the dataset represents a collection of news articles, research papers or any written material that we want to separate into groups of similar documents – probably similar meaning that they talk about the same topic. We will see that the notion of similarity is essential in the field of clustering.

Clustering is of unsupervised nature as the data that is clustered is unlabeled meaning that the true natural clustering of a dataset is unknown. Without labeled data there is no error or reward to evaluate a solution. Therefor, in clustering the solution is usually evaluated based on the shape and of the solution. Two different things to look at is compactness and separation of a clustering. *Compactness* is when each cluster consists of samples more similar to each other than to those of other clusters and *separation* when the clusters are well separated from each other. Intuitively the solution should be as compact and separated as possible as we want to find groups that is not on top of each other and contains elements more similar to each other than other points.

In the field of clustering different algorithms are categorized based on their input and output [8]. Similarity-based clustering is when the input is a distance (or dissimilarity) matrix D of size $N \times N$ where each element is the distance or similarity between two data points and feature-based clustering when the input is a $N \times D$ feature matrix X. N is the number of samples and D the feature dimensionality.

When the output of a clustering algorithm is a single assignment of each data point to a cluster the algorithm is called flat clustering. Common for flat clustering algorithms is the need to select the numbers of clusters before the clustering, which is a problem when the data has no intuitive clustering or when there is no prior knowledge of how many clusters that is the most natural for the data. This problem of *selecting the numbers of clusters* will be discussed further as it is a common and hard tackled problem in unsupervised clustering.

Hierarchical clustering does, unlike flat clustering, reveal more of the structure of the data as it outputs a hierarchical representation in which clusters at each level is created by merging two clusters on the direct lower level. One of the main reasons to prefer hierarchical clustering over flat clustering is that it does not require the number of clusters to be selected in advance [10]. It only requires a method of calculating dissimilarity between clusters, also called a linkage criteria which is further explained in the section about hierarchical clustering.

Soft clustering is when the output of the clustering algorithm is for each sample a distribution over clusters. Unlike the flat clustering where each sample belongs to only one cluster this output reveals more information about each samples similarity to other clusters.

Besides the methods explained in this thesis there are numerous ways of clustering. Some of the other popular approaches is dimensionality reduction such as PCA, affinity propagation, self-organization maps and density-based clustering such as the DBSCAN algorithm.

2.1.1 Measure of dissimilarity

A measure of dissimilarity (or similarity) is essential in clustering as clustering is based upon grouping samples based on similarity. The measure should reflect how close or how similar two samples are. Usually a *distance* measure is used to measure how far away two samples is from each other.

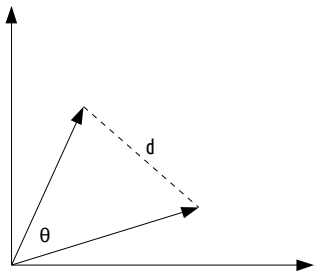


Figure 2.1: Illustration of two 2-dimensional vectors. Euclidean distance is marked as d and the angle with θ

A distance, d, is a function with only non-negative values and qualifies as a metric if, for every x,y,$z \in X$

- Identity: $d(x,y) = 0$ if x=y
- Triangle inequality: $d(x,y) + d(y,z) \geq d(x,z)$
- Symmetry: $d(x,y) = d(y,x)$

When dealing with numeric vectors the most common measure used is the squared euclidean distance [10]. The Euclidean distance is the direct distance between two points and is the same distance as given by the well known Pythagoras formula. To show this and make it more intuitive Fig 2.1 shows the distance d between the vectors that is the euclidean distance. In a n-dimensional space the euclidean distance between u and q is calculated by:

$$d(u,q) = \sqrt{\sum_{j=1}^{P}(u_j - q_j)^2} = \|u - q\| \tag{2.1}$$

and the squared Euclidean distance by:

$$d(u,q) = \sum_{j=1}^{P}(u_j - q_j)^2 = \|u - q\|^2 \tag{2.2}$$

The Euclidean distance is a standard metric as it fulfills the criteria above. Its alternative, squared Euclidean distance, is useful when objects that are farther apart should be of greater weight. Using Euclidean distance will value magnitude of the samples and length between them.

Cosine similarity is a *similarity* measure between two vectors using the cosine of the angle between the vectors. In Fig 2.1 the angle is given by θ. The cosine similarity is bounded in the positive space by [0,1] where the similarity is 1 if two vectors have the same orientation. Between the two vectors u and v it is given by:

$$sim(u,v) = cos(\theta) = \frac{u \cdot v}{\|u\|\|v\|} \tag{2.3}$$

where $u \cdot v$ is the dot product of u and v. Cosine similarity can be converted to a distance in the positive space by:

$$d(u,v) = 1 - sim(u,v) \tag{2.4}$$

The reason to convert cosine similarity into a distance is to get a measure which grows with dissimilarity between samples as similarity-based clustering needs such input. The cosine distance is thou not a proper distance metric thou as it violates triangle inequality but used when a proper metric is not a constraint.

2.1.2 K-means algorithm

K-means is the most popular algorithm in cluster analysis [33] and was first introduced already in 1967 [34]. The algorithm tries to partition the input dataset into k different clusters where each sample belongs to the cluster with the closest mean. Each cluster is represented by its sample mean, which doesn't need to be one of the samples in the dataset. The measure of similarity used to when comparing samples to clusters is the squared Euclidean distance [7].

The first step of the algorithm is to initialize k centroids. Iteratively each sample is compared to each centroid and assigned to the centroid which yields the least within-cluster sum of squares. This is the same as assigning each sample to the nearest centroid when using the squared euclidean distance or the euclidean distance as both will have the same minimum. This, the assignment step, is also refereed to as the *expectation* step.

After each sample has been assigned to the nearest centroid the centroids are updated as the new sample mean of the cluster. Since the centroids are sample means it does not have to be one of the samples but could be any arbitrary point representing the mean. The arithmetic mean is also minimizing the within-cluster sum of squares. This is the update step also refereed to as the *maximization* step.

The algorithm will terminate and return the final centroids and the cluster assignment of the samples when the centroids have converged i.e. they do not change with more iterations. There is no guarantee that k-means terminates at a global optimum hence it is usually run several times and the solution with lowest within-cluster sum of squared distances is usually chosen. As the mean can be any point different runs of the algorithms might produce different solutions that has converged differently, but could be rather similar. Naturally one understands that the closer the initialization of the centroids is a local optimum the faster the algorithm will converge.

The output K-means produces is a flat clustering and the algorithm is a variant of a more generalized algorithm called Expectation-Maximization (EM) [7]. The EM algorithm is probabilistic and will produce a soft clustering instead of the hard clustering by k-means.

Pseudocode for k-means is found in algorithm 1. The efficiency of the expectation step of K-means is $\mathcal{O}(\mathcal{NKD})$ where N is the number of samples in a D-dimension with K clusters.

Algorithm 1 Basic k-means algorithm. With inspiration from [8].

procedure KMEAN(X, k)
  Initialize k centroids $c_{1,\dots,k}$
  while centroids not converged **do**
    for all $x_i \in X$ **do** ▷ Expectation
      $r_{ik} = \begin{cases} 1 & \text{if } k = argmin_k \|x_i - c_k\|^2 \\ 0 & \text{else} \end{cases}$
    end for
    for all centroids c_j **do** ▷ Maximization
      $c_j = \frac{\sum_{i=1}^{n} r_{ij} x_i}{\sum_{i=1}^{n} r_{ij}}$
    end for
  end while
  return For each x_i, return last k where $r_k == 1$ and $c_{1,\dots,k}$
end procedure

Different ways of doing the initialization has been proposed as fast convergence means faster termination. The standard initialization is that the centroids are drawn uniformly from the dataset [35]. In [35] k-means++ was proposed, which is an algorithm to better choose the first initialization of centroids. K-means++ works such as it initializes the k centroids uniformly from the dataset, as in the normal case. But before proceeding with the standard k-means the centroids are updated by choosing new samples at random using a weighted distribution. Each point is chosen with a probability according to their squared distance to the closest centroid. In the original paper [35] they show how k-means++ outperforms the standard k-means in terms of low total within-cluster sum of squared distance and convergence time.

As we have seen in Algorithm 1 the centroids is just the sample mean in each cluster. This can be a disadvantage as the sample mean is known to be sensitive to outliers. Outliers is samples in a dataset with abnormal values, and thous very far away from the true values. As how k-means works they will be assigned to a cluster hence move the centroid in its direction. The median of a sample will, unlike the mean, be able to cope with outliers which in statistic is called a robust statistic. There exists alternative algorithms to K-means such as K-medoids and K-median that are more robust to noise and outliers.

2.1.3 K-medoids algorithm

The commonly phrased advantages with K-means is its efficiency whilst the use of euclidean distance as a similarity measure can be inappropriate in some applications [8]. Such application has proven to be text document clustering where cosine has shown to be superior to euclidean distance [31].

The K-medoids algorithm is a similarity-based clustering model that takes as input a distance matrix D. This allows the use of other similarity measures than Euclidean.

Instead of representing each clusters centroid by the sample mean of the samples assigned to that clusters each centroid is represented by one of the samples in each cluster. More specifically the sample with the lowest total within-cluster distance to all other samples i.e. most centrally located. This makes K-medoids less sensitive to noise and outliers than K-means and it minimizes pairwise distances instead of sum of squared Euclidean distances. The pseudocode is shown in algorithm 2.

In terms of efficiency K-medoids takes $\mathcal{O}(n_k^2)$ work per cluster update, which is computationally heavier than the linear update in K-means.

Algorithm 2 K-medoids algorithm. Based on [8].

procedure K-MEDOIDS(D, k)
  Initialize k centroids m_j
  while centroids not converged **do**
    for all $i \in D$ **do**
      $r_i = \underset{k}{\operatorname{argmin}}\, d(i, m_k)$
    end for
    for all centroids c_j **do**
      $m_k = \underset{i}{\operatorname{argmin}} \sum_{j:r_j=k} d(i,j)$
    end for
  end while
  return r_i, m_k
end procedure

2.1.4 Hierarchical Agglomerative Clustering

There are other clustering algorithms than those who produce a flat clustering. Hierarchical Clustering is one of them and as the name tells they output a hierarchy of clusters.

In this section only one of the two common strategies of Hierarchical clustering is explained, namely the agglomerative approach. The other approach, divisive, works somewhat similar but is less popular hence not explained [8]. In Hierarchical Agglomerative Clustering (HAC) each sample of the dataset is initialized as a cluster. At each step the two most similar clusters are merged together until only one cluster remains. The output of a hierarchical clustering is a linkage matrix that explains the merging process. The algorithm is given in Algorithm 3. The linkage matrix can be visualized in a dendrogram as shown in Fig 2.2. At the lowest level of the dendrogram every sample is its own cluster. Every time two clusters are merged together the tree is joined until there is only one cluster. The height of the branches corresponds to the dissimilarity between the two clusters joined. As a dendrogram visualizes the merging in HAC sub clusters can be found. Cutting the dendrogram in Fig 2.2 slightly around distance 0.8 gives a flat clustering with approximately 12 clusters.

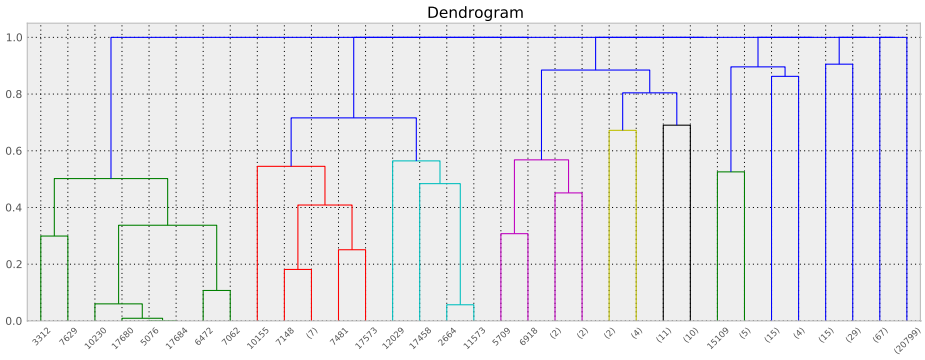


Figure 2.2: Dendrogram of agglomerative complete hierarchical clustering on a test dataset

But, as of the nature of the hierarchical clustering algorithm it will always produce some clustering of the data, even if the data only consists of noise. It is also hard to evaluate the quality of the clustering as there is no objective function that is being optimize that could be compared between models. It is therefor hard in some applications to find a good number of clusters even in the Hierarchical clustering, same problem as with the K-means. A indication of a good number of clusters is if the dendrogram could show a gap in lengths of the links. This could indicate a natural clustering as the length of the branches represents similarity.

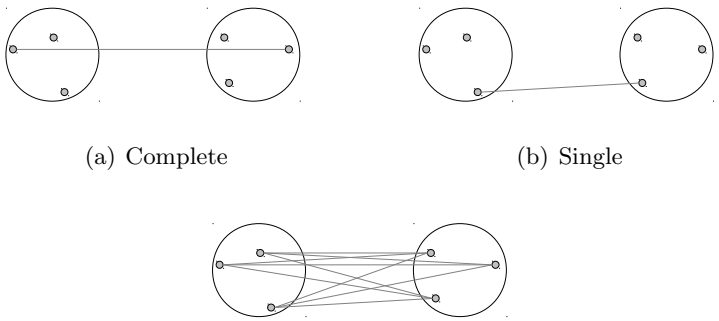
There are multiple variants of HAC depending on how similarity between two clusters is defined. Single, complete and average are the most commonly used and they are called linkage criterion [10]. All of the different methods criteria require a measure of similarity (as described in 3.1.1) between two samples, $d(a,b)$, and can produce quite different merging. [8] Later, when referring to the shape of the clusters yielded by different linkage criteria it is the shape of clusters at different hierarchies that is refereed to.

Single linkage

The single linkage criteria is also called the nearest neighbor clustering as the distance between two clusters, A and B, is defined as the smallest distance between two samples in the clusters:

$$D(A,B) = \min \{ d(a,b) : a \in A, b \in B \}. \tag{2.5}$$

As this criteria only require one pair of samples to be close to each other in order for two clusters to be merged together, single linkage can produce clusters where the biggest distance in the same cluster is big. This phenomena is called chaining and this means that the clustering might not be compact.



(a) Complete (b) Single

(c) Average

Figure 2.3: The different linkage criteria used for cluster similarity in HAC. Each big circle is a cluster with the small points being the samples. The line in between represents what distance is accounted for in the different linkage criteria.

Complete linkage

Complete linkage is the opposite extreme of the single linkage: similarity between two clusters is defined as the biggest distance between two samples.

$$D(A,B) = \max\{\, d(a,b) : a \in A, b \in B \,\}. \tag{2.6}$$

Complete linkage is also called the furthest neighbor clustering and will produce clusters where the biggest in-cluster distance is small hence compact clusters. Even though complete linkage avoids chaining that occurred in single linking it might produce clusters in which samples are closer to samples in other clusters than in the same. This violates the criteria of separation between clusters as mentioned before.

Average linkage

A compromise of single and complete linkage is the average linkage. As the name tells the similarity of two clusters is calculated as the average of the pairwise distances between samples:

$$D(A,B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a,b). \tag{2.7}$$

By measuring the average distance of all the pairwise distances the resulting clusters will be relatively compact as all object in the cluster contributes to the similarity. The major disadvantage of average linkage over complete and single is that it is sensitive to the magnitude of the distances [8]. The average linkage is also called Unweighted Pair Group Method with Arithmetic Mean (UPGMA)

Algorithm 3 Agglomerative Clustering. With inspiration from [8].

procedure HAC($matrix\ D, method\ d(A,B)$)
  Initialize a cluster per sample $\forall i \in x_i : C_i = i$
  Initialize available cluster $S = \{C_1, \ldots, C_n\}$
  while S not empty **do**
    $C_j, C_k = \text{argmin}_{C_j, C_k \in S}\, d(C_j, C_k)$ ▷ Closest two clusters
    $C_z = C_j \cup C_k$, $S = S \setminus \{C_j, C_k\}$ ▷ Merge and mark unavailable
    if C_z not all samples **then**
      $S = S \cup C_z$
    end if
    $\forall C_i$: update $D(C_i, C_z)$ ▷ Update distances
  end while
end procedure

2.1.5 Cluster evaluation

One important and hard problem of the unsupervised learning clustering is to evaluate the performance of the clustering. Due to the complete unsupervised nature of the problem in this thesis only so called internal cluster evaluations is applicable. Internal cluster evaluation is when the result evaluated is only based on the data used in the clustering. No ground truth exists to be able to compare the results to. In external clustering the results is evaluated on data that has not been clustered such as data with known class labels.

Evaluating clustering using internal methods can be used as comparison between different algorithms or for parameter tuning such as k in k-means. It is important to notice that a high/good value in an internal cluster evaluation does not imply high information gain, but can indicate if a model is preforming worse or better than another.

Silhouette score

Silhouette score was first introduced in [36] and is a measure of how well grouped the data in the clusters are. The idea is that similar samples should be in the same cluster and that they should be more similar to each others than to other clusters. The similarity, or dissimilarity, here is also defined using a distance or similarity measure, most commonly the same as used in the clustering.

The silhouette score is a measure on each sample in the dataset. When comparing models the overall average silhouette score is calculated or when feasible a silhouette plot is given, showing the silhouette score per sample.

For each sample i, $a(i)$ is defined as the average distance/similarity to all other samples in the same cluster and $b(i)$ the average distance to the samples of the closest cluster. The closest cluster to i is the cluster with the lowest average distance to i that i is not a member of.

The silhouette score of a sample i is defined as

$$s(i) = \begin{cases} 1 - \frac{a(i)}{b(i)} & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1 & \text{if } a(i) > b(i) \end{cases} \tag{2.8}$$

The silhouette scores ranges in the interval

$$-1 \leq s(i) \leq 1 \tag{2.9}$$

When the silhouette score is -1 we see from Eq 2.8 that $a(i)$ is much greater than $b(i)$ which means i lies closer to another cluster than it has been assigned to. Vice versa for a score of 1.

2.2 Classification

Classification is unlike the previous explained data mining method of supervised type. The goal is to learn the relationship between the input data x and the class $y \in \{1, \ldots, C\}$ where C is the number of classes. Supervised learning works on data with corresponding label data, y.

A popular example of a classifier is the email spam filter. The data x consists of emails and they are classified with the outcome $y \in \{0,1\}$. y takes value $y = 1$ if the input email x is spam or $y = 0$ if non-spam. The spam filter learns what characterizes a spam respectively a non-spam email by training on historical emails which somehow have been labeled.

It is important to emphasize that in the setting of training a classifier on data labeled with the outcome of clustering the inaccuracies in the clustering will correlate to inaccuracies in the resulting classifier. The class labels on the data is not so called ground truth and evaluations on classification models such as accuracy does not have the same importance.

The most obvious application of a classifier in a setting with clustering might be to classify a new user sessions which of the clusters the session most likely belong to. In this application the classifier is not used to predict the outcome of new samples but used to learn the relationship between the sessions in a cluster and all other samples.

There exists numerous of different classifications methods such as support vector machine (SVM), perceptron, naive bayes classifier, decision trees, logistic regression etc. [8] All of which works more or less differently (the difference between SVM and perceptron is small) and has its benefits in different settings.

2.2.1 Logistic regression

Logistic regression (LR) is a form of classification although the name indicates it being a regression model. The name originates from its similarity to linear regression and is

the first probabilistic model explained. [8] Linear regression is one of the most widely used regression models [8] and it assumes that the output y is a linear function of the input, commonly described:

$$y(x) = \sum_{j=1}^{D} w_j x_j + \epsilon$$

W is called weight vector and ϵ the residual error between the observed and the estimated value. Under the common assumption that the residual error ϵ is Gaussian [8, 10] it can also be modeled probabilistically as a conditional probability density

$$p(y|x, \theta) = N(y|\mu(x), \sigma^2)$$

Where μ is the linear combination of the input $\mu = w^T x$ and the noise σ^2 is constant. The parameters is $\theta = (w, \sigma^2)$. It should be clear that this is a regression model and not a classification model as the normal distribution is a distribution over the real numbers. Logistic regression is a binary classifier, meaning that the outcome is $y \in \{0,1\}$. By replacing the Gaussian (normal distribution) with the Bernoulli distribution in the Linear regression the outcome becomes binary. Remember that the Bernoulli distribution is famous for only taking on values 0 and 1, just as the logistic regression, with the famous example of flipping a coin. This gives

$$p(y|x, w) = Ber(y|\mu(x))$$

In the Bernoulli distribution the mean values, $\mu(x)$, is a probability namely the one of the outcome being one hence we need to ensure that $0 \leq \mu(x) \leq 1$ in order for $\mu(x)$ to be a probability. This is ensured by passing $\mu(x)$ through the logistic function (also called the sigmoid function) given in Eq 2.10.

$$sigm(n) = \frac{1}{1 + exp(-n)} \tag{2.10}$$

The sigmoid function has the important property of mapping the whole real line, meaning all linear combinations of input $w^T x$, to [0,1] and hence is interpretable as a probability. Fig 2.4 shows the sigmoid function where it can be seen that $sigm(-\infty) = 0$ and $sigm(\infty) = 1$. Putting it all of this together gives the logistic regression:

$$p(y|x, w) = Ber(y|sigm(w^T x)) \tag{2.11}$$

Interpreting logistic coefficients

In the logistic coefficients, the weight vector w, is what is being estimated to describe (learn) the given data. Each coefficient, w_i can be interpreted as the effect in the x_i variable on the predicted logits with all other x held constant. That is how one change in unit effects the log of odds when all other variables are constant.

The coefficient is related to odds ratio from statistics. Odds ration (usually OR) is a way of quantify how associated the presence (or absence) of some property A is with the

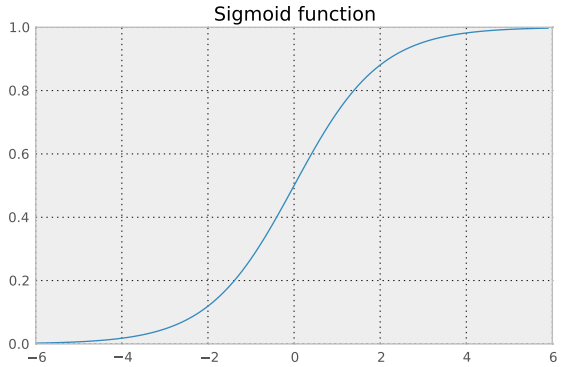


Figure 2.4: The S-shaped sigmoid-function from Eq 2.10.

presence (or absence) of another property B in a population. The ratio can, with genres as properties, describe the association between the presence of "Pop" and presence of "Rock". Calculating the odds ratio from the logistic regression coefficients b_i is done by taking the exponential function of the coefficients such as e^{b_i}.

Parameter estimation

In equation 2.11 the parameter, weight vector, w have to be chosen somehow and maximum likelihood estimation (MLE) is the most common way of estimating parameters in a statistical model [8]. In MLE the principle is that the most reasonable value for the parameter, usually denoted θ, is the one for which the probability of the observation is largest. As the name of MLE reveals the likelihood, $\mathcal{L}(\theta; x)$, has to be calculated. Due to convenience the logarithm of the likelihood is taken, turning the multiplication into addition and replaces the likelihood with the log-likelihood. This does not change the MLE since log is a strictly monotonically increasing function. Assuming independent and identically distributed (iid) examples gives the following derivation of the MLE $\hat{\theta}$ when having N observations:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \log p(D|\theta) \tag{2.12}$$

$$\log p(D|\theta) = \sum_{i=1}^{N} \log p(y_i|x_i, \theta)$$

For many model, e.g. logistic regression, no closed form expression exists to the maximization problem hence an optimization method is needed to numerically find the MLE. Different optimization methods can be used such as gradient descent, Newton's method etc. The implementation of logistic regression used in this project uses Newton method [37]. The negative log-likelihood for the logistic regression is derived in [7, 8, 10]

as:

$$NLL(w) = -\sum_{i=1}^{N}[y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)] \tag{2.13}$$

One problem with parameter setting using ML estimation is the risk of overfitting. The selection of parameters is made as good as possible for the given training data but this might have lead to the model describing the noise in the training data and not the general underlying relationship. The efficiency of a regression or classification model is evaluated on how well it performs on unseen data and a model that has overfitted will very well fit the seen data but not the unseen hence usually yield worse performance than a model which has not overfitted. Different methods to avoid overfitting exists such as regularization and cross-validation.

Regularization

Regularization is a method where complex parameter selections are penalized in order to force simpler solutions [8] [38] by adding a regularization term, $R(\theta)$. This changes the optimization problem seen in Eq 2.12 to the following regularized version:

$$\underset{\theta}{\text{argmax}} \sum_{i=1}^{N} \log p(y_i|x_i, \theta) - \alpha R(\theta) \tag{2.14}$$

Naturally, no regularization, $R(\theta) = 0$, is the same equation as before. The term $\alpha \geq 0$ is the tuning parameter of how hard or loosely to regularize the model and originates from Eq 2.14 being the Lagrangian of a constrained optimization problem and α is the Lagrangian multiplier.

Different regularization methods exists depending on how $R(\theta)$ is defined. The most common ones is when calculating the ℓ_1 or ℓ_2 norm of the parameter θ.

$$\begin{aligned} \ell_1 \text{ regularization:} \quad & R(\theta) = \|\theta\|_1 = \textstyle\sum_{i=1}^{N} |\theta_i| \\ \ell_2 \text{ regularization:} \quad & R(\theta) = \|\theta\|_2^2 = \textstyle\sum_{i=1}^{N} \theta_i^2 \end{aligned}$$

In the setting with more features than samples, Andrew Ng has shown in [38] that ℓ_1 regularization is superior to ℓ_2 in logistic regression. ℓ_1 regularization will produce a sparse parameter vector and thus perform feature selection by weighting irrelevant features to zero [8]. This is useful in the setting of using a logistic regression to explain the relationship between features and the clusters as only the most descriptive features is interesting.

Cross-validation

To avoid overfitting in a supervised model it is common practice to hold out a part of the available data to test how well the model performs on this unseen test data [10]. K-Fold Cross-validation (CV) works as it splits the data into K-equally sized pieces;

then, for each piece $k \in \{1, \ldots, K\}$ we train the model using all pieces but k and test the model on the unseen k. This is done for each piece of the data and the final score is the averaged error. Typical values for K is 5 or 10 [8, 10]. K = N is a special case and is called leave-one-out cross-validation.

Cross-validation is a common technique in model selection and can e.g. be used to compare different models of logistic regression to find a good value for the regularization parameter α.

2.3 Related work

In this section previous research is explained and discussed. No methods will be explained with same level of details as the previously explained clustering algorithms and the reader is advised to read up on the referenced material for more information.

As the amount of data grows manual analysis becomes infeasible. Manual analysis has been replaced with computational methods and theories to assist in extracting knowledge/information from large datasets. These theories and methods are all subject of the field of knowledge discovery in databases (KDD).

KDD is not restricted to any specific industry or application but is a general field applicable to any problem dealing with large datasets. The application ranges from Fraud detection, Marketing to understand user behavior and modeling. It doesn't serve as an off-the-shelf method to use but prior knowledge about the data and understanding of the application domain is needed.

The definitions of knowledge discovery, data mining and knowledge discovery in databases (KDD) varies in literature. The definition of Fayyad is used in this thesis:

> "KDD refers to the overall process of discovering useful knowledge from data, and data mining refers to a particular step in this process." [4]

The KDD process contains all the steps from extracting the data from its source, usually database, to transforming it into the right format of the algorithm that is applied to the data in the data mining step. The process involves the steps of selecting what data to use, remove noise and treat missing data as preprocessing and transforming the data to suite the data mining algorithms. The output of the data mining algorithms are later evaluation and interpreted. The application of this method in this thesis can be found in chapter 3.

Following this you will find a number of different fields which relates to the problem of clustering user-sessions with selected research presented. The reader can find a survey on text mining with clustering algorithms explained in [6] and the text books [7, 8, 9, 10] gives a good summary of the field.

2.3.1 Web usage mining

Web usage mining is an application of the KDD process focusing on the problem of extracting usage patterns from Web data to better understand and serve the users of

web-based applications. It is of interest to this thesis as the data used in web usage mining is of similar form, it deals with the problem of defining user sessions and focuses on understanding user behaviors.

In the field of web usage mining an user session is an ordered sequence of page requests or interactions by an user [11]. This is similar to an user session of music listening history which is a list of tracks listened subsequent to by a specific user. A important problem in web usage mining is the one of user identification [12]. Identifying different users in the raw web log data can be difficult because of the lack of unique identifiers of users as the IP numbers can be shared between multiple users connecting through proxies. This problem is not shared as the Spotify data contains unique usernames.

Research in web usage mining presents alternative definitions of an user session. The most widely used is to define a session as subsequent event with at most a threshold of time between two occurring event [12]. The time-gap would represent inactivity and therefore treat the interactions after an inactivity as a new session. The most used threshold in web usage mining is 25.5 minutes but naturally the choice of time threshold is highly domain specific.

A clustering of user session in web usage mining is performed in [13] using a matrix-factorization algorithm. The matrix decomposed is a similarity matrix $S = n \times n$ where n is the number of sessions and each element is the similarity between two session $s(i,j)$. The algorithm is found in detail in [14] but can be basically explained with two steps. First the similarity matrix is rearranged such that similar sessions are also closely located in the matrix and then the matrix is decomposed into sub-matrices by finding a dividing point of the matrix. Each sub-matrix is a cluster of sessions and this is done recursively on all sub-matrix until no more cluster can be found. Each sub-matrix will contain the similar sessions as of the rearrangement of the matrix. The similarity metrics used in the experiments are very primitive. They compare three different similarity measures in total: one is based upon how many pages two sessions have commonly visited, one that takes in to account the frequency of commonly visited pages and the last which considers the amount of time spent on each page.

This approach is applicable in the setting of clustering user music sessions but the presented algorithms lacks comparison and evaluation against traditional clustering algorithms, such as those previously explained. The use of matrix factorization in clustering such as the non-negative constrained matrix factorization (NMF) [15] has been used in other fields such as document cluster [16]. The problem of clustering user session will later on be shown how it can be modeled as a document clustering.

Also in the field of web usage mining, [17] shows a clustering of user sessions using a probabilistic model called probabilistic latent semantic analysis model (PLSA) [18]. PLSA is utilized to find the the hidden relationship (called latent usage pattern) between the user sessions over web pages. It is assumed that there is a latent factor space $Z = \{z_1, z_2, \dots z_k\}$ where each latent factor z_i corresponds to some usage pattern such as showing interest in sport or fashion etc. The clustering is performed on the conditional distribution from the PLSA model as there is one cluster per latent factor and a session s_i is clustered to cluster k if the conditional probability of the corresponding latent factor is

over a threshold μ: $p(z_k|s_i) \geq \mu$. The data about the session is represented in a session-pageview-matrix where each session is a n-dimensional vector $s_i = \{a_{i1}, a_{i2}, \ldots a_{in}\}$ where a_ij denotes the pageview for page p_j in session s_i. This also shows the similarity to document clustering where a document is represented as a n-dimensional vector and each element being the occurrences or term-frequency of a word.

There has been shown in [19] that PLSA and NMF optimizes the same objective function though they are different algorithms, meaning that there is a clear connection between these two approaches. In [19] they experimentally show that they are different algorithms as they converge to different solution when initialized the same.

2.3.2 Music listening data

Session-based user research can be found with music related data as well. The problem of recommending the right track to the user based on what she has been listening to is a popular research topic. Recommender systems are typically divided into two different approaches: Collaborative filtering and content-based filtering. Collaborative filtering (CF) [20] uses data about user preferences and taste from many users (collaborating) to make automated predictions (filtering) based on the similarity to other users. A lot of the contributions to the research of CF is based on movie data from the competition Netflix prize[1].

In [21] they compare a session-based collaborative filtering (SSCF) with promising results compared to the normal CF. The intuition behind the session-based approach is that the user want different recommendations based on the last played tracks, i.e. the active session. The normal CF approach differs from this as it calculates a set of tracks based on the entire listening history of the user. So instead of finding a similar user to recommend music from it finds similar sessions. They look at the k most similar sessions and how those users have rated the tracks in order to rank what tracks to recommend. The sessions are defined as in the web usage mining examples above and represented as vectors where the elements corresponds to the number of times a track occurs in a session and the features are the tracks. To measure similarity between sessions they use the cosine similarity. The data used is from a Korean streaming service, Bugs Music, and the time threshold for the session splitting is set to 30 minutes without further evaluation on the number. Also, interestingly, they choose to clean out all tracks which the user had not listened to at least 80% of. The relevancy of [21] is how sessions are defined and the choice of similarity measures. It also motivates the session-based approach as the recommender system benefited from it.

Another use of session-based collaborative filtering is in [22] who extended the work in [21] by adding temporal properties to the sessions. As the sessions are represented by vectors the construction and definition of these vectors will effect which sessions are similar and not. Instead of letting the sessions be vectors where each feature is the track in the session and the element the frequency of the track in the session they added temporal information of sessions such as time of day, weekday, day of month, which

[1]Netflix prize: `www.netflixprize.com`

month and the song diversity of the session. Song diversity is the ratio of different songs played in a session (1 for all song being different and 0 repeats the same songs). With this, two session will be more similar if they occur more closely in time with the same song diversity unlike in the previous definition where two similar sessions consisted of the same tracks. The different recommendation approaches that they evaluate is the SSCF previously explained, temporal session-based collaborative filtering (TSSCF) and a latent Dirchlet Allocation (LDA) [23] based collaborative filtering. In TSSCF the sessions are first clustered using the Expectation-Maximization (EM) algorithm which assigns a probability distribution to each session indicating the degree of membership to each cluster. The sessions are later represented as its probability distribution over clusters and two sessions with similar distributions will be treated as similar. They use the Kullback-Leibler divergence [24] to compare probability distributions.

Latent Dirchlet allocation (LDA) is an extension of PLSA to deal with the problem of overfitting and to clarify how to assign a probability to a sample outside of the training set [23]. LDA is a model that allows a group of observations (ex words in documents) to be explained by latent variables (ex topics) that explain why some parts are similar. In the LDA-based collaborative filtering in [22] they treat the each session as a documents with the songs being the words in the document. They do not add the temporal property to the definitions of sessions but instead lets the LDA implicitly find the temporal properties by inferring a topic distribution for the sessions. Similarity between two sessions is treated in the same way as in the TSSCF but with a distribution over topics (which can be seen as clusters).

Interestingly they evaluate the time gap in session generation and measures its impact on the recommendation system. They find that a time gap of 20 minutes performs the best with 15 being close in performance. The LDA-based recommender system shows best performance followed by TSSCF and SSCF.

In [25] they show how one could define statistical models to describe patterns of song listening in a music streaming service or community. They define two models. One to capture what they call the music taste from listening data and one to capture the listening moods of the users by using sessions of listening data. In the taste model they define a LDA model where the latent factors are said to represent different 'tastes', the documents the playlists of users and the words of the documents the songs. I.e. they cluster the users music playlists and refer to the different clusters as various taste profiles. In the session model they define another LDA model where the latent factor is to represent the different moods and the documents the time-thresholded sessions with songs as words. The data used is from a music community called Zune social and associated with the song data is a two-level genre hierarchy[2]. In the LDA model one has to select the number of topics t in advance, as in k in k-means. To select the resulting number of topics they calculate the perplexity of each model. The perplexity is a measure of how well a probability distribution predicts an unobserved sample and is commonly used to evaluate LDA models [23]. A comparison between the two models is also given in the problem of what they call the playlist generation. That is that they

[2]http://social.zune.net

try to predict the genre of the next song that the user want to hear. Interestingly is how they visualize their results. They show the resulting clusters of moods as combinations of genres. One cluster consists of sessions with songs of genre Gospel etc. Some of the mood clusters represents the genre hierarchy well and some not. By treating the genre hierarchy as clusters and with using the Mallow distance [26, 27] as similarity between two clusters they measure whether the resulting moods clusters are new collections of genres or just the previous known. Both models yield clusters different from the original genre structure.

2.3.3 Text document clustering

The vector representation of sessions could be represented with the bag-of-words-model commonly used in document representation [8]. The bag-of-words model represents a sequence of words as a vector with features the number of occurrences of words or the term-frequency value, tf-idf, which intends to measure how important a word is to a document given a set of documents (corpus) [9]. The vector representation deletes the order of the words and takes into consideration how often different words occur. As the vector representation with bag-of-words model grows in dimension with each unique word one wants to reduce the number of redundant words. Stop words such as ”a”, ”the”, ”as” does not help distinguish the context of to text so they are usually removed. Stemming is also used to reduce the words to its base form (e.g thinking becomes think) which also reduces the number of dimensions without additional loss of information.

In text document clustering each document has a clear start and ending whilst in cluster of user sessions each document would have fuzzy boundaries that is determined by the rules in session generation. Changing the session generation would yield a new set of documents and could hence affect the clustering output.

The objective in text document clustering is usually to group a large collection of documents into smaller groups containing those document that treat the same content. Two different approaches is using a soft clustering, which assigns a distribution over clusters to each sample, versus hard clustering where each sample belongs to exactly one cluster.

A experimental study of document clustering techniques is presented in [28]. The algorithms used in the study is K-means algorithms, bisecting k-means and hierarchical clustering. The hierarchical clustering that shows best result is the agglomerative when using average linking (UPGMA).

The study is evaluated over eight different datasets when measuring with two external quality measures, entropy and F-measure. Their conclusion is that bisecting k-means is superior to normal K-means and to the best hierarchical clustering, UPGMA. The regular K-means is reported to be generally better than the hierarchical clustering and they also give a possible explanation for hierarchical clustering to perform poorly.

The explanation in [28] to why agglomerative hierarchical clustering using the cosine similarity performs poorly is due to the nature of language namely the distribution of words. Documents consists of sequence of words and what distinguishes text of different class is the frequency with which words it consists of. Since two documents of differ-

ent class may share many of the same words they could be nearest neighbors (closest regarding to a similarity measure). This will lead the hierarchical clustering to make "mistakes" early on. K-means can deal with the problem of mixed nearest neighbors as it relies on a global property. The global property is used implicitly with the centroids as computing the similarity to a centroid is the same as the average similarity to all the clusters document. The reasoning is interesting and of interest for this thesis but due to the way sessions would be modeled as document the decision on what to use as "words" is up to us. Unlike normal documents where the words are chosen according to rules of the language used.

Agglomerative hierarchical cluster have been shown to be perform better than K-means in other evaluations such as [30] but being computationally slower and therefor being infeasible for larger datasets.

The previously explained probabilistic model latent Dirchlet Allocation (LDA) was originally proposed as a topic modeling algorithm i.e. produces clusters of document [23]. LDA does not use the vector representation with bag-of-words but assumes that each document is represented by a mixture model of various topics. Each topic is a probability distribution over words. This representation relates to the bag-of-words model as each individual word can be exchanged. LDA was showed to cluster text documents representing music sessions in [22] and its predecessor PLSA in web usage mining in [17].

As for the important problem of choosing similarity measure, Cosine similarity and Jaccard coefficent is to be preferred in document clustering according to the experimental study in [31]. The most common measure in web usage mining is reported to be the Euclidean distance [12]. The choice of similarity measure is dependent on the dimensionality and structure of the data. Cosine is more efficient to compute in sparse high dimensional data and therefore sometime preferred and Euclidean captures the magnitude of the vectors which is also sometimes preferred.

3

Methodology

As we've previously seen, a lot of different steps are needed to produce knowledge out of a big dataset. This section maps the KDD process to how this thesis was carried out and gives a detailed explanation of the problem setting of data structure, session generation etc.

The goal was to extract human-interpretable information from a large dataset of music listening histories and to evaluate if the approach of clustering user session with the new descriptive data could help achieve a good understanding of music moments.

As of the problem with finding popular moments, clustering algorithms were applied to the transformed session data. These clusters grouped together similar sessions into moments with the biggest being the popular ones. To understand and be able to characterize these clusters a classification model (Logistic Regression) was used to learn the relationship between the features of each cluster.

The process of extracting knowledge and patterns from data is by nature iterative. No one knows the right question to ask in the first iteration. Each analysis and attempt gives new insights of the data that might change the original question or assumptions made in previous steps. By working with the data, analyzing and experimenting with different ideas and approaches one will hopefully end up with the information sought for. The KDD process welcomes this iterative methodology as illustrated in Fig 3.1.

3.1 Data structure

The raw user data was stored in log files collected by information sent by the clients. This data infers the music listened to by the user and involved information such as what track was played, how long time it was played, at what time the message was sent and an unique identifier of the user. An example of raw user data is shown in Table 3.1.

A problem with data of this kind is that the users intent nor rating is captured. The only information available is that the user by purpose, or by accident, played a track and

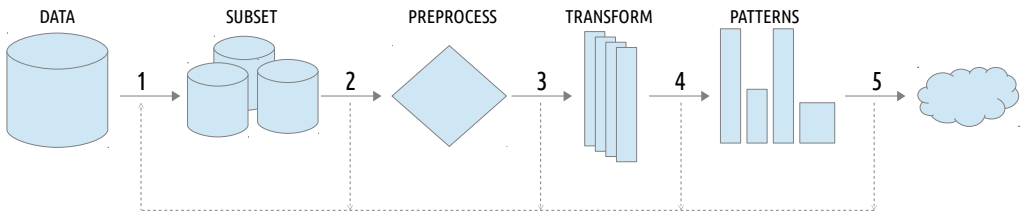


Figure 3.1: The iterative KDD process. Each number represents one of the steps in the process. (1) is Selection of data, (2) preprocessing, (3) transformation, (4) data mining and (5) interpretation. Illustration made with inspiration from [4].

Timestamp	Userid	Platform	Ms	Track id
2013-12-09 23:34	User2	phone	30530	spotify:track:3SjXx3rbNGk8nCho8YEoz5
2013-12-09 23:41	User1	desktop	50530	spotify:track:3SjXx3rbNGk8nCho8YEoz5
2013-12-09 23:51	User1	phone	30400	spotify:track:3SjXx3rbNGk8nCho8YEoz5

Table 3.1: Example of raw user data in log form. Only some, selected, attributes are shown.

not if the user liked the track or not. This problem is referred to as implicit feedback in Information Retrieval (IR).

Only a subset of user data was selected and analyzed to obtain a feasible data size and only above mentioned information about an user play was considered relevant. The size and information in the dataset is in detail explained in the result chapter.

The new descriptive additional information about each track, further on refereed to as metadata, was stored separately and had to be joined in with the selected user data. The relevant available metadata about tracks were track name, genre categorization and a content-categorized tag called mood.

3.1.1 Track metadata

Each track in the Spotify catalog has a set of extra information labeling or explaining the track. This data about data is called metadata and has previously been refereed to as the new rich descriptive data. This section aims to give the reader an understanding of the structure of the metadata used in the experiments in this project.

Metadata about a track usually contains standard information such as the name of the track, the artists and to what album the track belong to. This information is not based on the actual content of the tracks but serves as additional information to help users differentiate the different tracks. The new descriptive information about tracks include tags that categorize the tracks. Genres assigns different tracks to categorizes of music based on cultural and traditional rules whilst moods categorize different tracks based on the rhythmic and temporal context in the tracks.

The assignment or categorization of tracks into different genres has been manually curated whilst the categorization in mood has been assigned based on a content-analysis system. This means that they treat the track from two different angles making the combination attractive in the use to cluster user sessions.

Genres

Music tracks are usually categorized into different music genres in terms of style and conventions. The categorization of genres is arbitrary and highly subjective to all experts in music. Especially categorizing tracks into sub-genres. The genre structure available in this settings was a multiple level hierarchical structure with the most general genres in top such as Pop, Jazz, Rock, Metal, House and the most granular genres as leafs. Such leafs could be regional-genres as Brit pop (Indie rock) or Goa trance (Electronica).

Each track was not only assigned a single genre but multiple with a corresponding weight indicating how much this track is of that genre. Only the leafs, usually granular genres, were assigned to tracks but they did not have to a very granular genre as the depth differed. This can be in Fig 3.2 which is an illustration of one genre hierarchy namely indie rock. Also, some of the leafs represented not a new more granular genre but could have been a more specific differentiation of that track, such as if it is male or female singing.

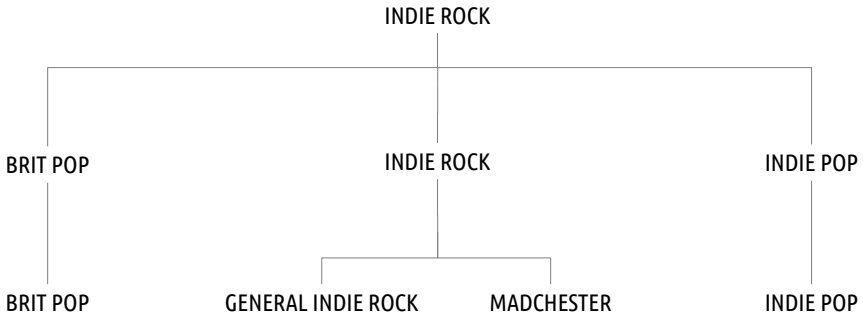


Figure 3.2: Example of how the Indie Rock genre-tree could look like.

Moods

One application of music information retrieval is automatic categorization of tracks based on its content. This can be done by analyzing characteristics of a track such as its melodic, rhythmic and harmonical structure. One application is to derive the mood or emotional of the song [39]. Such classification can be based on a track being peaceful, romantic, sensual or upbeat, fiery and energizing.

The mood tag was assigned to each track based on its content and had, as the genres, a multiple level hierarchical structure. A track could be assigned multiple different moods

with a corresponding probability of how much each mood was represented in the track. This weight were a probability unlike the weights in genres. As for genres only the leafs were assigned to tracks.

To give an understanding of what the naming of the mood could have been, Fig 3.3 shows two hierarchies.

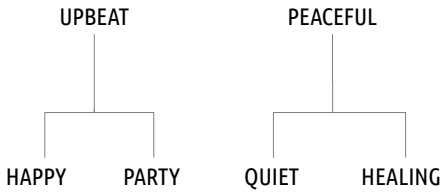


Figure 3.3: Example of how two hierarchies in the mood structure could look like and how the moods could be named.

3.1.2 Session

The user data and the metadata was aggregated and divided into sessions in order to differentiate different listening moments of the user. A session was defined as a sequence of consecutive played tracks with at most a time threshold of inactivity using the same platform. This means that any track listened to after an inactivity longer than the threshold of 15 minutes or on a new platform would have been assigned a different session than previous track. A illustration of the structure of two user sessions is shown in Table 3.2. If two users would share the same account but listen on different devices, their sessions would have been separated, and the same goes for different moments from the same user which occurred with a pause of 15 minutes.

Session id	User	Track	Session length	Track-data
0	User1	0	5 min	{Id, Name, Genre, Mood}
		1		{Id, Name, Genre, Mood}
		2		{Id, Name, Genre, Mood}
			⋮	
1	User2	0	15 min	{Id, Name, Genre, Mood}
		1		{Id, Name, Genre, Mood}
		2		{Id, Name, Genre, Mood}

Table 3.2: Example of two user sessions

3.1.3 Data extraction

Each day Spotify store about 1.5 TB compressed data sent from users. To process this Spotify has a Apache Hadoop[1] cluster of 694 nodes used for offline processing of data. As of the size of all data stored, both about users and tracks, all data were stored in a distributed file system, Hadoop Distributed File System (HDFS). HDFS is preferred for large datasets as it works in tandem with the parallel processing framework MapReduce[40] and because of its scalability – scalable as more servers means more storage capacity. MapReduce processes large amounts of data at the location of the data rather than moving the data to a computational location. This means MapReduce can process without being affected by network bandwidth.

The dataset used in this thesis were stored in different sources and a chain of MapReduce jobs were needed to select a subset of users, join in the data from different sources and lastly to construct the sessions.

MapReduce is a simple model implemented with two functions; a mapper and a reducer. The mapper takes as input the raw data and produces a set of key/value pairs that the Apache framework groups together such that the same all the pairs of each key gets into the same reducer. The reducer function takes a key and a list of values as input and merges the data before outputting in the wanted format. An example of one of the MapReduce jobs created in this thesis was the one of joining user data and track data.

In the job from Fig 3.4 a mapper per row in the user and track sources is spawned. The mapper function yields the data with the unique track identifier as key. One reducer per yielded track id is spawned and joins the fields together and outputs as a new data source. After the job has outputted the final results another job is needed to create correct sessions from this unsorted source of user-track-data.

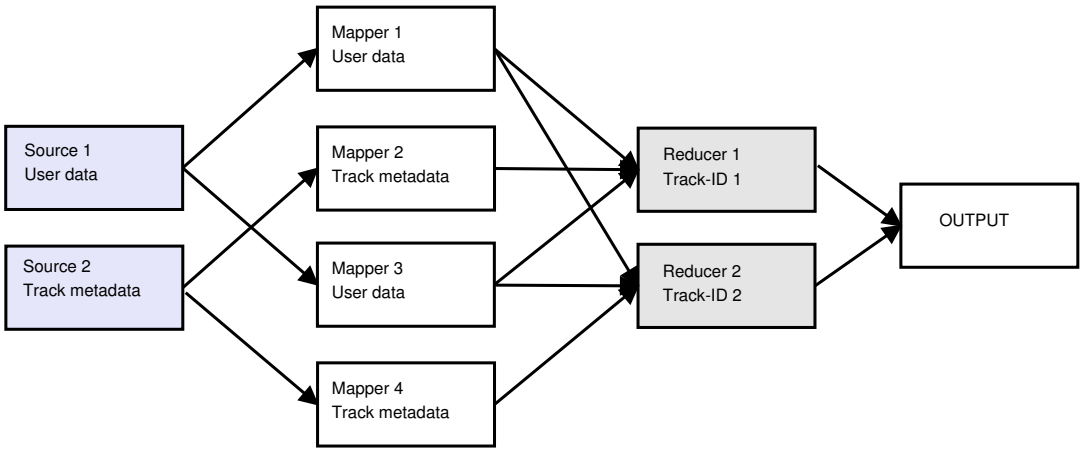


Figure 3.4: Illustration of how the MapReduce paradigm works with an example on joining user data with metadata.

[1] http://hadoop.apache.org/

3.2 Data preprocessing

The step of doing data preprocessing and cleaning is removing noise, points with abnormal values and missing value in the data to increase quality.

In the selection of user data only tracks streamed more than 30 seconds were considered as a play. This number has been set internally at Spotify and is the same used to determine when to pay an artist for a stream. Therefor user data with lower than 30 seconds were treated as noise probably caused due to wrong logging from a client or user actively skipping the track.

Sessions treated as too short were cleaned away and treated as noise. Since of the problem being modeled as document clustering a session represents a document. A document consisting of five or less word would be to short to be able to derive their meaning from. The same goes for sessions and therefor sessions of smaller length than 5 tracks were treated as noise and cleaned away.

All the preprocessing and analyzes of the dataset was done with the Python library pandas [41] and its DataFrame objects.

3.3 Data transformation

The raw session data can itself not be directly plugged into the algorithms used in the step of data mining. What attributes, feature, to use and transformation of the textual data into a vector representation to support the algorithms is first needed. This chapter explains how the previous problem was modeled as a text document problem and hence how a vector representation of the data was created.

3.3.1 Vector representation

The algorithms explained in the theoretical chapters takes as an input a numerical matrix, either a feature matrix or a distance matrix. Some algorithms, e.g. K-means, are limited to input of numerical values only as for the assignment step. The same goes applies to other similarity measures explained, hence a vector representation of the textual data is needed.

A feature matrix is a matrix of size $n\_samples \times n\_features$ where each row is a vector representation of a session and a distance matrix is a $n\_samples \times n\_sample$-sized matrix where each element is the distance, measured by a similarity measure or distance measure, between two sessions.

The bag-of-words model, commonly used in Natural Language Processing to represent text documents, were used on the sessions as in Table 3.2 to represent them as numerical vectors. In the bag-of-words model of a text document the occurrence of each word is treated as a feature and it does not take into consideration the ordering of the words.

The sessions were treated as documents where the words were the chosen metadata-information about each track. Instead of counting the number of occurrences of each

feature the weight assigned to the metadata information was used.

A session consisting of tracks in various rock-related genre with corresponding metadata tag could have had the following document representation:

$$S_1 = \text{"ROCK INDIE ROCK METAL} \ldots \text{ENERGIZING FIERY} \ldots \text{"}$$

Where each word in the session, either being a genre or mood information, occurred as many times as the weight to the metadata information. As the structure of the metadata were hierarchical, each level was added as a feature and not just the most granular one. This was done to explain the hierarchical structure to the clustering algorithm. E.g. if a track would be Brit pop with weight 50 and Indie pop with weight 50 its vector representation would be (with the genres from Fig 3.2)

$$\begin{array}{lccc} & \textit{Brit pop} & \textit{Indie pop} & \textit{Indie rock} \\ \text{s} = & <50, & 50, & 100> \end{array}$$

This is important to note because this explains the hierarchy to the cost of correlation between features. A special case is with very rare genre or moods which will introduce extra features that always take the same values. This increases the dimensionality without adding extra information to the algorithms.

With the bag-of-words-model the session data is represented as a sample-feature matrix in Fig 3.5. Each row corresponds to a session and each column to a feature (metadata tag). The attributes $a_{1,i}$ is the weight of the $i : th$ feature in session 1.

$$S_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

Figure 3.5: The sample-feature matrix. Each row is one session and each column is the features selected to use. Each element is the total weight of that feature in that session.

To account for the different length of sessions the $S_{m,n}$ was row vise normalized by the number of tracks in that session. Different techniques of adjusting the data such as standardization, normalization or whitening exists. Before running basic K-means the data were normalized per feature by dividing with its standard deviation, also known as whitening.

The session-matrix were stored in a Python numpy array [42] generated by the bag-of-words vectorizer from Scikit-learn [43]

3.4 Data mining methods

The clustering algorithms evaluated were K-means, k-medoids using cosine similarity, average and complete hierarchical clustering both of which using cosine similarity.

First the different clustering methods performance were evaluated when using only genre as feature. This was chosen as first evaluation as the results would be easier to manually detect noise or bad clustering, due to it being easier to interpret what the feature names mean and represent. We wanted to be confident that the method of choice would yielded interpretable and reasonable solutions in the genre case and when moving on to adding mood-tag.

The evaluation of clustering methods was based on total average silhouette score, distribution of cluster sizes and average silhouette score per cluster. The total average silhouette score reflects the overall clustering but does not take into consideration the individual clusters silhouette score. As it is a mean one big cluster score can hide lots of small negative, hence the average silhouette score per cluster. Also, the silhouette score is not a metric of information gain and measuring the amount of information that can be extracted from a model is hard. The distribution of cluster sizes reveal the shape of clusters produced by each method and is of interest as different methods might find different patterns in the data.

Then per each chosen, and motivated, clustering method the logistic regression experiment was trained. For each cluster a logistic regression model was trained and the estimated coefficient weights were used to explain the clustering. The regularization parameter, C, in the logistic regression was first chosen by 10-fold cross-validation on the parameters $[0.001, 0.002, 0.003, 0.005]$ and the one with the highest accuracy was chosen, per cluster. If the resulting solution did not show any positive descriptors, meaning the regularization was to hard, it was repeated but with parameters multiplied with 10 until regularization yield descriptors. The chosen values for the C-parameter to search in was due to the want of hard regularization meaning a feature selection making the results interpretable but the same level of hardness is not applicable to all cluster sizes.

3.4.1 Materials

The agglomerative hierarchical clustering used in this thesis was the one from the SciPy-library [42]. It comes with methods to cluster using the different linkage criteria, visualize the linkage matrix in dendrogram and different methods of cutting the trees into flat clustering. As HAC is a similarity-based clustering a distance matrix using the cosine distance was generated from the feature-matrix.

The approaches of forming flat clustering from a linkage matrix used in this thesis is the cut based on distance and maximum number of clusters. The distance based will form a flat clustering so that the cophenetic distance between two samples have not a greater distance than a threshold t The maximum number of cluster will try to find a minimum cophenetic distance that forms no more than the defined k number of clusters.

The standard K-means algorithm used was the one from Scikit-learn [43]. Each run was repeated 90 times with different centroid seeding and initialized with k-means++ to speed up convergence. The number of repetitions, 90, was determined after experimentally testing when the output seems to have converged to a good (local) optimum.

As for the K-medoids algorithm an implementation from the library PyCluster [44, 45] was used. The distance matrix used was created using an equivalence to cosine

similarity, namely uncentered Pearson correlation [45]. The K-medoids was repeated 90 times to converge to a good value.

The silhouette score was calculated using the implementation in Scikit-learn. The same distance measure was used as in the clustering that was evaluated.

The implementation of Logistic regression in [43] were used. The regularization parameter in this library is the inverse of regularization strength meaning $C = \frac{1}{\lambda}$. The dataset, labeled as the cluster being described or not the cluster, is under/oversampled proportional to the size of the cluster. This is beneficial when the cluster is very large or very big.

4

Results

In this section the details on the results of applying the data mining methods is presented. First the dataset used in the experiments is explained to give the reader an understanding of what the information were in the dataset.

For each of the different clustering algorithms the experiments are reported and compared. The best model is furthered explained with the logistic regression model. A discussions about the obtained results and used method is given in the next chapter.

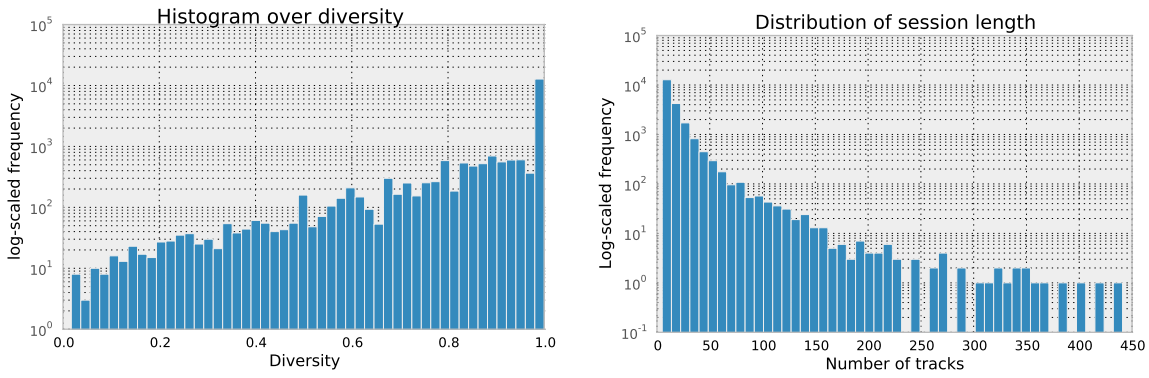
4.1 Dataset

The data consisted of a subset of Swedish users in the date range 2013-12-09 23:41 - 2013-12-10 23:59. In total the dataset consisted of 20991 sessions with 10089 unique users.

The coverage of metadata on tracks were not complete. Meaning that all tracks streamed by users did not necessarily have a genre or mood-tag. This was not seen as a problem as the percentage of tracks assigned with genre were 85% and for mood 60%. All sessions were guaranteed to have at least one track with genre-information due to the preprocessing step. No attempt to fill in this missing data was made as the percentage was sufficiently high enough.

Naturally, the different session varied in length. A histogram of how many tracks each sessions had can be seen in sub figure (b) in Fig 4.1. The distribution is highly skewed and the y-axis has been log-scaled for visual support. The majority of the sessions had around $5 - 50$ tracks. As stated in methodology, all sessions under five tracks were cleaned away.

No experimental study were made with other inactivity thresholds than 15 minutes when generating the sessions. Most of the sessions were ended due to the inactivity constraint or being the last track of that user in the analyzed dataset. The constraints of ending an user session were the user being inactive for a period of time, changing



(a) (b)

Figure 4.1: Histograms illustrating the sessions of the dataset. a) Song diversity b) Number of tracks.

platform, client or stop playing (last track). The full data for session ending is found in Table 4.1.

The majority of the sessions in the dataset had a song diversity close to one as can be seen in sub figure (a) in Fig 4.1. The song diversity of a sessions is the ratio of how many unique tracks an user listened to during a session. Diversity close to 0 means that the user only listened to the same tracks and close to 1 means that the user listened to different songs in the session. This shows that almost every sessions in the dataset is a collection of unique tracks and not just repetitions of the same. Even thou a repetitive session could be a moment, song-diversity close to one yields more combinations of genres and moods hence more information.

Reason ended	
Inactivity: (15)	13378
last track	6682
user changed platform	827
client changed	104

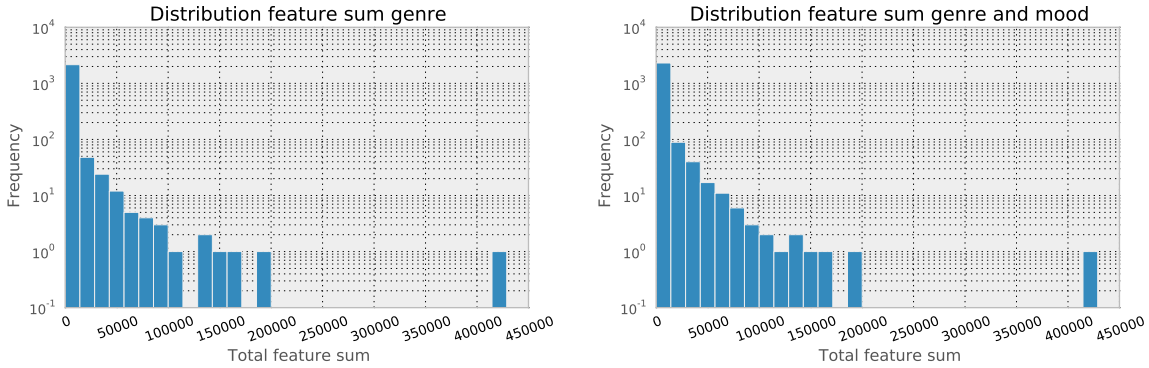
Table 4.1: Data of reasons why sessions were ended.

4.1.1 Feature selection

The creation of the session-feature matrix $S_{m,n}$ depended on what information about each track were included and not. When creating the matrix with only genres as features, including all the steps in the hierarchy, the size were $(20991, 2255)$. All sessions were represented since sessions with no track with genre information were cleaned away. The high dimension, 2255, were the total number of unique genre-features represented in the sessions.

When using both genre and mood the dimensionality slightly increased with resulting size $(20991, 2482)$. The dimensionality is the sum of the previous one and the unique mood tags from the sessions, in this dataset being 227.

The distribution of feature weights for both cases can be found in Fig 4.2. The feature weight is the total sum of a feature over all sessions. It is the same as the sum of one column in $S_{m,n}$. Also, these distributions were skewed and the y-axis has been log scaled. Sub figure (a) show the distribution of feature sum with genre only. The extreme values were the popular top-level genres such as pop, rap, dance & house, rock and Indie rock. Sub figure (b) show the distribution with both genre and mood. Mood did not, as seen, have a big impact on the distribution.



(a) (b)

Figure 4.2: Distributions of feature sums in session matrix

4.2 Clustering evaluation

For complete hierarchical clustering and k-means the result is only presented with genres only. They were rejected early on in the project and therefore not evaluated with update feature set.

The dendrogram visualizing the complete hierarchical clustering is shown in Fig 4.3. It is plotted only in range distance $0.8 - 1$ as the tree at distances below only shows a "wall" of branches and with truncated samples to condense the dendrogram. Only the last 500 nodes of more than one observation in the linkage and the leaf nodes are shown in the dendrogram. The rest were contracted into leaf nodes. Nodes are clusters at different levels in the linkage. By doing this, the dendrogram shows the linkage structure of the clustering but leaves out information such as how many samples each branch consists of.

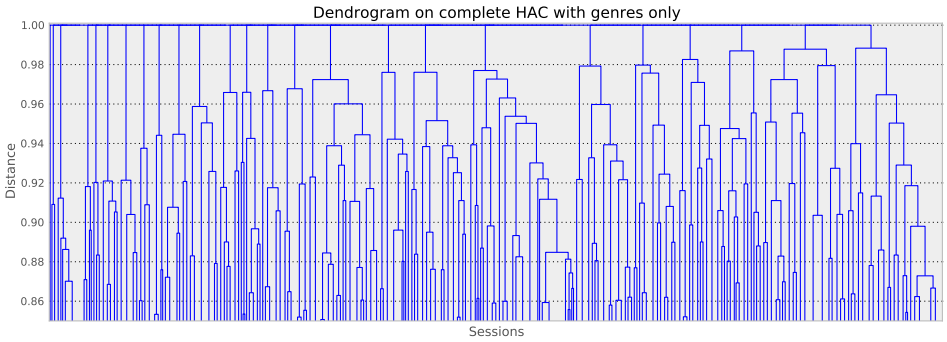


Figure 4.3: Top of the dendrogram of complete hierarchical clustering on data with genre only.

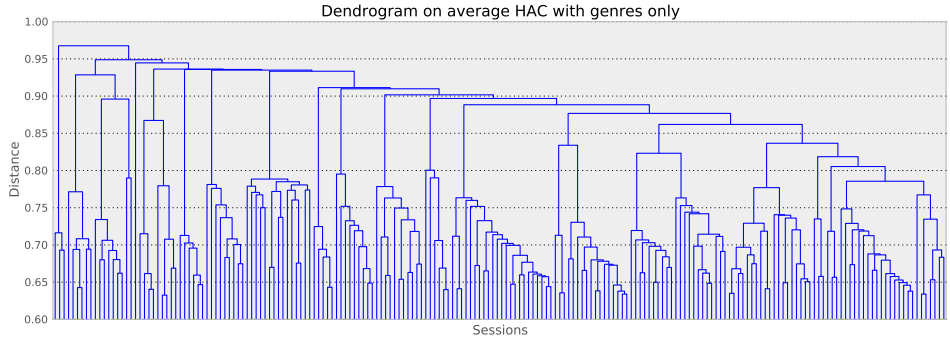
Two important things to note from the dendrogram in Fig 4.3 is the fact that least number of clusters after 1 is 25 and that the heights of the branches in general are small as can bee seen by the ticks on the y-axis. This shows that the different clusters are similar which could occur when the hierarchical clustering is not able to find natural clusters and results in noisy clusters. The dendrogram accompanied with further plots explaining the clustering can be found in Appendix A.

K-means was the only algorithm that used the Euclidean distance and not the cosine. The silhouette score were high for low k such as 2,3 but with further inspection on distribution of the clusters sizes (in terms of number of samples in the clusters) the clustering were skewed with one big cluster, including almost all samples and one small. This is one of the problems with relying on the silhouette score as it will only measure how well the samples fit the cluster it is in compared to the other clusters and not the information gain from such clustering. The silhouette score oscillated around 0 for $k = 2 - 20$. A plot of the silhouette score from K-means is also given in Appendix A. The sub figure (a) show the silhouette score at different k and (b) the within-cluster sum of squares WCSS at different k.

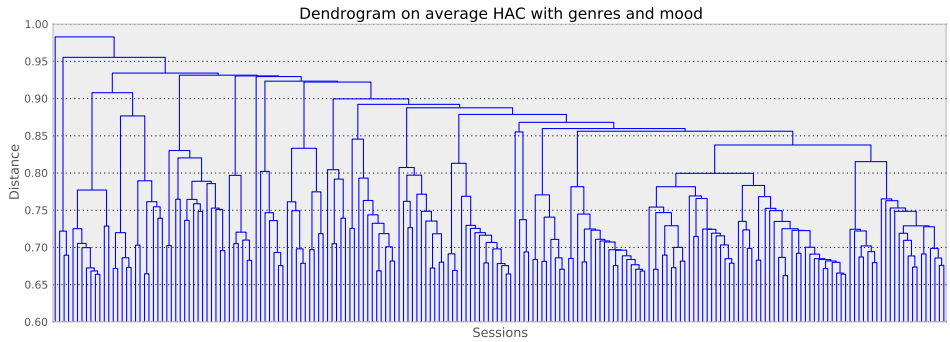
Average hierarchical clustering

The average linking hierarchical clustering showed signs in the dendrogram, shown in (a) in Fig 4.4, of being able to separate the data, unlike the complete hierarchical recently seen. As the dendrogram in the complete HAC it is zoomed but at lower distance 0.6 and the dendrogram is rendered using same truncation but now only showing the last 200 nodes. The dendrogram shows clearly how the merging at each hierarchy was made by the algorithm and how all sessions ended up in the last cluster. The lengths of the branches indicates dissimilarity between clusters merged which could indicate good cutting points. The longer, the less similar the merged clusters are. Further explanation about cluster sizes and silhouette score at each distance in the dendrogram is shown in Fig 4.6.

The silhouette score peaks around distance 0.8 which as of the dendrogram looks like a reasonable cutting point. This does not indicate a good information gain but gives us more confidence that the clustering might be of natural and useful information. The maximum silhouette score is approximately the same as for the the one of complete linkage, namely 0.199580. But with the information of the linking seen in the dendrogram the clusters seems more natural and more reliable than the complete hence we will conclude that this approach works better. The silhouette score per cluster is found in Fig 4.5. These results are given for both genre and genre and mood.



(a)



(b)

Figure 4.4: Dendrogram of average hierarchical cluster with (a) genres (b) genres and mood. Zoomed at $d = 0.6$ and truncated only showing the last 200 nodes

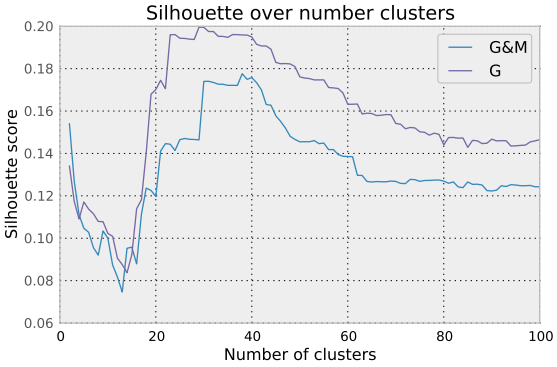
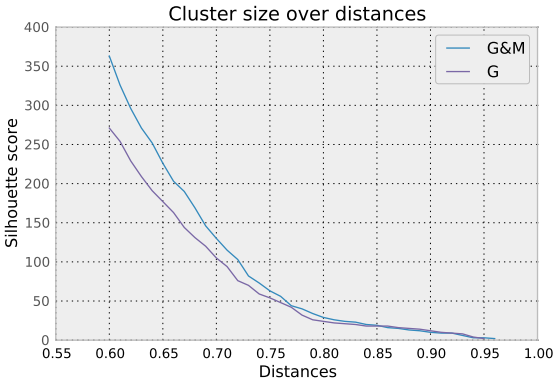
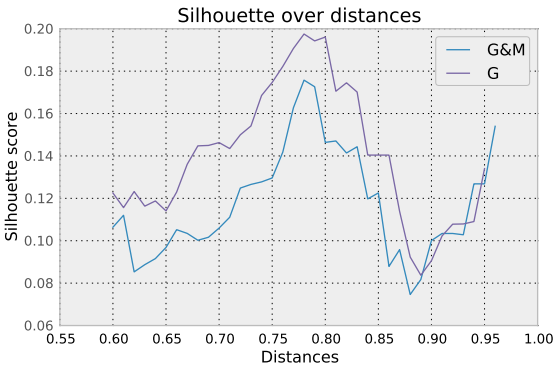


Figure 4.5: Results of hierarchical Average clustering on genre (purple) and genre and mood (blue). Shows the silhouette score when flattening using different number of clusters.



(a)



(b)

Figure 4.6: Results of hierarchical Average clustering on genre (purple) and genre and mood (blue). Sub figure (a) the cluster sizes at each distance. (b) the silhouette score at each distance.

We can in Fig 4.5 see a peak of silhouette score around $20 - 40$ clusters which is a sufficiently high number of clusters to be interesting. Moreover we are interested in how the cluster sizes and silhouette score per cluster changes when increasing K to get knowledge about how the clustering algorithm evolves. Looking at the behavior of the clustering in HAC with four different number of clusters, illustrated in Fig 4.7, its visible that for high number of clusters, $k = 50 - 150$ the silhouette score per cluster stays positive per cluster but there is a skewness in the distribution. Noticeable is the biggest cluster which has roughly the same size through increasing k. Each sub figure in Fig 4.7 is the cluster size distribution when cutting the average HAC at a different number of clusters. (a) is 50 cluster, (b) 100, (c) 150 and (d) 200. At $k = 200$ the biggest cluster is of size 5267 and there are 104 cluster of size lesser or equal to 10 and there are negative silhouette scores per clusters. This is of interest in the evaluation as deciding number of clusters is a hard problem and after an obtained clustering we are interested in what would happen if we increased or decreased k.

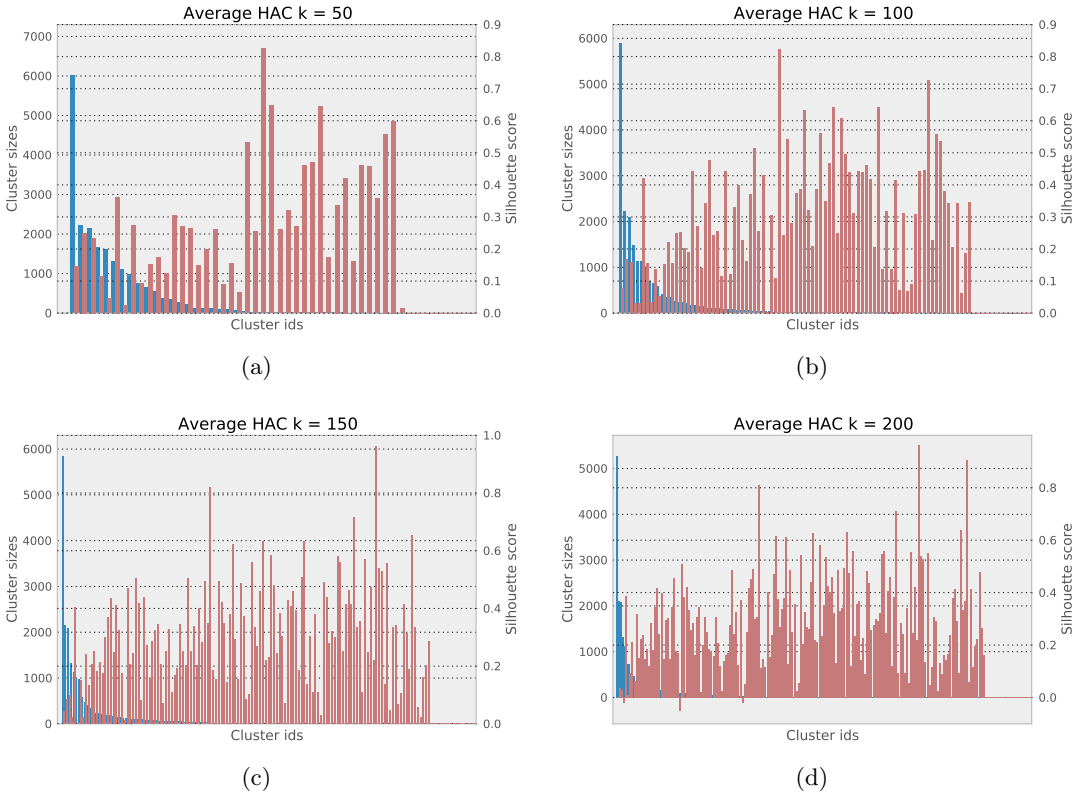
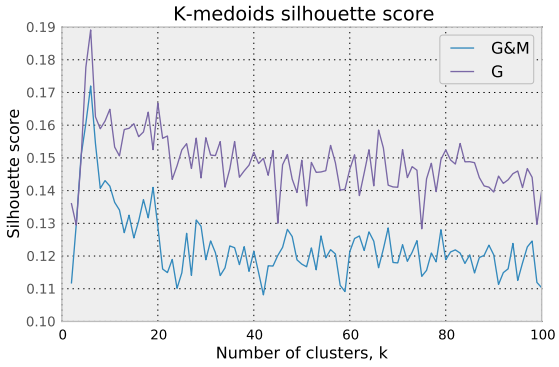


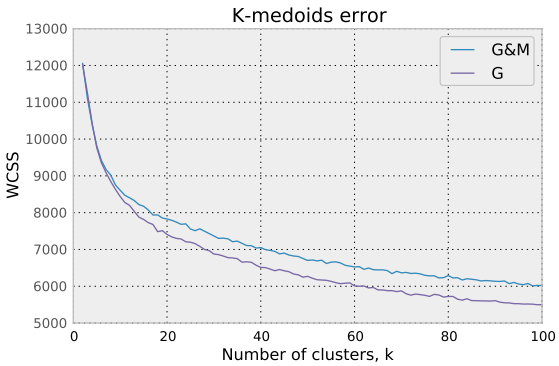
Figure 4.7: Distribution of cluster sizes when cutting at four different number of clusters, 50 in (a), 100 in (b), 150 in (c) and 200 in (d). The blue bars represents the cluster size and the red bars the silhouette score.

K-medoids

The K-medoids, the outlier robust alternative to K-means who is clustering a distance matrix based on cosine, performs nearly as good as the average hierarchical clustering in terms of silhouette score. But, as stated before, the silhouette score is not a measure of information gain but rather sample cluster-similarity compared with other clusters. The silhouette score-plot of K-medoids can be seen in Fig 4.8. Its oscillation is due to K-medoids not guaranteeing global optimum and the plot being generated out of an average of three runs over the range of k. More runs would smoothen the curve. K-medoids was, as stated previously, ran 90 times for each k, and the result with lowest within-cluster sum of squares (WCSS) was chosen. Both genre only and with mood peaks at $k = 6$ and then slightly decreases. The difference in silhouette score between genre and genre and mood seems to be approximately the same difference as in average HAC.



(a)



(b)

Figure 4.8: K-medoids silhouette score with both genre and genre & mood (a) shows the silhouette score over k and (b) the within-cluster sum of squares. The values are average of 3 runs.

The same analysis as from the average HAC of cluster shape behavior at increasing K but for K-medoids is given in Fig 4.9. It is by the plot clear that the distribution is more even in K-medoids than in average hierarchical clustering. It seems like K-medoids is able to avoid one big cluster, but to the cost of individual silhouette score per cluster being negative. This could indicate that breaking the big cluster that the average HAC was able to make sub clusters of was the wrong thing to do.

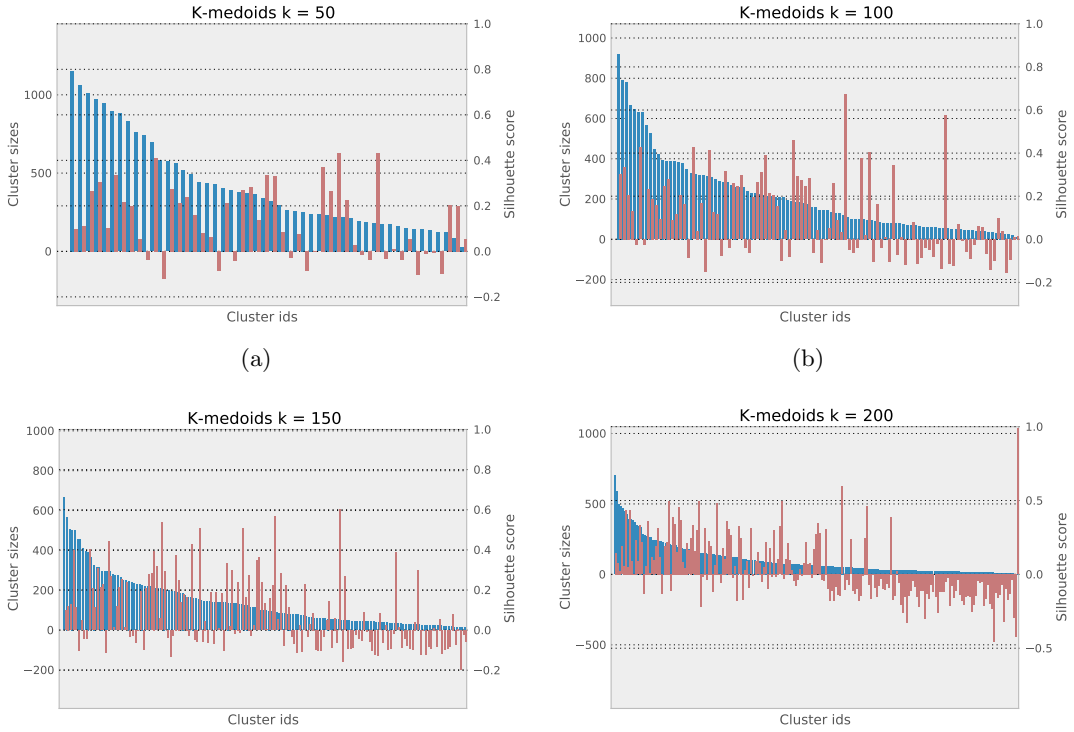


Figure 4.9: Distribution of cluster sizes when k in k-medoids changes levels, 50 in (a), 100 in (b), 150 in (c) and 200 in (d). The blue bars represent the cluster size and the red bars the silhouette score.

4.3 Genre clusters

After inspecting the metrics showing the differences between the two superior algorithms, K-medoids and Average HAC, it's time to try to interpret the results. As for the cluster interpretation the genre only will explained first. This is aligned with the reasoning of how the final results were obtained.

By training a logistic regression for each cluster we obtain a logistic regression with feature weights, as previously explained. The result of each logistic regression is visualized with a bar plot of height being the weight of each feature coefficient and the label

being the name of the feature. Only the relevant figures is shown here, in the result chapter, and the rest is left in Appendix for the reader.

K-medoids

The silhouette score evaluation of K-medoids seen in Fig 4.8 showed a peak of silhouette score at $k = 6$ which makes this interesting to investigate further.

Looking more closely at the clustering with $k = 6$ in Fig 4.10, with genres only, where the blue bars and the left y-axis is the cluster sizes per cluster and the right y-axis is the silhouette score with red colored bars. It can be seen that the silhouette score per cluster is strict positive in each cluster and that the sizes of the clusters is rather evenly distributed but with one big cluster. The big cluster is of size 6800 and this cluster, and the rest of the cluster due to small k, could be containing interesting granular sub clusters. $k = 6$ is a very low number of clusters when looking at a dataset of this size and with the natural intuitive feeling that there should be more specialized groups of sessions.

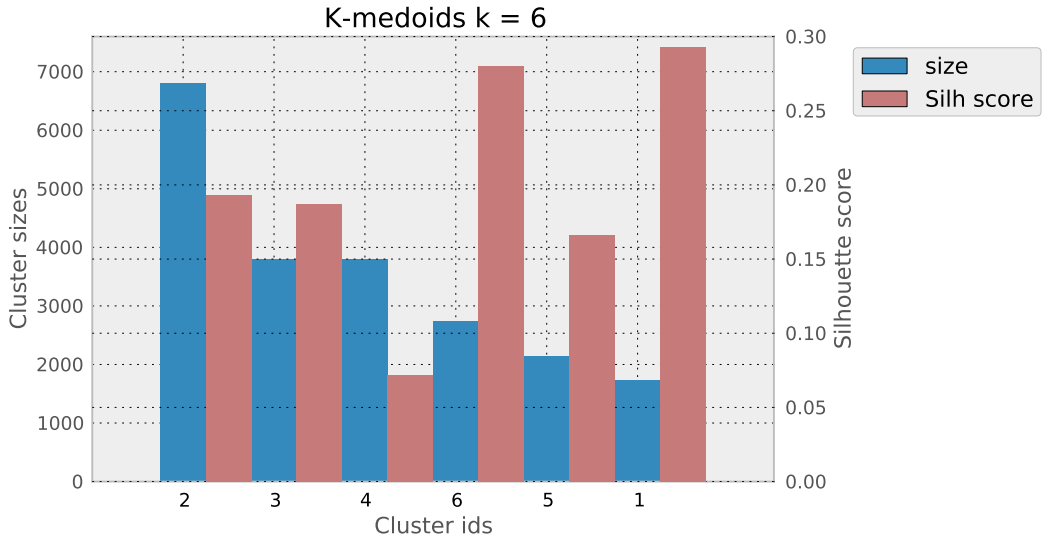


Figure 4.10: K-medoids, k=6, with genres only. The left y-axis is the size of each cluster, blue bar, and the right is the silhouette score, red bar.

The clusters in order of decreasing size were named Pop, Dance & House, Rock, Rap, Indie Rock and Holiday. Plots of the logistic regressions for each cluster can be found in Appendix B.

Average HAC

As of the plot in (c) Fig 4.6 the maximum silhouette score achieved, with genres only, is at max with 29 number of clusters. Investigating this clustering further shows a skewed

distribution of clusters sizes with a big cluster of size 6000, as previously indicated. In Fig 4.11 the x-axis represents each cluster, the left y-axis, cluster size, has been log-scaled to visualize the sizes in the tail of the distribution, and the right y-axis and the red bars are the silhouette score for the cluster.

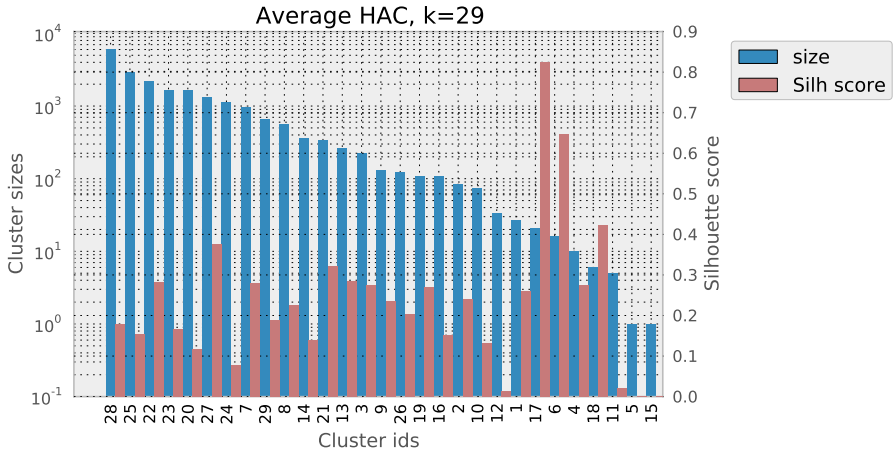


Figure 4.11: Cluster sizes and silhouette scores per cluster. Average hierarchical clustering cut at 29 clusters with only genres.

The different clusters were, in order of size, named by the highest weighted feature; Pop, Dance & House, Rap, Indie Rock, Rock, Holiday (Christmas), Funk Rock, Metal, R&B, Punk, Country & Folk, Reggae, Soundtrack, Classical music, Jazz, Children's, Comedy, Latin, New-age music, Christian Pop, Niched electronics and Funk/Blues. All of the weight-plots can be found in B.1 in Appendix B. Two clusters of size 1 was found, id 5 and 15, they were removed. Clusters 18 and 11 where of too small size and resulted in only one positive weight vector. For 18 this was Spoken & Word and Traditional for 11.

The resulting clusters of average hierarchical clustering can be seen in Fig B.28. Full details such as chosen C-values for each cluster, accuracy-score and information about each cluster from running the logistic regression on the average HAC with 29 clusters can be found in the table in B.1 Appendix B.

4.4 Genre and mood clusters

No K of competing size with Average HAC was found for K-medoids that yielded positive silhouette scores for each individual cluster. Therefore only average hierarchical cluster was used for both genre and mood clusters.

The silhouette score in average HAC peaked at $k = 38$ (from Fig 4.6). The cluster size distribution and silhouette score per cluster can be seen in Fig 4.12. This shows the same tendencies as the genre plot with $k = 29$ but there are more clusters. In total there

were 4 clusters of size 1, namely Ids $10, 23, 26, 38$, which were removed. All obtained clusters had a strict positive silhouette score.

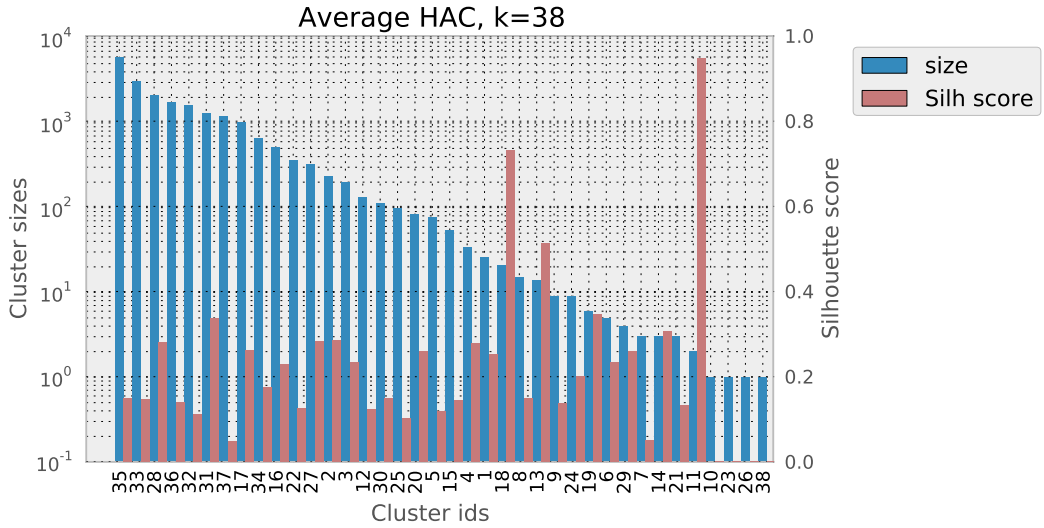


Figure 4.12: Average hierarchical clustering cut at 38 clusters with both genres and moods.

A subset of the clusters from the genre and mood with average hierarchical clustering is explained below. Only a subset is explained due to the large number of clusters and a subset is enough to prove the point of this thesis. The other clusters are of course not excluded but found in Appendix C and a full summary of the clusters with cluster ids, sizes and naming is found in Table 4.2.

Size	ID	Naming
5869	35	Pop
3116	33	Dance & House
2098	28	Rap
1762	36	Indie Rock
1632	32	Rock
1301	31	Holiday
1188	37	Alternative
1013	17	Metal
655	34	R&B
518	16	Classic U.S. Punk
363	22	Country & Folk
327	27	Spanish Pop
234	2	Progressive Electronic
199	3	Classical
132	12	Jazz
114	30	Children's
97	25	Latin
84	20	Comedy
77	5	New Celtic Trad (New Age)
54	15	Religious
34	4	Solo instrument
26	1	Funk/Blues

Table 4.2: Summary view of the clustering by Average HAC k=38 on Genre and mood. Low clusters were filtered away and the name is from the highest weighted feature coefficient in the logistic regression.

The top eight biggest clusters represents, by a human, different distinguishable music tastes. Pop, Dance & House, Rap, Indie Rock, Rock, Holiday, Alternative, Metal and R&B. As for their visualization with the logistic regression, all of them have a mix of similar genres or sub genres as positive and due to the similarity only Pop and Dance&House is shown here, the rest is found in Appendix C.

The Pop-cluster can be seen in Fig 4.13. It is the, by far, biggest cluster with 5869 sessions. The positive coefficients is of a mix of similar pop-like genres such as Indie dance, Dance, European Pop and the negative of genres we would interpret as different from Pop such as Holiday, Children's, Metal etc.

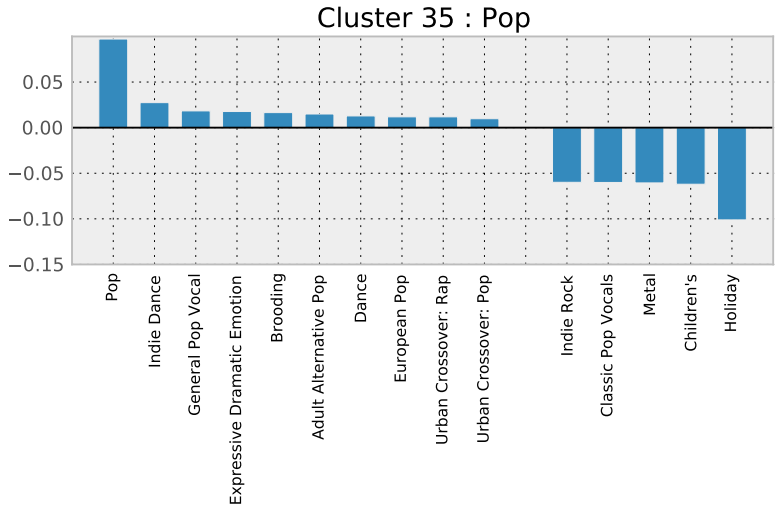


Figure 4.13: Pop-cluster. Size: 5869 Silhouette score: 0.148939409677 C: 0.005

The second most biggest cluster is the Dance & House-cluster and is shown in Fig 4.14. With the size of 3116 sessions the positive coefficients ranges over a mix genres similar to what Dance & House music could represent. The positive coefficients represents genres named Electronica, House, Dance Pop and Pop Electronica and some mixes of energetic moods can also be found.

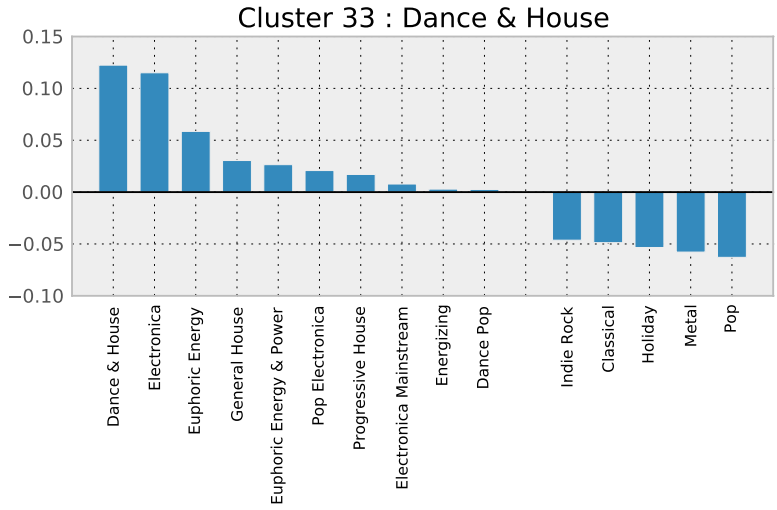


Figure 4.14: Dance & House-cluster. Size: 3116 Silhouette score: 0.146052179499 C: 0.005

As for the clusters with lower size than the eight biggest ones they seem to represent sessions in more niched genres such as Holiday, Children's, Latin, Comedy, Religious, Punk, Classical music and Electronica. Niched in the sense that they are intuitively less

popular than the biggest eight or representing some session that most probably is not the only type of session of that user.

The rest of this chapter will be dedicated to show the clusters named Holiday, Children's, Latin, Comedy and Religious. The reader is encourage to read Appendix C as a complementary to the naming in Table 4.2 to get a better understanding of what sessions the different clusters represent and to understand the clusters not explained here.

The Holiday-cluster of size 1301 session seem to be a cluster of Christmas sessions. It can be seen in Fig 4.15 and the positive features seen is Holiday, Easy listening, Christmas and Pop Christmas etc.

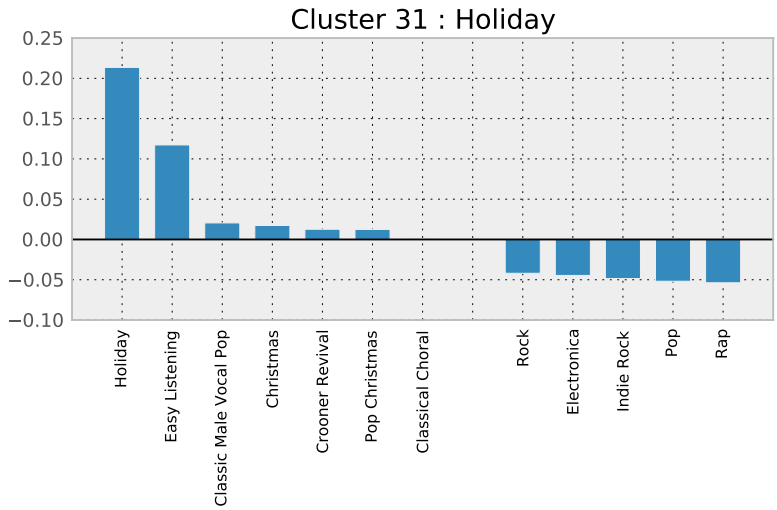


Figure 4.15: Holiday-cluster. Size: 1301 Silhouette score: 0.33772172736 C: 0.005

A cluster named Children's after a genre can be seen in Fig 4.16. What can be seen is a mix of Children's music, sentimental and melancholy tracks unlike the negative coefficients Rap, Rock and Dance & House.

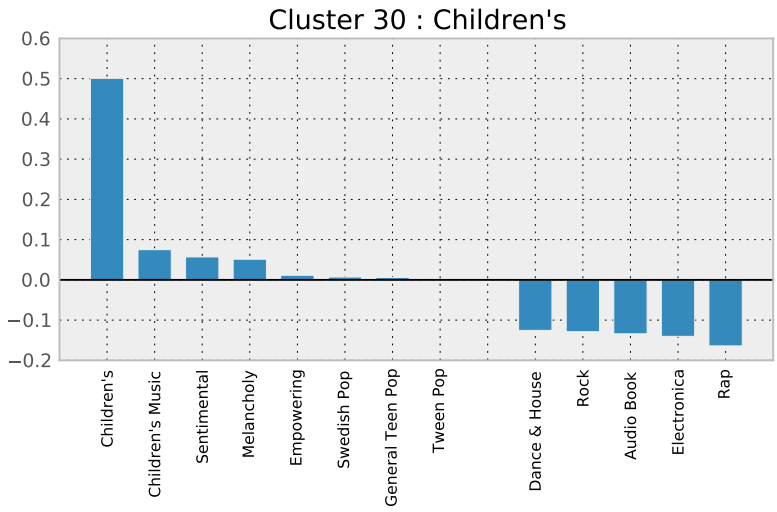


Figure 4.16: Children's-cluster. Size: 114 Silhouette score: 0.149199821065 C: 0.5

Also, an interestingly cluster found is the one named Latin. It can be found in Fig 4.17 and is a clear combination of Latin alike genres. It is reasonable to believe Latin pop is Fiery and Energizing as seen in Fig 4.17.

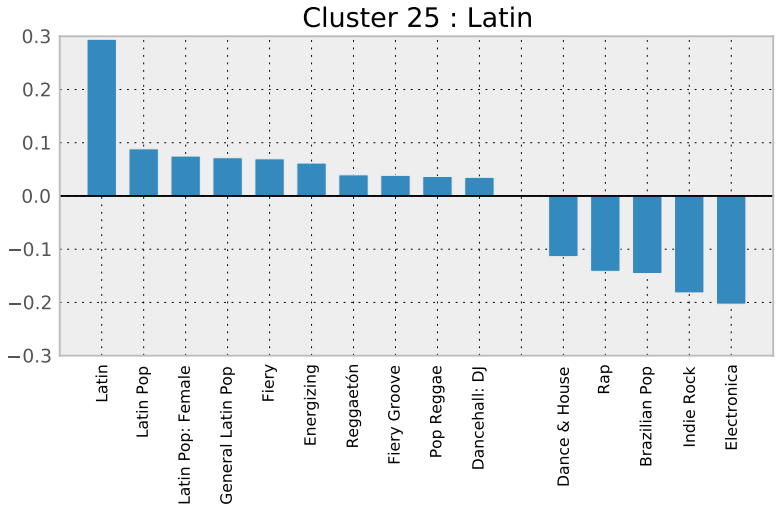


Figure 4.17: Latin-cluster. Size: 97 Silhouette score: 0.101858307823 C: 0.5

By now, the cluster size has decreased down to below 100 sessions. The cluster in Fig 4.18 is a mix of Comedy music, Swedish pop and Comedy rock.

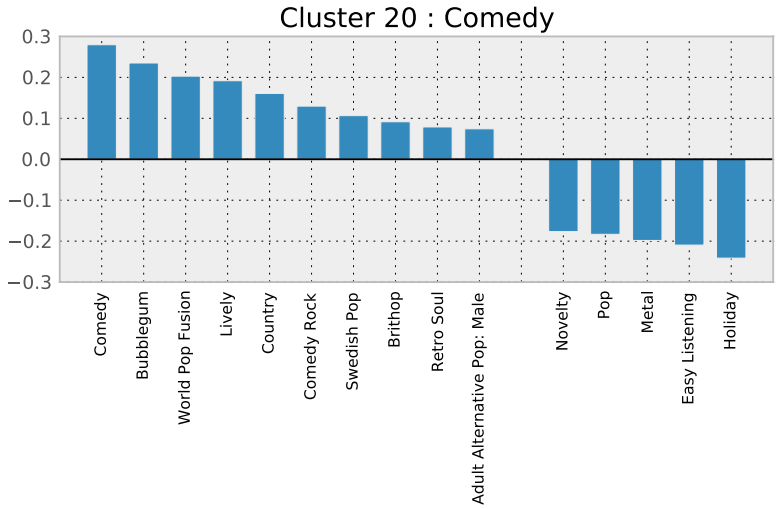


Figure 4.18: Comedy-cluster. Size: 84 Silhouette score: 0.259850151927 C: 2

Lastly a cluster named Religious is shown in Fig 4.19. What can be seen is a cluster of sessions with tracks in different Christian genres such as Christian Pop, Christian Metal etc.

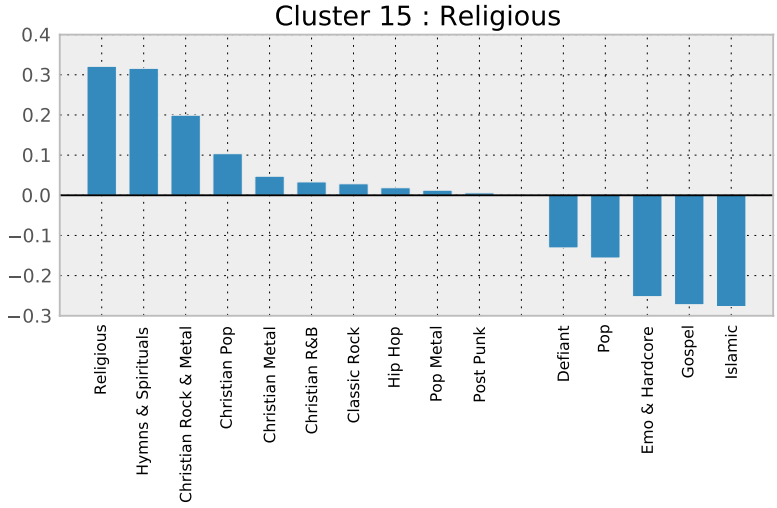


Figure 4.19: Religious-cluster. Size: 54 Silhouette score: 0.145533996878 C: 2

5

Discussion

Going back to the objective of the thesis; answer the question if it is possible to use the new descriptive data to find music moments and if so, answer what the popular moments amongst user are. Music moments is a very fuzzy term relating to some choice of music to accompany an environment or situation. When interpreting user music moments as groups of sessions we have shown that the new rich descriptive data can very well help find combinations of genres and moods. Due to the small dataset, compared to the total dataset at Spotify, used in this project we can not conclude that the eight biggest clusters found here also represents the eight most popular moments by the user. They will most probably change depending on time frame, country and size of the dataset.

As for the comparison between the two superior models – average hierarchical clustering and k-medoids – average hierarchical clustering was concluded to be superior in terms of positive silhouette score per cluster at high k. The biggest clusters in average HAC was thou also found by K-medoids but in the later it was grouped together with cluster that average HAC seemed to be able to separated. The biggest clusters represented different top level genres, as illustrated with the Figures of the Pop-cluster and Dance & House in Fig 4.13 and Fig 4.14. This result is a good indicator that the model works. We expected to find a lot of music moments within the same genres and as the model was able to separate Pop sessions from Rock session etc. we see tendencies that it has managed to cluster what a human represent as similar sessions.

Another example of a cluster found by both K-medoids and average HAC was the Christmas cluster. But, in the K-medoids the found Holiday/Christmas cluster were larger and the positive coefficients showed genres such as Jazz, which we can see in Table 4.2 that the average HAC was able to separate into its own cluster.

Surprisingly in the results is the size, or popularity, of the Christmas sessions. This result proves the usefulness of session-based analysis and the new descriptive data. Christmas music is most probably not listened to equally distributed over the entire year, but very concentrated around December. This means that users Christmas sessions repre-

sent so called moments specific to time and the fact that the model manages to find such sessions is very positive results.

Another surprising finding was The Children's cluster. The cluster could represent a moment where parents play music to their children's or the session of an account belonging to a child. Anyhow, these sessions also most probably represent a rare session of a user, imagining that a user does not only listen to Children's music but only at some specific occasion. A special music moment.

The cluster of Latin music with Latin Pop, Fiery, Energizing and some positive coefficients with Pop Reggae showed, unlike some of the other cluster, how useful the logistic regression proved to be to help understand the combinations of features in clusters. When looking at the biggest clusters, only seeing very similar genres that could all be in the same branch of the hierarchy, the Latin-cluster shows a mix of genres and moods that a human could understand belonged to the same moment even thou not being in the same hierarchy.

The comedy cluster contains genres such as Bubblegum, Comedy, Swedish Pop and Birthpop and a mood feature: Lively. Comedy music is artists such as Tenacious D, The lonely island and the flight of the concords. There probably exists user who only listens to these sessions but this kind of music might be special music moments that is not the most popular moment of each user who has listened to them.

Last cluster shown was the one named Religious. It contained genres related to Christian music. Christian Rock & Metal, Pop, Metal, R&B are all shown in cluster. What connects these sessions is the lyrics related to religion whilst instrumentally they are similar to their corresponding top-level genres such as Pop, Metal etc. It is interesting that the model manages to find a cluster of sessions indicating a music moments of mixed genres but connected by the top level genre Religious. Also, the positive features shows other genres such as Hip Hop and Pop metal, which could be due to tracks being both Christian Metal and Pop Metal as of the structure of the genre data, or it could be that the user listened to tracks not being categorized as religious.

The difference between clustering with genres only and both genres and moods were not as biggest as initially hoped. As the moods tags is tagged based on the content of a track whilst genres is manually assigned the hope was that moods would be able to differentiate two by genre-looking similar sessions whilst they would be contently different. One example would be two pop-sessions, one upbeat and one more slow in tempo. All tracks were of type Pop but the mood tag would have been different, making the model separating these into two clusters. This might have happened but it is very hard to tell from the clustering. Looking at the cluster results with mood the biggest cluster is of size 5869 whilst with only genres it is of size 6010. Obviously, with genre and mood this bigger cluster is smaller but its hard to tell if this is due to increasing k in genre and mood or that the mood-tag was able to separate sessions.

5.1 Clustering algorithms

One could prior have questioned the use of hierarchical clustering in a such high-dimensional and large dataset. But as this project was carried out in an exploratory way and with the advantage of not needing to predetermining the number of clusters in HAC it was an attractive method to try. Also, it showed interesting properties with increasing number of clusters that other alternatives, such as K-medoids did not show. The conclusion was that average hierarchical clustering performed better than the alternatives. This goes in line with the modeling as a text document clustering problem and the previous work in [28, 30].

Not much hope was given in Euclidean distance due to the size of dimensionality and the popularity and proved performance of cosine similarity in similar domains e.g. text document clustering. The work in [31] showed that Cosine and Jaccard coefficient were superior in the document domain. And looking at the nice grouping of similar genres and moods in the clustering, cosine seamed to have captured the human-definition of similarity between sessions as well.

To generalize the results from this work the dataset needs to be scaled up. Clustering of subsets in the larger dataset is recommended in subsets where the average HAC is feasible to run. Remember that the average HAC is heavy to run and less efficient than the other proposed methods, but as of the results show it is an interesting method worth further investigation in. It might, with lack in quality, be possible to use K-medoids for a bigger dataset of size over 100 000 sessions.

5.2 General method

The research methodology used in this project has been discussed and explained extensively in both Related work and Methodology chapters. Besides following a research, step-by-step methodology, the work has been carried out by creating minimum valuable products (MVP). First iteration was to get just a clustering result, no matter what technique was used or parameters was set. Then, in the next iteration improved by tweaking parameters according to the give knowledge and everything was repeated again. This has helped learning what time each step will take and introduces potential problems and choices that had to be made early on in the thesis. One can spend almost all available time finding new ways of cleaning the data or alternative clustering algorithms to test but as a thesis has its time constraint the methodology has helped staying focused and delivering results.

As for the discussion, I can not exclude to mention the use of Python as a scientific programming language. Thou IPython, Scikit-learn, Pandas, Numpy etc. has been excellent and very fun to use the use of R might have simplified and speed up the work in the thesis. The choice of Python was due to this being the standard language for data crunching at Spotify and to support the community of Data mining and Machine Learning in Python.

As previously mentioned with unsupervised learning. There is no perfect or obvious

way of evaluating a solution. The approach in this thesis has been to rely on domain knowledge about what we could expect from music data, a metric for model selection in silhouette score and a will of finding compact clusters that explain a previously unknown type of session. K-medoids showed clear clusters with $k = 6$ but that k was too low when looking for more specific sets of sessions hence it was natural to continue with Average HAC that produced positive silhouette score clusters at higher K. There are more metrics, such as Dunn Index and such that could have helped indicating good number of clusters, but due to lack of time and implementations in used libraries this was not tried in this thesis.

The silhouette score is not a good metric to only rely on in evaluation of clustering but it gives an understanding and guidance if a model is doing really bad or descent. The resulting silhouette scores, all around or below 0.2 in this thesis is, according to literature, rather low. But the clusters anyhow show a clear grouping as of the naming. We believe Pop sessions are separated from Metal and classical music, as shown in the clustering. This thesis shows that a low silhouette score does not imply bad information gain, as previously stated.

Also, the use of a classification model, and more specifically logistic regression, to explain clustering results is to the author not known to be widely used, but with this work indicated to be useful in a similar problem setting of course. One of the benefits with this work was the semantic understanding of the names of the features. Decision trees has been successfully used to explain important features but logistic regression were more attractive as they provided negative coefficients and hence yielding better interpretation of the clusters.

6

Conclusion

This thesis has focused on the exploratory problem of finding groups of user sessions in a large dataset of music listening behaviors by modeling the problem as in previous research in text document analysis. The goal was to find music moments defined by users and understand if a new set of descriptive data could group sessions of music and make them humanly interpretable.

By digging into raw user data from Spotify, sessions were generated and transformed into a format applicable to data mining methods. Basic preprocessing of the data was applied with rules on what tracks to determine as streams and which sessions were long enough to contain interesting information.

It is shown in this thesis that user session-based clustering with the new descriptive data is able to cluster user sessions that capture what a human define as similar sessions. The findings of cluster such as those named Christmas, Children's and Comedy rock are moments that normal approaches that take into consideration the entire listening history might not catch and proves the need for session-based clustering.

A range of different, basic, clustering techniques was evaluated in the setting and both complete hierarchical clustering and basic k-means was discarded due to bad performance. The choice of cosine similarity as distance measure showed useful as of interpretable clusters that seem within genres. I.e. with the resulting clusters the cosine similarity seems to have captured the similarity that we wanted.

By looking at silhouette score, cluster size distributions and average silhouette score per cluster the different obtained clusters were evaluated and compared. The models indicating best performance were selected and further investigated showing interesting clusters.

This work shows yet again the difficulty of unsupervised learning, usefulness of domain knowledge and the power of being able to semantically interpret results. The decision not to apply dimensionality reduction techniques and similar that could remove the semantical understanding proved useful as a lot of new insights were gained

on interpreting the obtained clusters.

By treating each cluster as a type of moment, moments with music sessions of Rap, R&B, Dance & House. These three are genres a human would say is rather similar but the clustering method managed to separate these sessions into some more detailed explanation. There seem to be different moments of Rock, Metal, Indie Rock, Alternative and Punk indicating popularity in these genres and proves the correctness of the model. Classical music is also found in three different clusters which are the Classical, Solo instrument and the New age clusters. Besides these groups there is Christmas, Blues, Pop that according to this project is important to support in the client.

We conclude that this new descriptive data can help find user moments as what was asked for. And we can also conclude that the popular moments of users seems as of this project to be within the top-genres Pop, Dance & House, Rap, Indie Rock, Rock, Holiday, Alternative and Metal. All of which with the average HAC were clusters of over 1000 sessions.

6.1 Future work

There is a lot of extensions and alternative ways of treating this problem. Most closely to this work would be to further manually investigate the sessions in each obtained cluster. Such as the session lengths, number of unique users, most occurred tracks and other interesting metrics to get more information on if the obtained clustering is good or bad and understand what sessions it contains.

As for the preprocessing there are different ways this could be extended. Experimentally evaluating the difference in change of time threshold in the session generation would be interesting. The choice of 15 minutes was based on previous work at Spotify and also previous research indicating that its reasonable but as the generation of sessions is such a cornerstone in this approach that it would be interesting. Since there is no metric or measure indicating how good the resulting clustering were, it is hard to evaluated such change in parameter.

Skipped tracks were not filtered away nor treated differently in this work. It is of interest to see the impact and change when removing such tracks. A skip is when the user actively choses not to listen to that song more. If it is because the song is actually over and not playing anymore sound or if the suddenly realizes it wants to listen to a similar but different track is not told by the field but still of interest.

The context from where the user was playing from was not considered. If the user only shuffled a playlist, artist page or some top list and treating the service as a radio that only serves the user different type of music with no intent behind the choice of tracks. Also, sessions from the radio feature was included, which means the choice of tracks were based on a recommendation algorithm and not a moment.

Doing the same analysis but in a different time and country could show how generalizable this results is. This was not done due to time constraints and an eager to investigate the given dataset further. Also, going even further down in a dataset looking at only a couple of users could reveal interesting clusters. I.e. almost personal data but

with longer time frame.

As for clustering algorithms to apply to this problem it might be obvious that a soft clustering model could be useful. By not treating a session as only being able to belong to one cluster and instead give it a distribution of clusters could help filtering out sessions that seem to not belong to any cluster, find clusters representing different intents but consisting of similar sessions. Two such entities could be work out-sessions and Friday-night-sessions which might both be of similar house/dance/techno-genre styles but be different on some genres. These differences would be interesting to analyze.

Lastly, the approach of taking temporal information further into the model by e.g. treating time of session as a feature could differentiate two similar sessions that occur at completely different time of the day. This could help differentiating the two above mentioned entities, work out and Friday-night, and show interesting time-based clusters.

Bibliography

[1] Spotify, Spotify offical numbers, spotify.com, cited May 15th 2014 (2013).
URL "http://press.spotify.com/us/information/"

[2] M. Kamalzadeh, D. Baur, T. Möller, A survey on music listening and management behaviours., in: ISMIR, 2012, pp. 373–378.

[3] K. Roe, Swedish youth and music listening patterns and motivations, Communication Research 12 (3) (1985) 353–362.

[4] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From data mining to knowledge discovery in databases, AI magazine 17 (3) (1996) 37.

[5] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, P. Lamere, The million song dataset, in: ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida, University of Miami, 2011, pp. 591–596.

[6] A. Hotho, A. Nürnberger, G. Paaß, A brief survey of text mining., in: Ldv Forum, Vol. 20, 2005, pp. 19–62.

[7] C. M. Bishop, et al., Pattern recognition and machine learning, Vol. 1, springer New York, 2006.

[8] K. P. Murphy, Machine learning: a probabilistic perspective, MIT Press, 2012.

[9] A. Rajaraman, J. D. Ullman, Mining of massive datasets, Cambridge University Press, 2012.

[10] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, R. Tibshirani, The elements of statistical learning, Vol. 2, Springer, 2009.

[11] J. Srivastava, R. Cooley, M. Deshpande, P.-N. Tan, Web usage mining: Discovery and applications of usage patterns from web data, ACM SIGKDD Explorations Newsletter 1 (2) (2000) 12–23.

[12] Z. Pabarskaite, A. Raudys, A process of knowledge discovery from web log data: Systematization and critical review, Journal of Intelligent Information Systems 28 (1) (2007) 79–104.

[13] J. Xiao, Y. Zhang, Clustering of web users using session-based similarity measures, in: Computer Networks and Mobile Computing, 2001. Proceedings. 2001 International Conference on, IEEE, 2001, pp. 223–228.

[14] J. Xiao, Y. Zhang, X. Jia, T. Li, Measuring similarity of interests for clustering web-users, in: Proceedings of the 12th Australasian Database Conference, ADC '01, IEEE Computer Society, Washington, DC, USA, 2001, pp. 107–114.
URL `http://dl.acm.org/citation.cfm?id=545538.545551`

[15] D. D. Lee, H. S. Seung, Algorithms for non-negative matrix factorization, in: Advances in neural information processing systems, 2000, pp. 556–562.

[16] W. Xu, X. Liu, Y. Gong, Document clustering based on non-negative matrix factorization, in: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, ACM, 2003, pp. 267–273.

[17] G. Xu, Y. Zhang, J. Ma, X. Zhou, Discovering user access pattern based on probabilistic latent factor model, in: Proceedings of the 16th Australasian database conference-Volume 39, Australian Computer Society, Inc., 2005, pp. 27–35.

[18] T. Hofmann, Probabilistic latent semantic indexing, in: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 1999, pp. 50–57.

[19] C. Ding, T. Li, W. Peng, Nonnegative matrix factorization and probabilistic latent semantic indexing: Equivalence chi-square statistic, and a hybrid method, in: Proceedings of the national conference on artificial intelligence, Vol. 21, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006, p. 342.

[20] J. S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.

[21] S. E. Park, S. Lee, S.-g. Lee, Session-based collaborative filtering for predicting the next song, in: Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on, IEEE, 2011, pp. 353–358.

[22] R. Dias, M. J. Fonseca, Improving music recommendation in session-based collaborative filtering by using temporal context, in: Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on, Vol. 1, IEEE, 2013, pp. 783 – 788.

[23] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, the Journal of machine Learning research 3 (2003) 993–1022.

[24] S. Kullback, R. A. Leibler, On information and sufficiency, The Annals of Mathematical Statistics (1951) 79–86.

[25] E. Zheleva, J. Guiver, E. Mendes Rodrigues, N. Milić-Frayling, Statistical models of music-listening sessions in social media, in: Proceedings of the 19th international conference on World wide web, ACM, 2010, pp. 1019–1028.

[26] C. Mallows, A note on asymptotic joint normality, The Annals of Mathematical Statistics (1972) 508–515.

[27] D. Zhou, J. Li, H. Zha, A new mallows distance based metric for comparing clusterings, in: Proceedings of the 22nd international conference on Machine learning, ACM, 2005, pp. 1028–1035.

[28] M. Steinbach, G. Karypis, V. Kumar, et al., A comparison of document clustering techniques, in: KDD workshop on text mining, Vol. 400, Boston, 2000, pp. 525–526.

[29] I. S. Dhillon, D. S. Modha, Concept decompositions for large sparse text data using clustering, Machine learning 42 (1-2) (2001) 143–175.

[30] B. Larsen, C. Aone, Fast and effective text mining using linear-time document clustering, in: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 1999, pp. 16–22.

[31] A. Huang, Similarity measures for text document clustering, in: Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand, 2008, pp. 49–56.

[32] G. E. Box, N. R. Draper, Empirical model-building and response surfaces., John Wiley & Sons, 1987.

[33] K. J. Cios, W. Pedrycz, R. Swiniarsk, Data mining methods for knowledge discovery, Neural Networks, IEEE Transactions on 9 (6) (1998) 1533–1534.

[34] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Vol. 1, California, USA, 1967, p. 14.

[35] D. Arthur, S. Vassilvitskii, k-means++: The advantages of careful seeding, in: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[36] P. J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Journal of computational and applied mathematics 20 (1987) 53–65.

[37] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: A library for large linear classification, Journal of Machine Learning Research 9 (2008) 1871–1874.

[38] A. Y. Ng, Feature selection, l 1 vs. l 2 regularization, and rotational invariance, in: Proceedings of the twenty-first international conference on Machine learning, ACM, 2004, p. 78.

[39] R. Typke, F. Wiering, R. C. Veltkamp, A survey of music information retrieval systems., in: ISMIR, 2005, pp. 153–160.

[40] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.

[41] W. McKinney, Data structures for statistical computing in python, in: S. van der Walt, J. Millman (Eds.), Proceedings of the 9th Python in Science Conference, 2010, pp. 51 – 56.

[42] E. Jones, T. Oliphant, P. Peterson, et al., SciPy: Open source scientific tools for Python (2001–).
URL `http://www.scipy.org/`

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

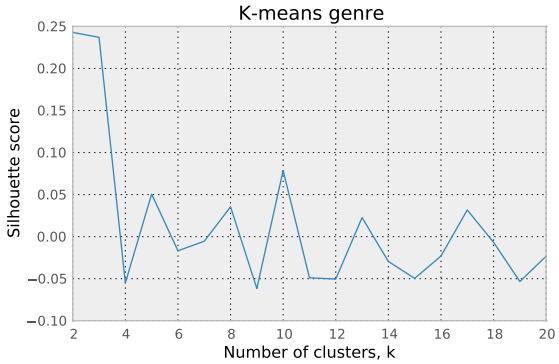
[44] M. J. de Hoon, S. Imoto, J. Nolan, S. Miyano, Open source clustering software, Bioinformatics 20 (9) (2004) 1453–1454.

[45] M. J. de Hoon, S. Imoto, J. Nolan, S. Miyano, The c clustering library, cited 21 May 2014 (August 2013).
URL `http://bonsai.hgc.jp/~mdehoon/software/cluster/cluster.pdf`

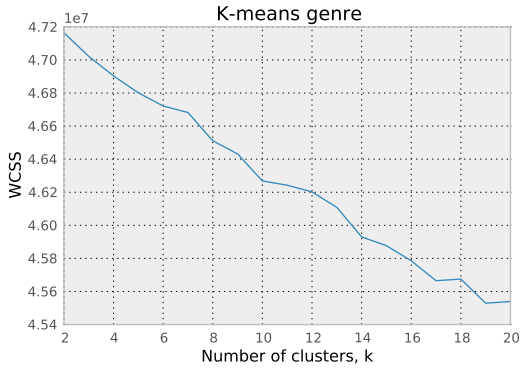
Appendix A
Algorithm evaluation

Below are the plots and results of the clustering evaluations for the models which were rejected. Both of the models, K-means and Complete Hierarchical clustering where only evaluated using only genre as features.

A.1 K-means



(a)



(b)

Figure A.1: K-means evaluation plots when using genre only. (a) is the silhouette score over k and (b) is the total within-cluster sum of squares

In Fig A.1 the silhouette score and within-cluster sum of squares (WCSS) is shown for K-means on the whitened dataset. The peak at low K in silhouette score is due to a very skew distribution of samples with almost all sessions in one cluster.

A.2 Complete HAC

The evaluation plots for the complete agglomerative hierarchical clustering is found in Fig A.2. A dendrogram of the merging is found in sub figure (a) and the height of the branches shows low ability to create natural clusters.

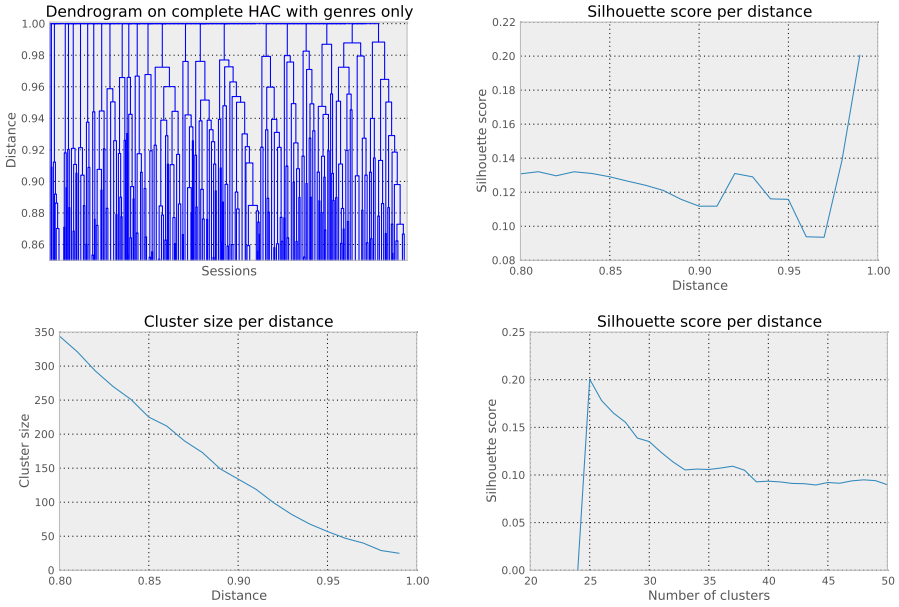


Figure A.2: Results of complete shierarchical clustering. Plot of the dendrogram in (a), the silhouette score when cutting the tree at appropriate levels in the linkage matrix in (b), a plot of how the cluster sizes of the different sizes in (c) and last in (d) is the silhouette score when performing flat clustering with different maximum number of clusters, starting at 25 due to being least number of clusters but 1 as seen in (a).

Appendix B
Cluster plots genres only

In this chapter the plots of the logistic regression on clusterings with genres only is shown. First the K-medoids, K=6, and lastly the Average Hierarchical clustering, K=29. Both are given with plots and a table of complete information.

B.1 K-medoids

The following six clusters were the result of the K-medoids, K=6, clustering on genres only. The clusters were named as following: Pop (Fig B.1), Dance & House (Fig B.2), Rock (Fig B.3), Rap (Fig B.4), Indie Rock (Fig B.5) and Holiday (Fig B.6).

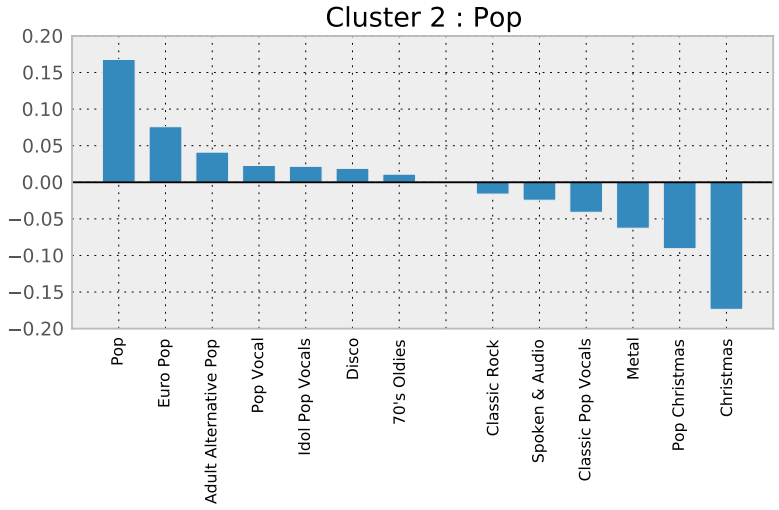


Figure B.1: Pop-cluster. Size: 6800 Silhouette score: 0.193300399677 C: 0.005

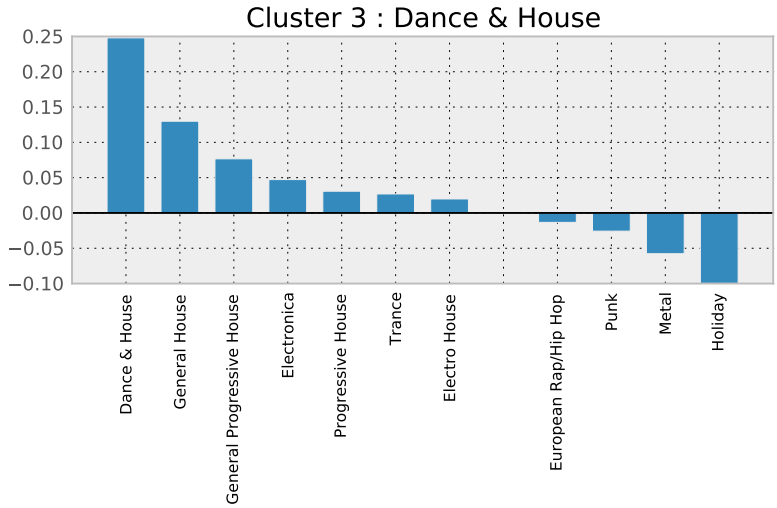


Figure B.2: Dance & House-cluster. Size: 3794 Silhouette score: 0.186877368005 C: 0.005

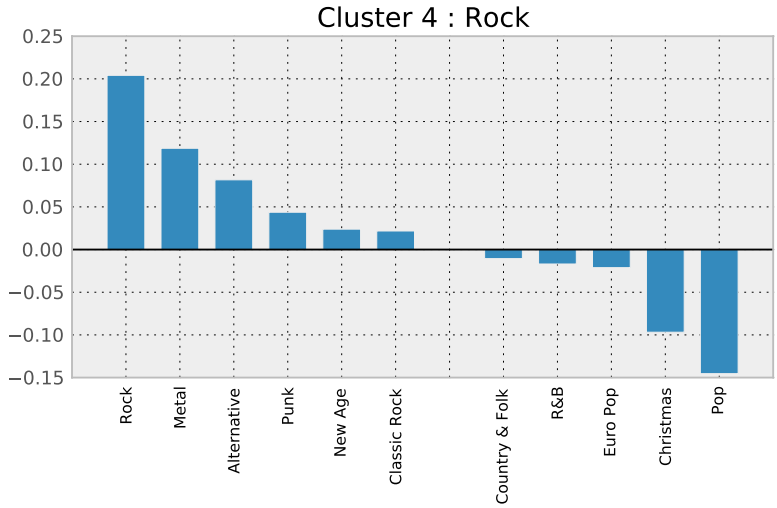


Figure B.3: Rock-cluster. Size: 3794 Silhouette score: 0.0721125193448 C: 0.005

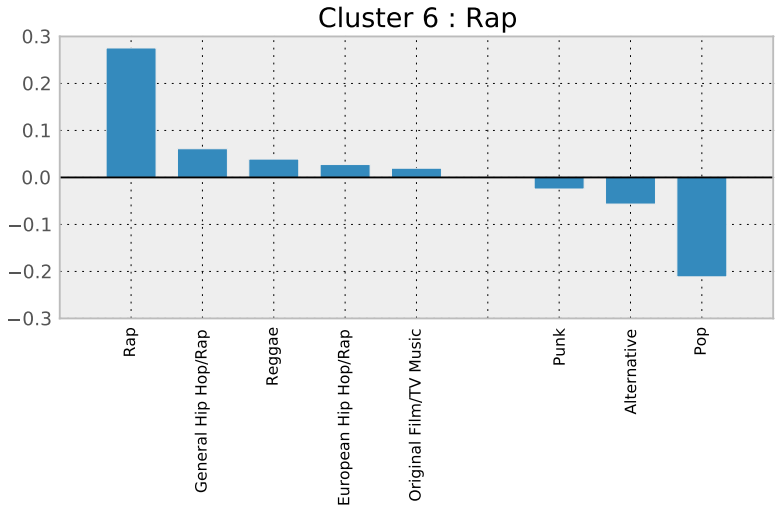


Figure B.4: Rap-cluster. Size: 2733 Silhouette score: 0.280001407094 C: 0.005

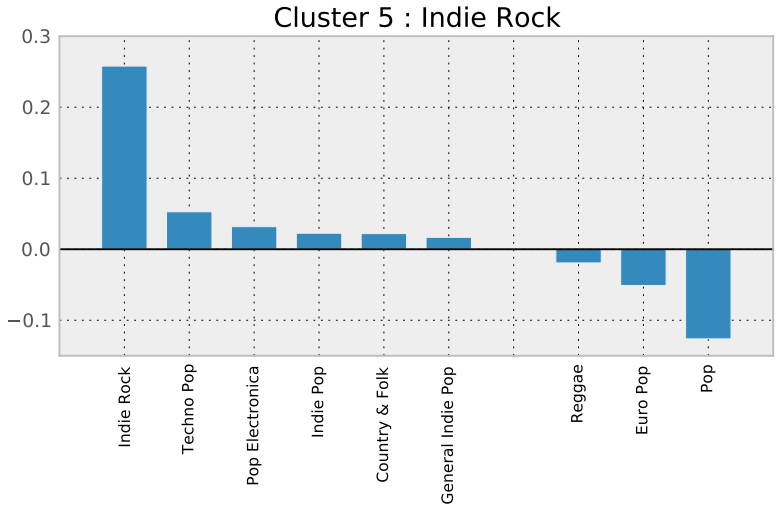


Figure B.5: Indie Rock-cluster. Size: 2138 Silhouette score: 0.166133118092 C: 0.005

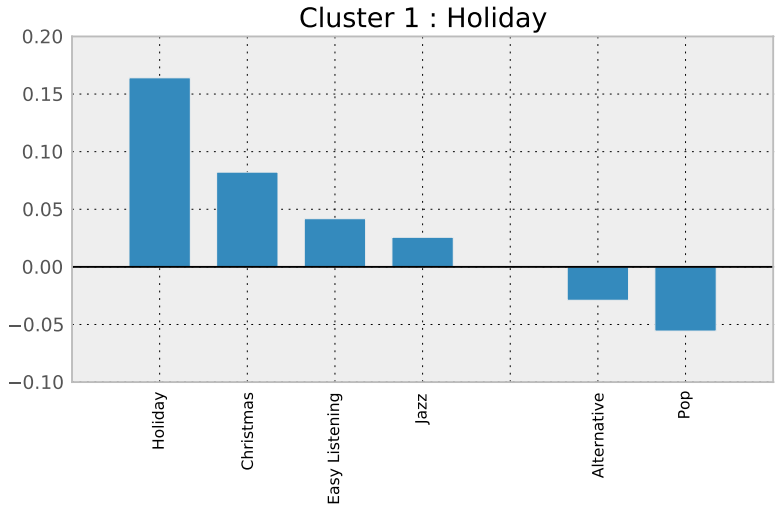


Figure B.6: Holiday-cluster. Size: 1732 Silhouette score: 0.292976433667 C: 0.001

And the following table shows a summary of the naming of each cluster, the regularization parameter and cross validation accuracy.

Cluster size	Cluster id	CV-accuracy	C	Silhouette score	Naming
6800	2	0.943261397742	0.005	0.193300399677	Pop
3794	3	0.95469486923	0.005	0.186877368005	Dance & House
3794	4	0.966652374827	0.005	0.0721125193448	Rock
2733	6	0.975418036301	0.005	0.280001407094	Rap
2138	5	0.961078557477	0.005	0.166133118092	Indie Rock
1732	1	0.984564813491	0.001	0.292976433667	Holiday

Table B.1: Complete information about logistic regression of K-medoids k=6 clusterings

B.2 Average HAC

In the following pictures characterization of cluster from average HAC on genres when cutting at 29 clusters is shown. Only clusters with more than one descriptive feature and of sufficient size is shown. In total, 22 clusters are shown below. The clusters were named Pop (Fig B.7), Dance & House (Fig B.8),Rap (Fig B.9), Indie Rock (Fig B.10), Rock (Fig B.11), Holiday (Fig B.12), Funk Rock (Fig B.13), Symphonic Progressive Rock (Fig B.14) R&B (Fig B.15), Punk (Fig B.16), Country & Folk (Fig B.17), Reggae (Fig B.18), General Film Music (Fig B.19), Classical (Fig B.20), Jazz (Fig B.21), Children's (Fig B.22), Electro (Fig B.23), Latin (Fig B.24), New Celtic Trad. (Fig B.25) Christian Pop (Fig B.26), Progressive Electronic (Fig B.27) and lastly: Funk (Fig B.28).

Lastly Table B.2 summarizes all the naming and parameters used.

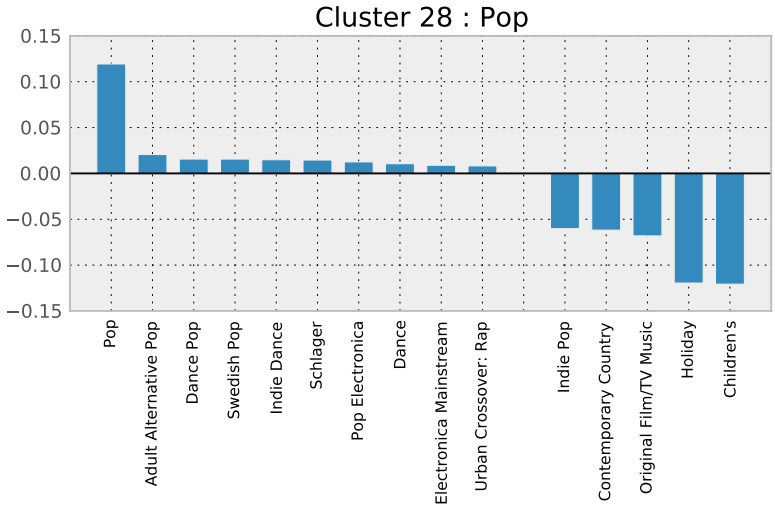


Figure B.7: Pop-cluster. Size: 6010 Silhouette score: 0.17899183912 C: 0.005

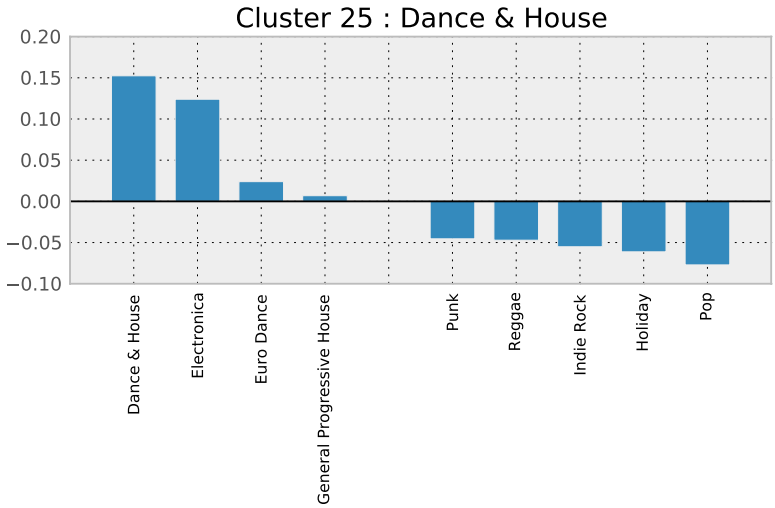


Figure B.8: Dance & House-cluster. Size: 2881 Silhouette score: 0.154786627704 C: 0.005

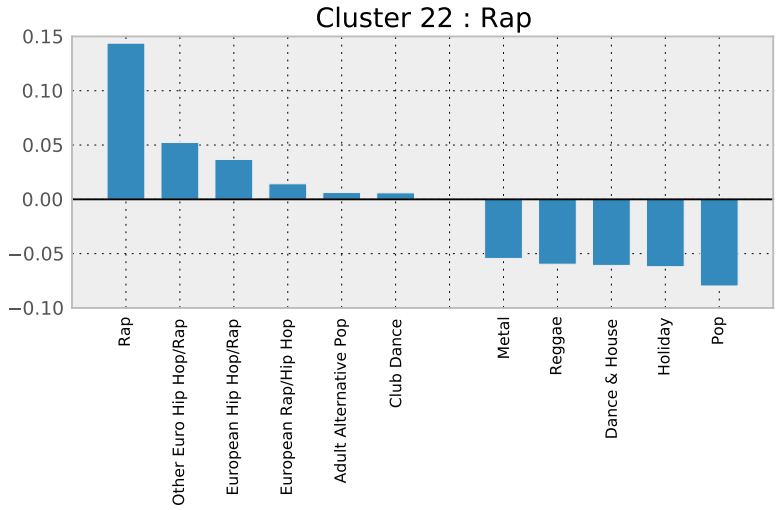


Figure B.9: Rap-cluster. Size: 2225 Silhouette score: 0.282989496078 C: 0.005

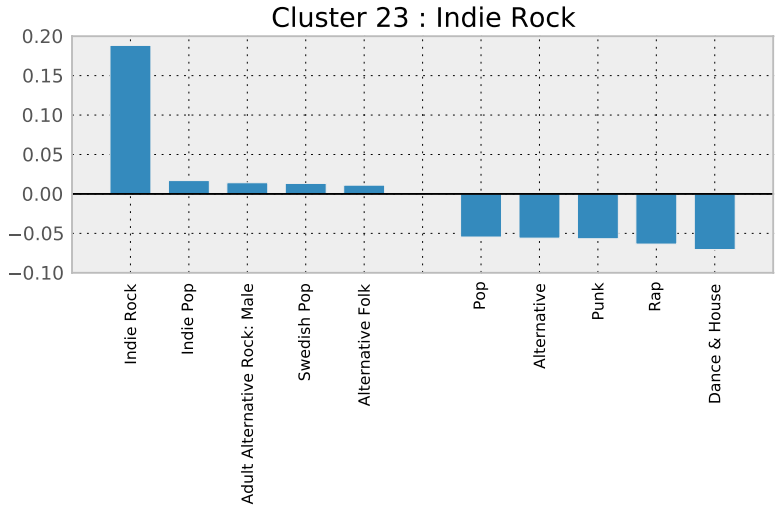


Figure B.10: Indie Rock-cluster. Size: 1655 Silhouette score: 0.165745331341 C: 0.005

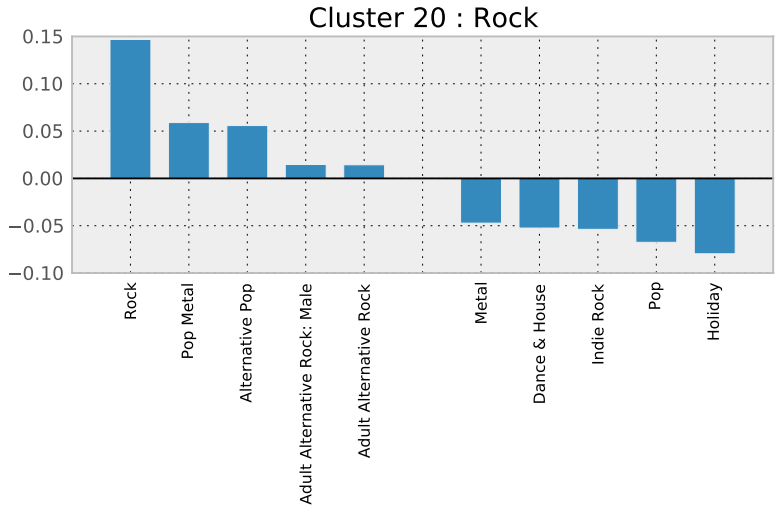


Figure B.11: Rock-cluster. Size: 1632 Silhouette score 0.1181337549 C: 0.005

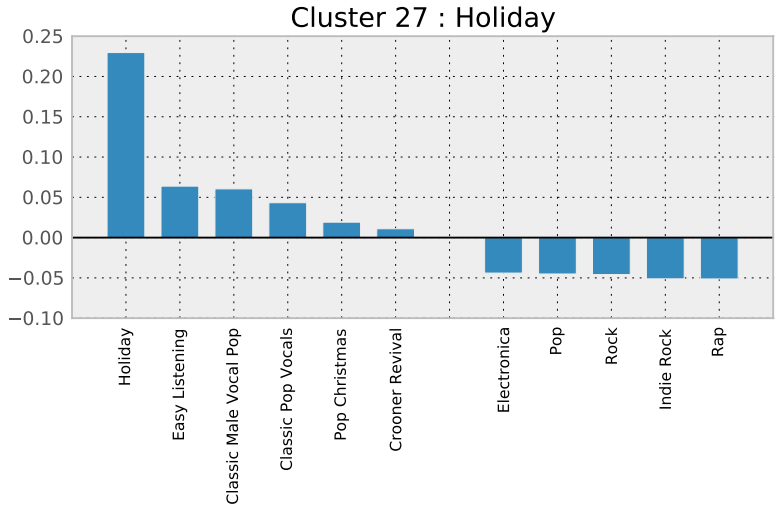


Figure B.12: Holiday-cluster. Size: 1304 Silhouette score 0.374839636028 C: 0.005

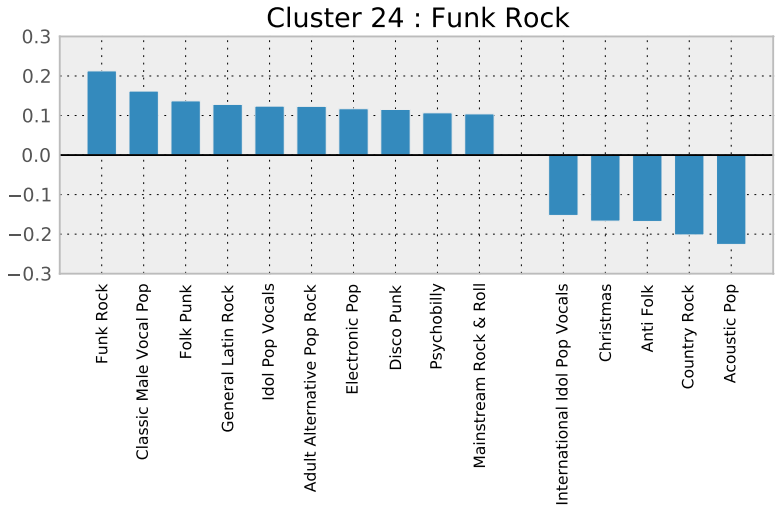


Figure B.13: Funk Rock. Size: 1144 Silhouette score: 0.0786773527471 C: 0.3

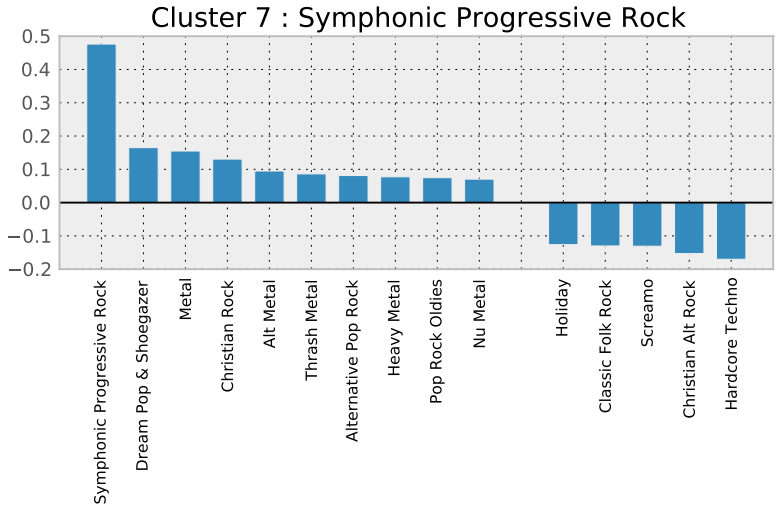


Figure B.14: Symphonic Progressive Rock-cluster. Size: 971 Silhouette score: 0.279747731966 C: 0.3

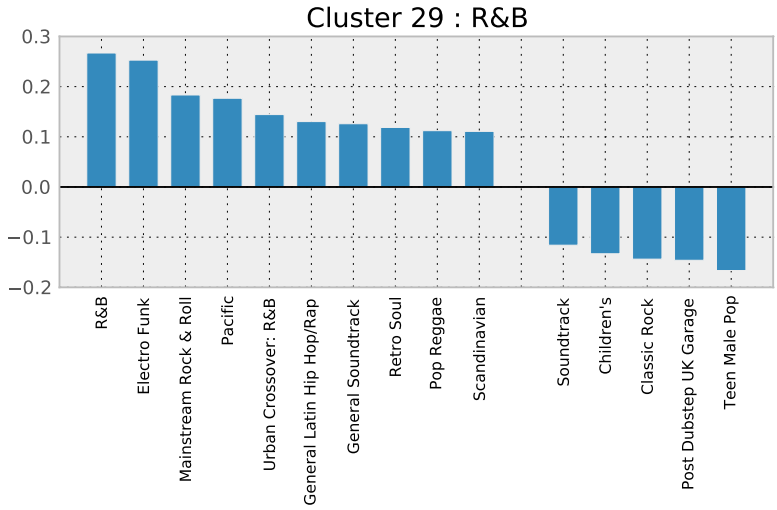


Figure B.15: R&B-cluster. Size: 667 Silhouette score: 0.188287849436 C: 0.3

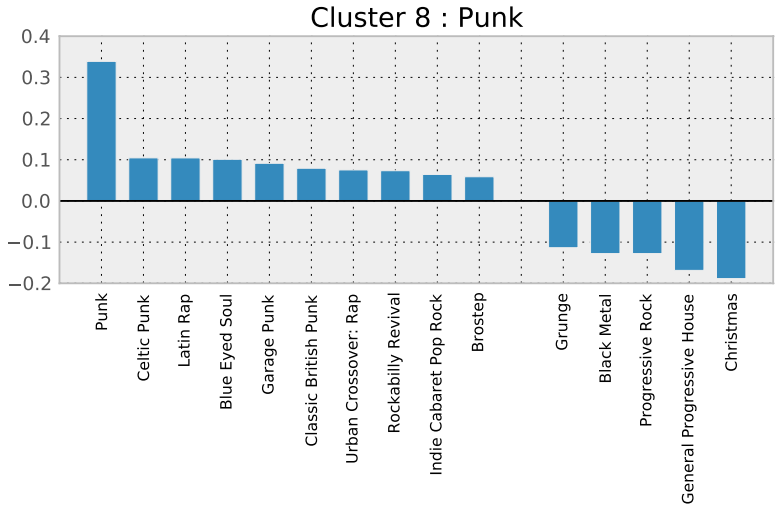


Figure B.16: Punk-cluster. Size 563 Silhouette score: 0.225465311073 C: 0.3

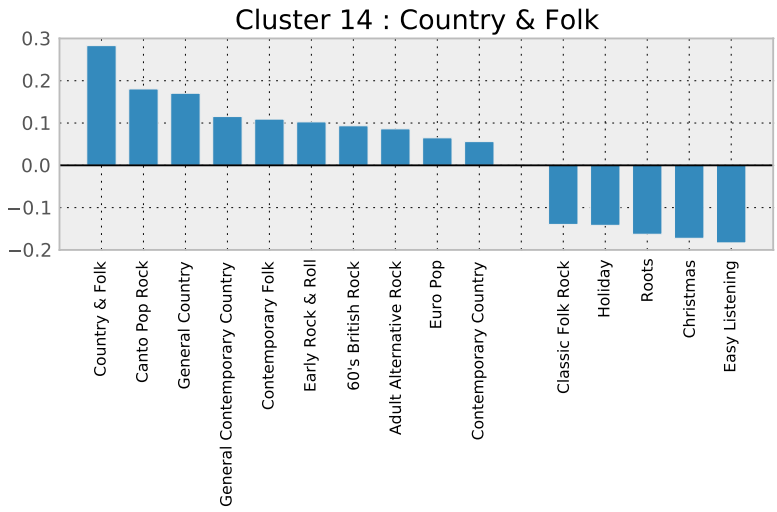


Figure B.17: Country & Folk-cluster. Size: 365 Silhouette score: 0.138335965553 C: 0.3

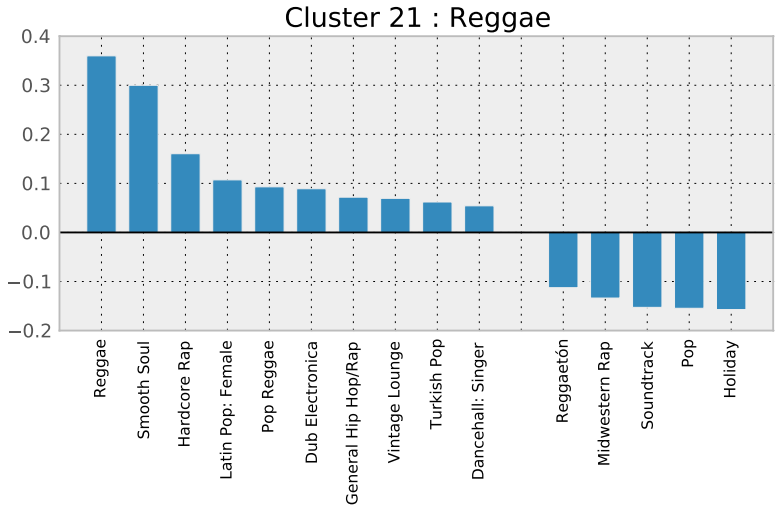


Figure B.18: Reggae-cluster. Size 338 : Silhouette score: 0.322601284276 C: 0.3

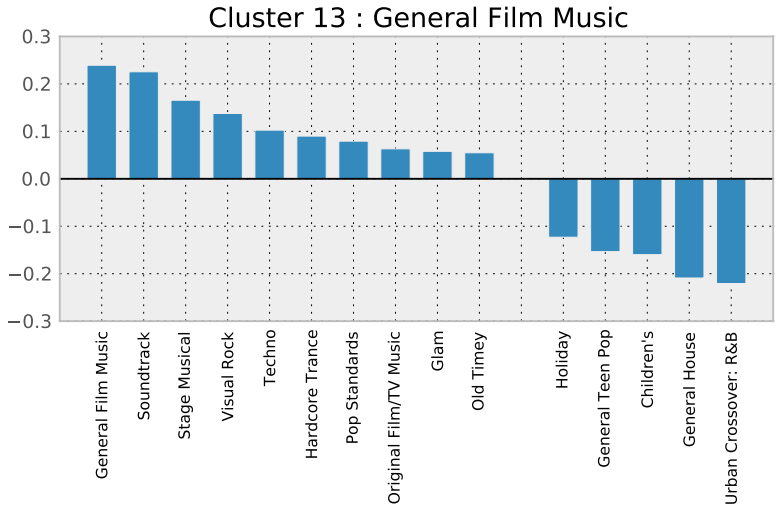


Figure B.19: General Film Music-cluster. Size: 265 Silhouette score: 0.284605546435 C: 0.3

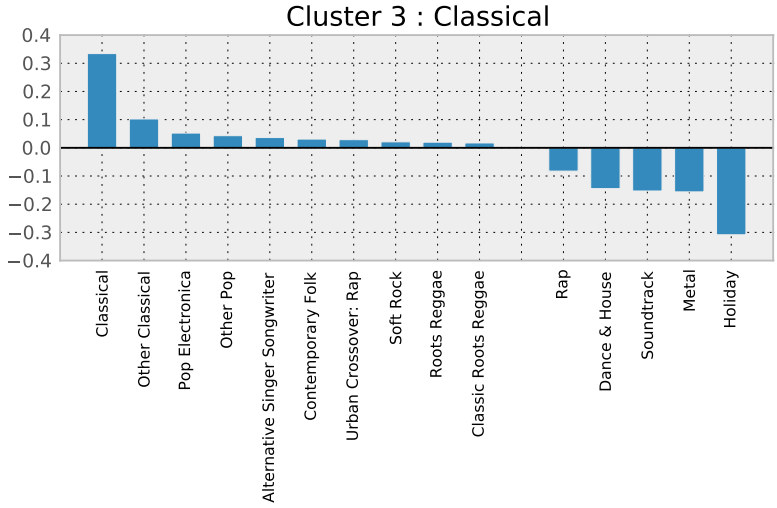


Figure B.20: Classical-cluster. Size: 223 Silhouette score: 0.274005887373 C: 0.3

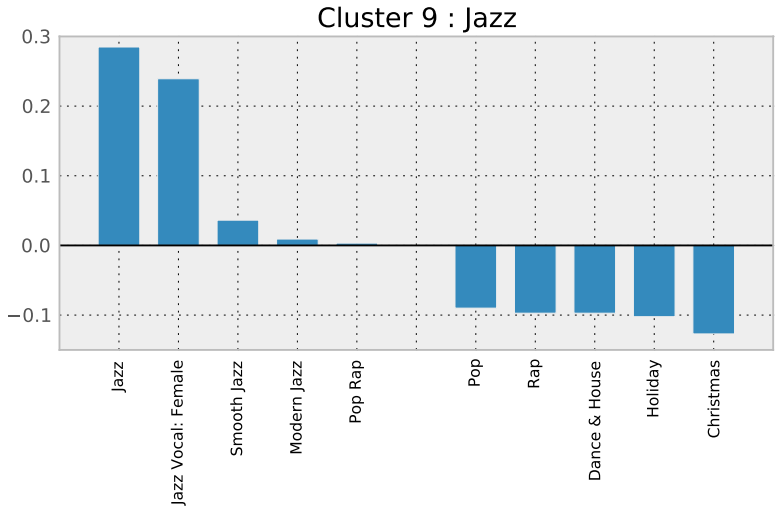


Figure B.21: Jazz-cluster. Size: 131 Silhouette score: 0.235674984706 C: 0.2

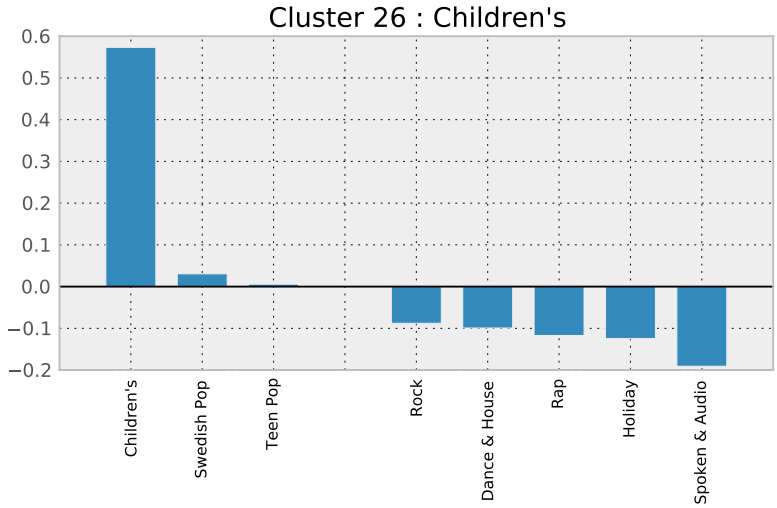


Figure B.22: Children's-cluster. Size: 122 Silhouette score: 0.202630326311 C: 0.2

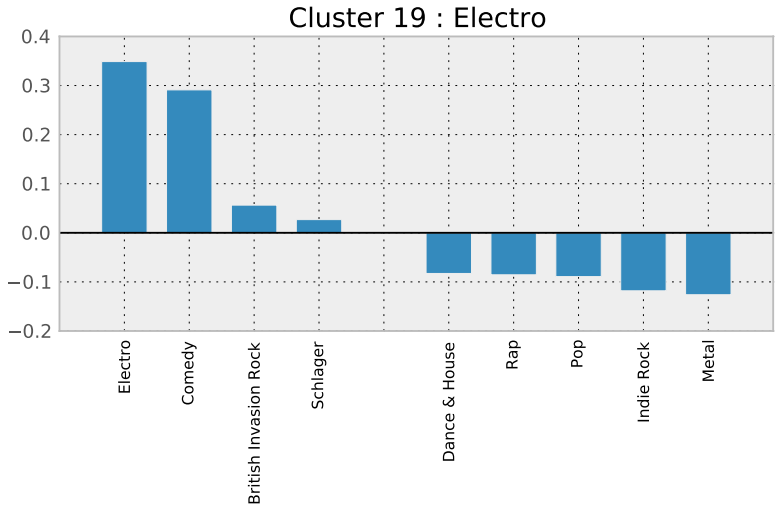


Figure B.23: Electro-cluster. Size: 109 Silhouette score: 0.271004241855 C: 0.3

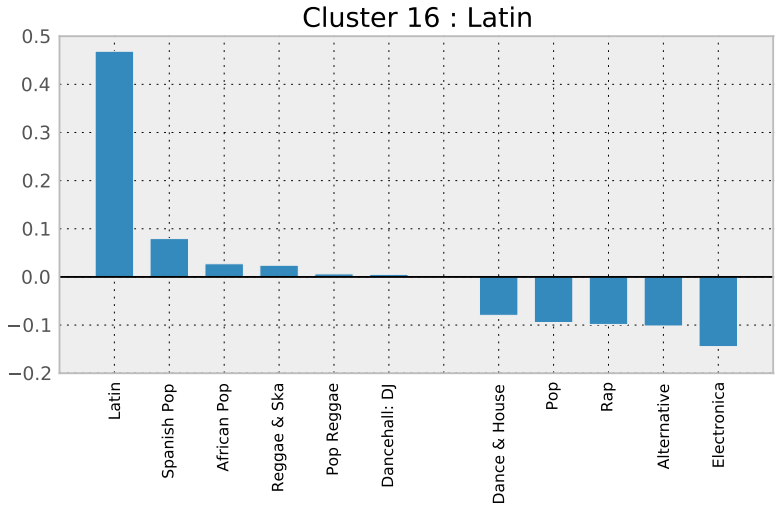


Figure B.24: Latin-cluster. Size: 107 Silhouette score: 0.151506236076 C: 0.2

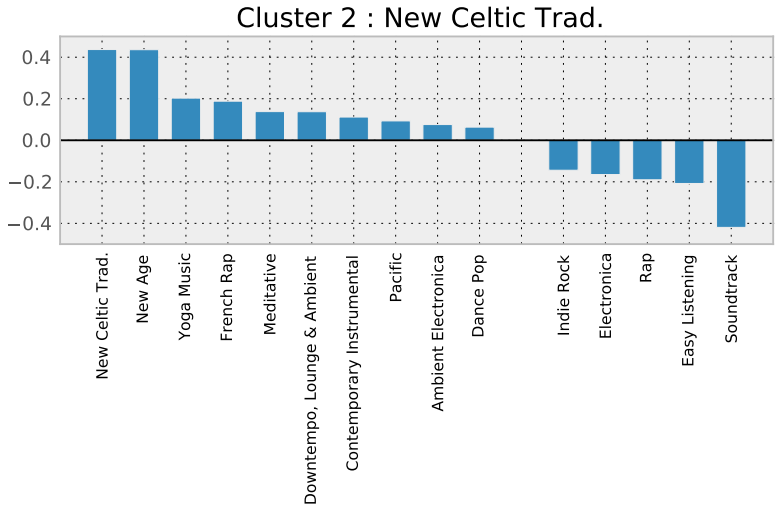


Figure B.25: New Celtic Trad-cluster. Size: 84 Silhouette score 0.239328464257 C: 2

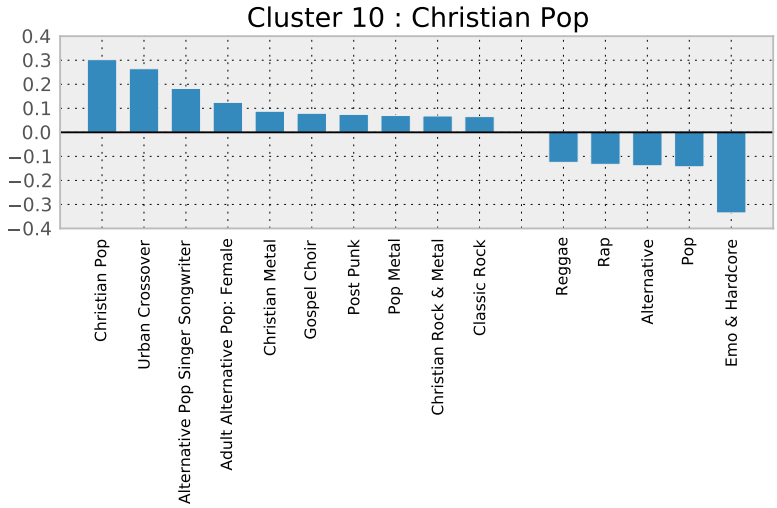


Figure B.26: Christian Pop-cluster. Size: 75 Silhouette score: 0.133099807276 C: 2

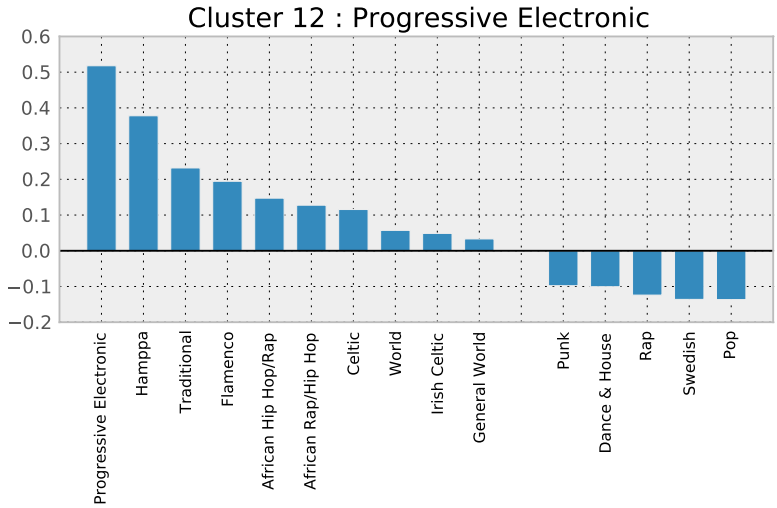


Figure B.27: Progressive Electronic-cluster. Size: 33 Silhouette score: 0.0125138688726 C: 2

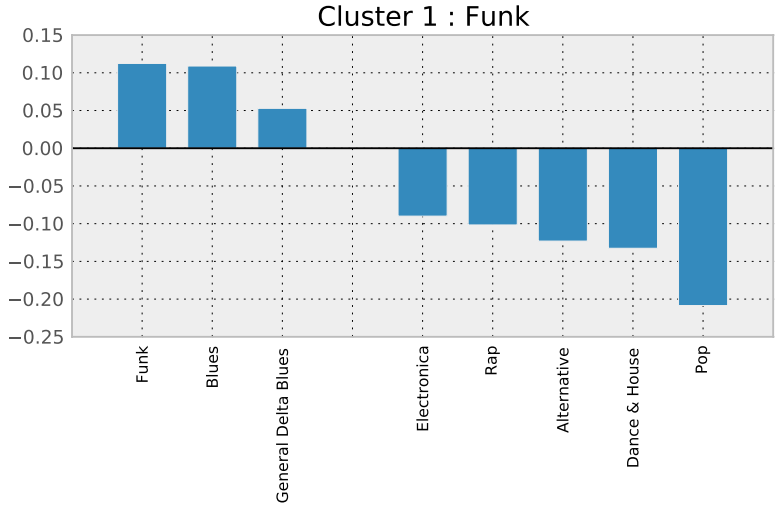


Figure B.28: Funk-cluster. Size: 27 Silhouette score: 0.259619028568 C: 2

A summary of all clusters, cross-validation accuracy, c-value and so on is found in Table B.2.

Cluster size	Cluster id	CV-accuracy	C	Silhouette score	Naming
6010	28	0.906340812729	0.005	0.17899183912	Pop
2881	25	0.93020818446	0.005	0.154786627704	Dance & House
2225	22	0.947215473298	0.005	0.282989496078	Rap
1655	23	0.949835643847	0.005	0.165745331341	Indie Rock
1632	20	0.93935496165	0.005	0.1181337549	Rock
1304	27	0.978752798819	0.005	0.374839636028	Holiday
1144	24	0.945119336859	0.3	0.0786773527471	Funk Rock
971	7	0.978752798819	0.3	0.279747731966	Symphonic Progressive Rock
667	29	0.961650231051	0.3	0.188287849436	R&B
563	8	0.985993997427	0.3	0.225465311073	Punk
365	14	0.988471249583	0.3	0.138335965553	Country & Folk
338	21	0.986994426183	0.3	0.322601284276	Reggae
265	13	0.987232623505	0.3	0.284605546435	General Film Music
223	3	0.993949788004	0.3	0.274005887373	Classical
131	9	0.993854509075	0.2	0.235674984706	Jazz
122	26	0.995140774618	0.2	0.202630326311	Children's
109	19	0.989138202087	0.3	0.271004241855	Electro
107	16	0.992425325139	0.2	0.151506236076	Latin
84	2	0.997856224096	2	0.239328464257	New Celtic Trad.
75	10	0.997427468915	2	0.133099807276	Christian Pop
33	12	0.997379829451	2	0.0125138688726	Progressive Electronic
27	1	0.998570816064	2	0.259619028568	Funk
21	17	0.999761802677	2	0.82488269842	Audio Book
16	6	0.997141632128	2	0.647256327638	General Unclassifiable
10	4	0.997665666238	2	0.276014250322	Other Easy Listening
6	18	None	2	0.422117931328	Exercise
5	11	None	2	0.0219265488301	Traditional
1	5	None	None	0.0	
1	15	None	None	0.0	

Table B.2: Full information of clustering with Average HAC, k=29, genres only. The most descriptive name is the name of the feature with highest weight.

Appendix C
Cluster plots genres and mood

In this section the complete results from the Average Hierarchical clustering with $k = 38$ is shown. First, each cluster is explained using the logistic regression-method and then a table of full details is given in Table C.2, as previous cluster plot appendix.

The Pop-cluster seen in Fig C.1 was the biggest cluster that we previously saw in the results chapter.

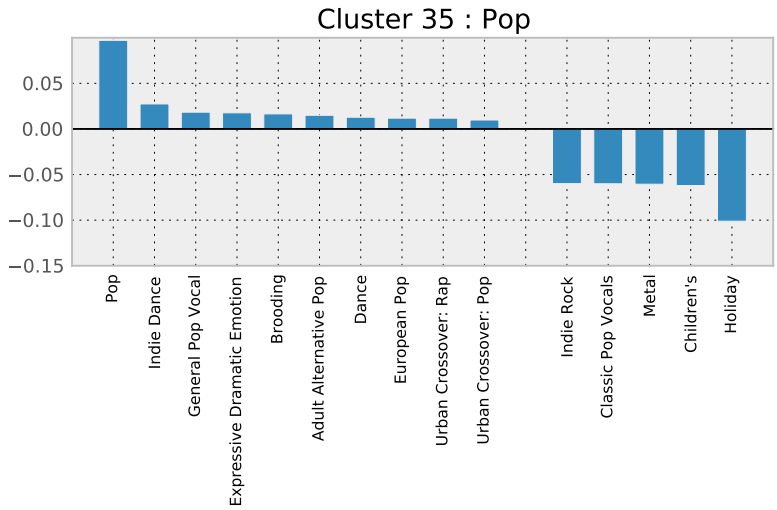


Figure C.1: Pop-cluster. Size: 5869 Silhouette score: 0.148939409677 C: 0.005

And Dance & House-cluster in Fig C.2 was also showed before.

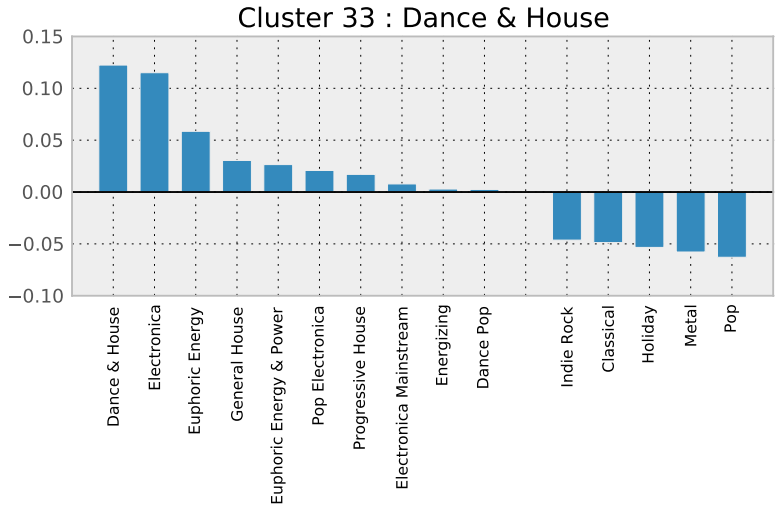


Figure C.2: Dance & House-cluster. Size: 3116 Silhouette score: 0.146052179499 C: 0.005

In Fig C.3 the third biggest cluster is shown, namely Rap. It is a mix of rap sub genres, European rap, and "cool" music. European hip hop might occur with high weight in this cluster because of the region of the dataset. Swedish hip-hop with bands such as Daniel Adams-ray and Timbuktu is two popular artist in the European hip hop genre.

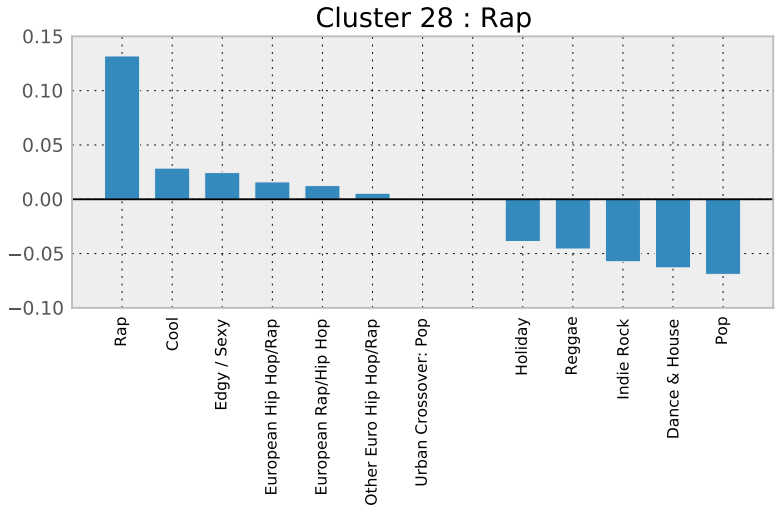


Figure C.3: Rap-cluster. Size: 2098 Silhouette score: 0.280391212148 C: 0.005

Indie Rock, being separated from Rock, is shown in Fig C.4. The very niched feature "Adult alternative rock: male" is from the lowest level of the hierarchy serving more as a working title to categorize tracks and by adding the term male differentiate from other adult alternative rock.

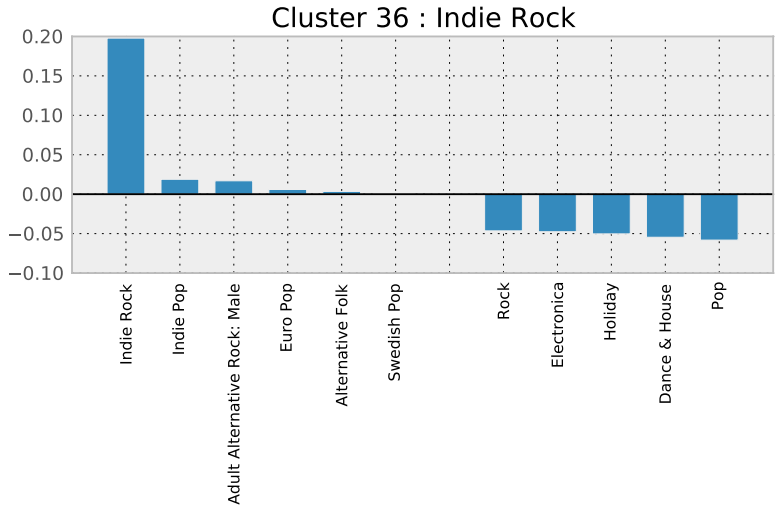


Figure C.4: Indie Rock-cluster. Size: 1762 Silhouette score: 0.140640395479 C: 0.005

Rock-cluster is seen in Fig C.5. The second most weighted feature is not a rock genre but from the metal-hierarchy. But as of similarity between genres, Rock and metal is rather similar, and they showing up together is no surprise.

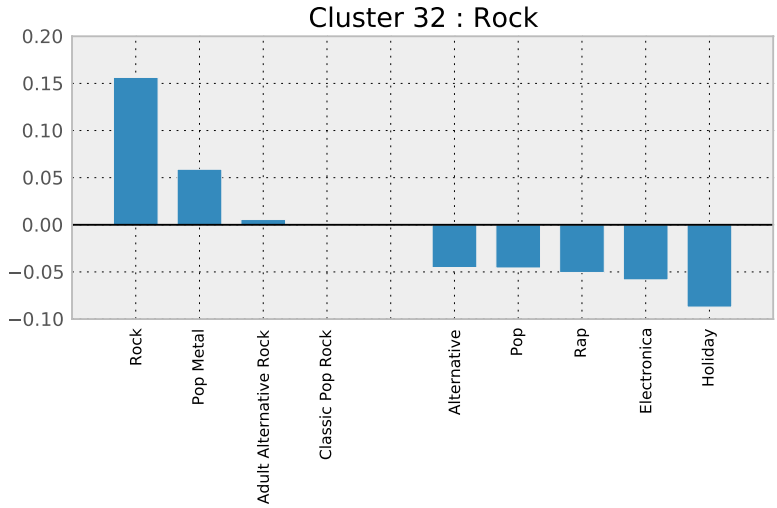


Figure C.5: Rock-cluster. Size: 1632 Silhouette score: 0.112049499506 C: 0.005

Christmas-cluster of 1300 sessions is found in Fig C.6. Categorizations such as Vocal pop and moods as Easy listening seems to be related to Christmas music.

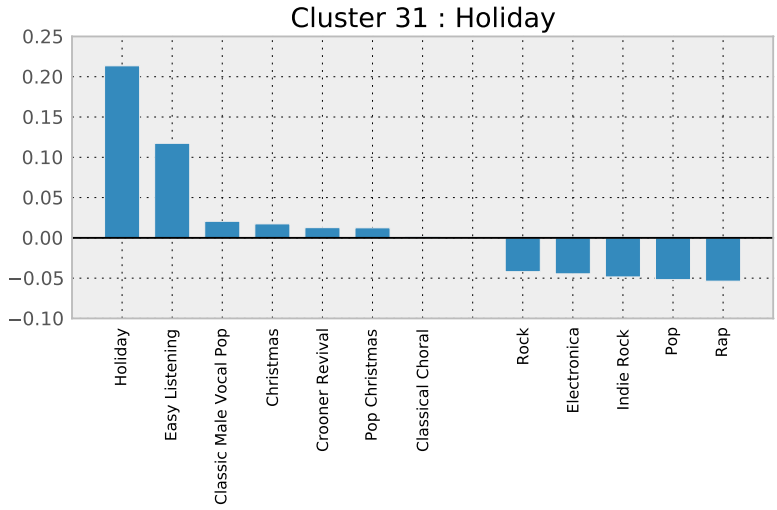


Figure C.6: Holiday-cluster. Size: 1301 Silhouette score: 0.33772172736 C: 0.005

To a non-music-expert the genre "Alternative" (from Fig C.7) might seem fuzzy but it is actually a common top-level-genre with sub genres such as Alternative Dance/Pop/Rock or rockabilly. Alternative music is closely related to Independent music (Indie).

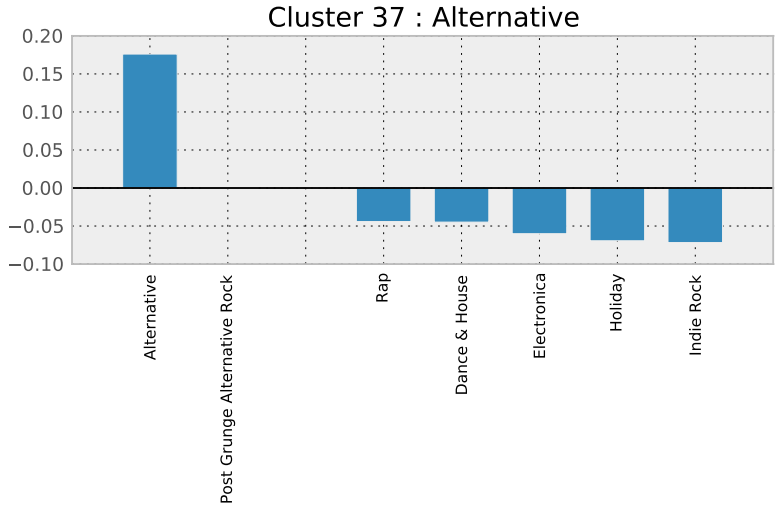


Figure C.7: Alternative-cluster. Size: 1188 Silhouette score: 0.0484131923941 C: 0.005

Interestingly in the Metal-cluster from Fig C.8 Metal and its sub genres is connected with aggressive and defiant music. This is maybe just a description of metal music but as of the logistic regression, the odds of a track belonging to this cluster increases if the track is aggressive.

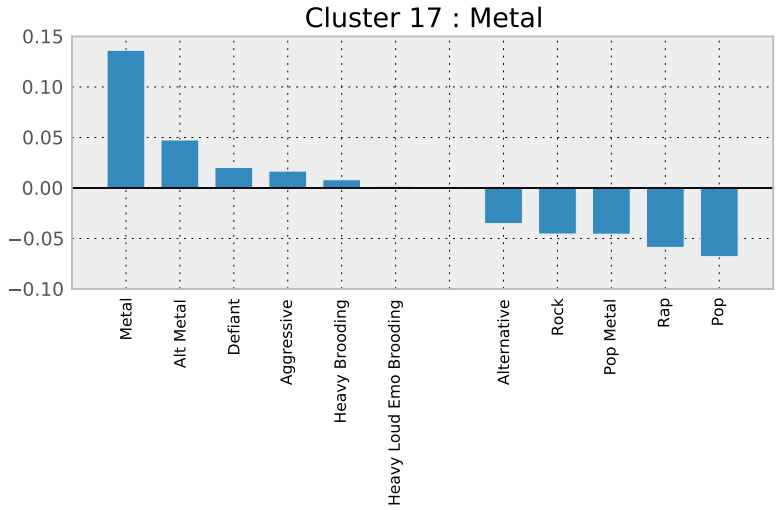


Figure C.8: Metal-cluster. Size: 1013 Silhouette score: 0.261917470475 C: 0.005

R&B in Fig C.9 is separated from the similar genre clusters Rap and Dance & House with a cluster of a mix of R&B hierarchical levels and urban crossover.

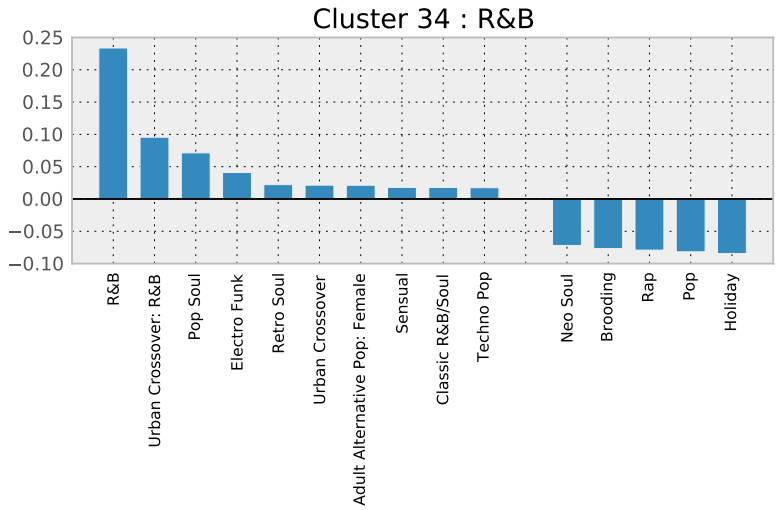


Figure C.9: R&B-cluster. Size: 655 Silhouette score: 0.175428811726 C: 0.05

In Fig C.10 a wide mix of punk genres can be found. Apparently a track with genre Classical U.S. Punk would increase the odds of a session being in this cluster the most.

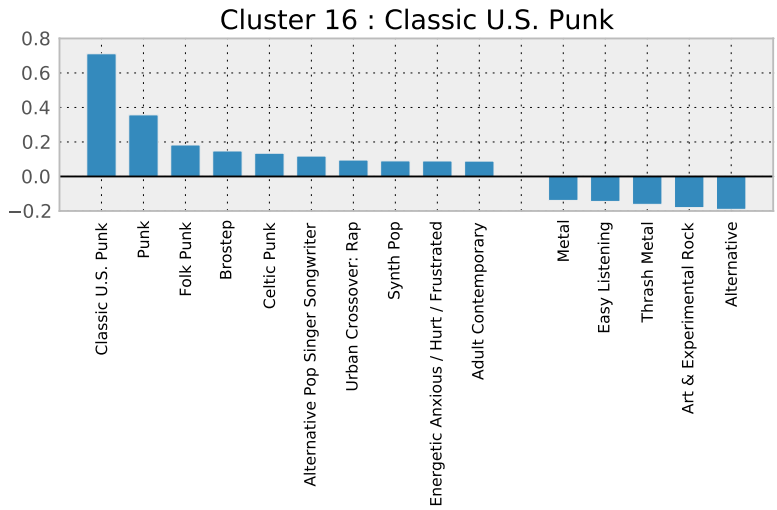


Figure C.10: Punk-cluster. Size: 518 Silhouette score: 0.228691120612 C: 0.5

Out of the 20000 sessions, 363 were separated and grouped in what the logistic regression would explain as Country & folk. The cluster can be seen in Fig C.11.

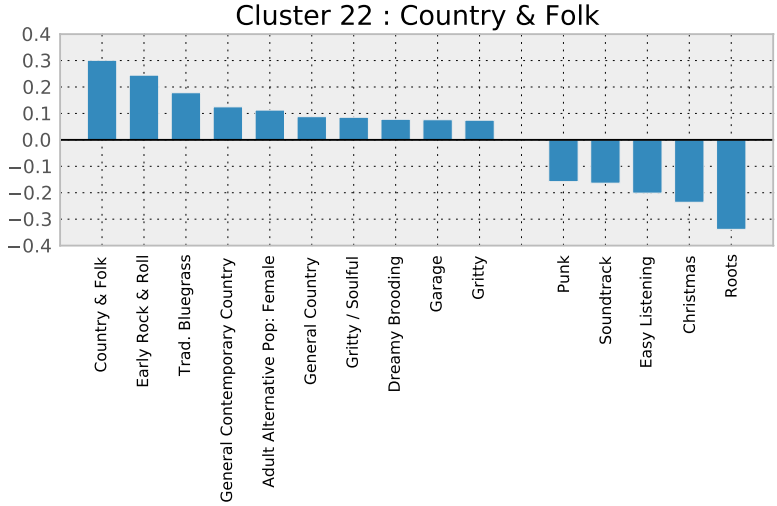


Figure C.11: Country & Folk-cluster. Size: 363 Silhouette score: 0.125178693609 C: 0.5

What is named the Spanish Pop cluster in Fig C.12 is a mix of Techno pop, Electro Funk, Spanish Pop, house and Pop rap and the moods cool and down tempo. All of which seems like rather popular genres. The silhouette score is sufficiently high to believe this might is a good cluster representing a mix of popular sessions by the users.

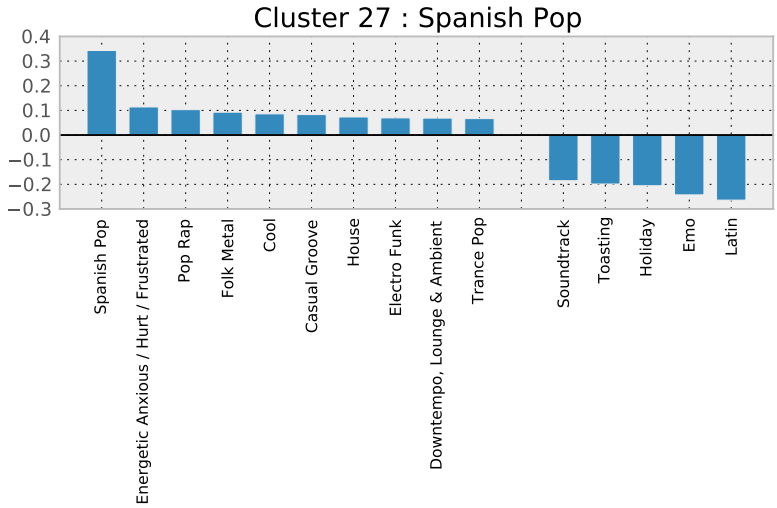


Figure C.12: Mixed Pop-cluster. Size: 327 Silhouette score: 0.281940137752 C: 0.5

As the size of the clusters decrease also the mix of genres explained goes from the ones one might call the most popular ones to less popular. E.g big cluster with the Pop is more popular than the smaller genres such as Jazz and Classical music. Such less common genre is Progressive Electronic and Soundtrack as has been captured in the cluster in Fig C.13. As of the positive coefficient weights it looks a mix of Soundtrack, Film/TV music and Electronica music. Could be Electronica tracks also belonging to a soundtrack hence tracks with both genres.

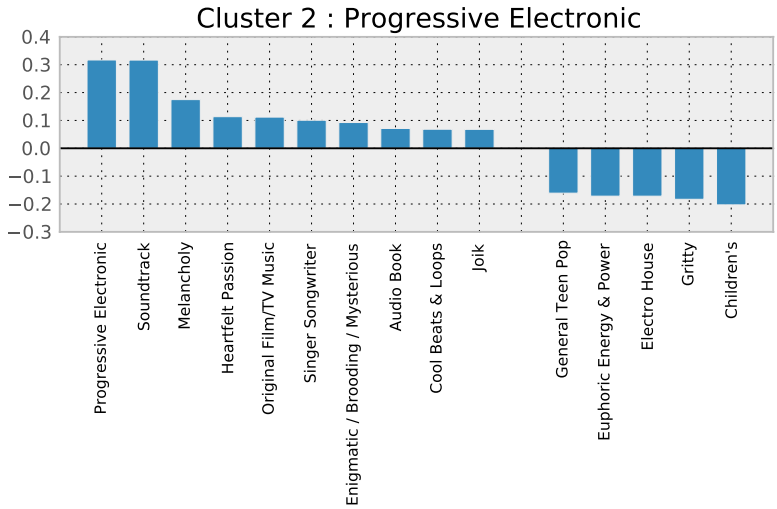


Figure C.13: Soundtrack/Electronica-cluster. Size: 234 Silhouette score: 0.285635406315 C: 0.5

Classical music such as Celtic new age. The tracks looks like they are categorized as classical instruments, quite and romantic as can be seen in Fig C.14. The weight of classical is much higher than the others.

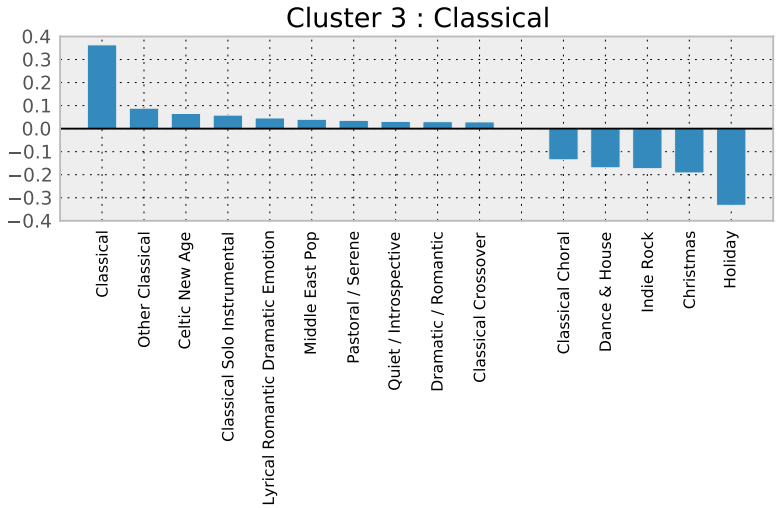


Figure C.14: Classical-cluster. Size: 199 Silhouette score: 0.23374599038 C: 0.5

Jazz-cluster with genres such as big band & swing and African pop and stage musical has been captured in a cluster of size 132 in Fig C.15.

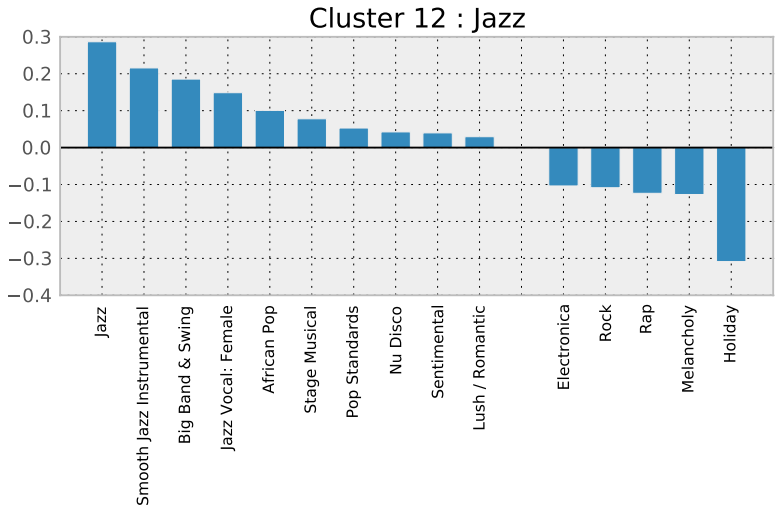


Figure C.15: Jazz-cluster. Size: 132 Silhouette score: 0.124271538529 C: 0.5

As previously shown, the Children's cluster in Fig C.16. It should be noted that music classified as Children can be music composed by adults to children's or educational tracks

for children's.

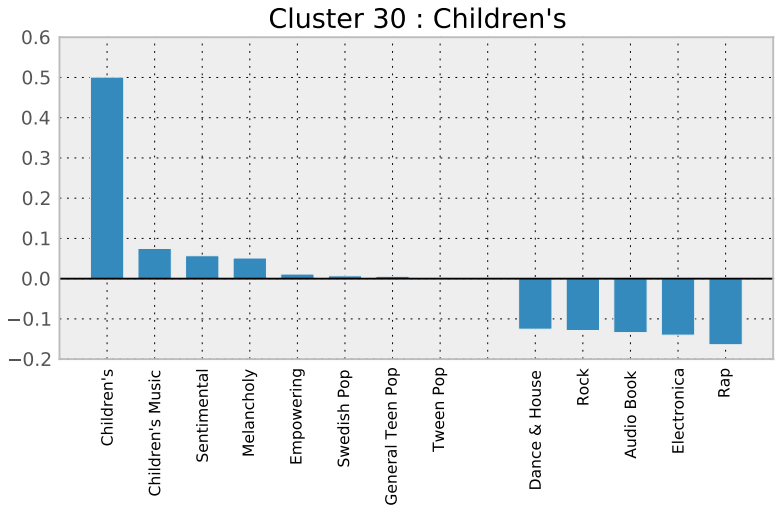


Figure C.16: Children's-cluster. Size: 114 Silhouette score: 0.149199821065 C: 0.5

Also as previously shown, The Latin-cluster is again give here in Fig C.17. Popular artists in Latin pop is Shakira, Ricky Martin, Pitbull etc.

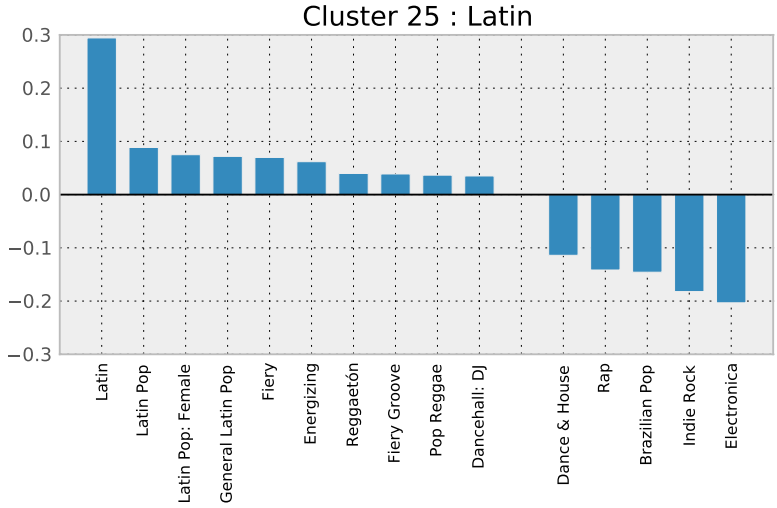


Figure C.17: Latin-cluster. Size: 97 Silhouette score: 0.101858307823 C: 0.5

The cluster size has decreased down to below 100 sessions which would not show popular but more niched moments by the users. The comedy-cluster is such and again given in Fig C.18.

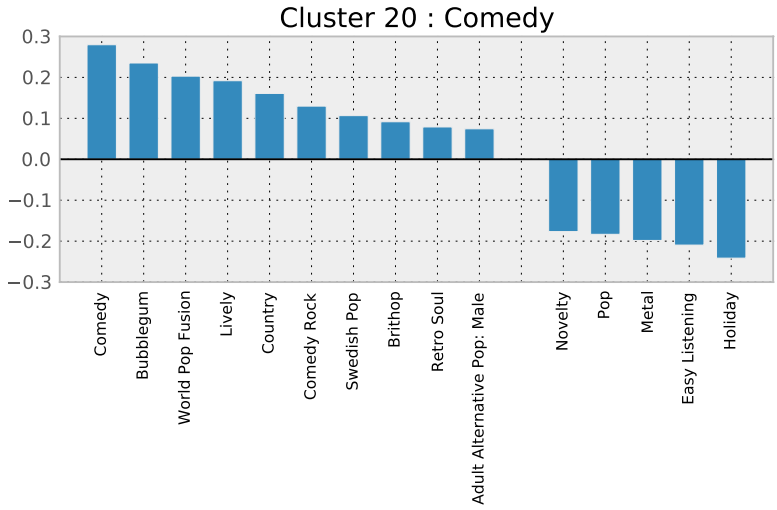


Figure C.18: Comedy-cluster. Size: 84 Silhouette score: 0.259850151927 C: 2

Healing, peaceful and lite melancholy music such as New age music has its own cluster as can be seen in Fig C.19. It apparently has French rap as well. The most famous new age artist might be Enya and of new Celtic Traditional the artists The Dubliners and Celtic Woman is famous. Both Enya and Celtic Woman has music that characterizes as Healing and Melancholy.

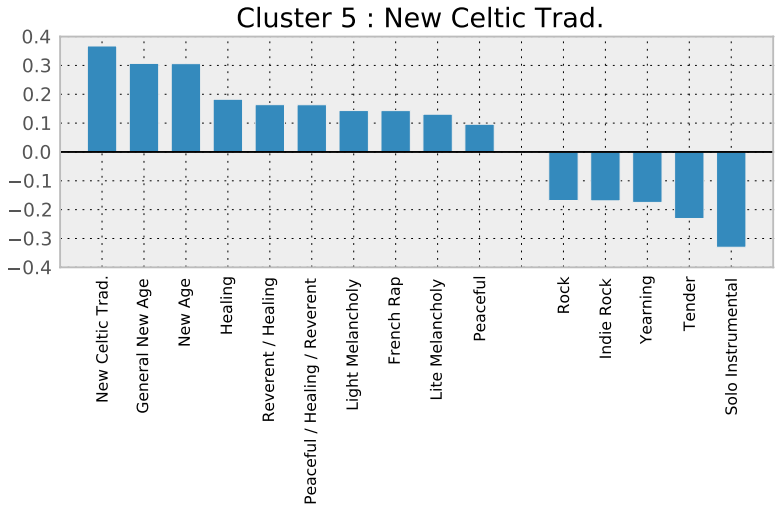


Figure C.19: New Age-cluster. Size: 77 Silhouette score: 0.118224435319 C: 2

Again, the Religious cluster is given in Fig C.20.

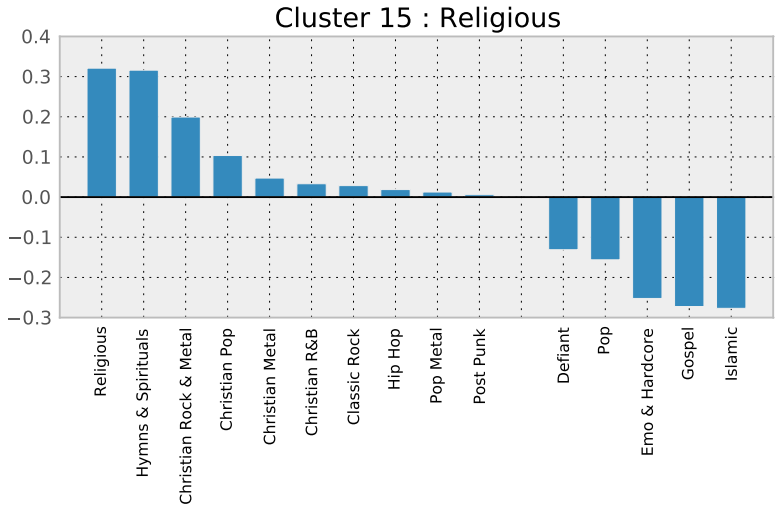


Figure C.20: Religious-cluster. Size: 54 Silhouette score: 0.145533996878 C: 2

The small cluster of solo instrumental music in Fig C.21 is in terms of genre hierarchy very similar to Classical music. Solo instruments is usually tracks of musical compositions such as Fur Elise etc. This cluster has as shown in the logistic regression something related to Synth pop as well.

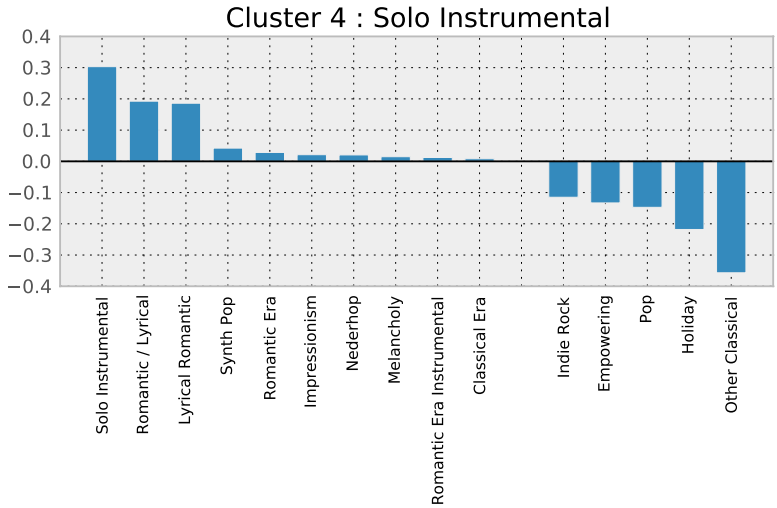


Figure C.21: Solo instrument-cluster. Size: 34 Silhouette score: 0.279228777653 C: 2

And the last cluster that made sense to visualize were named Funk and consisted of a collection of Blued-like genres. This cluster be seen in Fig C.22

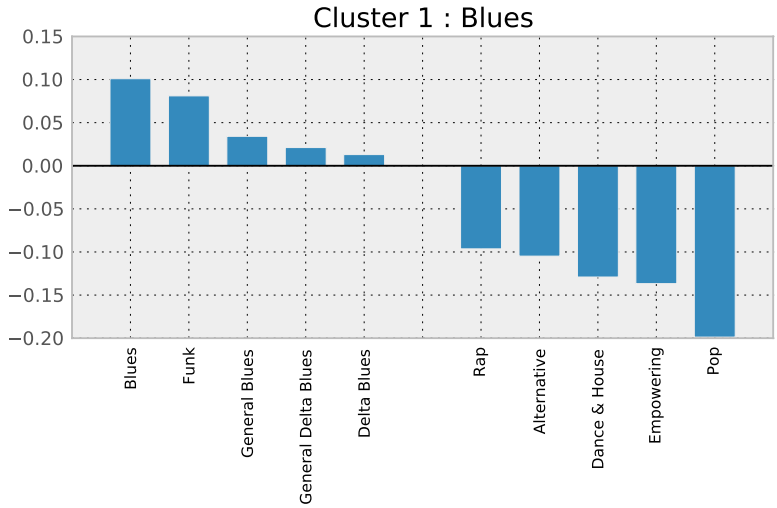


Figure C.22: Funk/Blues-cluster. Size: 26 Silhouette score: 0.253271467363 C: 2

Cluster size	Cluster id	CV-accuracy	C	Silhouette score	Naming
5869	35	0.898289743223	0.005	0.148939409677	Pop
3116	33	0.929112476776	0.005	0.146052179499	Dance & House
2098	28	0.947406031156	0.005	0.280391212148	Rap
1762	36	0.943785431852	0.005	0.140640395479	Indie Rock
1632	32	0.939736077366	0.005	0.112049499506	Rock
1301	31	0.977561812205	0.005	0.33772172736	Holiday
1188	37	0.931113334286	0.005	0.0484131923941	Alternative
1013	17	0.971987994855	0.005	0.261917470475	Metal
655	34	0.960697441761	0.05	0.175428811726	R&B
518	16	0.987566099757	0.5	0.228691120612	Classic U.S. Punk
363	22	0.989995712448	0.5	0.125178693609	Country & Folk
327	27	0.987851936544	0.5	0.281940137752	Spanish Pop
234	2	0.9903291887	0.5	0.285635406315	Progressive Electronic
199	3	0.995855366586	0.5	0.23374599038	Classical
132	12	0.99495021676	0.5	0.124271538529	Jazz
114	30	0.996569958554	0.5	0.149199821065	Children's
97	25	0.995426611405	0.5	0.101858307823	Latin
84	20	0.993330474965	2	0.259850151927	Comedy
77	5	0.997808584632	2	0.118224435319	New Celtic Trad.
54	15	0.998094421419	2	0.145533996878	Religious
34	4	0.997903863561	2	0.279228777653	Solo Instrumental
26	1	0.998570816064	2	0.253271467363	Blues

Table C.1: Full information of clusters explained with logistic regression in the average HAC, k=38, with both genres and moods

Cluster size	Cluster id	CV-accuracy	C	Silhouette score	Naming
21	18	0.999857081606	2	0.730871449618	Audio Book Children
15	8	0.998380258206	2	0.149123810404	Celtic
14	13	0.996522319089	2	0.511898336375	Latin Pop: Female
9	9	None	2	0.136508104702	World
9	24	None	2	0.201008832082	Brazilian Pop
6	19	None	2	0.345179105333	Exercise
5	6	None	2	0.234617300101	Flamenco
4	29	None	2	0.259647433062	Cheerful / Playful
3	7	None	2	0.0504956630475	Norwegian
3	14	None	2	0.305604778888	Islamic
3	21	None	2	0.13273960261	Soft Soulful
2	11	None	2	0.94703015798	Suave / Sultry
1	10	None	2	0.0	
1	23	None	2	0.0	
1	26	None	2	0.0	
1	38	None	2	0.0	

Table C.2: Full information of clustering not explained with logistic regression clustered with Average HAC, k=38, with both genres and moods.