Natural Language Processing and Clustering Optimization for Music Album Recommendation

Nurzhan Kanatzhanov
nurzhan.kanatzhanov@wustl.edu

Pranav Jain
pranav.jain@wustl.edu

Abstract

With the big boom of AI recommendation systems, there has been a proliferation of companies that use the so-called "collaborative filtering" algorithmic approach that predicts what users might enjoy based on their similarity to other users. For instance, Netflix, the popular American streaming service, went as far as announcing a prize of one million dollars for anyone who would improve their recommendation engine by a mere 10% back in 2009 [1], when their service was mostly centered around DVDs rather than streaming. Nowadays, popular music streaming services like Apple Music, Spotify, Amazon Music, follow a similar approach and collect user historical activity in order to determine clusters of similar users to produce individual music suggestions. While current music streaming services utilize a ton of data like user ratings of music tracks, length of listening periods, and song metadata such as mood, energy, and genre, this research focuses on a niche part of song structure, particularly its lyrical content and semantics. In this project, we perform clustering analysis to find similarities in the songwriting of individual albums with the intention of developing a novel recommendation system for users to explore new music.

Keywords: Recommendation system, K-Means, clustering, LDA, PCA, LSA, TF-IDF, NLP.

1 Introduction

"Music gives a soul to the universe, wings to the mind, flight to the imagination, and life to everything."
— Plato

Research in music recommendation systems has gained a lot of popularity thanks to the rapid rise of online music streaming services in the recent years. Services like Spotify, Pandora, and Apple Music have enormous catalogues as they put tens of millions of songs, podcasts, and even audio-books in the hands of their users. Like with anything else, it is hard to crack music fans' tastes and musical needs, as these are subject to a plethora of different factors such as mood and emotional state, lyricism, musicality/melodiousness, and overall genre. Due to this multitude of factors and a vast music inventory, it is important for music streaming services to offer a helping hand to their users in order to lower the cognitive load when trying to choose what to listen to for entertainment. Filtering through millions of songs to recommend a select few is an art, as it is paramount to limit users' decision fatigue (also known as "overchoice") to increase user satisfaction, thus supporting the business model. As one of the top reasons to use a streaming service is to discover new music, almost all music streaming services offer some kind of a music discovery tool for its users. For instance, Apple Music has "New Music Mix" whereas Spotify posts "Discover Weekly" for every single paid customer every Monday morning, which is a "...curated playlist of 30 songs from a variety of artists" [2].

Through the lens of the trendy research in music recommendation systems, this project aims to find out the quality of suggestions being made to users based only on one important feature in music: lyrics. The goal is to analyze various approaches to clustering musical preferences by looking into songwriting choices of various artists in their individual studio albums, ultimately hoping to optimize the strength and validity of those clusters and recommendations. Two experiments are ran in this project: one with latent dirichlet allocation (LDA) and principal component analysis (PCA), and the other with latent semantic analysis (LSA) and two vectorizing methods (TF-IDF and Hashing). Both perform K-Means clustering analysis.

Figure 1: Album cover of The Beatles's "Abbey Road"

2 Related Work

There is a lot of previous work examining music recommendation systems, whether they are based on individual songs, artists, or whole genres.

Some authors focus on clustering analyses (K-Means) as well, but use a multitude of features that they acquire from music service APIs, including information on duration, energy, liveness, valence, tempos, and metadata such as year, language, genres, and eras. They then reduce the dimensionality of all of their data using principal component analysis (PCA) and run K-Means clustering on the dataset first, and then try using a hierarchical clustering model (with Euclidean, Manhattan, and cosine distance metrics), showing that K-Means outperformed hierarchical clustering by about 7% [3].

Ballaney [4] took another approach to music classification—the authors decided to train a five-layer Auto Associative Neural Network (AANN) that would perform an identity mapping on the input space to effectively build a music genre classification system specifically, effectively showing that their model faced difficulties with the size of the feature vector for each music genre.

An extremely interesting approach was in Thompson's [5] paper about lyric-based music genre classification using not just lyrics, but also hidden features such as rhyme density, readability, syllabic metrics, and the occurrence of profanity. The author aimed to improve upon existing literature by hand-crafting own features using the Malmi Rhyme Factor, which is a function that calculates rhyme frequency in lyrics, and the Flesch reading ease score, which is a formula that assesses readability/difficulty of an English text passage. These intuitive features could be readily obtained from music lyrics alone, and they do not require inferred features from a deep-learning model.

3 Methodology

3.1 Data Collection

There was no readily available public dataset that contained a variety of individual artist studio albums, so a new dataset had to be constructed to support our research. A total of 71 randomly generated albums spanning the most popular English music genres (rock, country, pop, R&B, rap, hip hop, etc.) were picked for our project, and all of the lyrics in the albums were scraped from a music lyric website `AZLyrics.com` and stored in `.txt` files. To control for variability in music album length, the dataset was mainly comprised of LP's, or studio albums, which are at least 8–10 tracks long. The average word length of an album in the dataset was $5,037$ words (min. $= 1,341$ words, max. $= 14,691$ words).

3.2 Data Pre-Processing

Following standard procedures in natural language processing, the whole dataset was cleaned up before features were extracted and token frequencies were calculated. The following steps were performed to ensure an equitable system:

i. All lyrics converted to lowercase
ii. Punctuation marks stripped (question/exclamation marks, commas, periods, etc.)
iii. Square bracket text removed (in music lyrics, text within square brackets indicates either a start of a verse or chorus, a specific artist's verse when a song contains multiple different artists, or when a line is repeated multiple times)
iv. Filtered out any "stopwords" using an NLP toolkit containing a list of the most common stopwords (i.e. "the", "you", "I", "and", etc). It is imperative to remove stopwords from text as they do not provide any useful data on the actual topic modeling [6].

After the above steps were performed, token frequencies for all 71 albums were calculated by converting each album into the bag-of-words (BoW) format—a list of tuples with each token (word) and its total count. After that, we used an open-source library for unsupervised topic modeling and natural language processing in Python called `gensim` to train an LDA (Latent Dirichlet Allocation) model (see Section 3.3 for further explanation).

3.3 Latent Dirichlet Allocation (LDA) Model

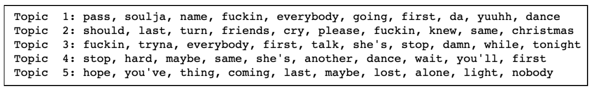
In natural language processing, Latent Dirichlet Allocation (LDA) is a three-level hierarchical Bayesian probabilistic model that discovers abstract topics in text (also known as topic modeling). It works by finding natural groups of topics based on frequency probabilities. That is why an important pre-processing step is to remove stopwords that do not carry any important topic information. However, unlike the standard definition of clustering, LDA uses "soft" or "fuzzy" clustering, meaning that data points can belong to more than just one cluster. As mentioned before, one of the important assumptions in LDA is that each document is simply a bag of words, meaning that the order of the words and their sentence roles (verbs, nouns, etc.) are not considered. Before training any LDA model, a required parameter is the number of topics to look for (`k` number of topics is pre-determined), just like how in K-Means the number of clusters to make is an input as well [7].

The main part of the LDA algorithm is actually calculating the probability of certain words belonging to some abstract topic. The following is a quick guide of how exactly the algorithm works in the scenario of our 71 album dataset after the pre-processing step:

i. Iterate through each of the 71 albums (`.txt` documents), and assign each word to one of the **`k`** topics.
ii. For each album **`a`**, look at each word **`w`** and calculate the following probabilities:
   (a) Proportion of words in a single album **`a`** that are assigned to one of the **`k`** topics **`t`**:
   $$Pr[\text{topic } t \mid \text{album } a]$$
   Intuitively, if a lot of words from some album **`a`** belong to a topic **`t`**, it is more probable that word **`w`** belongs to topic **`t`** as well.
   (b) Proportion of assignments to topic **`t`** over all albums that come from word **`w`**:
   $$Pr[\text{word } w \mid \text{topic } t]$$
   This probability tries to see how many albums are in a specific topic **`t`** because of word **`w`**.
iii. After the above steps have been performed, the algorithm updates the probability of some word **`w`** belonging to a particular topic **`t`**, following the Bayes Theorem:

$$Pr[\text{word } w \text{ \&\& topic } t] = \\ Pr[\text{topic } t \mid \text{album } a] \times Pr[\text{word } w \mid \text{topic } t]$$

For instance, during one of the iterations with **`k`** $=$ 5, the top 10 words in each topic could be recovered from the LDA model (see Fig. 1), as well as each albums distribution of topics by percentage (see Fig. 2):



```
Topic  1: pass, soulja, name, fuckin, everybody, going, first, da, yuuhh, dance
Topic  2: should, last, turn, friends, cry, please, fuckin, knew, same, christmas
Topic  3: fuckin, tryna, everybody, first, talk, she's, stop, damn, while, tonight
Topic  4: stop, hard, maybe, same, she's, another, dance, wait, you'll, first
Topic  5: hope, you've, thing, coming, last, maybe, lost, alone, light, nobody
```

Figure 2: LDA Model's top 10 words in each topic (1st experiment)

```
Elton John Blue Moves:
Topic 4 (99.88%)

Michael Buble Christmas:
Topic 2 (99.95%)

Elliot Smith Figure 8:
Topic 2 (86.60%), Topic 4 (12.44%)

Amy Winehouse Back To Black:
Topic 2 (90.24%), Topic 4 (9.69%)

Baby Keem Die For My Bitch:
Topic 3 (99.95%)
```

Figure 3: Topic breakdown for a subset of 5 albums in the dataset by percentage (1st experiment)

3.4 Silhouette Coefficient

How does one measure goodness of fit of a clustering technique? Unlike supervised machine learning algorithms such as support vector machines (SVM), k-nearest neighbors (k-NN), and decision trees that have tons of metrics checking the discrepancy between observed and expected values like accuracy, sensitivity, and correlation coefficient, what is the approach with clustering algorithms like K-Means? The answer is the silhouette coefficient.

The silhouette coefficient, or score, is a metric that measures how good of a result a clustering algorithm returned. It measures consistency and validity within clusters of data by measuring how similar a data point is to its own assigned cluster compared to other clusters. The silhouette coefficient ranges from -1 to $+1$, where a score of -1 indicates that clusters were assigned in a completely wrong way, $+1$ indicates that clusters are perfectly distinguished, and 0 indicates that clusters are indifferent—the distance between clusters is insignificant [8].

Thus, to create a good music recommendation system based only on lyrics, this research focuses on maximizing the silhouette coefficient. To calculate it, we use Python's `sklearn` package's silhouette function that computes the mean silhouette coefficient of all samples using the mean intra-cluster distance and the mean nearest-cluster distance for each sample. Mathematically, the silhouette coefficient is calculated as such:

$$\text{Silhouette Coefficient} = \frac{b - a}{max(a,\ b)}$$

In the above formula, a is the average intra-cluster distance (distance between each data point within a single cluster), and b is the average inter-cluster distance (distance between all of the clusters). Since this project calls for use of PCA (Section 3.5) for dimensionality reduction, the Euclidean distance metric was chosen for a simple 2D projection.

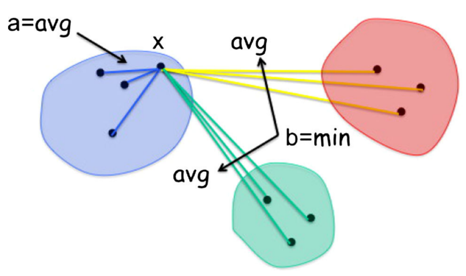


Figure 4: Visual representation of the silhouette coefficient calculation

3.5 Principal Component Analysis (PCA)

Due to the innate nature of the LDA model used in this research, the dataset contains multiple features that are the given number of topics that the LDA model uses for topic modeling. That is the curse of dimensionality: as the number of dimensions/attributes grows, so does the complexity of a model, and thus error increases and meaningfulness and interpretability of results get worse [9]. In the context of this research on clustering optimization, how does one visualize a multitude of dimensions to see where clusters of music albums would end up?

That is where Principal Component Analysis (PCA) excels. PCA is a dimensionality reduction technique that transforms datasets with a large set of variables into smaller sets that still retain a big chunk of the information in the original large dataset. However, PCA sacrifices a bit of accuracy when reducing dimensionality for simplicity and interpretability of models—exactly what is needed to visualize cluster data. PCA works in the following way:

i. Initial variables are standardized. Standardization is an important first step for PCA since there could be large differences between the ranges of initial variables, so standardizing transforms them to the same scale. The standardization technique is as follows:

$$z = \frac{c - \mu}{\sigma}$$

where z is the new standardized value (z-score), c is the original value of each variable,

μ is the mean of the feature/attribute, and σ is its standard deviation.

ii. Covariance matrix is computed. Computation of the covariance matrix tells PCA if there are redundant dimensions that are highly correlated with each other that could be potentially reduced. The covariance matrix is a symmetric $n \times n$ matrix, where n is the number of dimensions, which in the scenario of this project is the number of topics in LDA k.

iii. Principals components are identified. After getting a covariance matrix for the variables, eigenvalues are calculated in order to determine principal components of the data, which are the new variables that are mixtures of the original variables and explain a maximal amount of variance. The new variables are essentially a compressed version of the larger set of original variables.

iv. Feature vector is formed and original data is reoriented along the principal components axes. As a final step in principal component analysis, the feature vector, which is a reduced vector of all the principal components found in the previous step, is calculated. The original input data, however, is still set to its original axes (dimensions), so this final step aims to recast that data to the axes represented by the principal components. This is performed by taking the transpose of the original dataset (that was standardized first) and multiplying it by the transpose of the feature vector. The formula is like this:

$$\text{final data} = \text{orig. data}^T \times \text{feature vect.}^T$$

Ultimately, the high-dimension dataset is reduced to just a few (we use only 2 principal components to visualize music clusters in 2D).

3.6 TF-IDF and Hashing Vectorizer

TF-IDF (Text Frequency - Inverse Document Frequency) is a technique to preprocess text data for machine learning and statistical analysis. It is a smart way to measure word frequency but at the same time extract actual key words and topical focus for each sample of text data it examines. It does this by measuring two values:

i. Tokenizing each word and counting its document frequency (frequency per sample), proportionally increasing the number value assigned to each word with logarithmic scaling.

ii. Weighting each word based on how frequently it appears across all documents (entire dataset)—negatively weighting words that appear across all documents and do not seem to be the primary topical focus of that specific document.

To get the final TF-IDF value, these two terms are multiplied. Formally, this can summarized in the below mathematical formula with N being the number of documents (size of the dataset), D being the entire set of documents, d being the specific document of interest, and t standing for the word of interest within document d [11]:

$$log(1 + freq(t, d)) * log(\frac{N}{count(d \in D : t \in d)})$$

To implement the above, the `tfidfvectorizer` package from the `sklearn` library was used in conjunction with the hashing vectorizer [13]. The hashing vectorizer is a way to tokenize the text in an efficient and scalable way. This is used because the matrix here works well for adjusting high dimensional data for most of the clustering algorithms where they would otherwise perform relatively poorly.

3.7 Latent Semantic Analysis (LSA)

The TF-IDF and hashing vectorizer methods are used as an additional pre-processing step for Latent Semantic Analysis (LSA). LSA is used here for two primary reasons:

i. First is to extract the semantic meaning of individual words using rigorous statistical analysis. LSA is able to use a words contextual use within the rest of the document to make reasonable assumptions about the relational meaning between words. This account of not only relative topical focus but specific word frequency should greatly boost the performance of clustering.

ii. Second, LSA can take the place of PCA in reducing the dimensionality of the dataset and optimizing it for best use within our clustering algorithms. This can be quite useful if the TF-IDF pre-processing step is taken as PCA has a tough time handling sparse matrices.

The methodology of LSA is to take our data matrix and use SVD decomposition to further decompose it into three matrices. Due to the nature of matrix decomposition, this step is used to reduce dimensionality as well by only using a select few rows and columns from the decomposition. The principle here is that we can still get a close approximation of the original matrix even using this limited information. Word trees are then constructed to derive semantic meaning and the matrices are updated, recombined and re weighted to reflect contextual information for each word within each sample [12].

It should be noted that LSA like PCA, sacrifices accuracy in order to reduce dimensionality. It also typically has a comparatively poor silhouette score as a result of condensing a large amount of information within two dimensions

3.8 Main Analysis

We combined all of the pieces described in previous sections to form two experiments. For the first experiment, after original dataset pre-processing, filtering, and clean-up, an LDA model was trained on multiple different number of topics. The LDA matrix was reduced to 2 dimensions using principal component analysis and then fit on the K-Means clustering algorithm (with a pre-determined number of clusters $k = 5$, and a pre-set random start state for reproducibility of results, i.e. making randomness deterministic). The experiment was repeated 100 times to average out results when trying to optimize for the best number of topics used for topic modeling. At the end, the silhouette score for each iteration was calculated and recorded.

For the second experiment a TF-IDF and hashing vectorizer were run on the raw data to get an interpreted high dimensional sparse matrix. This matrix was then run through LSA to extract and apply semantic meaning and reduce dimensionality. The results of this experiment were then optimized by tuning for a variety of parameters based on the silhouette score, including the number of clusters specified, the features extracted by the hashing vectorizer, and the clustering algorithm used between K-Means and Gaussian EM.

4 Results and Discussion

4.1 First Experiment

After the experiment described above was performed, over 100 iterations, the total time taken to run it was **758.8** seconds (12.6 minutes). The number of topics that maximized the silhouette score for the formed clusters was **9**, with an average silhouette score of **62.86%**, indicating that the clustering configuration is appropriate.

Baby Keem - Die For My Bitch	**Amy Winehouse - Back To Black**
Danny Brown - Atrocity Exhibition	The Eagles - Hotel California
Eminem - Slim Shady LP	Linkin Park - Hybrid Theory
Nirvana - In Utero	Elton John - Blue Moves

Ariana Grande - Dangerous Woman	**David Bowie Blackstar**
NIKI - Moonchild	The Beach Boys - Little Deuce Coupe
Olivia Rodrigo - Sour	Behemoth - The Satanist
Gorillaz - Demon Days	The Beatles - Sgt. Pepper

Table 1: Recommendations based on sparse matrix similarity scores

From the visual presentation of the 5 K-Means clusters, we could see that clusters are significant and consistent in their shape (Fig. 5). In the provided visual, each color blob represents a different cluster, each colored dot represents a single album (data point) within that cluster, and each triangle represents the centroid (or, cluster center) of each cluster. The "shadows" or shapes of each cluster were drawn with the help of the convex hull (`SciPy` package), which is the smallest set of connections between data points to form a polygon that encloses all the points. It is not extremely accurate but provides good insight on the contour of each clustered dataset.

Table 2 (see below) portrays the constituency of each cluster. Under closer inspection, the music album recommendation system based on each cluster would perform an agreeable, valid job. However, there seem to be some albums that are placed non-intuitively; nonetheless, this intuition is based on our preconceived notions of musical genres, and there may be connections within the lyrics being detected that transcend those genres and focus more on topical relevance.

Using `gensim`'s sparse matrix similarity function, we are also able to provide recommendations as we get a vector of most similar texts. The function computes similarity against a collection of documents by storing the sparse index matrix in memory, calculating the similarity metric by using cosine distance between two vectors. The results are provided in Table 1 for a small subset of the albums and displaying only the top 3 similar texts.

4.2 Second Experiment

The next experiment was run with the same basic framework. As mentioned above pre processing was done with the TF-IDF and hashing vectorizer. The main focus of the first part was to tune the three parameters mentioned above. For the tuning of parameters all steps of the experiment, including pre processing and fitting had to be run from scratch. Feature extraction (which controls the size of the sparse matrix outputted by the vectorizer, essentially how much information from the text was included) was tuned with 20 values in the range of 200 to 100,000. The number of clusters was tuned with 12 values in the range 4 to 15.

Lastly, the two algorithms tried were K-Means (a version of this called Mini Batch K-Means was used in practice, which is a more efficient version of K-Means that works by clustering small batches of the dataset and then aggregating the results together) and Gaussian Mixture, the `sklearn` implementation of EM Gaussian Clustering. The first difference of note between the first and second experiments was the runtime. The above tuning implies that about **480** iterations of this experiment were run, and as opposed to the **12.5** minutes taken by experiment 1 for **100** iterations, experiment 2 took only under **1** minute for nearly 5 times as many iterations.

The silhouette score for the most optimal K-Means combination of parameters was **68.41%**. For the most optimal Gaussian mixture combination it was **67.68%**. These interestingly gave very different optimal parameters: **11** clusters and **400** features extracted for K-Means and **5** clusters and **4000** features extracted for Gaussian mixture.

The results shown in Figure 6 and Table 3 are for the Gaussian iteration of this experiment. This was shown as it is more comparable with experiment 1 and the difference in silhouette score between this and K-Means was negligible. What is first fascinating is the shape of the graph. The graph is arranged in a crescent moon like shape as opposed to the typical spread out clusters but still retains a comparable score. Examining Table 3's results also leads to a more intuitive grouping of albums, for example almost all of the hip-hop albums in the sample are in cluster 3. As mentioned before, the similarity to score may be as a result that transcends genre. Perhaps, experiment 1 extrapolates on this hidden factor, while experiment 2 is more in line with our own preconceived notions.

5 Future Work

Given the results of our analysis, it could be seen that purely lyrical analysis might not be the most lucrative solution to music album recommendation systems, and it is obvious that adding multiple other features described in previous related research increases recommendation accuracy and thus user satisfaction.

There are some future potential improvements we could implement: (1) adding additional features like profanity scores, syllable structures (syllables/characters per word), length of lyric lines, unique words used, and any other natural language processing aspects; (2) trying various clustering methods (DBSCAN, Mini-batch, Agglomerative, etc.); and (3) finding a way to pre-process data more accurately.

It would be quite fascinating to follow [5] and hand-craft features from lyrics alone, twisting Thompson's idea to extend it to music album recommendation rather than music genre classification. As multiple clustering methods have their own strengths and weaknesses, it would have been interesting to look at their performance on the same dataset, but this would be outside of the scope of this research. Ultimately, one of the interesting results we saw when looking at most frequent words used in all of the albums is that we saw the same words appearing twice, except that they were spelled differently. This problem stems from the lyrical phenomenon used in genres like rap and hip-hop, where words are often spelled phonetically. For instance, the word "coming" could have been displayed as "comin", or "yes" and "ye". The potential solution to that problem would be to cross-check all words with words in an English dictionary, but that could be too extreme and punishing for rap and hip-hop albums [9].

6 Contributions

Both authors of this paper contributed equally, dividing data collection, pre-processing, experimental, and write-up work fairly.

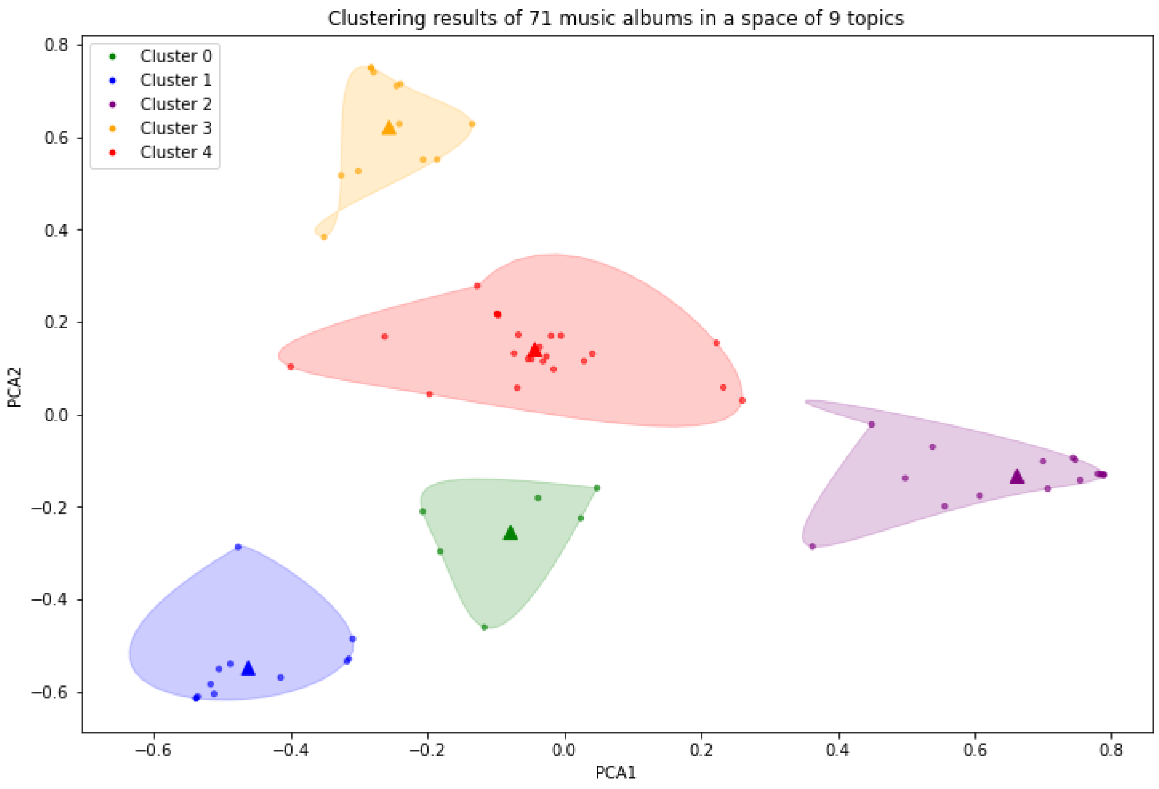


Figure 5: Resulting clusters on a 2D PCA plane (1st experiment)

Cluster 1	**Cluster 2**	**Cluster 3**	**Cluster 4**	**Cluster 5**
Drake - Scorpion	The Strokes - Room On Fire	Elton John - Blue Moves	Elliot Smith - Figure 8	Baby Keem - Die For My Bitch
Eminem - The Eminem Show	XXXTentacion - ?	Michael Buble - Christmas	Taylor Swift - Folklore	Behemoth - The Satanist
Britney Spears - Oops I Did It Again	Tyler the Creator - Igor	Amy Winehouse - Back To Black	Judas Priest - Defenders Of Faith	Soulja Boy - Souljaboytellem Com
Beatles - White Album	Turnstile - Glow On	ACDC - Back In Black	Kanye West - College Dropout	Radiohead - In Rainbows
Drake - More Life	Limp Bizkit - Significant Other	ACDC - Dirty Deeds Done Dirt Cheap	NIKI - Moonchild	Nirvana - In Utero
Sam Smith - The Thrill Of It All	ACDC - Blow Up Your Video	Ed Sheeran - +	Olivia Rodrigo - Sour	Future - HNDRXX
	Imagine Dragons - Night Visions	The Smiths - The Smiths	Ariana Grande - Dangerous Woman	Tame Impala - Currents
	Ozzy Osbourne - Speak Of The Devil	Linkin Park - Hybrid Theory	Gorillaz - Demon Days	Injury Reserve - By The Time I Get To Phoenix
	Prince - Purple Rain	The Eagles - Hotel California	Green Day - American Idiot	Amine - Twopointfive
	Bastille Bad Blood	Adele - 25	Kanye West - Graduation	David Bowie - Blackstar
	’Kesha Warrior’,	Beatles - With The Beatles	Fleetwood Mac - Then Play On	Wu Tang Clan - Enter The Wutang
	’Rina Sawayama Sawayama’,	Charli XCX - Charli	The Beatles - Sgt. Pepper	Big Sean - Dark Sky Paradise
	’Billie Eilish Happier Than Ever	Jaden Smith - Syre		Kanye West - Late Registration
		Neutral Milk - Aeroplane Over The Sea		Rich Brian - Amen
		Anderson .Paak - Oxnard		The Doors - Strange Days
		Chris Stapleton - Starting Over		Metallica - The Black Album
				Eminem - The Marshall Mathers LP
				Death Grips - The Money Store
				Eminem - Slim Shady LP
				Lil Tecca - We Love You Tecca
				Outkast - Aquemini
				The Beach Boys - Little Deuce Coupe
				Drake - Views
				Danny Brown - Atrocity Exhibition

Table 2: Music album breakdown of each cluster (1st experiment)

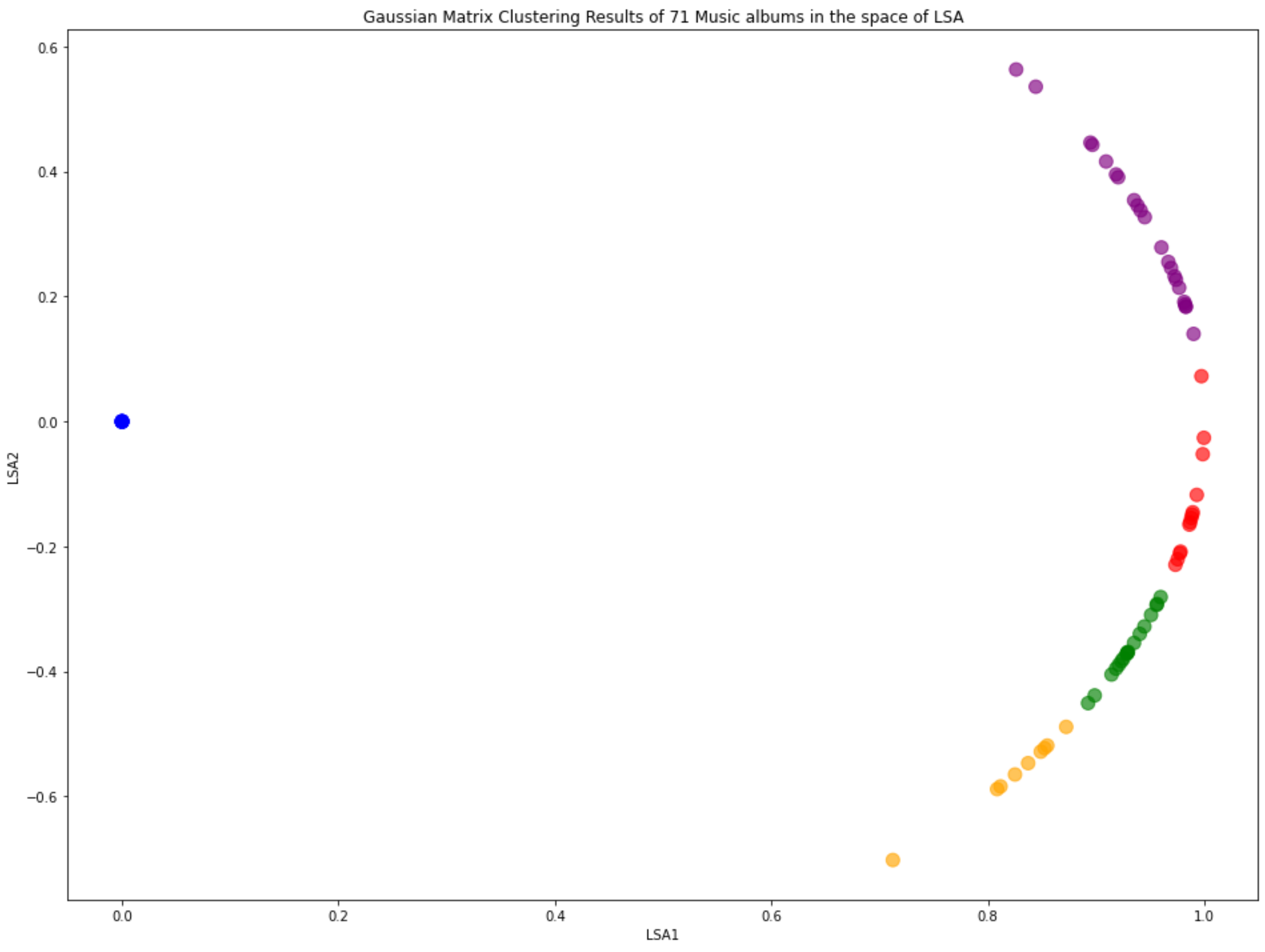


Figure 6: Resulting Gaussian mixture clusters on a 2D LSA plane (2nd experiment)

Cluster 1	**Cluster 2**	**Cluster 3**	**Cluster 4**	**Cluster 5**
The Beatles - With The Beatles	Behemoth The Satanist’,	Kanye Graduation’,	Beatles Sgt Pepper’,	ACDC Blow Up Your Video’,
Green Day - American Idiot	’Death Grips The Money Store’,	’Drake Scorpion’,	’Michael Buble Christmas’,	ACDC - Back In Black’,
Nirvana - In Utero	’Tyler The Creator Igor’,	’Drake More Life’,	’Imagine Dragons Night Visions’,	’ACDC - Dirty Deeds Done Dirt Cheap’,
Adele - 25	’Ozzy Osbourne Speak Of The Devil’,	’Drake Views’,	’Ed Sheeran +’,	’Beatles White Album’,
Gorillaz - Demon Days	’Neutral Milk Aeroplane Over The Sea’,	’Kanye Late Registration’,	’Bastille Bad Blood’,	’Billie Eilish Happier Than Ever’,
’Elton John Blue Moves’,	’Olivia Rodrigo Sour’,	’Kanye College Dropout’,	’Metallica The Black Album’,	’Charli Xcx Charli’,
’Tame Impala Currents’,	’Prince Purple Rain’,	’Eminem Slim Shady Lp’,	’Judas Priest Defenders Of Faith’,	’Chris Stapleton Starting Over’,
’Niki Moonchild’,	’Turnstile Glow On’,	’Eminem The Marshall Mathers Lp’,	’The Smiths The Smiths’,	’Limp Bizkit Significant Other’,
’Amy Winehouse Back To Black’,	’Elliot Smith Figure 8	’Eminem The Eminem Show’,	’Sam Smith The Thrill Of It All	’Britney Spears Oops I Did It Again’,
’The Eagles Hotel California’,	Bastille Bad Blood	’Danny Brown Atrocity Exhibition’,	Kanye West - Graduation	’Kesha Warrior’,
’David Bowie Blackstar’,	’Kesha Warrior’,	’Big Sean Dark Sky Paradise’,	Fleetwood Mac - Then Play On	’Jaden Smith Syre’,
’The Doors Strange Days’,	’Rina Sawayama Sawayama’,	’Soulja Boy Souljaboytellem Com’,	The Beatles - Sgt. Pepper	’Rina Sawayama Sawayama’,
’Radiohead In Rainbows’,	’Billie Eilish Happier Than Ever	’Future Hndrxx’,		’Ariana Grande Dangerous Woman’
’The Strokes Room On Fire’,		’Anderson Paak Oxnard’,		Rich Brian - Amen
’Linkin Park Hybrid Theory’,		’Xxxtentacion Question Mark’,		The Doors - Strange Days
’Taylor Swift Folklore’,		’Outkast Aquemini’,		Metallica - The Black Album
’Fleetwood Mac Then Play On’,		’Wu Tang Clan Enter The Wutang’,		Eminem - The Marshall Mathers LP
’The Beach Boys Little Deuce Coupe’		’Baby Keem Die For My Bitch’,		Death Grips - The Money Store
		’Rich Brian Amen’,		Eminem - Slim Shady LP
		’Amine Twopointfive’,		Lil Tecca - We Love You Tecca
		’Lil Tecca We Love You Tecca’,		Outkast - Aquemini
		’Injury Reserve By The Time I Get To Phoenix		The Beach Boys - Little Deuce Coupe
				Drake - Views
				Danny Brown - Atrocity Exhibition

Table 3: Gaussian mixture clustering with LSA (2nd experiment)

7 References

[1] Johnston, Casey. “Netflix Never Used Its $1 Million Algorithm Due to Engineering Costs.” Wired, Conde Nast, 16 Apr. 2012, https://www.wired.com/2012/04/netflix-prize-costs/.

[2] “Five Ways to Make Your Discover Weekly Playlists Even More Personalized.” Spotify, 1 May 2019, https://newsroom.spotify.com/2019-05-02/five-ways-to-make-your-discover-weekly-playlists-even-more-personalized/.

[3] Langensiepen, Caroline, et al. “Using PCA and K-Means to Predict Likeable Songs from Playlist Information.” 2018 UKSim-AMSS 20th International Conference on Modelling & Simulation. https://uksim.info/uksim2018/CD/p26.pdf

[4] Ballaney, A. K. et al. “Music Genre Classification using Auto-Associative Neural Networks.” 2007, DA-IICT.

[5] Thompson, Curtis. “Lyric-Based Classification of Music Genres Using Hand-Crafted Features.” International Journal of Undergraduate Research 14:2 (2021).

[6] Fan A., Doshi-Velez F., Miratrix L. “Assessing Topic Model Relevance: Evaluation and Informative Priors.” Stat Anal Data Min: The ASA Data Sci Journal. 2019;12:210–222. https://doi.org/10.1002/sam.11415

[7] Blei, David, et al. “Latent Dirichlet Allocation.” Journal of Machine Learning Research 3 (2003) 993-1022, https://dl.acm.org/doi/10.5555/944919.944937.

[8] Bhardwaj, Ashutosh. “Silhouette Coefficient: Validating Clustering Techniques.” Medium, Towards Data Science, 27 May 2020, https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c.

[9] Canicatti, Anthony. “Song Genre Classification via Lyric Text Mining.” Int’l Conf. Data Mining (DMIN’16), http://worldcomp-proceedings.com/proc/p2016/DMI8052.pdf.

[10] “Clustering Text Documents Using K-Means.” Scikit, https://scikit-learn.org/stable/auto_examples/text/plot-document-clustering.html.

[11] Stecanella, Bruno. “Understanding TF-ID: A Simple Introduction.” MonkeyLearn Blog, 10 May 2019, https://monkeylearn.com/blog/what-is-tf-idf/.

[12] Landauer, Thomas K. “An Introduction to Latent Semantic Analysis - LSA.” 1998, http://lsa.colorado.edu/papers/dp1.LSAintro.pdf.

[13] “Clustering Text Documents Using K-Means.” Scikit, https://scikit-learn.org/stable/autoexamples/text/plotdocument-clustering.html.