Data-Driven Harmonies: Exploring Artist Clusters in LastFM's Dataset

James Harvey

Abstract - **This paper explores the application of clustering algorithms to the HetRec LastFM dataset to identify and analyze similarities among musical artists. Utilizing a combination of data preprocessing techniques and advanced clustering methods, such as K-Means and DBSCAN, the study navigates through the complexities of user data, artist metadata, and social network information inherent in the dataset.**

Introduction

The music streaming industry represents a revolutionary shift in how music is consumed and distributed. This industry has rapidly evolved into a primary source of music consumption, offering vast libraries of songs and albums accessible anytime and anywhere. Dominated by key players like Spotify, Apple Music, and Amazon Music, making up 50.7% of the market share (1), music streaming has revolutionized how audiences engage with music, altered marketing strategies, and the overall discovery process of music. With a global revenue of $17.5 billion (2), streaming continues to redefine the boundaries of accessibility, personalisation, and global reach in the music industry.

In the digital age, where the volume of music available online is vast and continually growing, the ability to accurately recommend music has become crucial. On Spotify, for instance, over one-third of all new artist discoveries happen through "Made for You" recommendation sessions (3). Accurate music recommendation serves several essential purposes:

- **Personalized User Experience:** With an overwhelming array of choices, listeners can easily find themselves lost in a sea of musical options. Accurate recommendations help navigate this abundance, guiding users to tracks and artists that align with their tastes and preferences. This personalization fosters a more engaging and satisfying listening experience, increasing user retention and satisfaction.
- **Commercial Success and Sustainability:** For music platforms and streaming services, the ability to deliver accurate recommendations drives user engagement and subscription retention, critical factors in commercial success. By keeping users engaged and subscribed, these platforms ensure a steady revenue stream, which is crucial for their sustainability and, by extension, for the artists who rely on them for income.
- **Industry Analytics and Strategic Development:** The sophisticated algorithms behind accurate music recommendation systems generate a wealth of data reflecting user preferences and listening trends. Record labels, artists, and marketers can leverage these insights to tailor their creative and promotional strategies, aligning their offerings more closely with listener trends and demands. This data-driven approach enables the industry to adapt dynamically to changing tastes, ensuring that productions and marketing efforts resonate more effectively with target audiences.

Accurate music recommendation is now fundamental in the modern music ecosystem. It bridges the gap between the vast array of available music and the individual listener, supports the discovery and promotion of artists, drives the commercial success of music platforms, and provides valuable industry insights. As such, research and development in this area are both technologically and economically significant.

Background

Last.fm is a music streaming and recommendation service that has pioneered the online music industry. Founded in 2002 (4), it gained popularity for its unique approach to music streaming, combining social networking features with personalized music recommendations. The platform uses "scrobbling" to track users' listening habits across various music services and devices, creating detailed profiles of their musical tastes. These profiles are then used to recommend new music, generate custom radio stations, and connect users with similar music interests. Last.fm also offers a wealth of information about artists, albums, and tracks, making it a valuable resource for music discovery and exploration. Last.fm's rich dataset of user preferences and behaviours has been instrumental in the research and development of recommendation algorithms, exemplifying the impact of data analytics in shaping the digital music experience.

The Data

A. HetRec2011 Last.fm Dataset

The dataset is a summarized, sanitized subset of the original released at The 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems (5). The HetRec LastFM dataset is a rich collection of music-related data compiled from the LastFM website. This dataset is particularly valuable for research in music recommendation systems and social network analysis. It contains diverse information, including detailed user profiles, artist biographies, and extensive records of user listening habits. Additionally, it features social networking data, such as user connections, and user-generated content like tags assigned to artists and tracks. This multifaceted dataset allows researchers and

developers to explore and model complex interactions between users and their musical preferences. It provides a comprehensive framework for understanding music consumption patterns and enhancing personalized music recommendation technologies.

Hypothesis

Given the importance music recommendation has within the music industry we draw the following hypothesis:

1. The use of social tagging on music streaming platforms such as Last.fm can be used to identify similar artists and provide convincing recommendations.

2. Clustering techniques can be employed to group together musically similar artists.

Software and Techniques

A. t-SNE
t-Distributed Stochastic Neighbor Embedding (t-SNE) is a powerful machine learning algorithm for dimensionality reduction and visualization of complex, high-dimensional datasets (6). Renowned for its effectiveness in revealing the intrinsic structure and patterns within data, t-SNE operates by converting similarities between data points into joint probabilities and then minimizing the divergence between these probabilities in both the high-dimensional and low-dimensional spaces. The t-SNE algorithm effectively reduces complex, high-dimensional data into a more manageable two or three-dimensional space, facilitating clearer visualization and interpretation of the underlying patterns and relationships within the data (7). t-SNE has become a staple in exploratory data analysis, especially in fields like bioinformatics, image processing, and natural language processing, where understanding the underlying relationships within complex datasets is crucial.

B. Python
Renowned for its versatility and the extensive support of libraries for statistical analysis, machine learning, and data visualization. Its syntax is clear and intuitive, making it an accessible yet powerful tool for analysts and developers to perform complex data manipulation and analytical tasks efficiently.

C. Pandas
A versatile and efficient data manipulation tool within Python. Pandas provides robust capabilities for data cleaning, transformation, and analysis.

D. SciKit-Learn
SciKit-Learn is an open-source machine learning library for Python. It is known for its simplicity and accessibility, providing a wide range of supervised and unsupervised learning algorithms.

E. Microsoft Excel
Microsoft Excel provides a user-friendly interface combined with powerful analytical capabilities. Its wide array of functions, from basic data organization to complex statistical analysis, enables users to perform in-depth data exploration and gain actionable insights.

F. Last.fm API
The Last.fm API provides access to the extensive music data collected by Last.fm. This includes information about songs, artists, albums, and user-generated data like tags, playlists, and user profiles (8).

Data Cleaning

The HetRec2011 Last.fm Dataset had already undergone sanitisation. The sanitisation of the HetRec2011 dataset includes:

(a) Artist name misspelling correction and standardization

(b) Reassignment of artists referenced with two or more artist id's

(c) Removal of artists listed as 'unknown' or through their website addresses

Two dataset files were selected and preprocessed for use in this work:

1. **user_taggedartists.csv** A listing of 186,749 artist tag assignments by user. It contains the tag assignments of artists provided by each particular user.

2. **user_scrobbles.csv** containing 92,792 scrobble counts for 17,493 artists generated by 1,892 users.

The main benefit of using tags was that users already defined the features. The drawback of tags was that due to their user-defined nature, many weren't used by more than one user. For example, tag #4541 was "if this was a pokemon i would catch it", which was not likely to be commonly used enough to use as a prominent feature.

To deal with this problem a python script was used to group all of the used tags and then only use the top 100 tags amongst all users. The most used tag "rock" had 7,503 tags, while 100th most used tag "sad" had 315 tags. The entries that used other tags were removed from the dataset. After this operation 63% of user-artist-tag relations were preserved.

The next step was to ensure that there was only one entry for each user-artist relationship whilst preserving the tags associated with a user-artist relationship. This was done using one-hot encoding. A Python script was created to increase the dimensionality of the data, having a field for each of the top 100 tags. If a user-artist relationship was associated with a tag, the respective field was indicated with a one. If a tag was not associated with a user-artist pair, then

the field had a value of zero.

Once transformed, the dataset was grouped by artist, and the average of the tag's one hot value was taken. This resulted in a dataset where each artist has a value between zero and one for each of the top 100 tags describing that artist.

Item profiles are defined in terms of the top 100 tags. Depending on the artist's popularity, some artists had much more plays and tags than others. In order to remove bias, the items are normalized by dividing each artist's total tag count for a given tag by the number of unique users who tagged that artist.

In effect, this was saying, "Of all users who tagged artist X, how many included at least tag Y?" For instance, 66% of users who tagged Radiohead tagged them with "Rock".

Some artists had only been tagged by a single user so to gain consistent and reliable results the artists were ordered from most tagged to least tagged. Initial investigation into whether or not this information can be used to determine similar artist involved using the cosine similarity between two artists.

Data Analysis

Cosine similarity is a metric used to measure the similarity of vectors, regardless of size (9). Mathematically, it calculates the cosine of the angle between two vectors projected in a multi-dimensional space. A python script was written to create a vector representing the top 100 tags associated with an artist. The cosine between two vectors can then be calculated to measure the similarity. To test this metric, artists

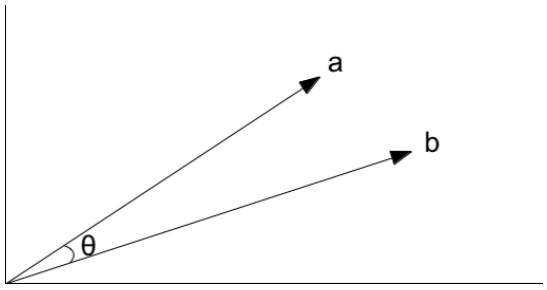


Fig. 1. Cosine Similarity

known to be similar were used. For example, Madonna and Kylie Minogue are similar (10), yielding a cosine similarity of 95.1%. In contrast, Madonna and Radiohead yielded a much lower similarity of 16.1%. A similarity matrix was created in order to examine all of the relationships. The similarity matrix calculates the cosine similarity between all possible artists and stores it in a matrix. This was again performed using a Python script. The result was then exported as a .csv file and examined within Microsoft Excel. A part of the similarity matrix can be seen in fig 2. Artists that are given a higher cosine similarity can be seen in green, and those that are different from each other in red. The results from the similarity matrix were promising upon visual inspection as artists know to be similar were assigned a green cell and artists know to be unalike were assigned a red cell.

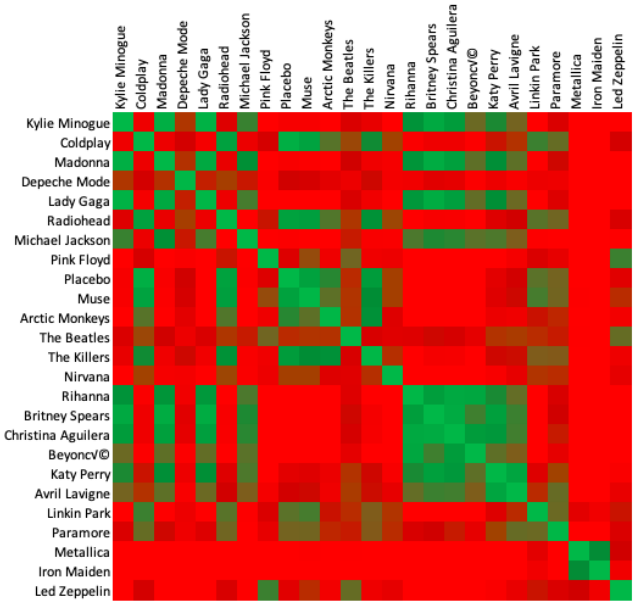
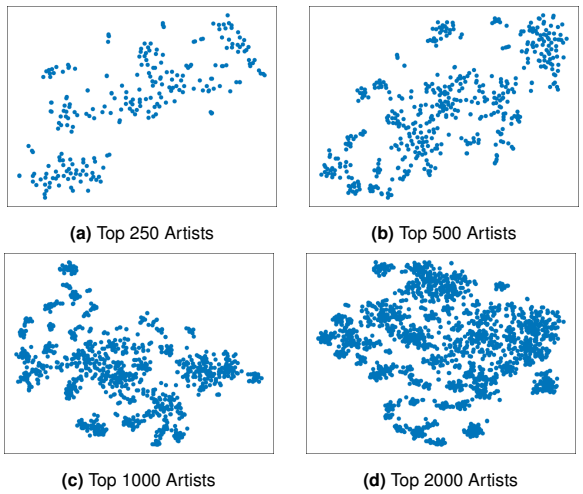


Fig. 2. A cosine similarity matrix for the top 25 artists.

As the information extracted from the dataset gave clear indication that the similarity between artists can be found from tags alone. It made sense to investigate the data further to see if it can be clustered such that all artists belonging to the same cluster are similar to each other.

To test this the t-SNE algorithm was used to visualise the data in two-dimensions. This was performed for the top 250, 500, 1000, 2000 artists. The results can be seen below in fig 3.



(a) Top 250 Artists **(b)** Top 500 Artists

(c) Top 1000 Artists **(d)** Top 2000 Artists

Fig. 3. Using the t-SNE algorithm to visualise the tag data in two-dimensions for a varying number of artists

The results from the t-SNE transformation were promising, as clear clusters could be identified for all numbers of top artists. Therefore, it was viable to investigate clustering

further. Where fewer artists were used, larger, more sparse clusters were present. As the number of artists increases, smaller dense clusters become present. As clusters have been visually identified, different clustering algorithms can be chosen to analyse the data.

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is an algorithm that groups points that are closely packed together while marking outliers, which lie alone in low-density regions (11). It is characterized by two main parameters: the minimum number of points required to form a cluster (*MinPts*) and the radius of the neighborhood around each point (ϵ). An advantage of using the DBSCAN algorithm is that the number of clusters is not a parameter. This is practical for the project as it is unknown how many groups of artists have a similar music profile. Other advantages of the DBSCAN algorithm include the agility to find unusually shaped clusters and the noise tolerance.

However, setting values for ϵ and *MinPts* is not intuitive, and the algorithm can be slow to execute, especially in higher dimensions.

t-SNE based clustering:
One approach was to cluster the data after the t-SNE transformation. As clusters can be visually identified, it made sense to perform this algorithm. The chosen clustering algorithm was the DBSCAN algorithm for the reasons mentioned previously. The algorithm meant the values for ϵ and MinPts had to be tuned accordingly, but the initial results were promising, as seen in figure 4. The main clusters are represented as coloured circles, whilst crosses represent the points identified as noise. This algorithm has identified all main clusters; however, there was a lot of noise, with 31 points not belonging to any cluster.

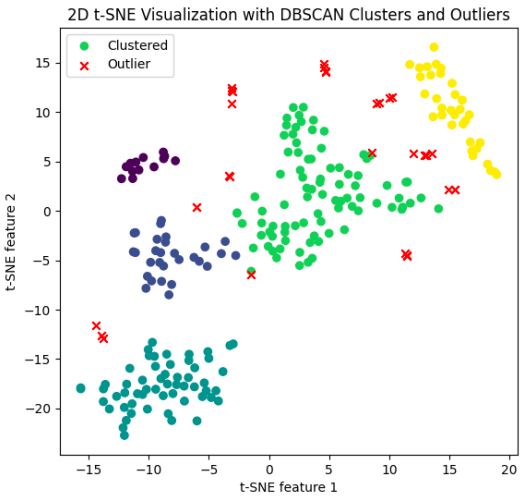


Fig. 4. Clustering into groups of similar artists using DBSCAN after t-SNE transformation

However, this clustering method is impractical as it becomes difficult to interpret how the algorithm clusters the data as the t-SNE transformation is complicated. Another issue with this method is assigning new points to a cluster. This is because the t-SNE transformation is unknown and changes when different points are considered. The t-SNE algorithm is better for identifying whether or not clustering is practical in the first place. t-SNE also incorporates a random component in its initialisation phase. This can lead to different results every time t-SNE is run, even on the same data. A random seed can be set to remove the random element; however, this is not good practice.

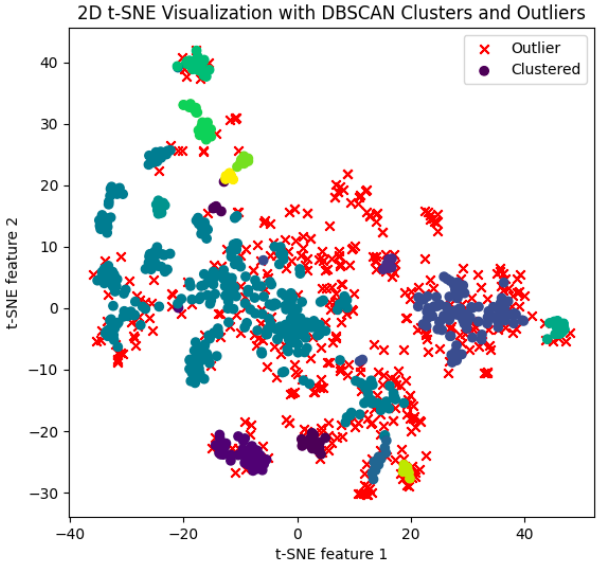


Fig. 5. DBSCAN for top 1000 artists (ϵ=0.65, *MinPts*=6)

DBSCAN:
After exploring clustering techniques on the t-SNE transformed data, the DBSCAN algorithm was applied to the original dataset. As the dataset has high dimensionality, the running time of this algorithm was slow. However, the algorithm was applied to the top 250, 500, 1000, and 2000 artists. The parameters ϵ and *MinPts* were updated accordingly for each dataset to reduce the number of data points identified as noise. The parameters were also adjusted to change the number of identified clusters.

After adjusting the parameters, the cluster assignments were visualised using the t-SNE algorithm described previously. An example clustering can be seen in figure 5. The paramters for this clustering are $\epsilon = 0.65$ and *MinPts*=6. The clusters identified with these parameters closely resemble the clusters that can be visually identified from the t-SNE transformation. However, many of the points are identified as noise. This is an issue as all artists identified as noise are not associated with similar artists.

Unfortunately, after testing many different values for ϵ and *MinPts*, all clustering had noise unless the parameters were adjusted such that there were only two clusters, which

is unlikely to be an appropriate clustering. An artist falling into one of only two categories is illogical.

Initial investigations into the clusters showed promising results. This was performed by manually analysising the clusters to see that artists of a genre appeared in the same cluster. In figure 5 we can see the results of the DBSCAN algorithm on the top 1000 artists with ϵ=0.65, *MinPts*=6. The results contain a lot of noise, however upon analysing a cluster we obtain the artists:

- Wolfgang Amadeus Mozart
- Johann Sebastian Bach
- Ludwig van Beethoven
- Georg Friedrich Händel
- Antonio Vivaldi
- Ludovico Einaudi
- Andrea Bocelli
- IL Divo
- Gabriel Fauré
- Johannes Brahms

This is a very successful cluster as the artists listed are renowned classical composers and, hence, are deemed similar to one another.

k-Means:
K-means clustering is an unsupervised machine learning algorithm that groups data into k-distinct clusters based on similar characteristics. The algorithm works as follows:

- **Initialisation:** Choose k initial cluster centres (centroids) randomly or based on some heuristic.
- **Assignment:** Assign each data point to the nearest cluster centre based on distance (usually Euclidean distance).
- **Update:** Recalculate the centroids as the mean of all points assigned to each cluster.
- **Iteration:** Repeat the assignment and update steps until the centroids no longer change significantly, indicating that the clusters have stabilised.

The k-means algorithm was performed on the original data using the Euclidean dis- tance metric for the top 250, 500, 1000, and 2000 artists. An issue with k-means clustering is that the number of clusters is not always apparent, like in this case. Therefore multiple values of K were used to find a suitable clustering. Another drawback of the k-means algorithm is that the final clustering depends largely on the initial choice of centers. Therefore it was also important to consider different random seeds when performing the algorithm. An example clustering showing the results from a k-means algorithm with K=11 on the top 1000 artists after transformation using the t-SNE algorithm can be seen in figure 6.

Results

A successful clustering is one where the size of the clusters is small, and the accuracy within clusters is high. It is easier to achieve a higher accuracy with larger clusters as it is more

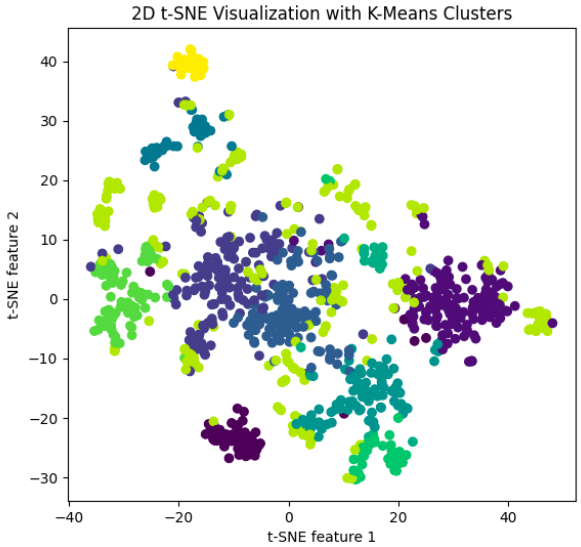


Fig. 6. k-means clustering on the top 1000 artist (K=11)

likely that an artist is within this cluster. It is more practical for a cluster to be small, as recommending a large number of similar artists does not solve the problem. The two metrics used to evaluate the clusterings in this project are the weighted average cosine similarity and the artist similarity based on the last.fm website. The cosine similarity validates the clustering against the original dataset whereas using the similar artists provided by last.fm provides validation independent of the original dataset, both providing valuable insights.

Cosine Similarity
The Cosine similarity is an appropriate metric to use to score the success of a clustering. The exact metric used to score a clustering was the weighted average of the average cosine similarity within a cluster. Mathematically, the average cosine similarity was calculated for each cluster. The final score was the sum of the averages multiplied by their respective size divided by the total number of artists. An example can be seen below for the DBSCAN algorithm applied to the original data as seen in figure **??**.

Cluster Colour	Average Cosine Similarity	Cluster Size
Cyan	0.74	13
Navy	0.54	29
Green	0.61	57
Purple	0.45	87
Yellow	0.46	33
Outliers	0.14	31

Table 1. Cosine similarity for the clustering in figure 3

From the average cosine similarity found from each of the clusters from the clustering we can calculate the weighted average for the entire clustering. The outliers are not included in this calculation.

$$\frac{0.74*13+0.54*29+0.61*57+0.45*87+0.46*33}{13+29+57+87+33}=0.54$$

Results for the cosine similarity for the different algorithms on a different number of artists can be seen in table 2. In the case of the DBSCAN algorithm, the score stated is the best for given parameters. For k-means, the accuracy is an average for the best k over multiple random seeds. Unfortunately, as the cosine similarity is a computationally complex operation, scoring clusters for the top 2000 artists was not feasible.

No. Artists	t-SNE	DBSCAN	k-Means
250	0.541	0.643	0.453
500	0.532	0.612	0.462
1000	0.490	0.603	0.402

Table 2. Cosine similarities achieved by different clustering techniques

The results show that the clusters found using DBSCAN on the original data yield stronger artist similarity than when the algorithm is performed on the t-SNE transformed data. The results suggest that the clusters identified by the DBSCAN algorithm yield a more substantial artist similarity than the ones found by the k-means algorithm. However, as the outliers are not included in the scoring for DBSCAN, it isn't easy to compare the two techniques.

This metric gives us a score on whether or not the similarities in the dataset are are represented within clusters however it is important to see if the results are representative beyond the dataset. Therefore a metric that is independent of the dataset should also be used.

Last.fm Similarity

A feature of the Last.fm website is the ability to view similar artists. Last.fm finds similar artists using a process called collaborative filtering, which is based on the listening habits of its user community. The algorithm analyses the songs and artists users listen to and identifies patterns and relationships among them. Using the Last.fm API and Python, a new dataset was formed containing all artists from the original dataset as well as three of the most similar artists as given by Last.fm. This new dataset provides a validation to assess how good the clusterings are.

The validation works as follow:

- Check if the artist is contained within the dataset.
- If the similar artist is in the dataset then score 1 if it is in the same cluster as the original artist and 0 otherwise.
- Repeat for all artist and their respective similar artists and calculate the empirical average.

This metric can then be used to compare different clusterings to see if the clusters contain similar artists.

After this metric had been established, different values of k for k-means clusterings could be evaluated to see which value was the most appropriate. Using a Python script, the k-means clustering algorithm was performed on the same data for values of k from 1 to 25. The accuracies were then calculated as described previously and plotted on a graph. An example can be seen in figure 7.

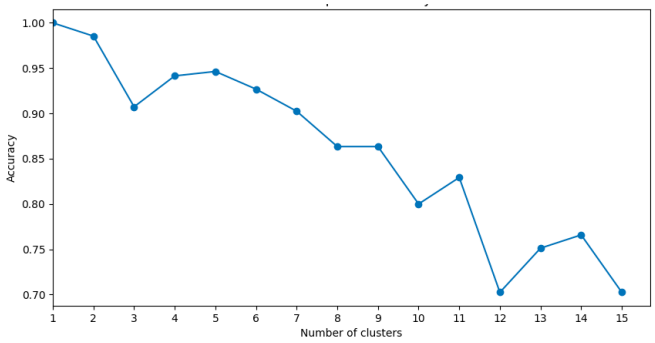
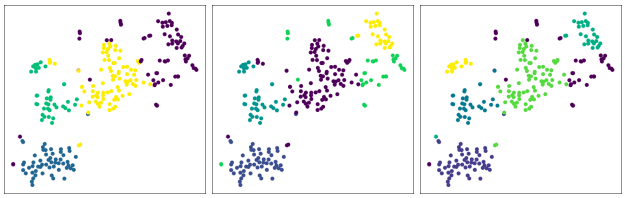


Fig. 7. K-Means accuracy for different numbers of clusters

For $k = 1$ the accuracy is 100%. This is because all artists belong to the same cluster; therefore, so do all of their similar artists. We can see that the highest non-trivial accuracies are achieved for $K = 4, 5, 6$. The clusterings for these values of K after a t-SNE transformation can be seen in figure 8. The algorithm has identified the clusters that can be visually seen from the t-SNE transformation. As each clustering has a similar accuracy of 93% we can say that K=6 is the best clustering as it breaks the artists into smaller subsets of similar artists and still maintains the same accuracy as $K = 4, 5$.



(a) K=4 **(b)** k=5 **(c)** k=6

Fig. 8. k-means clustering on the same data with different values of k

After performing this evaluation for a different number of top artists the best values of k were found along with their respective accuracies.

- Top 250: K=6, accuracy=0.92
- Top 500: K=13, accuracy=0.85 (See figure 9)
- Top 1000: K=11, accuracy=0.85
- Top 2000: K=8, accuracy=0.79

This metric was also used to evaluate the clusters obtained using the DBSCAN algorithm. The accuracies 0.90, 0.842, 0.853, 0.825 were achieved for the top 250, 500, 1000, 2000 artists respectively. The results suggest the majority of the time the clusters group together similar artists.

The Last.fm similarity metric does not consider artists that are classified as noise. Therefore, the k-means algorithm may be a more suitable algorithm for this task. This is because both algorithms score a similar accuracy; however, the k-means cluster all points, whereas the DBSCAN algorithm does not necessarily.

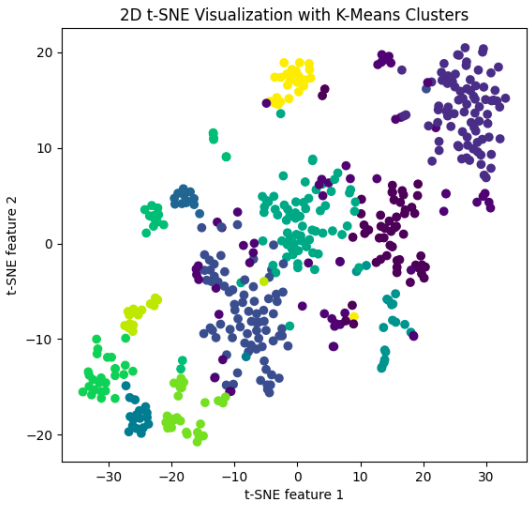


Fig. 9. k-means clustering for the top 500 artists (K=13)

Conclusions

It can be concluded that, as hypothesised, social tagging music artists can identify similar artists using methods such as cosine similarity and a variation of clustering techniques. Once identified, similar artists can be recommended. Although a model that fits all data was not achieved, the techniques and approaches outlined in this paper have defined methods to tune parameters and find clusters successfully.

Limitations

One of the key limitations of social tagging is the coverage. It is common that only the most popular items are described frequently by users, creating a compact description. On the other hand, long-tail items usually do not have enough tags to characterise them; This complicates the recommendation process (12).

As the number of artists considered for the clustering algorithm increases, the number of items in the largest cluster increases disproportionally. This is especially the case for the DBSCAN algorithm. This could be the case because the data becomes more dense; hence, more links are created between items, and larger clusters are found.

Extensions

As mentioned in the limitations section as the number of artists increase the largest cluster increases disproportionally. This could be because it is capturing a parent genre, whereas we wish to find subgenres. This could be investigated by using clustering techniques on each of the clusters, this could involve nested clustering (13).

Another area of interest would be to test the techniques and approaches described in the paper on a more extensive dataset by pulling more data using the Last.fm API. This would allow us to see if the patterns hold when considering a much larger number of artists.

Bibliography

1. Music streaming services worldwide - statistics facts. https://www.statista.com/statistics/653926/music-streaming-service-subscriber-share/. Accessed: 08/01/24.
2. Fabio Duarte. Music streaming services stats (2023). https://explodingtopics.com/blog/music-streaming-stats, 2023. Accessed: [Insert Access Date Here].
3. Made by you. https://found.byspotify.com/made-by-you. Accessed: 08/01/24.
4. Last.fm. https://www.last.fm/, . Accessed: 04/01/2024.
5. Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys 2011, New York, NY, USA, 2011. ACM.
6. Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.
7. George C. Linderman and Stefan Steinerberger. Clustering with t-sne, provably. *CoRR*, abs/1706.02582, 2017.
8. Last.fm api. https://www.last.fm/api, . Accessed: [insert date of access here].
9. Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*, chapter 8, page 500. Addison-Wesley, 2005. ISBN 0-321-32136-7.
10. Artists similar to madonna. https://www.last.fm/music/Madonna/+similar. Accessed: 09/01/24.
11. Adil Abdu Bushra and Gangman Yi. Comparative analysis review of pioneering dbscan and successive density-based clustering algorithms. *IEEE Access*, 9:87918–87935, 2021. doi: 10.1109/ACCESS.2021.3089036.
12. O. Celma. *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.
13. Xutao Li, Yunming Ye, Mark Junjie Li, and Michael K. Ng. On cluster tree for nested and multi-density data clustering. *Pattern Recognition*, 43(9):3130–3143, 2010. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2010.03.020.