Machine learning methods for content-based music recommendation systems

Milita Songailaitė[1], Tomas Krilavičius[1]

[1] Vytautas Magnus University, Faculty of Informatics, Vileikos street 8, LT-44404 Kaunas, Lithuania

Abstract

With the increased popularity of online music streaming, people find themselves spending more and more time choosing the content they like. This poses a problem of a fast and accurate music recommendation method, which would let the user ignore the large quantities of unwanted music and choose precisely what he likes. This work presents methods to compare music based entirely on its audio signal properties. For this, we used three different approaches (Gaussian mixture models, dynamic time warping and autoencoders) to calculate the similarity between the given signals. All three experiments were performed on a database consisting of 2511 most popular songs from 10 different genres. The methods were evaluated by comparing algorithms' results with the music similarity results given by the experts. The Gaussian mixture model was the best-evaluated method, while the worst was the autoencoder method.

Keywords

Machine learning, Gaussian mixture models, dynamic time warping, autoencoder

1. Introduction

The music industry has increased rapidly in online streaming services during the last decade. For example, in 2011, more than 80% of music sales revenues consisted of physical and digital records [1]; however, by the year 2021, the majority of the revenues were only from the music streaming services [2]. Moreover, the amount of music that people can find online is also increasing. For instance, *Spotify*, one of the largest music broadcasters, offers customers a selection of more than 70 million songs, supplementing this selection by 60,000 new songs each day [3]. In addition, another popular music broadcaster *SoundCloud*, uploads nearly 12 hours of music content online each hour [4].

In order to deal with vast amounts of data, companies use music recommendation algorithms. These algorithms can be grouped into two main categories: content-based and context-based recommendations [5]. Context-based recommendation systems use users' data (such as liked songs history or the list of favorite genres) as well as similar customers' choices. This may lead to the system recommending only the songs that other users find similar. In other words, the algorithm will start recommending the most popular songs among similar users and leave less popular songs behind. Content based recommendation systems avoid this problem because the recommendations are based purely on the signal's audio properties and nothing else.

In this work, we focus only on content-based recommendation systems. A features profile was generated to characterize each song, which consisted of various audio signal properties created by several different methods. The generated feature profiles were used as songs vectors in the created content-based recommendation systems.

The rest of the paper is organized as follows. Related work is provided in Section 2. The recommendation systems are discussed in Section 3. The evaluations of the proposed systems are provided in Section 4. The conclusions are given in Section 5.

2. Related work

The analyzed literature covers two main topics: content-based music recommendation systems and deep learning usage for audio signals feature generation.

Most of the papers ([6], [7], [8], [9]) on content-based systems used Mel-frequency cepstral coefficients (MFCCs) as the audio profile for the songs. The profiles were compared using K-means, Gaussian Mixture Model and Kullback-Leibler divergence measures, which distinguished the most similar profiles and recommended them as the most similar songs to the given queries. The system described in [10] simply used Short-time Fourier transform to capture the features of the signals. In order to compare the gathered time series, the authors used the Locality-sensitive hashing method.

IVUS 2022: 27th International Conference on Information Technology, May 12, 2022, Kaunas, Lithuania

milita.songailaite@stud.vdu.lt (M. Songailaitė);
tomas.krilavicius@vdu.lt (T. Krilavičius)

The method worked especially good when detecting song covers. A slightly different approach was introduced in the paper [11]. The authors used feature vectors consisting of 18 distinct features provided by *The Echo Nest* API. The feature profiles were clustered using the K-means algorithm to gather clusters of similar songs.

The second part of the literature review focused specifically on the use of deep learning in music recommendation systems. Some of the most popular approaches used deep learning to generate new sound representation vectors for the signals. Papers [12], [13] describe how the autoencoder architecture can be used to encode musical style, which will be later used to generate music. The authors in paper [14] tested two different architectures (OpenL3 and VGGish) for encoding audio signals. These encodings helped the authors distinguish the emotions of the songs and classify them. Finally, in [15] authors compared variational autoencoders with LSTM and Recurrent networks. The research showed that almost all deep learning networks outperformed baseline principal components methods for feature vector generation.

3. Recommendation system overview

After analyzing the literature and testing the most popular algorithms, we selected three main directions in which our research was carried out:

1. a Gaussian mixture model for feature space modelling;
2. vector similarity using dynamic time warping distance;
3. autoencoder network for features encoding.

The workflow of the experiments is depicted in Figure 1. First, we produced the Mel-Frequency scale cepstral coefficients, the baseline of the songs features profile. For each song, we generated the first 8 MFCCs at a sample rate of 22 050 Hz. After that, in order to create songs' audio profiles, the existing MFCCs were transformed using the three mentioned methods. In the end, created recommendation systems were evaluated using two methods:

1. finding the average number of the same genre, same artist and same album songs among the generated list of recommended songs;
2. comparing music sorting results of the algorithms with the human ability to sort by similarity.

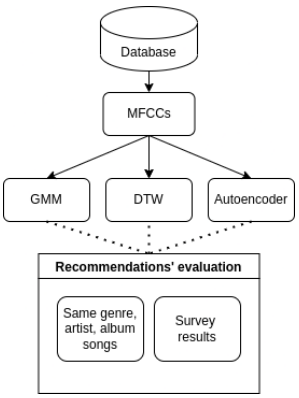

Figure 1: Workflow of the proposed system

Table 1
The distribution of songs number in each genre

Genre	Number of songs
Pop	275
Rock	246
Metal	236
EDM	257
Kpop	265
Country	236
Classical	203
Jazz	252
Blues	253
Rap	288

3.1. Dataset

A dataset consisting of 2511 popular songs was used to evaluate the developed music recommendation systems. The database consisted of songs created by 135 artists from 10 different genres. The distribution of songs across genres is presented in Table 1. All songs were converted and stored using uncompressed .wav format. This form of signal was the input for all models. In addition to the audio signals, we also stored unique id, song's artist, album, and genre for each song. However, the metadata was only used to test recommendation systems. Therefore, the signals' audio profiles consisted purely of the songs' audio data.

3.2. Methods

3.2.1. Gaussian mixture model

The first used music modelling algorithm was Gaussian Mixture Model (GMM). This method was initially mentioned in [8] paper. First, all 30 s audio signals are clustered into 5 clusters using a Gaussian mixture clas-

sifier. The original paper uses only 3 clusters, but testing has shown that a slightly larger number of clusters work better in the system. The fabricated Gaussian mixture models defined the characteristics of each audio signal in 5 Gaussian distributions. Next, these distributions were compared using Kullback-Leibler (KL) divergence. This divergence measure shows how one distribution is different from the other one. It is possible to calculate it using the (1) formula, where $p(x)$ and $q(x)$ are the probabilistic distributions [16]. Since the Gaussian mixture model describes each audio profile as a Gaussian distribution, we can compare several Gaussian models based on this measure.

$$D_{KL}(P,Q) = \int_{\mathbb{R}^d} \ln\left(\frac{p(x)}{q(x)}\right) p(x)dx \tag{1}$$

The proposed model describes the songs by 5 clusters, each characterized by 8 Gaussian distributions. Each distribution in the first cluster of the query is compared to each distribution in the first cluster of the song from the database. These calculations are repeated for each cluster, and the resulting eight estimates are averaged. Finally, we obtain a vector of 5 values, where each value reflects the similarity between the two clusters of audio signals being compared. These values are also averaged. Therefore, we get one value, which reflects the similarity of the Gaussian Mixture Models.

3.2.2. Dynamic time warping method

Dynamic Time Warping (DTW) was the second method tested to model songs' audio profiles. Unlike the Gaussian mixture method, the main point of this method was to directly compare MFCCs without further modification. Thus, the vectors of the generated MFCC matrices were compared by calculating the Dynamic Time Warping distance measure (2), where $\delta(w_k)$ is usually the Euclidean distance between the two time series [17]. This distance measure was chosen since it lets us find similarities between two different duration audio signals and gives us more freedom when capturing musical motifs across the audio signal.

$$DTW(S,T) = \min\left[\sum_{k=1}^{p} \delta(w_k)\right] \tag{2}$$

Since audio signals are multidimensional time series, the Dynamic Time Warping distance was modified to work with multidimensional data. There were two selected modifications: the dependent DTW (DTWd) and independent DTW (DTWi), both of them described in [18]. The result is an array of songs sorted in ascending order according to the calculated distances. Finally, the N most similar songs are selected and considered a recommendation for a given query.

3.2.3. Autoencoder model

The final tested model was the Autoencoder (AE) network. Autoencoder is a deep learning method, which learns a representation by attempting to encode it and then validates that representation by regenerating the original input from it [19]. Similar to the Gaussian Mixture Model, the main point of this algorithm was to compress the audio signals so that the created features would contain only the most essential information. In this case, both the songs' in the database and the queries' MFCCs are encoded with the created Autoencoder model. This way, we get new attribute vectors for each song and the query, and thus, we can now calculate the distance between newly generated features. For this step, we chose Euclidean distance. Finally, these distances are sorted in ascending order, and the top N closest to the query songs are selected to be a recommendation for the query.

The Autoencoder has encoding and decoding parts. The encoding part consists of four fully connected layers. These layers compress the flattened MFCC matrices into an array of 64 values. After that, the decoder part, which also has four fully connected layers, decodes the array back into its initial dimensions. We used the ReLU activation function in the hidden layers and the linear activation function in the output layer. Adam optimized was chosen to optimize the network. The network was trained with 100 epochs using a batch size of 100 songs.

4. Evaluation

The evaluation of music recommendations is a rather subjective matter. Many people perceive music similarity differently; therefore, the constructed tests must stay unbiased. In this work, we used two ways to test the recommendation systems:

1. Assuming that the songs from the same artist, album, or genre are similar, we count how many of those songs show up in the recommendation list.
2. Comparing music sorting results of the algorithms with the human ability to sort by similarity.

Table 2
Metainformation of the selected query

Name of the song	Kamikaze
Artist	Eminem
Album	Kamikaze
Genre	Rap
Start time	1:42
End time	2:12
Duration, s	30

4.1. Number of same artists, albums and genre songs in the recommendations list

This test assumes that the songs from the same artist, album, and genre are similar. Thus, we can say that similarity detection is good if the algorithm detects as many same artists, albums, and genre songs as possible. Such a test was performed for each created system. First, we selected a query to test the systems (the query is described in Table 2). Then we analyzed the list of top 50 songs recommended by the algorithms.

The results of the first test are given in Figure 2. It can be observed that the best algorithm in all three categories was the Gaussian Mixture model. No other algorithm had songs from the same album in their recommendation lists. However, this was expected since there are only 11 songs in the selected album, which are not necessarily similar to the query. The other remaining algorithms performed almost equally poorly, although the Autoencoder model found the smallest number of similar songs.

4.2. Music sorting results in comparison with the human ability to sort

In the second test, created algorithms were compared to the human ability to rank songs by their similarity. The experiment involved 23 volunteers who were given ten different query songs. The volunteers had to sort three given music clips from the most similar (first place) to the least (third place) for each query. As a result, relative locations of the lined-up songs were provided for each query. Generally, these calculations can be written with a formula (3), where r1, r2, and r3 are the percentages of volunteers who voted for the first, second, and third places, respectively.

$$\text{rank} = 1 \cdot r_1 + 2 \cdot r_2 + 3 \cdot r_3 \tag{3}$$

The algorithms were given the same task: they had to sort the given music clips by their similarity to the query. The positions of the songs sorted by the algorithms were denoted by integers (1, 2, and 3). In order to estimate the accuracy of the methods, the Mean Square Error (4) was used, where Y is the results of the surveys and $\hat{Y}$ is the ranking of the algorithms.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{4}$$

Figure 3 the accuracy of the methods for each query individually. Table 3 shows the total MSE estimates for all of the methods.

Similarly, as in the first test, the accuracy of the Gaussian mixture method was the highest. However, the accuracies of the two Dynamic Time Warping methods acted slightly differently in this test. If the independent DTW method had a higher score in the first test, the dependent DTW method generated better recommendations in the second test.

Pearson Correlation Coefficient was used to com-

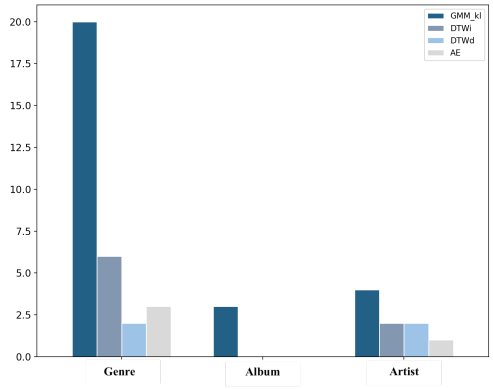

Figure 2: The number of songs by the same artist, album, and genre in the recommendations generated by each algorithm. The X-axis depicts the three performed tests and the Y-axis show the number of similar songs found. `GMM_kl` - Gaussian Mixture Model method; `DTWi` and `DTWd` - dependent and independent Dynamic Time Warping method; `AE` - Autoenoder method

Table 3
Overall estimation of the Mean Square Error for each method

Method	MSE
GMM_{KL}	**0.7345**
DTW_i	0.8452
DTW_d	0.7632
AE	0.9605

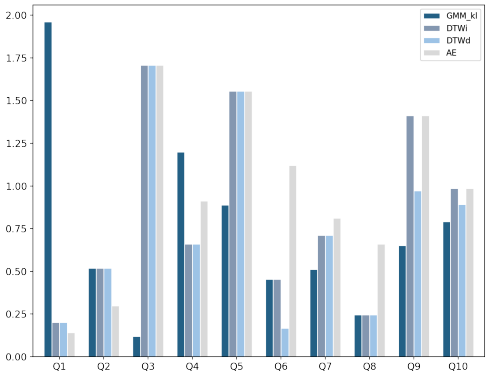

Figure 3: The Mean Square Error function of algorithms sorting results compared to the volunteers sorting. The X-axis depicts the ten performed tests and the Y-axis show the MSE estimates. `GMM_kl` - Gaussian Mixture Model method; `DTWi` and `DTWd` - dependent and independent Dynamic Time Warping methods; `AE` - Autoenoder method

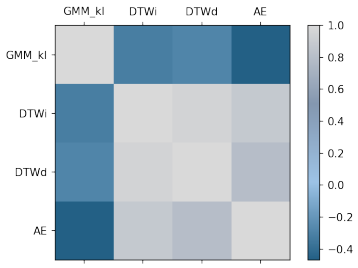

Figure 4: Methods correlation matrix heatmap. `GMM_kl` - Gaussian Mixture Model method; `DTWi` and `DTWd` - dependent and independent Dynamic Time Warping methods; `AE` - Autoenoder method

pare how similar are the created models. The correlation was calculated between the error functions for each tested query using the formula (5), where cov(X,Y) is the covariance matrix of the two features, and σ_X, σ_Y are the standard deviations of these features. The results are presented in a correlation matrix (see Table 4 and Figure 4).

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{5}$$

The methods correlation matrix showed that the DTWi, DTWd, and AE methods had a weak negative correlation with the GMM_KL method. This was due to several queries tests (e. g., Q1 or Q3 in the figure 3), as the results of the rankings of the algorithms in these

Table 4
Methods correlation matrix values

	GMM_{KL}	DTW_i	DTW_d	AE
GMM_{KL}	1.000	-0.319	-0.281	-0.469
DTW_i	-0.319	1.000	0.959	0.881
DTW_d	-0.281	0.959	1.000	0.795
AE	-0.469	0.881	0.795	1.000

tests differed significantly. Another finding is that all poorly rated algorithms (DTWi, DTWd, and AE) work reasonably similarly. The structure of the survey itself determined this phenomenon. Because for each query, the user and algorithms had only three songs available for sorting, there were few possible lineup options. Therefore, even if the algorithms misidentified the similarity, there was still a high chance that they would make similar recommendations to each other.

Finally, after the second test, it can be stated that the most accurate method for similar music recommendations was the Gaussian Mixture Model. The remaining algorithms, same as in the first test, gave inferior recommendations.

5. Conclusions and recommendations

5.1. Results

In this work, an analysis of existing content-based music recommendation systems was performed. Different systems for recommending similar songs have been developed using the best-found algorithms (Gaussian Mixtures Models, Dynamic Time Warping distance, and Autoencoder networks). Two types of accuracy evaluations were performed for the developed systems: finding the number of same artists, same album, and same genre songs; and algorithms music sorting comparison with the human ranked results.

5.2. Conclusions

The following conclusions were formulated based on the obtained results:

1. The analysis of the algorithms' recommendations showed that the best performing algorithm was the Gaussian mixture model. The GMM was able to identify similar songs in terms of an audio signal, and the recommendations generated

by the algorithm were in line with the majority of survey respondents.
2. Dynamic Time Warping methods performed slightly worse than the Gaussian Mixture Method. Both methods (dependent and independent) were able to find at least several songs from the same artist or genre. The accuracy of the independent method in the first test was higher due to the higher number of songs in the same genre in the recommendations list. In the second test, the dependent method was more accurate. The Dynamic Time Warping method provided almost as accurate recommendations as the Gaussian mixture method in this test.
3. The worst-performing method was modeling the musical features using Autoencoder networks. This method had the lowest accuracy of all the other algorithms tested in both tests. This result suggests that the Autoencoder network is too simple to model complex musical features.

5.3. Recommendations

The following are some recommendations for future plans that will allow us to improve the developed algorithms in the future:

1. Increase the collected database by the number of music signals in it and by the number of variables describing the characteristics of the collected signals.
2. Improve algorithms' testing methods. So far, the survey is relatively small (10 queries, three songs for each). However, it is planned to increase the scope of this test and increase the number of people participating in the study.
3. Since deep neural network methods are often used in the literature to solve the problem of music modelling, it is planned to extend the usage of these methods for music recommendation in the future. The tested Autoencoder method could be transformed into a more complex neural network structure. Since the modelling of music with Gaussian distributions has proven to be the most successful in this paper, a simple Autoencoder could be changed to a variational Autoencoder. In the variational Autoencoder, each music feature is described by a Gaussian distribution instead of encoding the data into simple vectors. Another potentially suitable variant of the neural network is the recurrent neural network. A recursive neural network would allow the exploration of excerpts from songs of different lengths (both longer and shorter), thus creating a universal model for identifying similar music.

References

[1] *U.S. Sales Database*. en-US. Available at https://www.riaa.com/u-s-sales-database/. (Visited on 12/10/2021).

[2] *MID-YEAR 2021 RIAA REVENUE STATISTICS*. en-US. Available at https://www.riaa.com/wp-content/uploads/2021/09/Mid-Year-2021-RIAA-Music-Revenue-Report.pdf. URL: https://www.riaa.com/wp-content/uploads/2021/09/Mid-Year-2021-RIAA-Music-Revenue-Report.pdf (visited on 12/10/2021).

[3] *Spotify Revenue and Usage Statistics (2021)*. en. https://www.businessofapps.com/data/spotify-statistics/. July 2021. URL: https://www.businessofapps.com/data/spotify-statistics/ (visited on 08/16/2021).

[4] *SoundCloud Revenue And Usage Statistics (2021)*. en. https://www.businessofapps.com/data/soundcloud-statistics/. Sept. 2020. URL: https://www.businessofapps.com/data/soundcloud-statistics/ (visited on 12/11/2021).

[5] Umair Javed et al. "A Review of Content-Based and Context-Based Recommendation Systems". In: *International Journal of Emerging Technologies in Learning (iJET)* 16 (Feb. 12, 2021). DOI: 10.3991/ijet.v16i03.18851.

[6] Terence Magno and Carl Sable. "A comparison of signal based music recommendation to genre labels, collaborative filtering, musicological analysis, human recommendation and random baseline". en. In: *ISMIR 2008 – Session 2a – Music Recommendation and Organization* (2008), p. 6.

[7] Beth Logan and Ariel Salomon. "A Content-Based Music Similarity Function". In: *Cambridge Research Laboratory Technical Report Series* (Dec. 2002).

[8] Jean-Julien Aucouturier and Francois Pachet. "Finding songs that sound the same". In: *Proc. of IEEE Benelux Workshop on Model based Processing and Coding of Audio*. 2002, pp. 1–8.

[9] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. "Deep content-based music recommendation". eng. In: *Advances in Neural Information Processing Systems 26 (2013)*. Vol. 26. Neural Information Processing Systems Foundation (NIPS), 2013. ISBN: 978-1-63266-024-4. URL: http://hdl.handle.net/1854/LU-4324554 (visited on 07/14/2021).

[10] Cheng Yang. "MACS: music audio characteristic sequence indexing for similarity retrieval". In: *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No.01TH8575)*. New Platz, NY, USA: IEEE, 2001, pp. 123–126. ISBN: 978-0-7803-7126-2. DOI: 10.1109/ASPAA.2001.969558. URL: http://ieeexplore.ieee.org/document/969558/ (visited on 07/15/2021).

[11] Malcolm Slaney, Kilian Weinberger, and William White. "LEARNING A METRIC FOR MUSIC SIMILARITY". en. In: *ISMIR 2008 – Session 3a – Content-Based Retrieval, Categorization and Similarity 1* (2008), p. 6.

[12] Kristy Choi et al. "Encoding Musical Style with Transformer Autoencoders". In: *arXiv:1912.05537 [cs, eess, stat]* (June 2020). arXiv: 1912.05537. URL: http://arxiv.org/abs/1912.05537 (visited on 08/23/2021).

[13] Cheng-Zhi Anna Huang et al. "MUSIC TRANSFORMER: GENERATING MUSIC WITH LONG-TERM STRUCTURE". en. In: *International Conference on Learning Representations (ICLR) 2019* (2019), p. 15.

[14] Eunjeong Koh and Shlomo Dubnov. "Comparison and Analysis of Deep Audio Embeddings for Music Emotion Recognition". In: *arXiv:2104.06517 [cs, eess]* (Apr. 2021). arXiv: 2104.06517. URL: http://arxiv.org/abs/2104.06517 (visited on 08/23/2021).

[15] Fanny Roche et al. "Autoencoders for music sound modeling: a comparison of linear, shallow, deep, recurrent and variational models". en. In: *IEEE SMC 2019* (2019), p. 8.

[16] Yufeng Zhang et al. "On the Properties of Kullback-Leibler Divergence Between Gaussians". In: *arXiv:2102.05485 [cs, math]* (May 27, 2021). arXiv: 2102.05485. URL: http://arxiv.org/abs/2102.05485 (visited on 03/31/2022).

[17] Donald J Berndt and James Clifford. "Using Dynamic Time Warping to Find Patterns in Time Series". en. In: *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining* (1994), p. 12.

[18] Mohammad Shokoohi-Yekta et al. "Generalizing DTW to the multi-dimensional case requires an adaptive approach". In: *Data mining and knowledge discovery* 31.1 (Jan. 2017), pp. 1–31. ISSN: 1384-5810. DOI: 10.1007/s10618-016-0455-0. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5668684/ (visited on 08/21/2021).

[19] Dor Bank, Noam Koenigstein, and Raja Giryes. "Autoencoders". In: *arXiv:2003.05991 [cs, stat]* (Apr. 2021). arXiv: 2003.05991. URL: http://arxiv.org/abs/2003.05991 (visited on 08/21/2021).