Self-organizing maps applied in a content-based music recommendation system

Ariadna Roman
Faculty of Mathematics and Computer Science
Babes Bolyai University
Cluj-Napoca, Romania
roman.ariadna@gmail.com

***Abstract*—In the context of a continuously expanding music industry, users have become increasingly dependant on curated recommendations. Most of the recommendation systems operate user data, but such approaches rely on the existence and the consistency of user feedback. This paper presents a system that merges content-based recommendations with an implementation of a self organizing map algorithm, aiming to eliminate the problems caused by relying on user data. The objective is testing the performance of a system that only uses audio features. The experiments showed that this method strongly depends on the characteristics of the tracks that are considered in the process of recommendation, therefore these results serve as a premise for future work to explore features extracted from the lyrics or the title of the songs, striving to bridge the semantic gap.**

***Index Terms*—self-organizing maps, content-based recommendations, music**

I. Introduction

As our methods of storing music developed from tapes and CDs to websites and applications, functionalities of music players had to keep up. Such functionalities are creating playlists, creating profiles for users that express their preferences through likes and dislikes, therefore creating also a consistent feedback system, sharing music over social media and finding the perfect music recommendations for each user.

Implementing a system for music recommendations raised multiple problems as there is no mathematical formula that would determine the relation between a user and his musical preferences. Multiple solutions were suggested, such as labeling songs ([6], [12], [9]), collaborative filtering ([1], [7], [3]) and extracting information from audio content ([15], [7], [10], [4]). Each method has its advantages and disadvantages, so many solutions proposed hybrid systems, combining different approaches.

1) Social tagging: This method relies on tags made by knowledgeable users to summarize the key semantic attributes of songs. The tags may refer to the genre, the geographic region of the artist, the mood the track inspires, the hidden meaning of the lyrics, the period of time in which the song was released and to many other characteristics that would be almost impossible to extract by any kind of algorithms. Therefore, the advantage is that social tagging offers terms of comparison between songs that are much more relevant for recommendations than audio analysis [6]. The disadvantages are presented in [6] by Lamere. There are certain situations in which the method of labeling or social tagging fails. The cold start problem: a song cannot be recommended if it has not been tagged. The noise problem: it often occurs that songs are labeled with a lot of tags that all have the same meaning. The hacking problem: people might maliciously tag songs, using words or phrases that have nothing to do with the track. There are some solutions for processing the tags in order to solve the noise and the hacking problems, but the cold start problem requires a complementary method that is able to make recommendations when the system fails to provide any tags (e.g. content-based recommendations).

2) Collaborative filtering: This approach relies on finding users with highly similar behaviour patterns. Therefore, it encounters the cold start problem as well. This method depends severely on user data and so it fails to make recommendation for a song that has just entered the system or for songs with niche audiences. Another issue of collaborative filtering, raised by Li et al. in [7] is the problem of non-transitive associations: two songs that have never been listened by the same user have no connection in the recommendation system. Having a comparable context with social tagging, the method can also be merged with content-based recommendations for better performance.

3) Content-based recommendations: This solution supports the idea of using various analysis of the item, rather than the usage data. As the item of interest is a track, characteristics can be extracted mainly from the audio content, the title and from the lyrics, if any. Although extracting information from audio content solves some of the problems listed in the previous subsections, as Aaron Van den Oord, Sander Dieleman and Benjamin Schrauwen state in [15], there is a semantic gap between the characteristics that can be extracted from audio signals and what makes a song a good candidate for a recommendation. Users do not prefer songs that only sound similarly to what they listen to, they might like a song for the theme or for the geographical location of the artist. The features that contain the meaning of a songs are harder to extract and harder to adapt to a mathematical formula.

The objective of this article is testing the performance of a system that only uses audio features. Choosing the audio features used as input for the system has a heavy influence over the result. These characteristics may refer to different aspects of the audio content - pitch, tonality, rhythm or to combinations of those traits.

A secondary key aspect of the proposed approach is the self-organizing map algorithm which concludes in a space where recommendations translate simply into neighbours. The output of the algorithm presents a two-dimensional map in which the distance between items is inversely proportional with the audio similarity. In order to present the results, a comparison is made between the output of a hexagonal and a rectangular map.

In comparison with related work, the system described in this paper is a content-based approach, because the input data relies completely on the audio features of the tracks, but the algorithm chosen to compute it is a different one. Most solutions use Matrix Factorization and Convolutional Neural Networks, whereas, in this paper, the Self-Organizing Map algorithm is used. However, because usage data of listeners is not taken into account, the performance of the system is expected to be quite low. The advantage is that, this way, the cold start problem or any other problems caused by user data is eliminated. As a result, it would be best if this approach was used in combination with a social tagging solution or a collaborative filtering one, so that the two solutions solve each others' shortcomings.

Self-organizing maps were used in different solutions regarding music, but the objective of the experiments and the input were quite different. [16] presents a system that strikes to organize textual documents - reviews about artists - in order to find similar artists and improve the quality of music recommendations. In [2] a quasi-supervised self-organizing map which receives features extracted from processing 20 seconds segments of music tracks is used for finding recommendations. In [10] self-organizing maps are used for organization and visualization of music libraries.

The results of the conducted experiments serve as a premise for future work to explore features extracted from the lyrics or the title of the songs, striving to bridge the semantic gap described before. Additionally, the method can be used for systems that rely mainly on usage data in order to solve the cold start problem.

Because the problem of finding music recommendations is not deterministic, the purpose is to create a pleasant experience for the user. A less abstract way to evaluate the solution is to see how well the system manages to categorize the songs into genres. As the output of the algorithm comes in the form of clusters of songs, the homogeneity of the clusters is calculated. The final results present, for each genre, how many clusters, which computed it as predominant genre, have homogeneity over 0.5 and how many have homogeneity 1.

The remainder of the article is arranged as follows: Section II presents the complete statement of the problem including input and output specifications, in Section III a discussion is made on different approaches that have been proposed in other papers, Section IV contains a presentation of the proposed solution of the article, Section V describes the experiments conducted and their results, Section VI presents the conclusion of the article and future work.

II. Problem definition

The problem of finding music recommendations belongs to Machine Learning area, more exactly, the solution presented in this article fits in unsupervised learning. The solving method is one of self-organization in clusters of elements that are closely related to each other.

Regarding the input and the output, the problem takes a list of songs characterized by multiple features and expects sets of songs that are similar and that can be recommended for one another. As a result, the method should build connections between the musical entities and use those connections to make recommendations.

The difficulty of the finding recommendations stays in realizing the connections considering a high-dimensional dataset, more precisely computing the distances between the tracks in a N (where N is the number of features) dimensional space and normalizing the results in order to obtain the sets of songs that are expected.

As far as the functional requirements are concerned, the system should allow the operation of finding a list of similar songs for a certain track.

Non-functional requirements should include a short time for executing this operation (a few seconds), easily manage the access to a big database (tens of thousands of songs) and the ability to execute the operations independently of the input data.

III. Related work

As the problem of music recommendation was approached in many ways, we have chosen two representative studies, made for two of the top companies of music industry - Spotify and YouTube. Aaron Van den Oord, Sander Dieleman and Benjamin Schrauwen raised in [15] the issue of making recommendations in absence of consistent user data, whereas Paul Covington, Jay Adams and Emre Sargin raised in [1] the issue of having to work with an immense dataset that is continually, rapidly updating. The input that was used in [15] was The Million Song Dataset, while for [1] the input used was formed by videos from YouTube. Although both of the studies approached a solution based on collaborative filtering and Deep Neural Networks, the architecture of the systems was completely different. The results were considered satisfying for both of the studies, managing to overcome the problems issued in the beginning of the articles.

A. *Deep content-based music recommendation*

Aaron Van den Oord, Sander Dieleman and Benjamin Schrauwen debate in [15] collaborative filtering and extracting information from audio content.

Collaborative filtering method uses users' profiles as well as their preferences. The main idea behind the method is that if two (or more) songs are included in a user's playlist and the same songs are included in many users' playlist along with a third song, that song is going to be recommended to the first person. In most cases, this method is going to do a lot better than using audio information extracted from the content, but it presents "the cold start problem". This issue appears when a song has not been played. If it has not been played, it cannot be recommended. Furthermore, a song that has a niche audience is not likely to be recommended as there is not enough data. The second method does not present the same issue [15].

The method based on audio content aims to find useful information, that is not usually known about a song. For example, the name of the artist, the name of the album and the year of its release will lead to predictable recommendations. As a result, the information that would make good recommendations is the mood the song inspires, the theme of the lyrics, but there are some traits much harder to extract like the genre or the instruments that are used. For all that, there are some characteristics that are impossible to find, such as the popularity of the artist or his geographical location [15].

The approach of Aaron Van den Oord, Sander Dieleman and Benjamin Schrauwen in [15] is to extract latent factors - that influence the users' preferences - from the audio content, evaluating the results in a recommendation system. The problem was considered to be a regression problem and was solved in two ways: a traditional approach - the aggregation of local characteristics extracted from audio signals in a Bag-of-Words representation ([18]) and an approach that uses Deep Convolution Neural Networks ([8]) on an intermediate time-frequency representation extracted from the audio signals. The training set was represented by the results of a Weighted Matrix Factorization algorithm used on a dataset containing the number of times a user listened to a song.

The conclusion of their experiments was that although many of the characteristics that influence users' preferences could not be extracted from audio content, the results of the recommendations were satisfying. Additional information about their results are presented in the tables below.

Model	mAP	AUC
MLR	0.01801	0.60608
linear regression	0.02389	0.63518
MLP	0.02536	0.64611
CNN with MSE	0.05016	0.70987
CNN with WPE	0.04323	0.70101

TABLE I
TABLE EXTRACTED FROM [15]

Table I presents the results of the first experiment, conducted on a subset of the 9330 most popular songs and data from 20000 users. Each line corresponds to a model: MLR - linear regression trained on a bag-of-words representation, used for the metric learning to rank algorithm, MLP - a multi-layer perceptron trained on the same bag-of-words representation, MSE - a convolutional neural network (CNN) trained on log-scaled mel-spectograms to minimize the mean squared error of the predictions, WPE - same CNN trained to minimize the weighted prediction error. The numbers represent the mean average presicion (mAP) and the area under the ROC curve (AUC) of the predictions [15].

Table II presents the results of the second experiment - a comparison between the results of linear regression on a bag-of-words representation of the audio signals, a CNN trained with the MSE objective on log-scaled mel-spectograms, the latent factor vectors randomized and a case when they are learned from usage data using WMF - weighted matrix factorization. The experiment was conducted on the full dataset that contains 382410 songs and 1 million users [15].

Model	mAP	AUC
random	0.00015	0.49935
linear regression	0.00101	0.64522
CNN with MSE	0.00672	0.77192
upper bound	0.23278	0.96070

TABLE II
TABLE EXTRACTED FROM [15]

B. Music recommendations for a dynamic system

Paul Covington, Jay Adams and Emre Sargin studied in [1] the problem of recommendations (video content) for Google, more exactly for YouTube. The obstacles that they encountered were the huge number of videos, the dynamics of the system that requires a responsive algorithm and the fact that there is not a reliable feedback system from users. The solution they used was Deep Learning systems as well, but they developed them using Tensorflow [1].

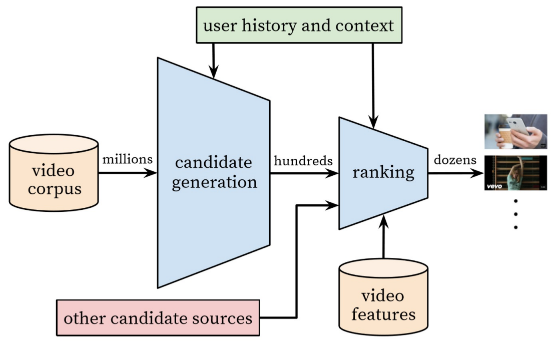

Fig. 1. Image extracted from [1]. The illustration presents the architecture of the system, underlying the method of choosing the candidates for recommendation.

They used two neural networks, one for generating candidates and a second one for ranking. The first one uses collaborative filtering and the second one finds the fittest suggestions, calculating for every candidate a score based on an objective function that uses a big data set of characteristics that describe the user and the video [1]. See Figure 1 for the architecture of the recommendation system.

The conclusion was that the solution surpasses the Matrix Factorization approach that was formerly used by YouTube [1]. See Table III for the results of the experiments.

Table III presents the results of different hidden layer configurations of Rectified Linear Units. In order to calculate "weighted, per-user loss" they used the positive or the negative reactions of the user (clicking or not clicking on the suggestions) on a single page. They considered "mispredicted watch time" the cases in which they had more negative reactions than

Hidden layers	weighted, per-user loss
None	41.6%
256 ReLU	36.9%
512 ReLU	36.7%
1024 ReLU	35.8%
512 ReLU 256 ReLU	35.2%
1024 ReLU 512 ReLU	34.7%
1024 ReLU 512 ReLU 256 ReLU	34.6%

TABLE III
TABLE EXTRACTED FROM [1]

positive ones. As a result, the value is computed by dividing the total amount mispredicted watch time by the total watch time [1].

IV. Proposed approach

In order to make connections between musical entities it is necessary to find the foundation of those connections. The collaborative filtering systems use as foundation the users and their listening habits. In content-based recommendation systems the foundation has to derive from the audio signals or from the semantic of the lyrics, if they exist.

The audio signals, as presented by Tzanetakis and Cook in [13], is a wide territory to explore when it comes to Music Information Retrieval. Their method approached three feature sets: timbral texture, rhythmic content and pitch content. The features of timbral texture contain different attributes (such as Spectral Centroid, Spectral Rolloff, Mel-Frequency Cepstral Coefficients and others) that can be used in music-speech discrimination. Rhythmic content is shown to differentiate between music genre using the beat histogram. The features that make this possible are the relative amplitude of the first and the second histogram peak, the ratio of the amplitude of the second peak divided by the amplitude of the first peak, the period of the first and second peak and the overall sum of the histogram. Finally, the pitch content features, computed from the folded and the unfolded pitch histogram, are used in identifying certain genres. For all that, the pitch content cannot fully characterize genres [13].

The proposed approach uses features derived from the features presented above to link the songs between each other. Creating a map of entities is difficult in a multidimensional space. Each item has to be positioned relatively to all the other items and then the items have to be clustered. This would usually take big time and space complexity. Furthermore, the solution must be one of self-learning, as the problem does not get as input any hint of what is expected or a model that should be followed. Therefore, the problem needs a solution in the sphere of Unsupervised Learning.

As the purpose is obtaining groups of tracks (with similar characteristics) as output, the algorithm should solve a clustering problem. Considering a large dataset with NF numerical features per item, the solution proposed is the Self Organizing Map algorithm. Vesanto et al. state in [17] that SOM's computational complexity scales linearly with the cardinality of the dataset. Besides, for memory it requires space only for the prototype vectors and the current training vector.

The SOM underlying mechanism actually involves an artificial neural network that uses the distance between neurons to reduce the dimension of the input data. Behind the SOM, the idea is using a two dimensional grid of map units (neurons). The neurons are represented with prototype vectors [5]:

$$m_i = [m_{i1}, ..., m_{iNF}] \tag{1}$$

NF being the dimension of the input data and m_{ij} being the value m_i unit (neuron) has for the j-th feature. The model vector ***m*** is initialized randomly or through an algorithm (e.g. Principal Component Analysis). This grid acts like an "elastic net" that is stretched to cover all items. The training is made in an iterative manner: at each iteration, a random candidate (*x*) is picked from the input set; for this candidate the best matching unit (BMU, notated as m_b) has to be found [5]:

$$||x - m_b|| = \min_i\{||x - m_i||\} \tag{2}$$

Afterwards the prototype vectors of all items are updated as a result of moving the BMU closer to *x* [17]:

$$m_i(t+1) = m_i(t) + \alpha(t) \cdot h_{bi}(t) \cdot [x - m_i(t)] \tag{3}$$

t - time
$\alpha(t)$ - adaptation coefficient
$h_{bi}(t)$ - neighborhood kernel centered on the winner unit:

$$h_{bi}(t) = exp\left(\frac{-||r_b - r_i||^2}{2 \cdot \sigma^2(t)}\right) \tag{4}$$

r_b - position of neuron b on the grid
r_i - position of neuron i on the grid
$\sigma(t)$ - the width of the kernel
considering that $\alpha(t)$ and $\sigma(t)$ are decreasing monotonically with time [17].

As presented in [5] by Teuvo Kohonen the grid of the map can be rectangular, hexagonal and even irregular, depending on the way the map units are connected to each other. Between the hexagonal and rectangular form, the hexagonal is much more preferred because of its isotropy. Besides, the hexagonal grid is considered to be also better for visualization as a unit of the map has more neighbours - 6 than those of the rectangular grid that have 4. It is much less rigid and allows better covering of the input data [5].

The output of the algorithm is the grid itself, which contains one or more items of the input data (a cluster of songs) in every unit of it. Therefore, for any given song, similar songs can be found in the cluster in which it belongs or in the clusters of adjacent units. This way, the list of songs that was given as input for the system translates into a map of songs in which the proximity of points is equivalent to audio similarity. Finally, in order to make recommendations for a given track, all that needs to be done is identifying its location and its closest neighbours.

V. Numerical experiments

The purpose of the conducted experiments was to test the system's ability to recommend similar songs based on audio features. A way to evaluate the results is observing if the clusters formed by the system contain tracks of just one genre.

The input data set used in the experiments was created using Spotipy - a Python web crawler developed for Spotify's API. Every song has 11 audio features (NF = 11) [11]:

- acousticness: real value between 0 and 1 measuring the likeliness that the track is acoustic;
- danceability: real value between 0 and 1 measuring how danceable the song is;
- energy: real value between 0 and 1 measuring intensity and activity;
- instrumentalness: real value between 0 and 1 measuring the likeliness that the song contains vocals;
- key: integer value between 0 and 11 - the key the track is in - integers map to pitches using standard Pitch Class notation (e.g. 0 = C, 2 = D and so on);
- liveness: real value between 0 and 1 measuring the likeliness that the song was performed live;
- loudness: real value between -60 and 0 measuring how loud the song is;
- mode: integer (0 or 1) indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived (major is represented by 1 and minor is 0);
- speechiness: real value between 0 and 1 measuring the likeliness that the song contains spoken words;
- valence: real value between 0 and 1 measuring whether the song inspires positive/negative emotions;
- tempo: real value expressed in beats per minute.

11 different genres (pop, rock, indie rock, soul, classical, disco, metal, hip-hop, funk, traditional country) were selected and, for each genre, a representative artist. The web crawler got the artist's most popular 10 songs and their features, and then went to similar artists and got their most popular 10 songs and so on. As a result, the final data set contained 27000 of songs of 11 different genres.

For the implementation of the self-organizing map algorithm a Python tool ([14]) was used. The parametrization was made as follows: the initialization of the prototype vectors was not made randomly, but through a Principal component analysis (PCA) algorithm, the data was normalized before entering the algorithm, the neighbourhood size was calculated through the Gaussian function and the map had a size of 50x50 units (which resulted in 2500 clusters). The normalization and the initialization through PCA is not mandatory, but they raise the performance of the system and fewer iterations are needed. The number of clusters was chosen so that the number of songs divided by the number of clusters to be approximately 10.

The first experiment was running the algorithm with a rectangular grid. The clusters of songs that were determined had dimensions between 1 (songs that did not match with other songs based on characteristics) and 30 (songs with a large number of possible recommendations).

The second experiment was running the algorithm with a hexagonal grid on the same data set. The results were clusters of songs with dimensions between 1 and 80 songs.

Computing the results of the experiments meant calculating a score: for each cluster the predominant genre was calculated; for each genre, the homogeneity of the clusters whose predominant genre was that one. The calculation of the homogeneity was the number of songs that have the predominant genre of the cluster divided by the number of songs of the cluster.

$$f_{g_c} = |\{s \in S_c, c \in C | \text{s has genre g}\}| \tag{5}$$

where f_{g_c} is the frequency of the genre among the songs included in S_c (the set of songs in cluster c)

$$g_c = \{p \in G | f_{p_c} = \max_{g \in G, c \in C} f_{g_c}\} \tag{6}$$

where g_c is the predominant genre of the cluster c, G is the set of genres and C is the set of clusters

$$h_c = \frac{|\{s \in S_c | \text{s has the genre } g_c\}|}{|S_c|} \tag{7}$$

where h_c is the homogeneity of the cluster c

The final score for each genre was how many of their clusters have homogeneity over 0.5 or equal to 1.0.

$$s'_g = |\{c \in C | g_c = g \text{ and } h_c \geq 0.5\}| \tag{8}$$

$$s''_g = |\{c \in C | g_c = g \text{ and } h_c = 1.0\}| \tag{9}$$

The results of the evaluation can be seen in Figure 2 and Figure 3.

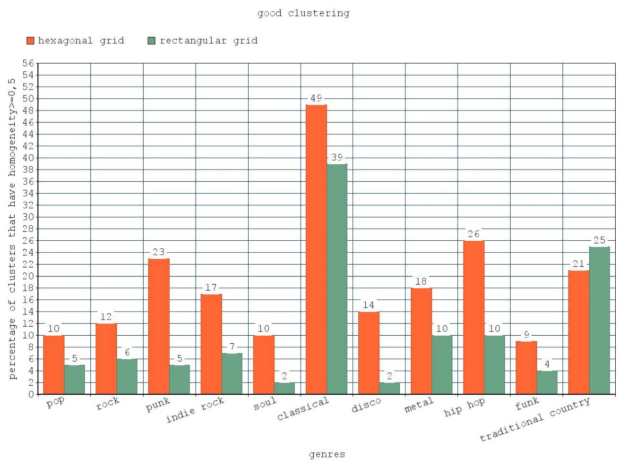

Fig. 2. This graph presents the percentage of clusters, per genre, that have homogeneity over 0.5. The orange bars correspond to the results scored by the SOM with the hexagonal grid and the green bars correspond to the results scored by the SOM with the rectangular grid.

The conclusions drew from the results are the following:

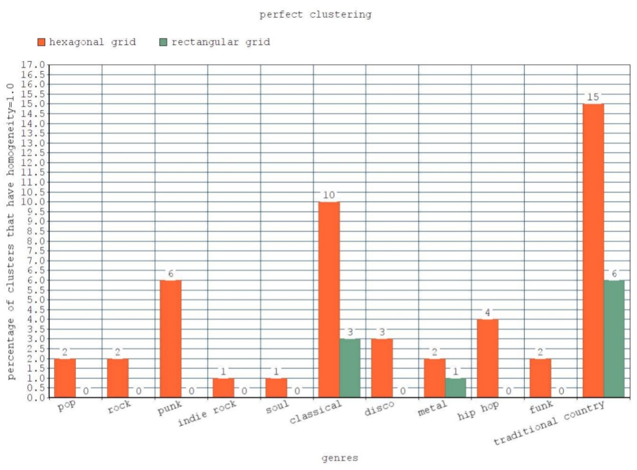

Fig. 3. Similarly to the graph above, the illustration presents the percentage of clusters, per genre, that have homogeneity equal to 1.0, more precisely the percentage of clusters that contain tracks of a single genre.

1) the algorithm parametrized with hexagonal grid scored better than the one parametrized with the rectangular grid - which strengthens the idea presented by Teuvo Kohonen in [5] that it is much better to use a hexagonal map;
2) as the score of the two algorithms is low, there is a high probability that the selection of the features for the input data did not contain the information needed to differentiate between the genres; however, clusters containing mostly classical music scored high in both graphs, which might mean that the proposed system would work better on genres that do no contain lyrics in comparison with genres that do;
3) as a consequence of point 2), other characteristics should be searched in the semantic of the lyrics (through Natural Language Processing algorithms) in order to raise the score of genres whose songs contain lyrics and also to strive to bridge the semantic gap.

VI. Conclusion

This article proposed a solution for music recommendation in absence of user data, more precisely an approach that uses Unsupervised Learning - Self Organizing Maps - on a large set of tracks defined by multiple audio features. The problem was defined as building a system that is able to create connections between musical entities that can lead to recommendations. Other approaches encountered the cold start problem, problem that the presented system managed to overcome. The downside is that the performance is not as good as the performance of a system that relies on usage data, but these two solutions can be combined in order to develop a system that solves the shortcomings caused by the inconsistency and the incompleteness of a feedback system and also those caused by a content-based approach. However, the conducted experiments show that further research needs to be made in selecting features related with the meaning of the songs in order to get better recommendations. Consequently, future work should examine more closely the lyrics, the title, the elements that compose the semantic side of music. Using these elements, recommendation systems may be able to extract the theme of each song, finally getting closer to bridging the semantic gap.

References

[1] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198. ACM, 2016.

[2] Kyle B Dickerson and Dan Ventura. Music recommendation and query-by-content using self-organizing maps. In *2009 International Joint Conference on Neural Networks*, pages 705–710. IEEE, 2009.

[3] Benjamin Heitmann and Conor Hayes. Using linked data to build open, collaborative recommender systems. In *2010 AAAI Spring Symposium Series*, 2010.

[4] Hyun-Tae Kim, Eungyeong Kim, Jong-Hyun Lee, and Chang Wook Ahn. A recommender system based on genetic algorithm for music data. In *2010 2nd International Conference on Computer Engineering and Technology*, volume 6, pages V6–414. IEEE, 2010.

[5] Teuvo Kohonen. *Self-organizing maps*, volume 30. Springer Science & Business Media, 2012.

[6] Paul Lamere. Social tagging and music information retrieval. *Journal of new music research*, 37(2):101–114, 2008.

[7] Qing Li, Byeong Man Kim, Dong Hai Guan, et al. A music recommender based on audio features. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 532–533. ACM, 2004.

[8] Tianyi Liu, Shuangsang Fang, Yuehui Zhao, Peng Wang, and Jun Zhang. Implementation of training convolutional neural networks. *arXiv preprint arXiv:1506.01195*, 2015.

[9] Alexandros Nanopoulos, Dimitrios Rafailidis, Panagiotis Symeonidis, and Yannis Manolopoulos. Musicbox: Personalized music recommendation based on cubic analysis of social tags. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(2):407–412, 2010.

[10] Elias Pampalk, Andreas Rauber, and Dieter Merkl. Content-based organization and visualization of music archives. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 570–579. ACM, 2002.

[11] Spotify. Get audio features for several tracks.

[12] Panagiotis Symeonidis, Maria M Ruxanda, Alexandros Nanopoulos, and Yannis Manolopoulos. Ternary semantic analysis of social tags for personalized music recommendation. In *Ismir*, volume 8, pages 219–224, 2008.

[13] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

[14] Ivn Valls Vahid Moosavi, Sebastian Packmann. Sompy.

[15] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.

[16] Shankar Vembu and Stephan Baumann. A self-organizing map based knowledge discovery for music recommendation systems. In *International Symposium on Computer Music Modeling and Retrieval*, pages 119–129. Springer, 2004.

[17] Juha Vesanto, Esa Alhoniemi, et al. Clustering of the self-organizing map. *IEEE Transactions on neural networks*, 11(3):586–600, 2000.

[18] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010.