Predicting Song Similarity using Deep Neural Networks

Sneha Shet
University of Illinois at Chicago
Chicago, Illinois
sshet6@uic.edu

Sivaraman Lakshmipathy
University of Illinois at Chicago
Chicago, Illinois
slaksh5@uic.edu

Sudheer Kumar Reddy Beeram
University of Illinois at Chicago
Chicago, Illinois
sbeera2@uic.edu

ABSTRACT

We intend to identify the possibility of predicting the similarity of songs using only its lyrics. A song consists of multiple components that uniquely identify it. We study the significance of the lyrics' role in two songs being considered similar to each other. Building upon the technique of one-shot learning using Siamese networks, we define a model to train on pairs of song lyrics to predict if they are similar. This similarity factor is defined using methods like collaborative filtering. The model is trained predominantly on lyrics in the English language. The results of the experiments suggest that lyrics, when used alone, are not a reliable indicator of the similarity quotient and could be co-dependent on other components of the songs.

KEYWORDS

songs, lyrics, neural networks, similarity

1 INTRODUCTION

In the era of digitization of music, the industry has gained massively by the adaptation of recommendation systems to identify user behavior patterns and provide tailored music that suits each individual. The invent of Music Information Retrieval that deals with the extraction of information from music to categorize, manipulate, or create new music signifies the potential of utilizing statistical models to study music. With the abundance of user data available in the present day, techniques like collaborative filtering have been successfully used in recommender systems.

Collaborative filtering is the process of making automatic predictions about a user's preferences by collecting and collaborating information about many other users. Applying collaborative filtering to the problem of predicting songs to users can be considered as an industry-standard today.

This system of recommendation is driven by the user's listening patterns, which may result in biases in terms of regionality, gender, or language. We consider that this approach could lend itself to a situation where popular songs might trend extensively and potentially ignore (less popular) music, which could be better candidates. Also, while collaborative filtering techniques perform excellently when historical data is available for songs, this reliance curbs the user from discovering new/less popular songs organically.

A song is made of a multitude of components: the artist, genre, audio features, and instrumentation, to name a few. When songs are evaluated to be similar or dissimilar, these components inherently contribute to the nature of similarity. All songs, excluding instrumental music, contain lyrics as the basic building block. This leads to the intuition that the lyrics must play a significant role in the similarity factor between songs. This serves as a motivation to explore the role of lyrics in deciding the similarity, either as a standalone factor or in tandem with existing recommendation system techniques.

Exploring the idea of studying the lyrics as a standalone feature to predict song similarity is the cornerstone of the set of experiments elaborated in the following sections. Lyrics are composed of a large number of words and their combinations, thus resulting in a feature space too complex to apply traditional Machine Learning algorithms. However, the advent of deep learning architectures provide a means to handle long sequences of data and extract information from them. We implemented an end-to-end deep learning architecture that can be trained to predict the similarity between two songs using only their lyrics.

Similar works in the field of images have utilized Siamese networks for the task. Adapting the technique for song lyric snippets, we design a twin network architecture with shared weights to process a pair of songs and generate a combined vector representation that is unique to the pair. This representation is fed as input to fully connected layers to train for the prediction task. The model is evaluated on the accuracy and loss value metrics to study the performance of the proposed models.

The experiments have not produced significant results to corroborate the initial hypothesis. However, the exploration processes involved in the experiments suggest that they could be improved upon in terms of complexity and design to investigate the hypothesis further.

2 RELATED WORK

Most of the music recommendation systems use Collaborative Filtering methods. These methods do not address the cold start problem of recommending new and unpopular songs. Zheng et al.[7] deal with the issue by proposing Deep Temporal Neural Music Recommendation Model (DTMNR) based on song metadata like length, genre, artist, and users' temporal preferences. They generate a rating value of an item for user outperforming seven baseline methods. The music metadata are encoded as one-hot vectors and projected to low-dimensional space by Deep Neural Networks to create hidden characteristics.

Soleymani et al.[8] address this issue by proposing a content-based music recommender system based on five attributes, namely, Mellow, Unpretentious, Sophisticated, Intense, and Contemporary (MUSIC) derived from psychological studies of music preference significantly outperforming genre-based and user-based recommendation on the root mean square error.

Many studies on song similarity majorly explore the audio content aspect of the song. The general method followed was first formulating a model for audio content and then evaluating a similarity model against the known ground truth. Eck et al.[5] get similarities between songs by mapping audio features onto Web-collected tags by running a set of boosted classifiers. Cunha, Caldeira, Fujii[4] have

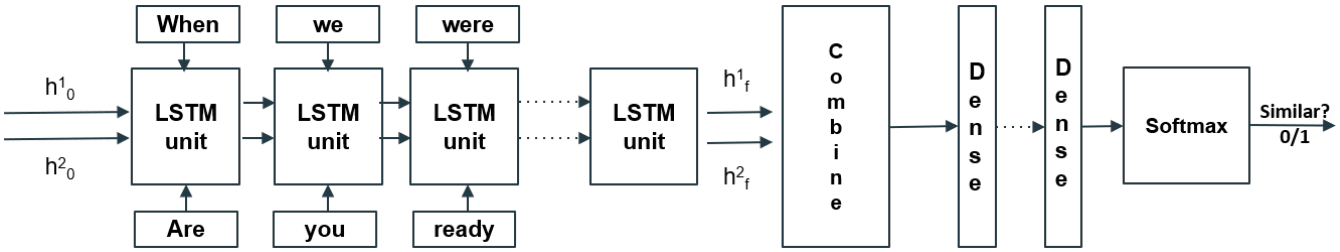

Figure 1: Network model design

built a recommendation system that implements machine learning techniques solely on song metadata (such as the performer, and song title) and on tags contributed by users. Kong, Feng, and Li Kong et al.[9] used pre-computed convolutional filters followed by an MLP to get 73% genre classification accuracy on the Tzanetakis dataset.

Oord et al.[11] propose a latent factor model that predicts latent factors from raw music audio using CNNs and found that it outperforms the traditional bag-of-words that uses engineered audio features such as MFCCs.

3 PROBLEM STATEMENT

Using pairs of song lyrics in the English language along with their similarity scores, we generate a *combined representation* for the pair using a twin network trained with shared weights. This is used to train LSTM/GRU networks to perform the classification task of identifying the pair to be similar or dissimilar. This approach is evaluated using accuracy score, and conclusions are drawn.

Prior to the training task, a custom word embedding is generated from a larger corpus of lyrics and trained as a Word2Vec model [10]. The effects of using word embeddings as opposed to using randomly initialized embeddings are studied.

4 TECHNICAL APPROACH

4.1 Baseline Model

The inspiration behind the idea of solving the problem of similarity between two songs is the concept of *one-shot learning*, which can be trained on a single example per each class. Though the setting in this problem is slightly different from a number of data points available for each class, an analogy can be made between the unique pairs of lyrics being considered as a one-shot classification task.

The Siamese networks were first discussed by *Bromley et al*[3] as an algorithm to solve the problem of verification of signatures. This work considered the problem of identifying the signature similarity as an image matching problem. The networks were designed with identical networks with similar time-delay neural networks to match the input image with the previously stored image by minimizing the distance measure between the source and target signature images.

Building upon the original paper, the experiments to use Siamese architecture for One-shot image recognition [6] is considered as the reference for the baseline model. The reference paper performs the task of classification of the Omniglot dataset by defining a siamese network to form twin networks for a given pair of images, with shared weight matrices at each layer of the network. Unlike the original paper, the weighted L_1 distance measure is calculated between the trained hidden features for the input pairs of images. This is mapped to a region of [0, 1] by applying a sigmoid function.

Similarly, the baseline model for this experiment utilizes a twin network model with shared weights to train on the input lyrics. The network model utilizes LSTM units to generate the feature representations of the input sequence of lyrics. The distance measure between the generated hidden layer representations of the twin network is calculated as the exponential negative manhattan distance between the outputs of the twin LSTM network and trained using a *AdaDelta* optimizer. Gradient clipping is used as a normalization technique in order to prevent the exploding problem during backpropagation.

4.2 Proposed model

Song lyrics are generally considered to be a sequence of words. Hence, the sequential nature could be one of the indicators of the similarity of songs. Notably, the sequences over time could lead to the similarity factor.

Vanilla Recurrent Neural Networks (RNN) serve the purpose of capturing the patterns in sequences. However, owing to the longer sequence lengths involved in songs, they are prone to vanishing gradient and gradient explosions. This makes the RNN network inefficient to train the lyric snippets, with a significant proportion of them utilizing a minimum of 70 distinct words.

The Long-Short Term Memory (LSTM) frameworks are considered to be an ideal candidate in order to build the twin network architecture with shared weights.

4.2.1 *Model architecture.*

The architecture of the proposed model, as given in Figure**1**, is as follows:

- Each song is represented as a sequence of words in the lyrics.
- This is converted to word vectors using embedding representations.
- A twin network with LSTM units are used to process the pairs of lyrics in parallel.
- Each song is unrolled one-time step at a time through the network, and the LSTM hidden cell is updated at each time step.
- At the final time step, two final outputs are recorded, corresponding to two input lyric sequences.
- This pair of trained vector representation is subsequently combined in one of the three ways, as elaborated upon in

	Data source	Data count	Data format	Comments
1	last.fm	584,897	SQLite database	Similarity score ground truth
2	musixmatch	153,462	JSON	Lyrics
3	Total downloaded lyrics	153,462	JSON	Lyrics (all languages)
4	English language lyrics	127,581	JSON	Lyrics (English only)
5	Prepared dataset	24,020	CSV	Lyrics pairs with similarity labels
6	Generated dataset	48,040	CSV	Final dataset

Table 1: Table summarizing the data extraction process

later sections, to result in a combined expression of the pair. Let this representation be labelled h_{final}.

- Now, this representation for the given pair of lyrics is fed as input to a subsequent network of fully connected (FC) layers.
- These dense layers utilize ReLU as the activation function.
- The final layer comprises as a single node with sigmoid activation to calculate the probability of the input pair as similar or dissimilar.
- The cost function minimized is the Cross-entropy loss.
- Adam optimizer is used to train the model for 25 epochs with the early stopping criteria applied to the validation loss.

4.2.2 *Combining twin network outputs.*

The twin networks generate representations of the input pair of lyrics. Let them be identified as h_f^1 and h_f^2 correspondingly. The *Combine* layer in the network performs one of the following operations to combine the vector representations into h_{final}. This final vector representation is expected to capture the similar features of the pair of song lyrics. The value d_h represents the dimension size of the representation generated by the twin network.

(1) **Concatenation:**

$$h_{final} = [h_f^1 + h_f^2]$$

- The concatenation of the pair of lyrics is expected to result in a unique representation for each distinct pair.
- Since the similarity scores are considered to commutative in nature, this representation might lead to unexpected results when the input song lyrics to the twin network are interchanged, i.e., ($Song_q$, $Songp$) instead of ($Song_p$, $Songq$).
- To address the same, each pair of song is represented twice with the same class label in the final dataset: ($Song_p$, $Songq$, $class_y$) and ($Song_q$, $Songp$, $class_y$).
- The resulting representation is of the size $2d_h$.

(2) **Element-wise subtraction:**

$$h_{final} = [h_f^1 - h_f^2]$$

- The final representation has dimension d_h.
- This representation is inspired by the original Siamese network, which involved calculating the absolute difference between the representations.
- It is observed that training this representation results in anti-symmetric representation and hence an even number of layers in the Fully Connected layers are preferred for the representation

(3) **Element-wise multiplication:**

$$h_{final} = [h_f^1 * h_f^2]$$

- The final representation has dimension d_h.
- This representation is symmetric in nature.

The idea behind the combination of the representations of the pair of songs is that this representation encompasses unique to the pair. This can hence be trained to learn the features to classify the songs. Adapting the approach explored in BalakrishnanDixit[1], the final model is built to train on multiple one-shot pairs to build discriminative models for pairs of songs.

5 EXPERIMENTAL SETUP

The following sections briefly describe the process of data preparation and the experimental model setup.

5.1 Data

A major component of the work described here involved the identification and preparation of the song lyrics snippets. This has resulted in a richly processed data corpus for tasks aiming to utilize song lyrics in the English language.

5.1.1 *Data Collection.*

The Million Song Dataset is a freely-available collection of metadata of popular music tracks [2]. It contains extensive information about the songs, including, but not limited to, the author, genre, and length. Though this dataset does not provide any audio or lyrics of the songs, it acts as a primary hub for similar communities that collaborate to provide additional information for the songs referenced by the Million Song Dataset.

The first such dataset utilized is the last.fm[1] that contains song-level tags and song similarity ground truths for the experiments elaborated in this work. The tags refer to the genre of the songs. The dataset provides 584,987 tracks with a variable number of similar tracks along with their similarity scores for each corresponding pair of songs. This data is available in two forms: JSON files and SQLite database, and the latter is used.

The second dataset is musixmatch[2] that provides the actual song lyrics in bag-of-words format. Since we intend to experiment

[1] http://millionsongdataset.com/lastfm
[2] http://millionsongdataset.com/musixmatch

with lyrics by considering them as a stream of words, an alternate approach is utilized to extract the lyric snippets of the songs. The source provider of the dataset, the MusixMatch website[3], provides APIs for developers to download the lyrics identified by their ids extracted from the Million Song Dataset. Downloading the lyrics from the website's databases resulted in 153,462 unique lyrics. The data is downloaded in the form of JSON files.

5.1.2 *Data filtering*.

Owing to the mismatch in the total number of unique tracks available from both the datasets, the tracks with similarity scores but no lyrics were identified and removed from the final dataset that combined the last.fm and musixmatch data. This combined dataset contains pairs of tracks along with their corresponding similarity scores as a CSV file.

Before the data could be processed, it was noted that the songs available in the dataset were of multiple languages. Since the primary objective of the work described here was to work with the English language, songs in foreign languages were identified[4] and removed from the dataset.

5.1.3 *Data preparation*.

For each unique track id, there exists a variable number of target tracks with similarity scores. Extracting all such unique pairs resulted in a large dataset over 100 million data points. In order to identify a manageable value and to retrieve maximum information, the top most similar and non-similar tracks for each unique track were identified. This resulted in 140,000 pairs.

Repeating this process to identify the tracks with at least 5 similar and non-similar tracks generated a new set of (source, target) pairs. This resulted in a more information-rich yet smaller dataset of size 24,010. The statistics presented in Table**2** pertain to this final dataset.

5.1.4 *Data pre-processing*.

The data went through a series of pre-processing steps in order to prepare it to be trained by the proposed model.

- The copyright information appended at the end of the lyrics was removed since it was not part of the original lyrics.
- All lyrics are converted to lower case and stripped of numbers and symbols.
- Stop words are removed.
- Since the lyrics contain long streams of words, a threshold is utilized to identify rare words. Words with occurrences less than the threshold of 100 across the entire dataset are identified and removed from the original lyrics.
- The maximum length of the input sequence of lyrics is limited to 60. This value is arrived upon by calculating the mean and standard deviation of the length of the lyrics in the entire data corpus and summing them
- The similarity scores are converted to labels as follows. All pairs of songs with similarity score >= 0.5 were labeled '1' to signify similar pairs while the rest were labeled '0' to signify non-similar pairs.

[3]https://www.musixmatch.com/
[4]https://pypi.org/project/langdetect

5.1.5 *Additional data generation*.

Each generated data point is in the form of (source, target) pair along with the similarity label. The similarity property of a pair of songs is considered to commutative. In order to represent the same, new data is generated by swapping the source and target tracks of each pair to generate (target, source) pairs with the same similarity label as the original pair. This effectively doubled the size of the dataset available to experiment.

5.1.6 *Data splitup*.

The finally generated data consists of 48,040 data points. It is ensured that there is an equal number of song pairs for the similar and non-similar classes to ensure that the distribution is uniform.

In order to perform experiments using the model, the data is split into training, validation, and test data using a `70 - 15 - 15` split. This is equivalent to training on 34,040 data points and validating and testing on 7000 data points.

	Train	Val	Test
# of similar lyrics	17,020	3,500	3,500
# of non-similar lyrics	17,020	3,500	3,500

Table 2: Split for Data for Train, Test and Validation set

5.2 Experiments

Each model was trained separately for all three combinations of representation vectors. Different configurations were explored to study the implementation of the proposed models and their performance.

5.2.1 *Word embeddings*.

Initially, the words were represented as vectors by randomly initializing the weights for the embedding layer and trained via backpropagation during training. The randomly initialized word vectors had a dimension size of 20.

Further down the development pipeline, the idea of utilizing word embeddings was explored by using Word2Vec embeddings. However, this did not lead to any significant improvements in the results.

On studying the effect of using word embeddings, it was identified that a significant fraction of the data was being marked as *unknown words*. Exploring the lyrics lead to multiple instances where the jargon was different from the regular language. Since the Word2Vec embeddings were primarily trained on formal language documents, this was identified to be less than a satisfying resolution to the idea of using pre-trained models for word embeddings. Hence, we ended up generating custom word embeddings by training Word2Vec models on the entire data corpus downloaded from *Musixmatch* database, *for 200 epochs*, for different dimensions: 10, 50, and 100. The 100-dimensional model was identified as more suited for the task and was used to train the network.

5.2.2 *Input sequence length*.

The input sequence length was fixed based on the mean and standard deviation values of lyric length in the final dataset. A mean

		Random Initialization		Trained Embedding	
Combining Method	No of FC layers	Accuracy (Train-Val-Test)	Loss (Train-Val-Test)	Accuracy (Train-Val-Test)	Loss (Train-Val-Test)
Multiply	1	0.86 - 0.57- 0.57	0.33 - 0.67 - 0.86	0.55 - 0.50 - 0.49	0.67 - 0.70 - 0.72
Subtract	1	0.85 - 0.56 - 0.57	0.34 - 0.68 - 1.1	0.49 - 0.49 - 0.50	0.69 - 0.69 - 0.69
Concatenate	1	0.83 - 0.55 - 0.56	0.35 - 1.19 - 1.17	0.52 - 0.50 - 0.50	0.67 - 0.73 - 0.73
Multiply	2	0.81 - 0.56 - 0.57	0.39 - 0.69 - 0.92	0.51 - 0.50 - 0.49	0.68 - 0.70 - 0.70
Subtract	2	0.83 - 0.56 - 0.56	0.38 - 1.00 - 0.99	0.52 - 0.50 - 0.50	0.68 - 0.71 - 0.71
Concatenate	2	0.55 - 0.51 -0.50	0.43 - 0.94 - 0.92	0.51 - 0.50 - 0.50	0.68 - 0.70 - 0.70

Table 3: Table summarizing the results for model with LSTM units

		Random Initialization		Trained Embedding	
Combining Method	No of FC layers	Accuracy (Train-Val-Test)	Loss (Train-Val-Test)	Accuracy (Train-Val-Test)	Loss (Train-Val-Test)
Multiply	1	0.83 - 0.52 - 0.52	0.58 - 0.70 - 0.70	0.51 - 0.51 - 0.49	0.69 - 0.69 - 0.70
Subtract	1	0.78 - 0.52 - 0.52	0.58 - 0.69 - 0.70	0.51 - 0.51 - 0.49	0.69 - 0.69 - 0.69
Concatenate	1	0.80 - 0.51 - 0.51	0.59 - 0.69 - 0.69	0.51 - 0.51 - 0.48	0.69 - 0.69 - 0.69
Multiply	2	0.50 - 0.50 - 0.49	0.69 - 0.69 - 0.69	0.51 - 0.51 - 0.49	0.69 - 0.69 - 0.69
Subtract	2	0.50 - 0.50 - 0.50	0.69 - 0.69 - 0.69	0.52 - 0.51 - 0.49	0.69 - 0.69 - 0.71
Concatenate	2	0.50 - 0.50 - 0.51	0.69 - 0.69 - 0.69	0.50 - 0.51 - 0.50	0.69 - 0.70 - 0.74

Table 4: Table summarizing the results for model with GRU units

Random Initialization		Trained Embeddings	
Accuracy (Train-Val-Test)	Loss (Train-Val-Test)	Accuracy (Train-Val-Test)	Loss (Train-Val-Test)
0.90 - 0.53 - 0.52	0.33 - 0.41 - 0.47	0.75 - 0.53 - 0.51	0.18 - 0.27 - 0.36

Table 5: Table summarizing the results for baseline(Siamese) model

value of 38 with a standard deviation of 22 resulted in a final sequence length of 60 to minimize the loss of data in the longer lyric snippets.

5.2.3 ***Dense layers***.

The generated *combined* vector representations were trained using fully connected dense layers to minimize the cross-entropy loss in the prediction task using Adam optimizer. The experiments explored utilizing one and two fully connected layers. For a single fully connected layer, the model was trained for 32-dimensional and 64-dimensional LSTM/GRU hidden unit output space. The model gave better results for 64-dimension LSTM/GRU hidden output space. In the case of two fully connected layer models, the training was repeated using 32-dimensional LSTM/GRU hidden output space, with the first layer containing ten hidden neurons. Both the network designs used a single sigmoid activated neuron in the final layer to predict the resultant class.

5.2.4 ***Hyperparameter tuning***.

A learning rate of 0.01 was determined to be an ideal value for training, with a batch size of 32. A dropout value of 0.1 is used for regularization in the network. The dropout factor is used between the input and LSTM/GRU layers, hidden units, combined vector representations, and the fully connected dense layers.

The model performed poorly for concatenation mode. This may be because of the fact that the other two methods (multiplication and subtraction) learn comparative information about the two-song vectors by combining individual features. The model performed well for element-wise multiplication. This is because the final combined representation is symmetric, and the element-wise operation captures important comparative information of the input vectors.

6 RESULTS

Table**3** shows that Multiply combining method performed better overall. The maximum test accuracy which we could achieve was

57% with the LSTM model. GRU (Table**4**) did not perform better than LSTM. However, as we can see from Table**5**, our best model performed better than the baseline model (52% test accuracy) for two sets of word embedding. Concatenate performed poorly than the other two methods. This may be because the concatenate technique does not combine individual song features.

Using custom embedding did not improve the results significantly. The results deteriorated and gave a maximum of 51% test accuracy for the GRU model. The baseline model, too, did not perform well on trained embedding. This may be because we just had 127,581 lyrics (English only). We set a threshold of 100 frequency and eliminated the less frequent words. This resulted in a vocabulary of the size of approximately 0.4 million only.

Despite our initial intuition that lyrics alone might be a stronger predictor of song similarity, our experiments here did not produce acceptable results. One reason for this could be that our model was trained on very little data. The overall distribution of positive and negative samples was highly uneven, which forced us to pick the best five positive and negative samples for each song and discarded the songs that had less than five samples to maintain an even fraction of positive and negative samples. This might have restricted the model from learning about a single song as each song had merely 10 data samples.

7 CONCLUSIONS AND FUTURE WORK

We explore a different approach to identifying song similarity in order to overcome the limitations of the collaborative filtering approach, which is predominantly used as the industry standard.

We design a Neural Network model inspired by Siamese networks and existing works using the network. The model is trained on song lyrics in the English language without utilizing other metadata. The results of the training experiments result in less than promising results, but we believe that the approach could yield significant results by refining it. The optimism stems from the success of adapting Siamese networks for the task of one-hot training of image classification, which has led to statistically significant outcomes.

One major factor influencing the experiments was the word embeddings utilized. The available embeddings are not suitable for the lingo in the lyrics, thus resulting in a large proportion of unknown words in the corpus. Trying to tackle this phenomenon by training custom embeddings on the lyrics that were downloaded for the experiments resulted in negligible improvements in the results. This could be resultant of the small size of the data corpus when compared to embeddings like Word2Vec and GLoVE. Thus, training custom embeddings for the lyrics could be a future branch of work.

We would also like to define more complex networks to study the significance of the learning encompassed in the fully connected layers of the proposed network model. Utilizing convolutional layers could also be useful, as evidenced by similar works in the domain of image processing.

Additionally, an ensemble model could be designed to combine the information gained from the lyrics along with the metadata of the songs to identify the similarity feature.

8 TEAM WORK DIVISION AND OVERALL EXPERIENCE

The entire project had the following major steps:

- Downloading 153,462 lyrics using musixmatch API.
- Data pre-processing and generating (source, target) pair.
- Building and training the baseline, LSTM, and GRU models.

Each member of the team generated their API keys to download roughly one-third of the entire data corpus, pre-processed the data, and generated the final dataset. Lakshmipathy built, trained, and evaluated the Baseline model while Shet was responsible for the LSTM model, and Beeram worked with the GRU model. Once the separate experiments were completed, the team worked together to identify the best hyperparameters across the models to improve them. The results presented here are the reflections of the cumulative efforts of the team to identify the best performing configurations in the proposed models.

REFERENCES

[1] Anusha Balakrishnan and Kalpit Dixit. 2014. *DeepPlaylist: Using Recurrent Neural Networks to Predict Song Similarity.* Master's thesis. Stanford University, US. https://cs224d.stanford.edu/reports/BalakrishnanDixit.pdf

[2] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. 2011. The Million Song Dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011).* http://ismir2011.ismir.net/papers/OS6-1.pdf

[3] Bentz James W Bottou Leon Guyon Isabelle LeCun Yann Moore Cliff Sackinger Eduard Bromley, Jane and Roopak Shah. 1993. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence* (1993). https://papers.nips.cc/paper/769-signature-verification-using-a-siamese-time-delay-neural-network.pdf

[4] Renato Luiz de Freitas Cunha, Evandro Caldeira, and Luciana Fujii. 2017. Determining Song Similarity via Machine Learning Techniques and Tagging Information. *CoRR* abs/1704.03844 (2017). arXiv:1704.03844 http://arxiv.org/abs/1704.03844

[5] Douglas Eck, Paul Lamere, Thierry Bertin-mahieux, and Stephen Green. 2008. Automatic Generation of Social Tags for Music Recommendation. In *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (Eds.). Curran Associates, Inc., 385–392. http://papers.nips.cc/paper/3370-automatic-generation-of-social-tags-for-music-recommendation.pdf

[6] Richard Zemel Gregory Koch and Ruslan Salakhutdinov. 2015. Siamese Neural Networks for One-shot Image Recognition. *ICML Deep Learning Workshop* (2015). https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf

[7] N. Liang A. K. Sangaiah Y. Jiang H.-T. Zheng, J.-Y. Chen and C.-Z. Zhao. 2019. *Applied Science* (1st. ed.). Vol. 9. University of Chicago Press, Chicago, 703. https://doi.org/10.1007/3-540-09237-4

[8] Frans Wiering Mohammad Soleymani, Anna Aljanaki and Remco C. Veltkamp. 2015. Content-based music recommendation using underlying music preference structure. In *IEEE International Conference on Multimedia and Expo (ICME).* http://www.cs.uu.nl/groups/MG/multimedia/publications/art/icme2015-preference.pdf

[9] Y. Li Q. Kong, X. Feng. 2014. Music Genre Classification Using Convolutional Neural Network. (2014). https://pdfs.semanticscholar.org/5477/9685b293a5afa2cd3107f479bf4503a99e82.pdf

[10] K. Chen G. S. Corrado T. Mikolov, I. Sutskever and J. Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS '13)*, Vol. 2. 3111–3119. https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[11] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2643–2651. http://papers.nips.cc/paper/5004-deep-content-based-music-recommendation.pdf