Music Genre Recognition Based on Lyrics

Akin Parkan
1752340
Eindhoven University of Technology
the Netherlands

Boris Rokanov
1396331
Eindhoven University of Technology
the Netherlands

Georgi Kostov
1396773
Eindhoven University of Technology
the Netherlands

Ivan Germanov
1483838
Eindhoven University of Technology
the Netherlands

Jeffrey Chan
1396005
Eindhoven University of Technology
the Netherlands

Abstract

Recognizing music genres based on song details is an instrumental task for many music-based companies. There have been many models that succeed in identifying songs based on both lyrical and auditory data.

This paper focuses on lyrical analysis. We propose two models that make use of network analysis - a *GCN* and *Node2Vec* model. We explain how we clean the dataset, show how we make the models, and discuss the achieved results.

Our *GCN* approach gives us a 0.7816 accuracy, while the *Node2Vec* model comes close at 0.7277. In the end, we conclude that our models are limited due to mainly using lyrical data. Our code is available in our GitHub repository [17].

Keywords: graph, neural networks, stemming, graph convolutional neural networks, music, genre, detection, text classification, node2vec

ACM Reference Format:
Akin Parkan, Boris Rokanov, Georgi Kostov, Ivan Germanov, and Jeffrey Chan. 2021. Music Genre Recognition Based on Lyrics. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/1122445.1122456

1 Introduction

How could music-oriented companies develop better recommendation algorithms? Due to the rapid development of technologies in 21st century, problems such as music classification and recognition have become popular. Many companies such as SoundCloud and Spotify stand to gain a lot of useful data based on such classification models. By creating algorithms to classify song genres, they can better understand what distinguishes a specific genre and then use that data to create a better recommendation system based on the genres their users listen to. For this task, we present two deep learning works to present a better recommendation system for songs. One of them is applying *GCN* (Semi-supervised learning) [11] and the other one is applying *Node2Vec* (Unsupervised Learning)[6] to the music data. In this paper, first, we will refer to the previous work that has been done in Section 2. Afterwards, in Section 3, we give a problem description. In Section 4, we present our pre-processing steps. In Section 5, we explain how we construct our graphs for the deep learning tasks. In Section 6, we present our methodology. In Section 7, we are going to present our experiments. In Section 8, we are going to discuss the limitations of the results, and possible improvements in Section 9. Lastly, in Section 10, the conclusion can be found.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

2 Related Work

2.1 Song Classification

Finding the genre of a given song is a well-researched problem, based on which many models have been proposed. Audio-based classification has proven to give excellent results [4].

However, lyric-based classification performs rather poorly [13]. Nevertheless, due to the development of *Natural Language Processing (NLP)* in recent years [24], classifications based on lyrics have been gaining traction.

For example, by making use of a *Hierarchical Attention Network (HAN)*, the meaning of different words and phrases of the lyrics can be related to the different genres [22]. This approach allows classifying numerous genres but still fails to reliably deliver high accuracy.

Other machine learning models have been used in lyrical classification as well. However, in general, they also fail to give a good result, generally, because of similarity between genres and the weak connection between words/phrases and their genres [24].

As a way to circumvent those problems, some machine learning models make use of both the lyrical and audio data

[2]. By doing that, the model can classify a given song with much higher accuracy to its genre.

However, all the example models for lyrics classification we found were based on *NLP* algorithms, and none were on graph data.

2.2 Graph Convolutional Network (GCN)

A *Graph Convolutional Network* (*GCN*) is a neural network graph model which is used for graph analysis [11]. *GCNs* preserve both the local and global structure of the graph.

The way that *GCNs* operate is relatively simple - for a graph G with an adjacency matrix A and an attribute matrix X, they calculate a degrees matrix D, where $D_{i,i} = \sum_j A_{i,j}$. Then based on D, they create a scaled-up adjacency matrix A'', in which each node has relatively the same importance (i.e., low degree nodes have more weight than high degree ones). Afterward, the matrix A'' is multiplied by X, in order to produce a new matrix that shows the average attributes of each node and all of its neighbors.

This process is repeated between each one of the layers in the neural network, thus making the number of layers represent "the maximal distance that attributes data can travel" (for n layers, we consider $n-1$-order neighbors).

Thanks to this process of calculating the weighted average of the neighbors and the node itself, we can produce a very accurate vectorization of each node, which shows the global and structural importance of the node. Because of that, *GCNs* are widely used for graph analysis tasks like node classification, link prediction, community detection, and network similarity.

Because of the robustness of networks, *GCNs* has been used in many fields. A particular area of interest is text classification, where *GCNs* produce good results due to their preservation of global information [9]. Another benefit of *GCNs* is that they tend to perform better than other text classification models when there is not as much data available. [25].

2.3 Node2Vec

Node2Vec [6] is a very popular model for embedding graph and node data. It works in 2 main steps - a **random walk**, followed by **Word2Vec** analysis:

Random Walk - The random walk used by *Node2Vec* is a 2nd-order biased random walk. A simplified description of the algorithm is that given an input of the graph $G(V, E)$, the starting node v, parameters p and q and the walk length w, it produces a list of randomly visited nodes (in the order of visit).

Based on the parameters p and q, the walk calculates a probability that from a given starting node v we will visit its neighbor u. The q parameter controls how likely we will not return to v after traveling to u, while p changes how likely we will visit v again.

After calculating the probabilities, a random neighbor of v is visited, and this whole process is repeated for the newly visited node. When the algorithm has visited w nodes, the walk is complete, and all the nodes visited are output.

Therefore, when we consider a smaller q, we expect that the walk will travel mostly around local clusters of nodes. However, if we select a bigger q, we expect that the random walk will visit more global clusters (i.e., nodes that are further away from v), and thus the walk will be more global.

Word2Vec - *Word2Vec* is a popular *NLP* algorithm, used to vectorize words. Given a text T input and a window size s, the algorithm starts analyzing the first word w of T. Next, it creates the context c, which contains all words that are within s words proximity (so the previous and next s words).

Next it applies one of 2 models, *Continuous Bag Of Words (CBOW)* and *Continuous Skip-Gram Model. CBOW* uses c to train a neural network to predict w. Skip gram works in the opposite way, where w is used to train a neural network to predict its context.

Both those models produce useful vectorization because similar words are used in a similar context (and similar context calls for similar word use).

Therefore, in the context of *Node2Vec*, after the random walk, the output of the walk is treated as words and fed into *Word2Vec* in order to define the meaning of each node. Thus, in the end, the nodes that have similar contexts are grouped.

3 Problem Description

Classifying genres using NLP tasks such as HAN[22] have been already researched. Also using audio classification too [2]. In this research, we extend our research's context using graph learning with NLP in order to take structural information that can be obtained from songs (see Figure 1) into account with the vector information of the songs to apply semi-supervised learning. Additionally, to obtain a more comprehensive model as well. Moreover, we researched an unsupervised learning task with constructing the graph which can be found in Figure 2. One can expect to have better results using the auditory data in order to classify genres. In this research, the main goal is to enhance genre classification using *NLP* task's performance by using additional structural information of the text data with applying *GCN*. Furthermore, this research focuses on exploring the structural information that can be obtained from music text data's effect on classifying genres with applying the *Node2Vec* method.

4 Pre-processing

The dataset which was used is called *Music Dataset: Lyrics and Metadata from 1950 to 2019* [15]. It provides a list of the lyrics of 28372 songs, describing music metadata as sadness, danceability, loudness, acousticness, etc. Even though these attributes were present, we focused only on the song lyrics. This was done to ensure that other future datasets which do

not contain auxiliary data such as sadness, loudness, etc. can also be used as input to the model. The pre-processing task consisted of several subtasks such as language distinction, removal of non-English songs, word stemming, and merging of genres of songs.

4.1 Lyrics Language Distinction

The initial pre-processing task was to distinguish the languages that were used in the *lyrics* field of the dataset. In order to accomplish this, we used the *googletrans* [5] library, which facilitated the language distinction. Its lyrics were used as input to Google's library for every song, which returned the detected language as output. Once the language was detected, we stored the returned value in an additional *lang* field (denoting the language of the specific song).

4.2 Removal Of Non-English Songs

Most of the songs in the dataset ($\approx 70\%$) were distinguished as English songs. The remainder of the songs were in other languages. However, each of the languages had a few songs for it. Because of this lack of data points, all non-English songs were removed by using a script. Due to the wide variety of languages within the dataset, multiple different stemming algorithms would have to be used. This would require more additional work for this paper.

4.3 Word Stemming

During this pre-processing task, we were only considering English songs. Word stemming is a traditional pre-processing task when working with words and text normalization. The aim of stemming is to extract the root of every word, such that every variation of a specific word maps to its stem. For example, the words *continued*, *continues*, and *continuing* would stem to *continue*.

To prepare the lyrics of the songs for further processing and to be used in the graph's structure, we used the steaming technique. The *PortStemmer* [19] of the *nltk* [16] library was utilized to stem the lyrics. *PortStemmer* is a simple and quick algorithm that uses suffix stripping to produce stems rather than following linguistic rules, which is why it sometimes produces stems that are not actual English words.

4.4 Merging Song Genres

After the removal of the non-English songs, an additional exploratory analysis was done on the dataset. It was discovered that all seven genres (pop, country, blues, jazz, reggae, rock, hip hop) were obtained using an AI model using the audio data of the songs. In order to centralize the relevant genre, we decided to merge the lyrics of some related genres into one genre. Jazz songs were combined with blues songs into a blues genre, and country songs were combined with rock songs into a rock genre.

5 Graph Construction

Once the pre-processing task was done, we proceeded with constructing the graphs. In this section, the two proposed graph constructions can be found. We used these graphs to run our experiments.

5.1 Graph 1 - Song And Word Nodes

For the first graph in Figure 1, we constructed our graph with only two types of nodes, which are the song and word nodes. Song nodes represent the song entity that we are trying to classify. Word nodes are the unique nodes that are in the full lyrics dataset and unique artist names. After we create the nodes, we construct the edges between-song nodes and word nodes according to the following rules: if the word can be found in the song's lyrics, the two nodes are connected in an undirected manner. Furthermore, every word node that specifies the artist is connected with the corresponding song nodes.

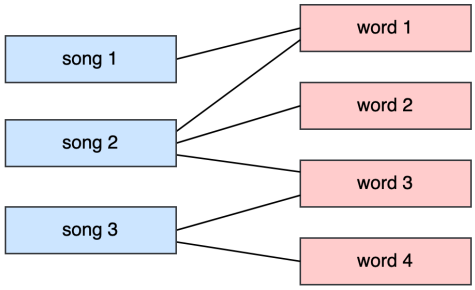

Figure 1. First Graph Structure

5.2 Graph 2 - With Attribute Nodes

For the second graph in Figure 2, we introduced a new attribute node type, similar to the work of [1]. While constructing this graph, we connected each word node with these attribute nodes according to which column they were obtained. For example, if a word node is obtained from the lyrics column, it is connected to a lyrics attribute node. Likewise, if a word node has the artist's name, it is connected with the artist attribute node.

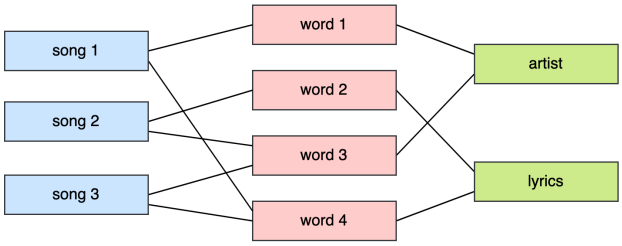

Figure 2. Second Graph Structure

6 Methodology

After obtaining the proposed graph 1, we constructed a *Word2Vec* model [14] to find the vector representations of the song nodes and word nodes. To do so, we discovered each unique word in our dataset for constructing our corpus; then, we fit the *Word2Vec* model with vector size = 100, window = 5. Afterward, for each song, we took the average *Word2Vec* vector from the unique words in the lyrics of that song, and for every word node, we took its vector representation as the attribute information.

For the *GCN* task, we used the *StellarGraph* [21] library. We constructed our model with 2 layers with 128 and 32 size, and we used *elu* for the first layer and *relu* for the second layer's activation function after applying hyper-parameter tuning as it can be seen in 4 and 5, with dropout rate of 0.5. Lastly, we used the *Adam* optimizer [10] with a learning rate of 0.01. We also fit our data with 100 epochs.

For the *Node2Vec* task [7] we used the *StellarGraph* [21] library again. We started our random walks just from the song nodes (our root nodes for this task were the song nodes only). We specified our maximum random walk length as 100, the number of random walks per root node as 10, p parameter as 0.5, and q parameter as 2.0 after applying hyper-parameter tuning as it can be seen in 2. Then, we constructed a *Word2Vec* model from our random walks, with $vector\ size = 128$, and $window = 5$ after applying hyper-parameter tuning as it can be seen in 3. Finally, we applied logistic regression to our *Node2Vec* embeddings to find out the accuracy of our task.

7 Experiments

In this section, we describe our findings after performing our proposed methods. We also ran a baseline that uses logistic regression with *Word2Vec* embedding to use as a comparison. The factors that we look at are the accuracy of the genre classification, and the *T-SNE* [23] representation.

7.1 Baseline Comparison

We compared our model with a logistic regression run on the average song word vector produced from the *Word2Vec* embedding.

The results of this approach were not very successful, as they only achieved an accuracy of 37% even after merging the genres. The *T-SNE* can be found in Figure 3. Unfortunately, the clustering of the genre was not successful at all.

7.2 Using GCN

The other proposed method was to use *Graph Convolutional Networks* [20] to classify our nodes. To achieve that, we used the first graph that we proposed in Section 5, and we obtained node embeddings applying semi-supervised learning.

This resulted in quite a good accuracy of 78% for classifying the genres. From the *T-SNE* representation in Figure 4, one can see the word node cluster, but it is not possible to see any other clusters.

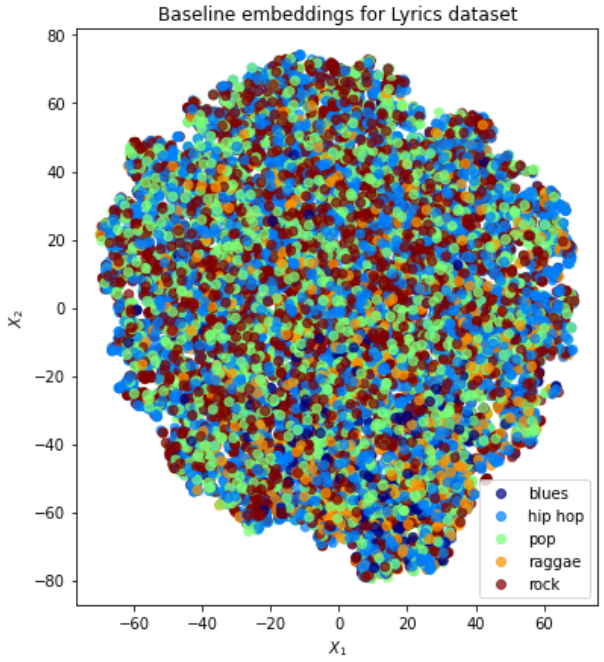

Figure 3. T-SNE baseline

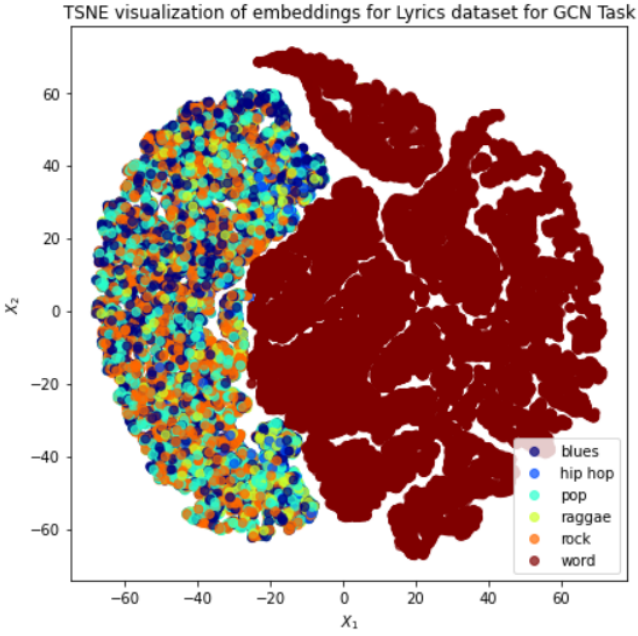

Figure 4. T-SNE GCN Task

7.3 Using Node2Vec

One of our proposed methods was to perform *Node2Vec* in order to classify our nodes. To achieve that, we used the second graph that we proposed in Section 5, and we obtained node embeddings by applying unsupervised learning.

This resulted in a good accuracy of 72% for classifying the genres. From the *T-SNE* representation in Figure 5, one is able to see 3 clusters.

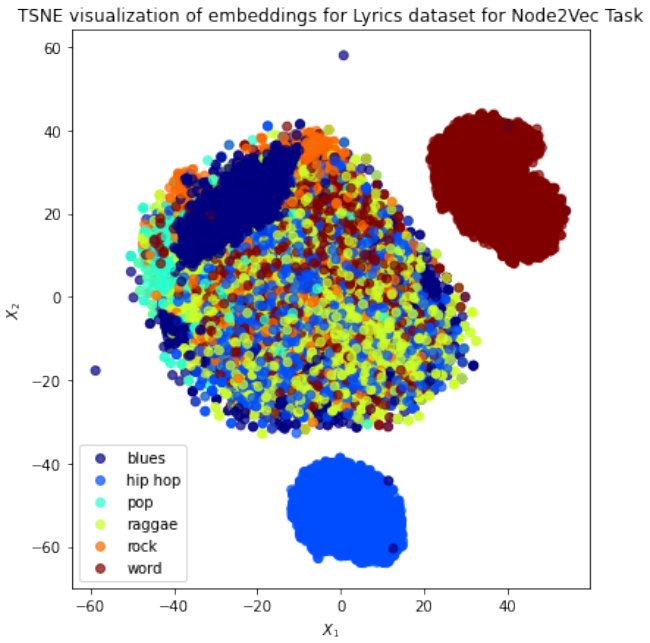

Figure 5. T-SNE Node2Vec Task

Model	Accuracy
Baseline(Logistic Regression)	0.372114
GCN	0.764213
Node2Vec	0.727784

Table 1. Comparison table of the different models' accuracy

7.4 Hyperparameter Tuning

In order to find the optimal parameters for our models, we experimented with different combinations.

Concerning the random walks, we attempted the parameters listed in Table 2. The values of p and q in the table take common values, similar to the ones in [8].

p	q	Accuracy
0.25	2.00	0.698819
0.5	2.00	0.727784
1.00	0.50	0.714567
1.00	1.00	0.709223
1.00	2.00	0.715973

Table 2. Accuracy for Random Walk hyperparameters

As to the window size s of *Word2Vec*, we experimented with the values listed in 3. We discovered that $s = 5$ was optimal.

window size	Accuracy
2	0.702755
3	0.705849
5	0.727784
10	0.677446

Table 3. Accuracy for window size hyperparameters

Regarding the layer sizes of *GCN* we have experimented with the configurations listed in 4. We discovered that using 128X32 is the most optimal for our case.

First Layer Size	Second Layer Size	Accuracy
32	32	0.7689
32	64	0.7528
32	128	0.7597
64	32	0.7650
64	64	0.7654
64	128	0.7522
128	32	0.7726
128	64	0.7623
128	128	0.7581

Table 4. Accuracy for GCN With Different Layer sizes

Concerning the Activation functions for the layers of *GCN* we have experimented with the configurations listed in 5. We discovered that using elu, relu with 128X32 layer sizes is the most optimal for our case.

Activation Layer 1	Activation Layer 2	Accuracy
relu	relu	0.7581
relu	elu	0.7626
elu	elu	0.7804
elu	relu	0.7816

Table 5. Accuracy for GCN With Different Activation Functions for Layer Sizes 128, 32

8 Limitations

8.1 Lyrical Data

Song classification based only on lyrical data is challenging to achieve reliably. The connections between the genre and the context of the lyrics are not unique, as similar lyrics can be used in many different genres. Therefore, a massive data set is required for more accurate results.

In order to bypass that limit, we merged some genres, as mentioned in the previous chapters. However, in an actual world application, the models will require a lot more data to classify more genres.

8.2 Song Language

Our dataset is cleaned by filtering out all non-English songs and then stemming all the words in the lyrics. This process is vital for the overall performance of the models, as it makes the words more unique, thus allowing for more concrete assumptions of the models.

However, if the dataset needs to contain songs with more than one language, the stemming algorithm will be far more challenging to implement, as each (natural) language has its own rules for finding word roots.

9 Improvements

9.1 Data Importance

However, if more data were available containing songs and their lyrics in the initial dataset, the model's performance could have been greatly improved over a more extensive set of genres. Therefore, the model would perform better on a larger dataset.

If other essential data are used in the analysis, for example, the duration of the song, the model's overall performance will be improved.

9.2 Model Improvements

Our model can be improved by using superior vectorizers such as *BERT* [3], *Doc2Vec* [12] or *ELMO* [18] as attribute of graph. Furthermore, adding the frequencies of the words in songs as edge weights may increase the performance of the model. Another approach would be to analyze a small audio sample alongside the lyrics in order to classify based on both lyrical and audio data, similar to [2].

10 Conclusion

Songs' identification has become a trendy topic in recent years. Many music-based companies develop recommendation algorithms to give a better experience to their users. Unfortunately, genres recognition algorithms based on lyrics are not yet efficient. By introducing two different graph models, we managed to identify better the songs' genres based on their lyrics. To experiment with our models, we applied two different methods, namely, *GCN* and *Node2Vec*. Eventually, after tuning the parameters of the different methods, we found their most optimal values. We discovered that the *GCN* method performs better than the *Node2Vec* in classifying the songs genres. However, the *Node2Vec* resulted in a better *T-SNE* representation of the genres of songs.

References

[1] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 1335–1349. https://doi.org/10.1145/3318464.3389742

[2] Tyler Dammann and Kevin P. Haugh. 2017. Genre Classification of Spotify Songs using Lyrics , Audio Previews , and Album Artwork.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[4] Dr. Arijit Ghosal, Rudrasis Chakraborty, Bibhas Dhara, and Sanjoy Saha. 2013. Genre Based Classification of Song Using Perceptual Features. *Advances in Intelligent Systems and Computing* 243. https://doi.org/10.1007/978-81-322-1665-0_26

[5] googletrans 2021. Googletrans Google Translate API. https://pypi.org/project/googletrans/.

[6] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 855–864. https://doi.org/10.1145/2939672.2939754

[7] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 855–864. https://doi.org/10.1145/2939672.2939754

[8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. *CoRR* abs/1607.00653 (2016). arXiv:1607.00653 http://arxiv.org/abs/1607.00653

[9] Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng WANG. 2019. Text level graph neural network for text classification. https://arxiv.org/abs/1910.02356

[10] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[11] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph Convolutional Networks. https://arxiv.org/abs/1609.02907

[12] Jey Han Lau and Timothy Baldwin. 2016. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. *CoRR* abs/1607.05368 (2016). arXiv:1607.05368 http://arxiv.org/abs/1607.05368

[13] Cory McKay, John Burgoyne, Jason Hockman, Jordan Smith, Gabriel Vigliensoni, and Ichiro Fujinaga. 2010. Evaluating the Genre Classification Performance of Lyrical Features Relative to Audio, Symbolic and Cultural Features. *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010*, 213–218.

[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL]

[15] Emanuel; Sampaio Vinicius; França Mardônio Moura, Luan; Fontelles. 2020. Music Dataset: Lyrics and Metadata from 1950 to 2019. *Mendeley Data* 2 (2020). https://doi.org/10.17632/3t9vbwxgr5.2

[16] nltk 2021. NLTK. https://www.nltk.org/.

[17] Akin Parkan, Boris Rokanov, Georgi Kostov, Ivan Germanov, and Jeffrey Chan. 2021. Music genre recognition based on lyrics. https://github.com/ivangermanov/graph-mining.

[18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR* abs/1802.05365 (2018). arXiv:1802.05365 http://arxiv.org/abs/1802.05365

[19] M.F. Porter. 1980. An algorithm for suffix stripping. *Program: electronic library and information systems* 14, 3 (1980), 130–137. https://doi.org/10.1108/eb046814

[20] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80.

[21] stellargraph 2021. StellarGraph. https://stellargraph.readthedocs.io/en/stable/README.html.

[22] Alexandros Tsaptsinos. 2017. Lyrics-based music genre classification using a hierarchical attention network. https://arxiv.org/abs/1707.04678

[23] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[24] Wengie Wang and Yezhou Li. 2019. Music Genre Classification Based on Lyrics. https://github.com/jwang862/Music-Genre-Classification-Based-on-Lyrics

[25] Liang Yao, Chengsheng Mao, and Yuan Luo. 2018. Graph convolutional networks for text classification. https://arxiv.org/abs/1809.05679