Music Genre Classification using Song Lyrics

Christina Gregis

Abstract

With the widespread availability of music available online, we are constantly bombarded with song recommendations catered to our interests. Music streaming services, such as Spotify, iTunes, and Youtube Music, rely upon user-based information, such as the songs a user has listened to in the past, in order to suggest songs a user may like. Streaming services often rely upon user-inputed data to classify songs and generate recommendations that music listeners may enjoy. While streaming services can make quick and accurate predictions, these approaches heavily depend on user-collected data rather than the actual song content. This also raises privacy concern. Often, users do not know if their personal information, obtained from these services, is sold to other companies or not. This leads music listeners to question: Is user data really necessary for song recommendations? Is there something inherent in music content itself that can make accurate genre classifications? This project will explore these questions by analyzing whether lyrics can predict a song's genre. Through various RNN models (including simple RNN, GRU, and LSTM), this project hopes to prove that accurate song classifications without using user-specific information. This project will also revisit the simple RNN model at the end of this analysis to see whether or not balancing the genres significantly affects classification results. Hopefully future researchers can use this non-user based approach to develop more complex models using music features, such as tempo and rhythm, in addition to lyrics, as an alternative classification method. .

1. Introduction

With the plethora of songs available today, it is difficult for music experts to manually classify songs on music-streaming platforms. Many streaming services have turned automatic classification methods to conveniently sift through songs and create recommendations music listeners may enjoy. For many music listeners, it is easy to guess what genre a song if someone where to simply say a song-lyrics aloud. For this project, I will analyze whether deep learning methods, like RNNs, can match this intuition.

Motivation: For this project I was interested in exploring Recurrent Neural Network (RNN) models. Intuitively, a person listening to a song for the first time may not remember every single lyric an artist sung. Most likely, the music listener remembered the most meaningful words an artist sung. [7] RNN models match this intuition using "feedback connections" that help keep track of lyrics that came before a particular lyric that is fed into the network. [6] As Andrej Karpathy, Tesla's AI director - phrases it, "there's something magical about RNNs" because these models try to capture repeated information in ways traditional neural networks fail to accomplish. [2] RNN models can map sequences, like song-lyrics, to generate outputs, like genre tags. Karpathy's enthusiasm greatly inspired me to implement several different RNN models to discover on my own the magic RNN models offer. Through this project, I hope to explore, through RNN models, how song lyrics can show how each song may fit into a specific music genre.

2. Related Work

Deep learning methods, such as Convolutional Neural Networks, to classify songs based on audio recognition.[16] Many of these CNN methods extract audio features and represent them as a spectrogram and treat audio features as an image classification task. [5] These classification methods typically focus on a song's rhythm, beat, and chord progression. Methods analyzing chord progression concentrate on how a song's frequencies fluctuates from chord to chord. However, many of these classification methods focus more on how a song sounds rather than how a song is written. Some researchers have tried combining CNNs and RNNs, using Paralleling Recurrent and Convolutional Neural Network (PRCNN), as an attempt to map both the spatial, audio features - as traditional CNN models do - while preserving the sequence of words, as RNN models accomplish.[4]

Many researchers have approached music classification methods focus on visualizing a song's lyrics rather than analyzing its text. While Eck et al.[3] used LSTM models to analyze long-term dependencies among chord progressions, further research is still needed to analyze how other music features map to one another. In sum, most researchers have focused on the way a song "sounds." This project is motivated on analyzing whether a the way a song is written is also important for music classification.

3. Proposed Method

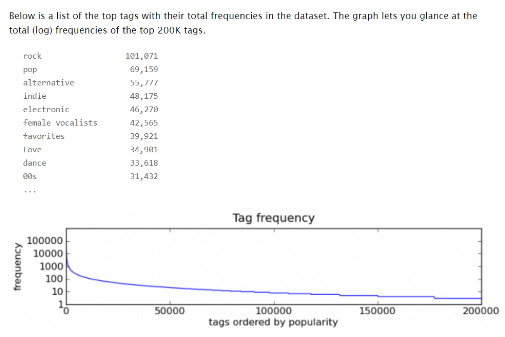
This project began with quite a rough sketch of the models I wanted to implement. Originally, I intended to implement a simple RNN model, given the complexity of natural language processing. By focusing on RNNs, I hoped to analyze a song's lyrics sequence, rather than using a method that solely cares about word frequency. Before embarking on this project, it was decided to create a model using song that had both genre labels and lyrics, in order to avoid errors in guessing in manually filling in incomplete data. However, in the final dataset used, only 220,000 of the original 1,000,000 songs had both lyrics and genre tags available.

While designing this experiment, only English songs were analyzed to streamline our analysis. This prevented translated words - such as "love" in English and its equivalent, "amor," in Spanish - from being counted as separate words. All words were also altered into lower-case to prevent capitalized and lowercase words - such as "go" and "Go" - from being marked as distinct words. These cleaning methods ensured that potential mis-classifications would not confound each model's performance - such as if I wrongly predicted a song's genre that originally contained a missing tag or if I mis-translated lyrics from one language to another. In addition, all punctuation and escape characters were removed from each lyrics, and the remaining words were separated by a single space. Stop-words were further removed, using Python's Natural Language Toolkit package (NLTK). By removing punctuation, escape characters, and stop-word from the data set, it helped differentiate each genre from one another by having less filler words and extraneous symbols that all genres may share. After the cleaning process was complete, 77,885 songs remained for analysis. This is still a smaller song subset compared to the original dataset of 220,000 songs with both tags and lyrics.

Music Genres: The genres selected based on the Last.Fm's top-ten list of genre, as shown in Figure 1. Initially, I planned to use just the top-ten The chosen genres were the following: Rock, Pop, Indie, Alternative, and Dance. The remaining songs were marked as "other" for analysis.

The performance metric for evaluation is the testing accuracy. A model was deemed successful if it had an accuracy of 80 percent or above. Testing accuracy can be used to estimate the generalize error of a model. The classification accuracy can be summarized as follows:

$$\text{Accuracy} \equiv \left(\frac{Number of Correct Predictions}{Total Amount of Pred}\right)$$



Below is a list of the top tags with their total frequencies in the dataset. The graph lets you glance at the total (log) frequencies of the top 200K tags.

rock	101,071
pop	69,159
alternative	55,777
indie	48,175
electronic	46,270
female vocalists	42,565
favorites	39,921
Love	34,901
dance	33,618
00s	31,432
...	

Figure 1. The graph above is provided by Last.Fm. and includes the top ten genres of the entire Million Song Dataset. The five analyzed genres for this project were selected using this graph.

4. Experiments

This project uses several Recurrent Neural Networks (RNN) models, using simple (aka vanilla) RNN, Long-Short-Term Memory (LSTM) and and Gated Recurrent Unit (GRU) architecture. Each model maps our input data - song lyrics - into a single genre label, using the "many-to-one" approach.[10]

Each analyzed, imbalanced RNN model involved the following implementations:

In order to analyze each song lyric, the lyrics first needed to be tokenized, or separated from their lyrics string. This allowed each word within a song to be counted in order to calculate the most frequent words. This allowed each word to be easily accessed. The max-song length (354), but was manually cut-off at 200, since most lyrics were quite shorter than dataset's actual max length. Unknown words marked. Then, a dictionary was created with the 25,000 most frequent words that appeared throughout the dataset; each key word was assigned to a particular value.

Next, the data was shuffled in order to randomize song selection within our models. Then, the data was randomly shuffled and split, with 50 percent used for training and 50 percent used for testing sets. The training set was further split so that 30 percent could be used for a validation sets. Each file was saved as a data-file to JSON file. The following steps closely follow Ben Trevett's Torch Text tutorial [15] and multi-class sentiment analysis [14] examples:

Next, data.fields were created for lyrics (denoted "TEXT") and its numeric genre tag (denoted "LABEL"). These pre-built functions allowed the creation of a unique vocabulary mapping that assigned lyrics to a numeric genre label. This format mostly ensured that the data could be passed into the RNN architecture, which heavily relies on these text-processing formats in order to easily read in the data. Perhaps this is because RNN models involve tedious text-processing to implement! Altering the RNN architectures from simple RNN to LSTM or GRU was easy and involved a one-line change, thanks to a pre-built Pytorch model, obtained using "torch.nn" package. The data-processing steps stated above are illustrated Figure 2. [8]

Balanced Model: For the balanced RNN model, I followed all of the steps above, except I only used 1,600 songs from each genre instead of simply using all of the songs. The balanced model had a total of 9,600 songs; this was approximately one-eighth of the total songs analyzed in the other models! Thus, the balanced case involved the creation of smaller, training/testing/validation sets to reflect this size differences.

Unless stated otherwise, each model was run using 5 epochs, has a mini-batch size of 64, uses an Adam optimizer, and has a learning rate of 0.001. By generally using the same hyperparameters, it makes it easier to draw comparisons to see which implemented RNN architecture best classifies music genres. All models had a random seed set to 123 in order to make reproducible results.

4.1. Lyrics Exploration

For the entire (imbalanced) dataset, the top ten most-frequent words were determined to be the following (written in lowercase to reflect its actual representation in the cleaned dataset):

Rock: i'm, know, love, like, got, oh, get, one, time, see
Pop: i'm, know, love, like, oh, baby, go, get, got, time
Indie: i'm, know, like, oh, love, never, time, see, go, got
Alternative: i'm, like, know, got, get, love, never, go, one, see
Dance: i'm, love, get, know, got, like, baby, oh, $\underline{\text{yeah}}$, go
Other: i'm, like, love, know, got, get, one, go, time, see

As shown above, not many of the top ten frequent words are unique to each genre- the only exception to this rule is the word "yeah", for Dance songs. As shown in Figure 3, many of these top ten words appear a disproportionate amount of times compared to other lyrics within the MSD. Despite this repetition, we can still use song lyrics to capture some of the variation songs have among genres. Also, it was discovered that if the dataset was balanced (see *Model Overview*!), the top ten words per category did not substantially vary and reflect the results shown above for the balanced data. This conclusion highlights a possible fault in this experiment - the choice of similar categories. Ideally, selecting genres that are quite different from one another (such as metal vs. country music) would have perhaps led to more interesting results when comparing lyrics among genres.

4.2. Dataset

This project uses song lyrics (scraped from songlyrics.com) paired with the Million Song Dataset (MSD). [1] The corresponding genre labels were obtained from the Last.Fm subset, featuring songs released up to 2012. The lyrics and tags were combined in a final dataset (containing 220,000 complete songs) under a single datafile, courtesy of Professor Raschka. Each song contained multiple genre labels that had been determined from Last.Fm listeners. To avoid multiple classifications of the same song, each song was reassigned a single genre based on the top five-occuring genres within the final dataset. As shown in Figure 4, most songs are classified as *rock*.

4.3. Software

Python NLTK package was extremely helpful for removing stop-words in text-processing. For visualizations, I used Python's Matplot to create the bar graph of the music genres (Figure 3), and I used Python's WordCloud to generate the image of the most frequent words that appear throughout the dataset (Figure 2). The other main package I used was Pytorch's torch.nn's built-in RNN, GRU, and LSTM architectures.

4.4. Hardware

Each model was run using Google Colab's GPU. The recorded epoch times (see *Results and Discussion!*) reflect how quickly a model ran using this hardware.

5. Results and Discussion

5.1. Revised simple RNN):

2,593,490 trainable parameters

Epoch 1: Training Accuracy = 74.02%
Epoch 1: Validation Accuracy = 74.83%
Epoch 5: Training Accuracy = 73.39%
Epoch 5: Validation Accuracy = 74.78%
Final Testing Accuracy: 74.38%

5.2. GRU:

I also implemented a GRU model as my first endeavor to see whether different RNN architectures would significantly impact the testing accuracy. GRU models attempt use a "update" and "reset" gates that helps remember the important song-lyrics when determining a song's genre **??**.These gates help prevent the vanishing-exploding gradient problems that many other RNN models experience. The results are displayed below:

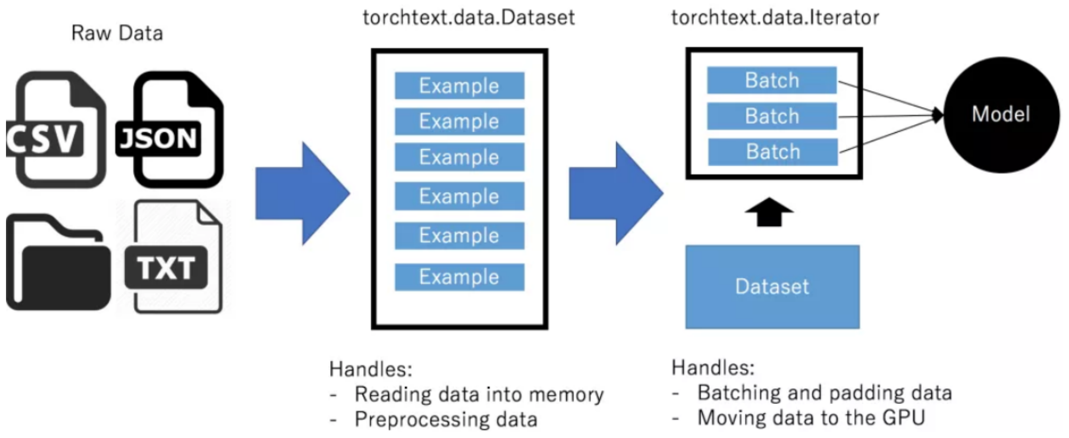


Figure 2. These are the top 100 words within the entire dataset. The proportion of the lyric captures how frequent it appears compared to all lyrics.

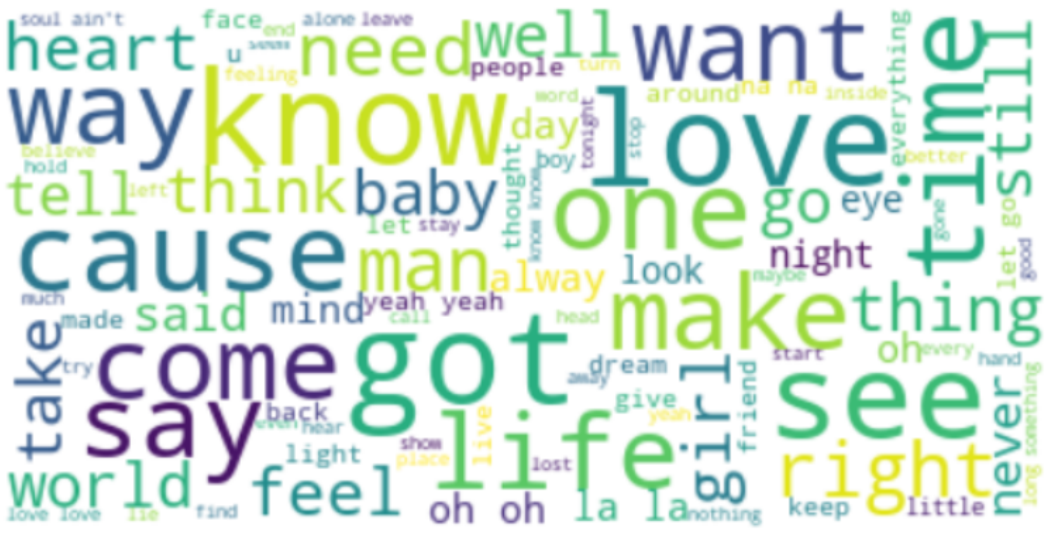


Figure 3. These are the top 100 words within the entire dataset. The proportion of the lyric captures how frequent it appears compared to all lyrics.

2,640,686 trainable parameters

Epoch 1: Training Accuracy = 73.95%
Epoch 1: Validation Accuracy = 74.83%
Epoch 5: Training Accuracy = 76.75%
Epoch 5: Validation Accuracy = 72.65%
Final Testing Accuracy: 71.99%

5.3. LSTM:

In essence, the LSTM model feeds the song's lyrics into an input cell that can remember the previous "time step," e.g. the lyric in a song before another lyric. The model has a "forget" gate, which discards the lyrics that are not deemed important for genre classification. The remaining lyrics are used to predict a song's class as the final output of our model. [11] Compared to the previous models, the LSTM architecture was the fastest to run, with an average of 5 mins per epoch; this was one minute quicker than the GRU's average time per epoch, but was 2.5 times as fast as the simple RNN model's run-time! The results are shown below:
2,640,686 trainable parameters

Epoch 1: Training Accuracy = 82.23%
Epoch 1: Validation Accuracy = 69.77%
Epoch 5: Training Accuracy = 98.30%

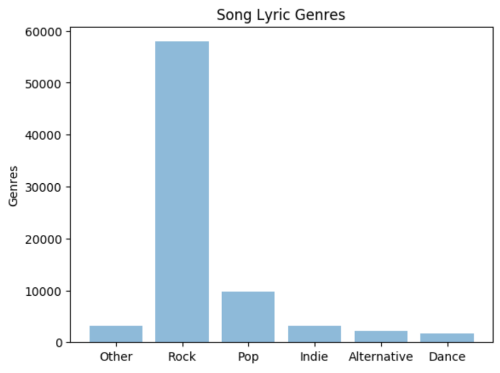


Figure 4. The above figure demonstrates how imbalanced the dataset is. There are a disproportinate amount of rock songs compared to all other genres.

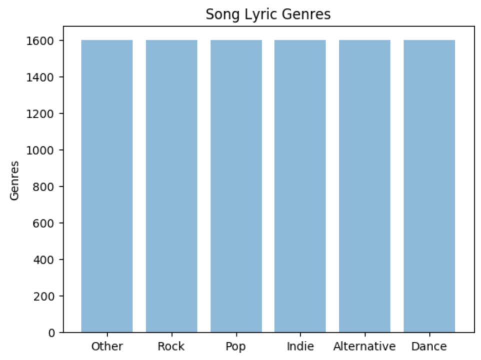


Figure 5. The graph above represents the balanced dataset. All song genres appear an equal amount of times.

Epoch 5: Validation Accuracy = 64.47%
Final Testing Accuracy: 63.64%

I also tried balancing the data to account for the high amount of rock songs.

5.4. Simple RNN (Balanced case):

For the balanced case, 1600 songs per genres were included, as shown in Figure 5. This is approximately the amount of songs in the lowest category, Dance (which had 1677 songs). By using a balanced case, I hoped to learn whether the disproportionate amount of rock songs affected the test accuracy in the models shown above.

For five epochs, the balanced RNN's the epoch time was approximately 50 secs - this is 12 mins faster than the original imbalanced simple RNN' epoch time! The prediction results are shown below:
2,593,490 trainable parameters

Epoch 1: Training Accuracy = 17.78%
Epoch 1: Validation Accuracy = 19.43%
Epoch 5: Training Accuracy = 20.11%
Epoch 5: Validation Accuracy = 17.87%
Final Testing Accuracy: 18.71%

Unlike the imbalanced simple RNN model, the training, validation, and testing accuracies highly varied - in some epochs, this difference was about 20 percent between training and validation accuracies! As a result, the balanced model had a low test accuracy of 18.71%. Another test was completed for 25 epochs, since the dataset was relatively small and easy to iterate through compared to the other attempted models with an imbalanced amount of genres:
2,593,490 trainable parameters

Epoch 1: Training Accuracy = 17.69%
Epoch 1: Validation Accuracy = 21.13%
Epoch 5: Training Accuracy = 68.04%
Epoch 5: Validation Accuracy = 17.19%
Final Testing Accuracy: 18.02%

Despite expectations, using more epochs did not led to a higher testing accuracy. For 25 epochs, however, there was approximately a 50 percent gap between the training and validation accuracy; it is unsurprising that the testing accuracy was quite low: 18.02%

The balanced case demonstrates that the imbalanced data does not significantly contribute to the overfitting all analyzed models demonstrate, with the exception of the simple balanced RNN model. While the dataset was relatively small for the balanced case (only 9,600 songs!), the model's performance worsened when more epochs were performed. By having an equal amount of songs per category, the low accuracies highlight flaws within this project's overall design. Perhaps limiting the project to only five genres led to its low performance, given how similar the top-most occurring lyrics seem to be among the genres. Perhaps more variables are essential to make better classification predictions, given the five genres selected. Alternatively, the low performance may indicate that this project should have evaluated genres that are more different from one another - such as metal vs. electronic music. By choosing a design with dissimilar genres, perhaps song lyrics would have varied more per genre and thus would have made more meaningful comparisons among genres.

Final Results: In sum, the best performing model based on test-accuracy is the simple RNN model.While the LSTM model had the highest training accuracy among all models, overfitting issues contributed to its low testing performance compared to the simple RNN model. None of these models achieved the desired testing accuracy of 80%, although

Model	Testing Accuracy
$SimpleRNN$	74.83%
GRU :	71.99%
$LSTM$	63.64%
$BalancedRNN(epochs = 5)$	18.71%
$BalancedRNN(epochs = 25)$	18.02%

Table 1. Model Comparison.

the LSTM model achieved this accuracy in training. Perhaps if this experiment was redesigned (see *Conclusions* section!), the LSTM model possibly would have been the "best" model after all for classifying genres based solely on lyrics.

6. Conclusions

While each RNN model had relatively high accuracies, according to the performance metric previously set, each model can definitely be improved upon in the future. Our data-cleaning methods. The songs were limited to only English songs; this was approximately a third of the songs within the cleaned dataset! While limiting analysis to a single language ensured that none of the lyrics would be inaccurate from mistranslations, this approach still marginalized the results to only English songs. Perhaps pop and rock lyrics written in another language, such as French, would have been more varied than they were for the English songs analyzed within this project. In addition, it is unclear how the Million Song Dataset was collected. Perhaps there was a bias in song-selection and that the songs were not randomized when creating the dataset. Potential bias within the dataset itself could also explain why many of the lyrics were similar to one another among different genres.

Genres: When classifying songs, many songs were pooled together as "other" that did not belong into the top five-categories. This means that music genres, such as "classical music," "punk," and "folk music" are analyzed as similar songs in all models! Music connoisseurs would probably cringe at how genres with quite different lyrics from one another are treated as the same type of music in this project. It also would have been beneficial if the dataset had genre labels determined by music experts. Realistically, this is not feasible, and the dataset does contain labels that have been determined by multiple listeners to classify each song's genre. User determined tags did also result in messy genre tags; some songs were labeled as "pop," "rock," and "indie," which are three distinct categories this project used for classification! As a remedy, the top-most frequent genre tag was used to classify overlapping genres. This method ensured that each song could be classified into a single genre and helped eliminate bias within each model.

Text Processing: While cleaning the lyrics, verbs in different tenses, such as "has" and "have" - were not combined. For instance, among the top most frequent words for all genres includes the word "get" and "got", which are essentially the same word but simply conjugated differently. In the future, it would be best to change all words into a single tense (e.g. change past-tense words like *had* into *has*) to avoid repetition of words with the same meaning. Having words in a single-tense could have altered the top-frequent overall words to help distinguish the five main genres from one another. Symbols, such as an exclamation mark, were also removed but perhaps would have conveyed useful sentiment for classification. The elimination of all symbols, however, did help make lyrics a more powerful predictor by looking at the words within each song itself and prevented repeated punctuation marks, like commas and periods, from skewing the classification results. Nevertheless, all of the limitations stated above provide guidelines of how to improve music classification projects in the future.

Overall: In general, the dataset was highly imbalanced; approximately 75% of our data consisted of rock songs! This created an unfair representation of the genres within the data. In further research, I would investigate adding momentum terms and altering learning rates to see if any of my model's generalized performance would significantly improve. [9] In the future, I would also implement *doc2vec* to map each lyric onto a vector and then use *T-SNE* to map the different genres. While time was limited, this method would have given a clearer representation of how different the lyrics are.[13] With the exception of the simple, imbalanced RNN model, each model had a relatively high training accuracy compared to the validation and testing accuracy. These training results show some hope for using lyrics as a classification basis. With more data and varied genres, this project demonstrates that you can make fairly decent genre predictions by using music content itself.

7. Acknowledgements

The datafile with song lyrics and genre tags was provided by my professor, Sebastian Raschka (*see Data Overview subsection!*) My previous group member, Wengie Wang, had cleaned the genre labels so each song lyrics had one corresponding genre name. She also created a separate column in the datafile that helped map each genre label to a numeric value (e.g. rock = 0, pop = 1, etc) that immensely helped with creating dictionaries for my RNN models. I used a Python Pandas blog, by data scientist Shane Lynn, as a guideline for text cleaning and visualization tips.[12] Otherwise, the architecture for all of my models closely follows text-processing examples by Ben Trevett, a Machine Learning PHD candidate at Heriot Watt University (see Experiment section!)

8. Contributions

See Acknowledgements!

References

[1] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[2] Colah. Understanding lstm networks.

[3] D. Eck. A first look at music composition using lstm recurrent neural networks. 2002.

[4] L. Feng, S. lan Liu, and J. Yao. Music genre classification with paralleling recurrent convolutional neural network. *CoRR*, abs/1712.08370, 2017.

[5] A. Foroughmand Arabi and G. Lu. Enhanced polyphonic music genre classification using high level features. pages 101 – 106, 12 2009.

[6] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] A. Karparthy. The unreasonable effectiveness of recurrent neural networks.

[8] K. Kurita. A comprehensive introduction to torchtext (practical torchtext part 1), Mar 2018.

[9] S. Raschka. Machine learning faq.

[10] S. Raschka. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow*. Packt Publishing, 2017.

[11] D. Seita. When recurrent models don't need to be recurrent.

[12] L. Shane. The pandas dataframe – loading, editing, and viewing data in python, Dec 2017.

[13] G. Shperber. A gentle introduction to doc2vec, Jul 2017.

[14] B. Trevett. Multi-class sentiment analysis.

[15] B. Trevett. Using torchtext with your own datasets.

[16] W. Zhang, W. Lei, X. Xu, and X. Xing. Improved music genre classification with convolutional neural networks. 2016.