Lyrics Classification with Verse-Wise Averaging Deep Network

Mor Buchnik, Ben Massarano

Abstract

Music genres are categories created by humans to classify songs and are an integral part of many music applications. Genre boundaries are often vague and overlap, making classification to a single genre impossible in some cases. The amount of music on the web is too large to be labeled by professionals and increases rapidly, so machine learning models are required for the task. While most other research attempts to solve this task with audio and its related features, in this paper, we aspire to solve the genre classification problem using the song lyrics only. We present a comparison between several models, the first being our implementation of Deep Averaging Network (DAN), the second our adaptation on DAN which takes the hierarchical structure of songs into account; we compare them against two state-of-the-art text-to-text models. While audio classification has the GTZAN benchmark, we had to build our own lyrics dataset. The complete dataset has 20K songs of 7 different genres, and our experiments were done on different genre subsets of it. Our model reached 34% accuracy on the complete dataset (45% by $BERT_{BASE}$) that has overlapping genres in it. When limiting the prediction to songs from one of Hip hop, Blues, Jazz we achieved 66% (75% by $BERT_{BASE}$).

1. Introduction

Classifying music into genres is a crucial task for many music related services, recommendation systems for example. The ever-growing music industry calls for intelligent tools for labeling songs. Genre classification of songs is also one of the most researched tasks in Music Information Retrieval (MIR) [1], as well as a classic natural language processing (NLP) task. To classify a song as a specific genre, one must capture the song's structure and its essence. We attempt to predict the genre of a song solely from its lyrics. This is a hard task even for a human, since the same song can be labeled as different genres when its melodies and tempos are changed. Moreover, genre borderlines are not clear, one song can belong to more than one.

When classifying songs, there are many relevant components. Previous MIR work was based on machine learning algorithms which often used features extracted from the audio[2], lyrics, and cultural data of the song. In particular, the audio contains lots of useful information- volume and frequencies, the instruments and the tone of the singer. NLP aims to understand passages of texts while assigning meaning and labels to words, so it is reasonable to assume that a state-of-the-art text-to-text model will be able to classify a song using its lyrics with relative success.

For lyrics classification, one needs to analyze long texts, in varying lengths. One approach to simplify a long text is by averaging the input words[3] and continue to process the fixed length average. Another is using Gated Recurrent Unit RNN[4] (GRU), intended for training the same model with different text lengths. We used both methods in our model, and compare it to averaging alone, and to several other state-of-the-art models. The model we present also utilizes the hierarchical nature of songs- a song is logically built from verses, that are constructed from lines which are made of words. Using the hierarchical structure of text has been done before[5].

We present a comparison between several models: DAN implemented as stated in the original paper, "verse-wise DAN" that average each verse on its own, then moves the verse representations through a GRU and linear layers, *Bidirection*al Encoder Representations from Transformers[6] (BERT) and T5 (used as-is). To batch the different samples, we pad the shorter ones and mask the padding. On the verse-wise DAN we pad on two levels- pad the verse to fixed word amount and pad the songs to a fixed verse amount. The DAN variations both start by embedding the text with GloVe[7] word embeddings to represent each word.

Finding a suitable lyrics dataset was discovered to be a difficult task. Therefore, a big part of our work consists of building our own database and preprocessing the data. The database includes lyrics and titles of over 20 thousand songs from 7 different genres. For each song we save its lyrics, title, and the genre it belongs to.

2. Prior work

Previous MIR work has focused on classifying genre[14], alongside mood[15] and annotations[16]. Since the amount of music in the web increase rapidly, genre classification can no longer be done manually. In this work we will showcase different NLP models that use lyrics as the sole representation of a song and compare their results. McKay et al.[17] showed that lyrical data performance is weaker compared to other forms of data when trying to classify genre.

The related research consists of mostly audio[8][9] files classification. Classifying audio is often done using spectrograms. Convolutional Neural Network (CNN) are widely used to classify images[11], and after Li et al. [10] declared that musical patterns with certain transformations are similar to images, CNNs were used to classify music as well[12].

Since lyrics have a fixed structure- words combine to form lines, lines form verses, and verses form a complete song, it can be explicitly used during training. The general term for this is hierarchical methods[19]. One hierarchical model is the hierarchical attention network[5] (HAN), that has been trained to classify documents. The main idea is to first represent each sentence using the low-level word representations, and then classify the document using the higher-level representation. HANs can be modified to classify lyrics[18]. In his paper, A. Tsaptsinos trained a HAN that calculates line embeddings from the word embeddings, and lyrics embeddings from the line ones. Both the sentence and the full lyrics embeddings are gathered from RNNs, when the first is Bi-Directional. RNN's advantage in relation to our task is the ability to process different length of samples. Another way to achieve this is by averaging. Deep averaging network[3] (DAN) embed a varying length-embedded masked input by averaging it. The output is of fixed length and can be ran farther through the network.

In our model, we combined those concepts- dividing the lyrics into verses and embedding each verse using averaging. The verse representations is then ran through a GRU to get a complete representation. We compare this model against a DAN, Text-to-Text Transfer Transformer[20] (T5) and Bidirectional Encoder Representations from Transformers[21] (BERT). The last two are text-to-text state-of-the-art models.

3. Data

3.1 Dataset

At firs, we searched for an existing dataset but found no appropriate corpus for song lyrics and the corresponding genres. We found some datasets with audio files but not many of them contained the lyrics in addition to the songs' genres. In particular, we encountered the GTZAN[24] benchmark for audio classification. We encountered some relevant APIs, but we couldn't use them since they only provided partial lyrics or their request-per-dat limit was too small. After examining 10+ APIs and datasets (i.e. Shazam[22] and Spotify[23] APIs, as well as Huggingface datasets) we decided to collect our own data. As was done in class, we decided to store the data as a Dataset object. At first, we used "letssingit"[25] website as our main data source for the songs. But as songs began to gather, we discovered the genre category is missing for lots of songs and decided to try a different approach. Another failed attempt was to cross data from different sites. That is, we used two websites, the first providing title/lyrics and the second title/genre. Finally, we found "lyrics"[26] website. This website contains over 1M lyrics of over 10 genres, and each song is attached to its matching genre; just as we wanted. In practice, the website quality control was limited at best and we could only use a fraction of those songs. There are several reasons why: First, when a song had more than one genre, the same song would appear once for each of them. We only wanted songs that has exactly one genre. Second, there were often different songs with the same title, and we wished to have only one lyrics per title in our dataset. Third, lots of songs weren't in English at all or had non-standard characters in them. We had to preprocess the raw lyrics before using them or else we would have had excessive noise in our data. All of the songs we used in our models are taken from this website.

3.2 Preprocess

As in every dataset creation, preprocessing was the only way to adjust the raw data for training on it. Since the task at hand is classifying a song to a single genre, we had to discard of songs with more than one genre in the site. We also followed some heuristics about what "real lyrics" looks like. We only accepted songs in which: the chars are only letters or several hand-picked ascii chars, the verses are separated with newline, they are purely in English and are not one verse long. We also replaced keywords like "CHORUS" with the actual chorus of the song, and same goes for "repeat x3", so the song will contain nothing but the lyrics as a human will interpret them. Many songs from the site couldn’t match these criterions, so we discarded them.

Final label-distribution in the dataset:

Total songs: 20406
Hip hop songs: 4.45% (910)
Blues songs: 6.22% (1271)
Pop songs: 12.7% (2593)
Funk songs: 18.2% (3732)
Jazz songs: 7.51% (1533)
Rock songs: 23.9% (4880)
Electronic songs: 26.8% (5487)

Since smaller genres were difficult to get a hand on, we couldn't use all our songs at once. When training on the full, non-uniform dataset, our models ignored the smaller genres completely. This constraint drastically shrinks our dataset. For the experiments we used several subsets of the dataset. This allowed us to use more of the songs of each genre when classifying between the big genres. For example, when using all 7 genres we only use 910 songs per genre, but when using the subset "Electronic, Rock, Funk" we use 3732 songs per genre.

In order to batch samples, we had to pad the songs to equal length. We did it by using zero padding and attention masks.

4. Models

4.1 DAN

Deep Averaging Networks (DANs) were first introduced in 2015 and were originally used for sentiment analysis. DANs work in the following way:

0. embed the input sequence.
1. average the embeddings.
2. pass that average through one or more feedforward layers.
3. Predict label using the final layer’s value.

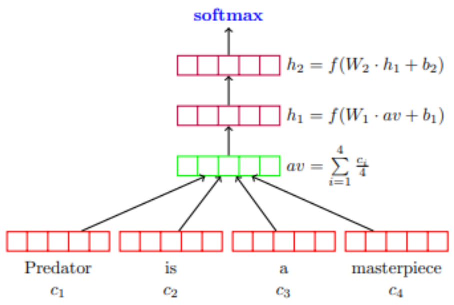
softmax
$h_2 = f(W_2 \cdot h_1 + b_2)$
$h_1 = f(W_1 \cdot av + b_1)$
$av = \sum_{i=1}^{4} \frac{c_i}{4}$
Predator
c_1
is
c_2
a
c_3
masterpiece
c_4

Figure 1: DAN arfhitecture

One unique trait of DAN is its' capability to transform relatively long passage of text to a short representation. Since songs' lyrics are long (often surpass three-hundred words), conciseness is crucial.

In one of the assignments, we trained a DAN to classify IMDB movie reviews to "positive" and "negative". We used that implementation as a prototype and modified it to suit our task. We embedded each song using the GloVe embeddings[6], then padded the songs to equal lengths so we can batch them together. We kept track of when the padding occurred to make the averaging only consider the actual samples. The padding didn't affect the sums since we padded with zeros. We then used dropout to avoid overfit, and trained the DAN described above with two hidden layers and Cross Entropy loss.

4.2 Verse-wise DAN

As humans, we don't see songs as one solid piece, but as a sequence of verses, each captures part of the song essence. Using DAN on the entire song ignores that. We wanted to refine the averaging part of our model, so we averaged each verse independently instead. After doing so, we are left off with "verse-embeddings", which we then feed as input to a Gated Recurrent Unit[4] (GRU) and continue with the linear layers used in the DAN on its' output.

We used a GRU to counter the fact that there is a variant verse amount in different songs. We didn't use a Bidirectional GRU verses only depend on the verses that appeared before.

The GRU is as follows:

$$r_t = \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr})$$
$$z_t = \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz})$$
$$n_t = \tanh(W_{in}x_t + b_{in} + r_t * (W_{hn}h_{(t-1)} + b_{hn}))$$
$$h_t = (1 - z_t) * n_t + z_t * h_{(t-1)}$$

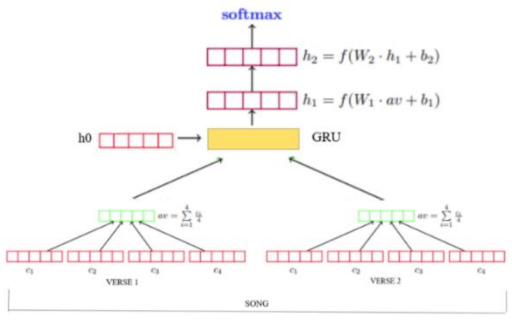
softmax
$h_2 = f(W_2 \cdot h_1 + b_2)$
$h_1 = f(W_1 \cdot av + b_1)$
h0
GRU
$av = \sum_{i=1}^{4} \frac{c_i}{4}$
c_1
c_2
c_3
c_4
VERSE 1
VERSE 2
SONG

Figure 2: Verse-wise DAN arfhitecture

4.3 BERT

BERT is a transformer-based machine learning technique for natural language processing. Context-free models such as GloVe, generate a single word embedding representation for each word in the vocabulary. In contrast, BERT considers the context for each occurrence of a given word. When it was first published it achieved state-of-the-art performance on a number of NL understanding tasks such as: GLUE[27], SQuAD[28], and SWAD[29].

BERT has many models: $BERT_{BASE}$, $BERT_{LARGE}$, $BERT_{TINY}$, $BERT_{MINI}$, $BERT_{SMALL}$ and $BERT_{MEDIUM}$. We used $BERT_{BASE}$, $BERT_{TINY}$ and $BERT_{MINI}$.

4.4 T5

Another model we worked with is T5. It is an encoder-decoder model pre-trained on a mixture of unsupervised and supervised tasks. Among the tasks are machine translation, document summarization, and question answering. Each task is converted into a text-to-text format. We trained T5 on a multi-classification task. Since all other T5 versions were too large to fit with our current resources we fine-tuned $T5_{small}$.

We failed to fine-tune it, even after consulting the course T.A and asking on the huggingface forum. The expected output was the matching genre for each song. Instead, the output was non-genre words and padding tokens. The text predicted was related to the song topic but wasn't the genres we wanted. It seemed like learning where to pad was sufficient for the model to reduce the loss. To tackle this, we even tried to change the loss function from inside the model, but with no success whatsoever. Since we couldn't limit the output tokens to genre names, we gave up on this model.

5. Experiments

5.1 Baseline models

We trained our models on several genre groups. The models used:

1. DAN
2. Verse-wise DAN
3. $BERT_{BASE}$, $BERT_{MINI}$, $BERT_{TINY}$

5.2 BERT

Before comparing against the non-BERT models, we wanted to compare different versions of BERT. With the resources we had, we could use all BERT sizes but $BERT_{LARGE}$, using GPUs. It can be seen clearly in [Table 1] that larger models performed better, so we compared our other models against $BERT_{BASE}$.

The table also confirm our assumption regarding some genres being similar, and even overlapping at times. We can see that all the models performed best on Jazz, Blues and Hip Hop, while performing poorly on Pop and Rock. Possible explanation for this is that Pop and Rock are similar to other genres (Rock, Pop and Funk often have similar lyrics). On the other hand, Jazz, Hip hop and Blues are quite different in their structure as well as in the words used and therefor easier to classify correctly.

GENRE	BERT-BASE	BERT-MINI	BERT-TINY
Pop	26%	2%	3%
Rock	35%	18%	13%
Electronic	31%	23%	11%
Jazz	66%	60%	62%
Funk	44%	23%	19%
Hip hop	51%	61%	52%
Blues	66%	55%	57%
Total	**46%**	**35%**	**31%**

[Table 1] Accuracy rate per genre for BERT models. The training dataset contained 5138 songs (734 songs per genre) and the evaluation dataset contained 1232 songs (176 songs per genre).

Furthermore, choosing different genre groups has a big impact on the model performance. A model can even perform better on identifying 4 genres compared to 3, if they aren't similar. As we found out, the better genre groups to choose are often the ones that were the most distinct from one another, to us as humans. In [Table 2] we can see that the choice of genre groups is crucial for classification.

GENRE	**BERT-MINI**	**BERT-TINY**
3 genres [Jazz, Hip hop, Blues]		
Jazz	68%	40%
Hip hop	90%	97%
Blues	53%	10%
Total	70%	49%
3 genres [Pop, Rock, Electronic]		
Pop	60%	30%
Rock	27%	75%
Electronic	36%	-
Total	41%	35%

[Table 2] Accuracy rate per genre of BERT models trained on different genre groups

5.3 DAN and Verse-Wise DAN

We trained DAN and Verse-wise DAN on numerous genre groups. On [Table 3] we can see that BERT performed best, while DAN out-performed Verse-wise DAN. We built Verse-wise DAN to refine the averaging using the structure of the song and expected it will improve predictions. The result is it had worsened the results rather than improving them. The difference between the DAN variations seem to stay about 10%, and both models perform well above random labeling.

	DAN	Verse-wise DAN	$BERT_{BASE}$
(A) 7 genres all genres	38%	34%	45%
(B) 5 most popular genres	40%	36%	47%
(C) 3 most popular genres	54%	49%	60%
(D) Hip hop, Blues, Jazz	69%	66%	76%

[Table 3] Results for comparing the accuracy of DAN and Verse-wise DAN on numerous tasks for 30 epochs. The train/ evaluation ratio was 80/20. Total song amount in each session: A- 6370, B-7665, C- 11196, D-2730

As expected, the less genres we have, the better the results are. It can be deduced since each model improves as we only use songs from less genres. Because some genres have more songs than others in the dataset, picking only the most popular genres allows us to train using more songs, even when the genres are similar.

Another aspect is picking the genres. As expected, using non-similar genres (to humans) causing the models to perform better. When viewing C and D we can see that in contrast to the fact C was trained using more songs (11.1k and 2.7k respectively), all the models achieved better accuracy in D. From this we conclude that picking the right genres is more important.

6. Conclusions and Future Work

In this paper we implemented different NLP models and compared how well they classified songs to genres based on lyrics only, instead of the usual classification by audio. Genre classification is a difficult task even for humans, and using only lyrics ignores most of the song information. We built our own dataset which contains 20k songs of 7 genres and preprocessed them.

We can see that $BERT_{BASE}$ preformed the best. This is not surprising considering it is a state-of-the-art model. In addition, both DAN variations performed better than $BERT_{TINY}$. However, DAN was the second-best. This is surprising since Verse-wise DAN (DAN to represent each verse followed by RNN) a refinement of the DAN that utilizes the song structure.

When classifying songs from Hip hop, Blues and Jazz, we achieved 69% accuracy using DAN, 66% using Verse-wise DAN, and 76% using BERT. As predicted, we noticed that using more songs improve accuracy. We also noticed that picking non-similar (to humans) genres improves accuracy significantly. The reason we need to choose genres wisely is inherent in the definition of the genres- songs can belong to multiple ones, and their boundaries aren't well defined. When using similar genres, one can't classify songs perfectly to a single genre. This is visible when classifying to all 7 genres, including similar ones like Rock and Funk- we reached accuracy of 34% using Verse-wise DAN and 38% using DAN.

As seen in past research, using the audio alone is better than the lyrics alone, so future might combine both to get more complete representation of the song. To humans, the tempo and the instruments used are crucial to classify the song. When making the lyrics explicit in the input, the models could use speech recognition more sparingly. One more direction for further work is to train a Line-wise DAN instead of Verse-wise since it performed better in similar task for documents[18].

We believe that the main issue with classifying songs is with the overlapping nature of genres. Redefining the task to multilabel classification may improve performance while also capturing the true genres of songs.
The source code can be found on github[30]

References

[1] G. Tzanetakis, P. Cook, "Musical Genre Classification of Audio Signals", IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING, VOL. 10, NO. 5, JULY 2002

[2] R. Castellon, C. Donahue, P. Liang, "Codified Audio Language Modeling Learns Useful Representations for Music Information Retrieval", arXiv preprint arXiv:2107.05677, 2021

[3] M. Iyyer**,** V. Manjunatha, J. Boyd-Graber and H. Daume, "Deep Unordered Composition Rivals Syntactic Methods for Text Classification", 2015.

[4] K. Cho, B. van Merrienboer, D. Bahdanau and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches", arXiv:1409.1259, 2014.

[5] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification", NAACL-HLT, pages 1480–1489, 2016.

[6] J. Devlin, M. W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", arXiv:1810.04805, 2018.

[7] J. Pennington, R. Socher and C. D. Manning, "GloVe: Global Vectors for Word Representation", Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014), 12:1532–1543, 2014.

[8] P.C. Chang, Y. S. Chen and C. H. Lee, "MS-SincResNet: Joint learning of 1D and 2D kernels using multi-scale SincNet and ResNet for music genre classification", arXiv:1409.1259, 2021

[9] R. Castellon, C. Donahue and P. Liang, "Codified Audio Language Modeling Learns Useful Representations for Music Information Retrieval", arXiv:2107.05677, 2021

[10] T. L. Li, A. B. Chan, and A. Chun, "Automatic musical pattern feature extraction using convolutional neural network", Proc. Int. Conf. Data Mining and Applications, 2010.

[11] D. C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification", IJCAI Proceedings-International Joint Conference on Artificial Intelligence, vol. 22, p. 1237, Barcelona, Spain, 2011.

[12] L. Feng, S. Liu, J. Yao, "Music Genre Classification with Paralleling Recurrent Convolutional Neural Network" arXiv:1712.08370, 2017

[13] A. Boonyanit and A. Dahl, "Music Genre Classification using Song Lyrics",

[14] R. Mayer, R. Neumayer, and A. Rauber. "Combination of audio and lyrics features for genre classification in digital audio collections", Proceedings of the 16th International Conference on Multimedia 2008, 2008.

[15] X. Hu and J. S. Downie. "Improving mood classification in music digital libraries by combining lyrics and audio", 10th Annual Joint Conference on Digital Libraries, JCDL 2010 - Gold Coast, QLD, Australia 2010.

[16] J. Nam, J. Herrera, M. Slaney, and J. O. Smith. "Learning sparse feature representations for music annotation and retrieval", ISMIR 2012, 2012.

[17] C. McKay, J. Ashley Burgoyne, J. Hockman, J. B. L. Smith, G. Vigliensoni and I. Fujinaga, "Evaluating the Genre Classification Performance of Lyrical Features Relative to Audio, Symbolic and Cultural Features", ISMIR 2010, 2010.

[18] A. Tsaptsinos, "Lyrics-based Music Genre Classification Using a Hierarchical Attention Network", arXiv:1707.04678, 2017

[19] R. Balyan, K. S. McCarthy and D.S McNamara, "Applying Natural Language Processing and Hierarchical Machine Learning Approaches to Text Difficulty Classification", Int J Artif Intell Educ 30, 337–370, 2020.

[20] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", Journal of Machine Learning Research 21 (2020) 1-67, 2020.

[21] J. Devlin, M. W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", arXiv:1810.04805, 2018.

[22] https://rapidapi.com/apidojo/api/shazam

[23] https://developer.spotify.com/documentation/web-api/reference/#/

[24] GTZAN Audio samples Dataset https://paperswithcode.com/dataset/gtzan

[25] https://www.letssingit.com

[26] https://www.lyrics.com

[27] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy and S. R. Bowman, "Glue: A Multi-task Benchmark and Analysis Platform for Natural Language Understanding", arXiv:1804.07461, 2018

[28] P. Rajpurkar, J. Zhang, K. Lopyrev and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text", arXiv 1606.05250, 2016.

[29] R. Zellers, Y. Bisk, R. Schwartz and Y. Choi, "Swag: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference", arXiv:1808.05326, 2018.

[30] https://github.com/BenM61/NLP_proj