Music Lyrics Classification & Generation using RNNs

Daniel Gordin and **Arnab Sen Sharma** and **Jaydeep Borkar**
Northeastern University

Abstract

In this work, we investigate the efficacy of different RNN-based language modeling architectures on the task of music lyrics classification and generation. Our custom dataset contains 80k song lyrics of the genres *Rock, Pop, Rap,* and *Country*. We evaluate the classification performance of three main architectures: Vanilla RNN, GRU, and a number of LSTM variations (including multilayer and bidirectionality). Our best-performing classifier was the GRU and achieved a macro-F1 score of 0.71 and validation accuracy of 72%. Furthermore, we trained the RNN, LSTM, and GRU architectures on the task of lyrics generation and found that LSTM consistently produced lyrics of higher quality. We also experiment with how the initial cell-state and the hidden-state can be used to *nudge* the LSTM-based model to generate lyrics in a specific genre.

1 Introduction & Related Work

Music information retrieval (MIR) aims to develop algorithms and systems for analyzing, organizing, and retrieving music-related data. One of the fundamental tasks in MIR is music genre classification, which involves categorizing music into distinct genres based on its musical characteristics such as melody, rhythm, and lyrics. Reliable genre classification is vital for a multitude of practical applications in music retrieval and analysis, music copyright and licensing, music education, and music recommendation. In particular, digital music platforms such as Spotify and iTunes are focused on harnessing genre classification to improve their recommendation systems and enhance their user experiences.

Language models are a popular tool for any kind of natural language processing task. The goal of language modeling is to build a statistical model that captures the patterns and dependencies of language and can in turn predict the likelihood of a sequence of words given some context. Recurrent Neural Networks (RNN) (Rumelhart et al., 1986) are a popular type of language model architecture that are designed to process sequential data by passing information from one time step to another. LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Chung et al., 2014) are variations (Figure 1) of RNNs that were developed to address the vanishing gradient problem and for tasks that involve longer-term dependencies. In this paper, we compare and contrast these different RNN language model architectures in their ability to classify lyrics by genre as well as generate new lyrics.

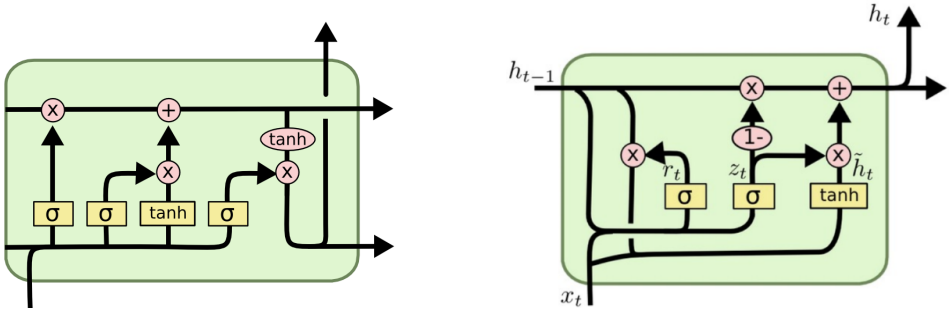
Several studies have investigated lyrics classification and generation using the aforementioned architectures. (Fell and Sporleder, 2014) conducted an analysis and classification of lyrics using n-gram models. (Tsaptsinos, 2017) explored lyrics classification using a Hierarchical Attention Network (HAN). (Chen and Lerch, 2020) attempted to generate melody-conditioned lyrics using SeqGANs. Similarly, (Yu et al., 2021) employed a conditional LSTM-GAN for this purpose, while (Fan et al., 2019) utilized a Hierarchical Attention Based Seq2seq Model for Chinese Lyrics Generation.

2 Data

We prepared a custom data set for this project that is made up of more than 80,000 song lyrics (from nearly 1000 different musicians) with approximately 20,000 songs (Figure 2) from each of the following 4 genres: *Rock, Pop, Rap,* and *Country*.

The data set was constructed by scraping the lyrics of `genius.com` using their python API[1]. Since many artists make music in multiple genres, we needed to select a list of artists for each genre by referring to artist genre labels assigned by media service providers such as *Spotify* and *Apple Music*, to mitigate subjectivity.

[1] `pypi.org/project/lyricsgenius`



(a) LSTM Cell (b) GRU Cell

Figure 1: Cell architecture for LSTM and GRU

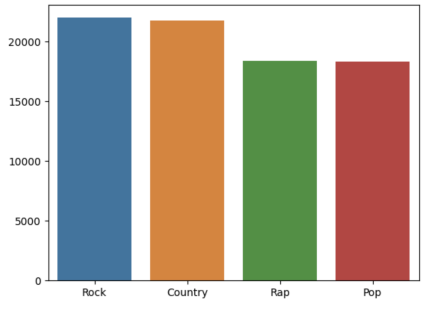


Figure 2: Genre Labels Distribution

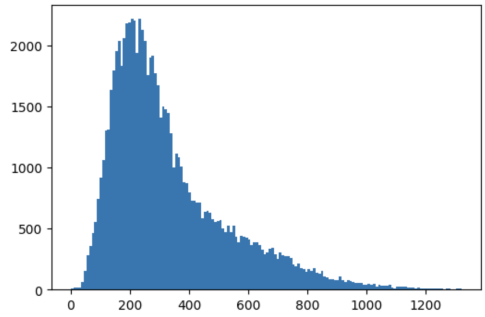


Figure 3: Lyrics Lengths Distribution

The raw scraped data was quite messy and several data-cleaning steps were required. Specifically, many of the lyrics contained promotional messages and song ID information that we removed using `regex` pattern matching as well as special unicode characters that needed to be decoded. We also deduplicated the dataset and filtered out any lyrics that were less than 50 words long or any non-English lyrics using the `langdetect`[2] library.

The preprocessing was mostly limited to converting the text to lowercase and removing punctuation/special characters. Parentheses were the only special characters that were kept as they play an important role in the lyrics for denoting backup vocals. Here is an example from the song *'I Want It That Way'* by the Backstreet Boys:

Ain't nothin' but a mistake
(Don't wanna hear you say)
I never wanna hear you say (Oh, yeah)
I want it that way

3 Experiment 1: Lyrics Classification

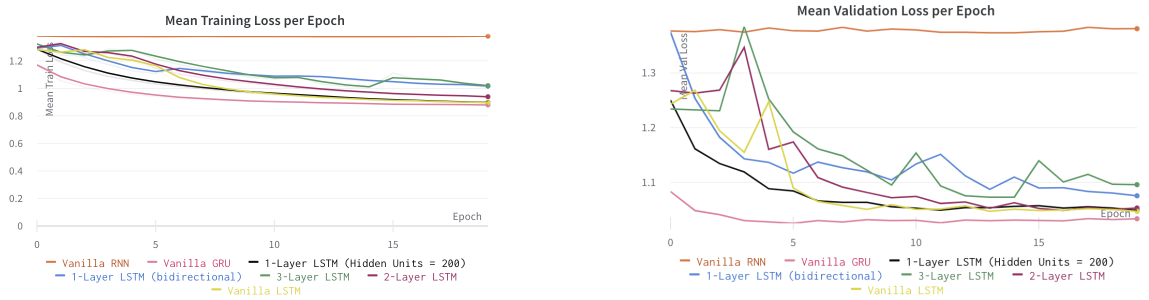
3.1 Data formatting

The preprocessed data was first split into a test set that contained 78% of the data and two subsets for validation and testing that made up around 11%. This particular split meant both the validation and test sets contained approximately 10k samples which is enough for accurately evaluating the performance of our classifiers.

Tokenization was performed using the `basic_english` tokenizer from the `torchtext` library. The vocabulary was constructed from the training set using the `torchtext` method `build_vocab_from_iterator`. Using a hyperparameter sweep, we found that the classifier performed best when omitting any words that appeared less than 5 times in the training set from the vocabulary.

On average, the song lyrics comprised around 300 tokens, although some songs reached as long as 1200 tokens (Figure 3). Using a hyperparameter sweep, we determined that capping the length of each song at 950 tokens yielded the best classifier performance. To accommodate the shorter lyrics, we applied zero back-padding to any with fewer than 950 tokens to ensure consistent feature dimensions.

[2] https://pypi.org/project/langdetect/



(a) Training Loss

(b) Validation Loss

Figure 4: Training and Validation Loss by Model

3.1.1 Methods

Finally, the genre labels were all one-hot encoded using the `MultiLabelBinarizer()` method from sklearn before being packaged into `pytorch TensorDatasets` and `DataLoader` objects alongside the index-converted lyrics tokens.

The model we decided to use as our baseline was a vanilla LSTM consisting of an embedding dimension of length 100 and a single LSTM cell with 100 hidden units followed by a dense softmax to convert the outputs into their final classification probabilities. We used a hyperparameter sweep to identify that a batch size of 40 and a learning rate of 2e-3 yielded the best classifier performance for this particular architecture. We also found that initializing the hidden and cell states with random numbers rather than zeros slightly improved performance. Moreover, we observed that using a weight decay of 1e-7 (L2 regularization) did a better job at preventing model overfitting during training than a dropout layer.

The classification performance of the baseline LSTM was contrasted with the following augmented architectures: 2-layer LSTM, 3-layer LSTM, a 1-layer bidirectional LSTM, a 1-layer LSTM with 200 hidden units, vanilla GRU, and vanilla RNN. The training results after 20 epochs for all these models were logged and visualized (Figure 4) using WandB, a developer tool for tracking and illustrating machine learning experiments.

3.1.2 Results

The results of this first experiment were quite surprising since the baseline vanilla LSTM actually achieved a higher test set classification accuracy (around 70%) than all the other LSTM-based models. One potential explanation for this outcome is that all those other models contained much larger numbers of parameters due to the extra layers or extra hidden units which can result in overfitting and poor generalization. This is evidenced by the results in the table below where the 3rd smallest model (200 HUs) performed the 3rd best and the largest model (3-layers) performed the worst. Alternatively, 20 epochs may not be enough time to properly train these larger networks.

Model	Hidden Units	Embeding Dim	Num Layers	Batch Size	Bidirectional	Accuracy	Macro F1
Vanilla GRU	100	100	1	40	FALSE	0.714	0.719
Vanilla LSTM	100	100	1	40	FALSE	0.702	0.705
200 Hidden U	200	100	1	40	FALSE	0.691	0.7
2 Layers	100	100	2	40	FALSE	0.686	0.69
Bidirectional	100	100	1	40	TRUE	0.665	0.671
3 Layers	100	100	3	40	FALSE	0.645	0.648
Vanilla RNN	100	100	1	40	FALSE	0.286	0.291

Figure 5: Model Parameters and Classification Results

The model that performed the best overall though was the vanilla GRU with a test set classification accuracy of nearly 72%. One possible reason for this is that music lyrics contain more short-term dependencies since they are usually separated into discrete groupings of verses and choruses. However, the accuracy boost could also be attributed to the fact that GRUs have less tunable parameters than LSTMs (which maintains the trend discussed above). Figure 4 demonstrates the loss for our models and Figure 5 summarizes our findings.

It was not surprising to find that the vanilla RNN performed only slightly better than random with a validation accuracy of just 28%. The primary reason for this is that it doesn't have any memory cells which are crucial for storing information and preventing vanishing gradient problems.

In order to identify the strengths and weaknesses of our best performing model, we generated a multiclass confusion matrix (Figure 6) that shows how well the vanilla GRU is predicting each genre and which genres are being misclassified more often. Specifically, we show that rap has a true positive

rate of nearly 90% and country has a true positive rate of nearly 80%. The themes and language used in rap lyrics are often very explicit, which makes them stand out from other genres of music. Similarly, country lyrics often features imagery that evokes the rural lifestyle, such as farms, fishing, trucks, and religion.

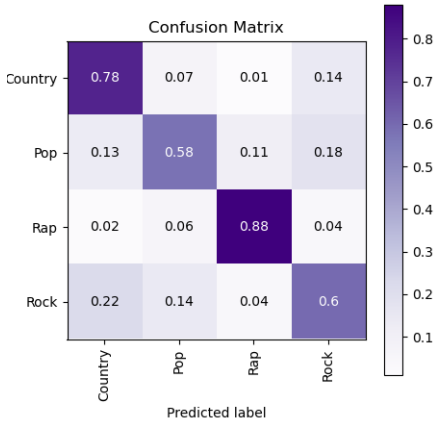


Figure 6: Confusion Matrix for GRU

On the other hand, rock and pop genres don't have the same degree of distinctive lyrical characteristics that define rap and country songs which is likely the main driver behind their much lower true positive rates of 60%. Interestingly, rock songs were most likely to be misclassified as country songs (22%) and vice versa (14%). This is understandable as the genres share some lyrical elements in common such as their focus on storytelling, romance and politics. Furthermore, rap lyrics were most likely to be misclassified as pop (6%) which makes sense as many pop songs actually feature rappers.

In order to provide some human-understable explanations for our networks' classification predictions, we applied the LIME (Local Interpretable Model-Agnostic Explanations) (Ribeiro et al., 2016) algorithm using the `LimeTextExplainer` method from the `lime_text` library. The algorithm works by creating a simpler "interpretable" model that approximates the behavior of the original model in the local region around a particular prediction and highlights the features (in this case words) most responsible for that prediction. We found that LIME generally highlighted expletives for many instances of rap and highlighted ecclesiastical

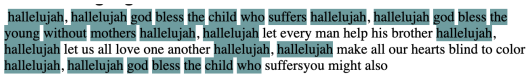


Figure 7: LIME Output for Instance of Country Lyrics

words for country music such as "Hallelujah" in Figure 7.

4 Experiment 2: Lyrics Generation

4.1 Preprocessing

We do not perform further preprocessing steps after data cleaning as all possible tokens are required for the fluency of a language model. We don't even lowercase the lyrics and remove punctuation marks as we did as part of preprocessing step before classification. We just add two special tokens `<s>` and `<\s>` at the beginning and end of a lyric respectively, to indicate its beginning and ending.

4.1.1 Tokenization

Our tokenizer is based on the pretrained tokenzier of `gpt2`[3] language model from `huggingface`. The gpt2 tokenizer was trained on `gpt2` training dataset, using Byte-Pair-Encoding strategy. The vocabulary of that pre-trained tokenizer was 50K. In order to reduce the parameter count, we build our own `Tokenizer` module that wraps the `gpt2` tokenizer and uses it as an intermediate step. The vocabulary size of our `Tokenizer` and thus the language models is $\sim 40K$.

4.1.2 Embedding

We trained a word embedding model with the help of `gensim`[4] package. The word embedding model was trained on a window of size 5 using skip-gram (Mikolov et al., 2013). The model was trained to embed each token in a vector of size 300.

4.2 Model Hyperparams and Training

For fairness of comparison, the hyperparameters were kept similar for all the language modeling architectures: RNN, LSTM, and GRU. The `encoder` module, which is basically the embedding matrix of our trained *Word2Vec* model maps a token to a vector of size $|E|$. The number of layers in the RNN/LSTM/GRU stack was 2. We experimented with different amounts of layers , but found that higher number of layers perform worse due to gradient diminishing during training. It was

[3] https://huggingface.co/gpt2

[4] https://radimrehurek.com/gensim/models/word2vec.html

interesting to observe that LSTM and GRU architectures also suffer from gradient diminishing after enough layers despite their reputation for being more robust against the problem due to their information storage capabilities. The RNN stack is followed by a `decoder` module that takes a vector of dimension $|E|$ and maps it to a logit distribution over the vocabulary space V.

The vocabulary size $|V| = 39903$ and embedding size $|E| = 300$. The hidden dimension of RNN/LSTM/GRU stack, $|H| = 4 \times |E|$.

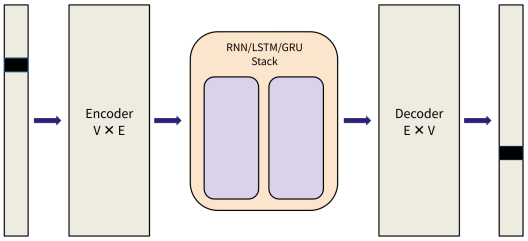


Figure 8: Model Architecture

All of the models were trained on 45K of our lyrics dataset for 10 epochs with batch size of 128. *Cross entropy* was used as the loss objective to minimize the difference between the target and predicted distribution. The models were tuned with `Adam` optimizer with a learning rate of 0.001. During training, the `encoder` module was kept frozen and all the remaining modules were tuned. A `Dropout` of 0.1 was used for regularization.

A set of 15K lyrics was used as the validation set and the perplexity of the models on that validation set was recorded after each epoch. The model loss and perplexity values during training LSTM is depicted in Figure 9.

4.3 Qualitative Comparison

2nd author (Arnab) generated 50 lyrics with each of the models and showed them to the remaining authors (Daniel and Jaydeep) for a qualitative assessment. Daniel and Jaydeep didn't know which lyrics were generated by which model. Both agreed that LSTM and GRU generated better-quality lyrics compared to RNN. Between LSTM and GRU, it was agreed that LSTM generated better lyrics. A selected sample of LSTM-generated lyrics is in Table 1.

4.4 Genre-specific Lyrics Generation

Inspired by recent works on light-weight finetuning (Houlsby et al., 2019), (Li and Liang, 2021), and (Lester et al., 2021) we wanted to experiment on how with the help of additional *soft contexts* we can steer our best performing LSTM model to generate lyrics for a given genre.

Each LSTM cell maintains 2 states; the *hidden* state and the *cell* state of size $|H|$. The role of the *hidden* state is to carry information from immediately preceding steps and the role of *cell* state is to remember relevant information from far back and function as a form of long-term memory. During the start of a sequence, it is normal practice to initialize the *cell* state ($\mathbf{c_0}$) and the *hidden* state, ($\mathbf{h_0}$) with zeroes. However, $\mathbf{c_0}$ and $\mathbf{h_0}$ can also help provide additional context and can be used for nudging the model to generate lyrics in a certain mood.

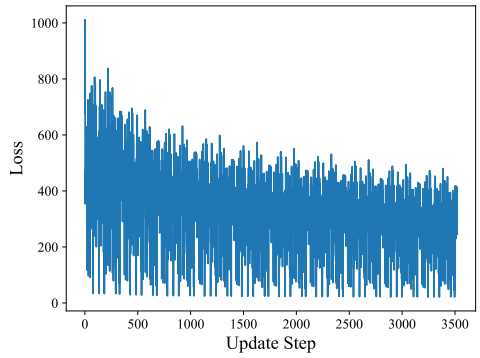
For a genre g, we train a genre-specific $\mathbf{c_g}$ and $\mathbf{h_g}$ as the *initial context* with genre-specific lyrics while keeping the parameters of our LSTM model frozen. This way the same LSTM model can be used to generate genre-specific lyrics just by feeding $\mathbf{c_g}$ and $\mathbf{h_g}$ as the $\mathbf{c_0}$ and $\mathbf{h_0}$ to provide genre-specific context.

$\mathbf{c_g}$ and $\mathbf{h_g}$ were optimized with similar training hyperparameters and objective as during the training of the lyrics generators. We do not notice a large loss reduction during training but the perplexity on the validation set goes down a little for each of the genres (Appendix 7).

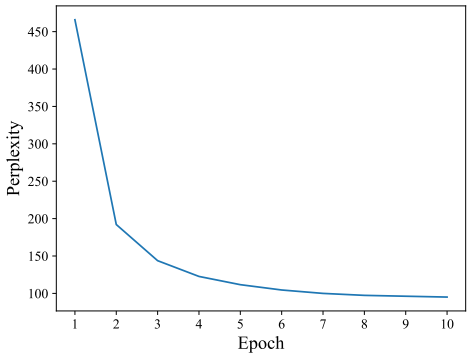
Similar to section 4.3 we perform a qualitative analysis of lyrics generated in different genres by feeding genre-specific $\mathbf{c_g}$ and $\mathbf{h_g}$ as the initial context to the LSTM model. Some of the samples are given in Table 2. While we do notice some subtle differences between the genres as expected, we could not be convinced of our success in genre-specific lightweight finetuning. The lackluster performance could be due to multiple reasons:

1. We do notice some bleedover among the genres in our dataset. The dataset was categorized into respective genres based on the artist. That means if a *Pop* musician produces a *Rock* song, the *Rock* lyric was categorized as *Pop*.

2. The genre-specific tuned *cell* state, $\mathbf{c_g}$ and hidden state, $\mathbf{h_g}$ together contains just $2 \times$ `num_layers` $\times |E| = 2 \times 2 \times 4 \times 300$ parameters. This might not be powerful enough to steer the LSTM model which contains about 7.86 million parameters in total.



(a) Training Loss



(b) Validation Perplexity

Figure 9: Training LSTM

You are going to have a time of the best time I can get along you and me But you are going to love me for a while And I will be right at the top of the · · ·
I have been looking up at the edge of my life Trying to find the answers to my soul There is no reason why I am leaving my life behind I do not think it is hard and · · ·
I am so excited, so I will tell you that I need to leave you alone You will never love me again I will always love you, baby, you are a star I will never know what you · · ·
When I was in a million years ago The day I saw my face And I felt so very younger But you still have to know I would never be the same I was a · · ·
I am a polarrified, I am a lineman case I am a writer, I am a whore for a rich gun, I can File a meal of weed and a sack of a gas, · · ·

Table 1: A selected sample of the lyrics generated by our LSTM model.

Lyrics	**Genre**
(Hey) Oh, yeah Oh, I have been doing this time for me to go (Uh, uh, uh huh, uh-uh) Yeah, yeah, I am about to tell you something (Yeah) But I am · · ·	Rock
(Hey) (Bno) (Hit it) You all *****s are not never seen nothing at all (You know I have been) I got a bad man, that is my boss I be like that · · ·	Rap
I want to tell you something you want to be in that world, babe You want to be my best friend, girl, what you want to be? I got a feeling of my life · · ·	Pop
A little boy, the boy, I can feel a little bit better Than you can see it in the night of a year in my life I can get my money back in her car, I got · · ·	Country

Table 2: A selected sample of the lyrics generated by our LSTM model.

5 Discussion

Our results show that LSTM and GRU architectures lend themselves well to the classification task because they can capture long-term dependencies and contextual information in the lyrics. Specifically, these models can learn the patterns and structure of language unique to each genre, including the use of certain words, phrases, and syntactic structures. We find that genres like Rap and Country were easier to classify over Pop and Rock because they have much more distinctive lyrical characteristics. Our best-performing classifier is a vanilla GRU with an accuracy of 72% and an F1 score of 0.71%. The vanilla LSTM achieves an accuracy of 70% while the vanilla RNN achieves only 28% accuracy. We believe it will be quite difficult to significantly improve on our classification results for the following reasons:

1. Genre boundaries are quite blurry and similar lyrical features can often be found across many genres. Moreover, genre labeling is actually quite subjective and there can be dis-

agreements on which genre a song should be assigned to.

2. Musical genres and their defining lyrical features are not static and can evolve over time thus a song may belong to multiple genres at different time periods.

LSTM and GRU also give improved performances in lyrics generation compared to RNN due to their architectural advantages over RNN. Albeit the training loss of GRU (Figure 11) is better than the training loss of LSTM (Figure 9), we found that LSTM-based model produces better lyrics in general. We tried to make our LSTM-based lyrics generator produce genre-specific lyrics by providing it initial context with genre-specific cell states and hidden states. However, we couldn't get satisfactory results there due to dataset limitations and low parameter count of cell and hidden states.

6 Future Directions

In the future, we aim to extend this work by experimenting with transformer-based models, which have become state-of-the-art at language modeling in recent years. BERT (Devlin et al., 2019) is one such encoder model of interest we believe its ability to capture context, pre-training on large corpora, transfer learning, and efficient handling of long sequences can result in a superior performance compared to GRU/LSTM for lyric classification and generation.

It is also possible to train multiple *soft prompts* for a specific task by taking advantage of the residual connections and attention mechanism of transformer-based models, thus overcoming the low parameter count bottleneck we faced while training *initial contexts* for our LSTM model [(Lester et al., 2021), (Li and Liang, 2021)].

7 Code and Data

Please find our custom dataset here and our code at this repo.

References

Yihao Chen and Alexander Lerch. 2020. Melody-conditioned lyrics generation with seqgans. In *2020 IEEE International Symposium on Multimedia (ISM)*, pages 189–196.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Haoshen Fan, Jie Wang, Bojin Zhuang, Shaojun Wang, and Jing Xiao. 2019. A hierarchical attention based seq2seq model for chinese lyrics generation.

Michael Fell and Caroline Sporleder. 2014. Lyrics-based analysis and classification of music. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 620–631, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier.

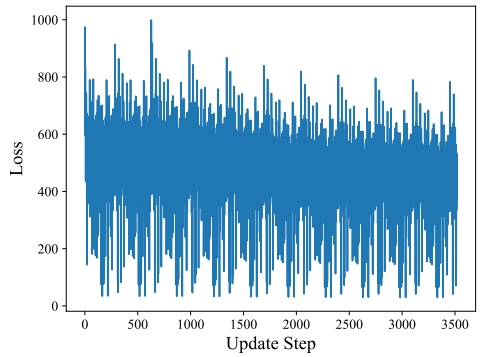
David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning internal representations by error propagation.

Alexandros Tsaptsinos. 2017. Lyrics-based music genre classification using a hierarchical attention network.

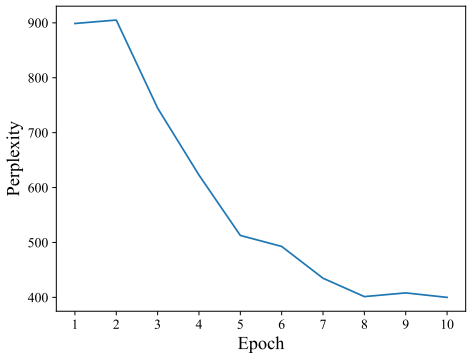
Yi Yu, Abhishek Srivastava, and Simon Canales. 2021. Conditional lstm-gan for melody generation from lyrics. *ACM Trans. Multimedia Comput. Commun. Appl.*, 17(1).

Appendix

A. Training RNN and GRU

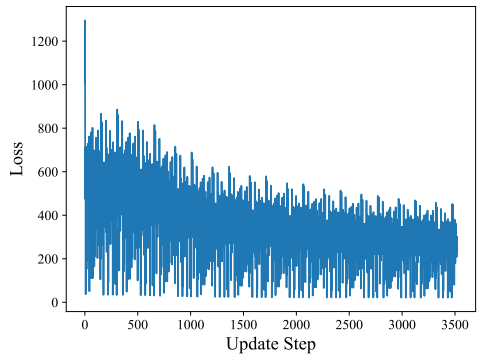


(a) Training Loss

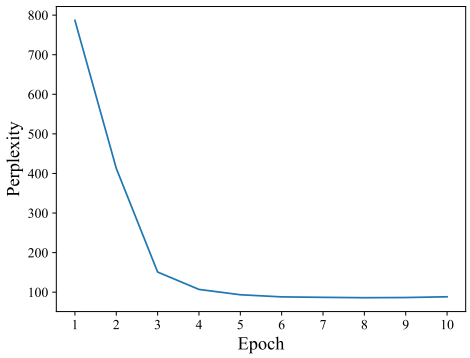


(b) Validation Perplexity

Figure 10: Training RNN



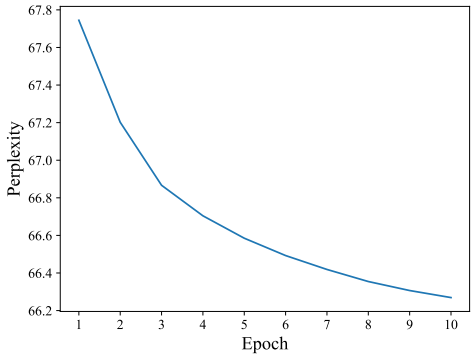
(a) Training Loss



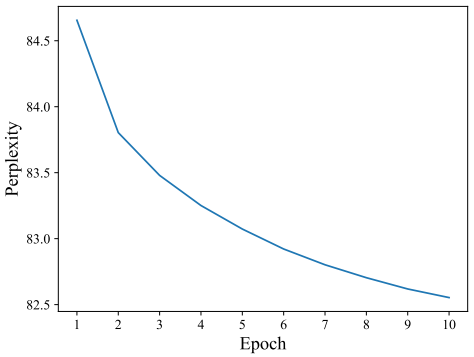
(b) Validation Perplexity

Figure 11: Training GRU

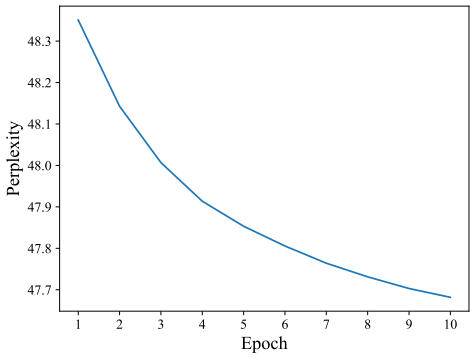
B. Genre Specific Light-weight Finetuning



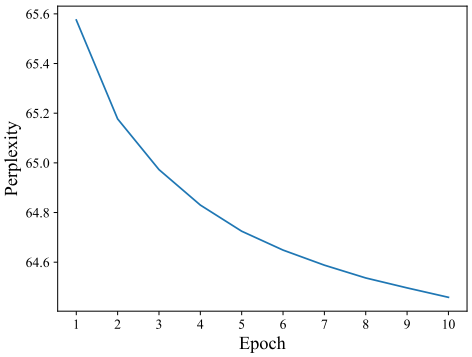
(a) Rock



(b) Rap



(c) Pop



(d) Country