Lyrics Generation using LSTM and RNN

Satyaki Mandal[1], Sandipta Bhadra[2]

Department of CSE, Vellore Institute of Technology Chennai, Chennai, India[1,2]

Abstract: This research examines the potential of contemporary deep learning techniques to enhance the song writing process by learning from artists and their substantial works.

By adjusting the model's hyper parameters, the neural network was pre-trained on the lyrics of certain composers and musicians. A multilayer LSTM-based training model with bidirectional neurons and BERT integration is used in this research. To generate a comprehensive set of lyrics, the lyrics supplied as input are divided down into a word and rhyme index. This model is generative, therefore each trial yielded a unique set of lyrics.

The model produces a loss of less than 0.06 when the parameters are set correctly. The differences in the results of permutations of different dropout positions were analysed. Some of the model's lyrics were determined to be of suitable quality.

Overall, our findings suggest that deep learning techniques may be utilised to support the creative process of song writing.

I. INTRODUCTION

The study focuses on creating a lyrics generator using LSTM at both the character and word levels (Long short-term memory). A generative model was used to assist the song writing process, which included a recurrent neural network (RNN) and transfer learning. Artists spend their entire careers perfecting their ability to write songs that are emotive, relevant, and innovative. This was a job that artificial intelligence might help with. Instead of having struggling artists sift through hours of viable material, what if we could automate the process by having a neural network create line after line of decent content that the artist may then choose from as inspiration?

The issue was significant since it is difficult to write appropriate lyrics that follow a strong rhyme scheme, but artificial intelligence may not only speed up but also make the process easier for artists. The outcomes were promising, and at times were indistinguishable from popular song writing approaches.

II. LITERATURE REVIEW

A. *Rhythm-based Lyrics Generation*. Tra-la-la lyrics is a system designed to generate lyrics based on pre-made melodies. Oliveira, Cardoso and Pereira studied the relationship between words, the musical rhythm of melodies and rhymes. They implemented various algorithms that could perform syllabus separation and determine the "syllable stress" of a particular word. They also created a dedicated database to store words and grammar types. Unfortunately, their result was of not so good quality. [1]

B. *Rap Lyrics Generation*. A rap lyrics generator developed by Nguyen and Sa. This program includes a database of about 40,000 existing rap lyrics. Then, a new lyric is created from the words and verses contained in the existing lyrics. The linear interpolation model approach was used to generate the lyrics. However, the results were assessed as lacking fluency. Thus, they change to the four-gram form. They also provided a database of rhyming words from two different sentences. In this way, they created sentences that rhyme with each other. Finally, all the phrases were reconstructed according to the structure and arrangement of the song. The generator works fine, but the lyrics don't make sense and don't relate to a particular topic. [2]

C. *Automated Poem Generation WASP* (Automated Spanish Poet) is the first poetry creation program that combines natural language generation techniques with artificial intelligence. It is a system that takes input from the users and they are used as the seed. The system is based on a system based on the forward inference rule. The results obtained were rated as mediocre and ineffective. [3]

D. *Semantic similarity in lyrics*. Logan, Kositsky and Moreno experimented with using lyrics to automatically identify and categorize music and identify artist similarity. The lyrics of the songs are collected from various sources on the web. Various techniques such as PLSA (Probabilistic Latent Semantic Analysis) and k-means clustering method have been used to analyse the content and semantics of the lyrics. Evaluation is done by combining the system with another sound system to check their similarity. Both techniques used have advantages and disadvantages. So a combination of the two techniques would probably be much better, but that has been left to future work. [4]

E. *Titular and LyriCloud*. Tilular and LyriCloud are two software tools developed with the aim of creating intelligent and interactive lyrics writing tools for musicians and musicians. Titular is a semi-automatic song title generation system while LyriCloud suggests related word cloud based on input seed. A model-based technique was used. Profanity and derogatory words were filtered out and the words, which appeared more frequently in the database, were the ones most likely to appear in the final title of the generated song. They get good results but sometimes the title has no semantic meaning and doesn't make much sense to the reader. [5]

F. *Natural Language Processing and Lyrics Generation*. Mahedero, Martinez, and Cano analysed the lyrics using basic natural language processing tools. Experiments were conducted on the text to identify the language, classify according to different themes extract structures, and look for similarities between them. The selection of 500 lyrics was selected from multiple websites. The result obtained was 92 curates. A Naive Bayes classifier was used. The Inverse Document Frequency (IDF) and Cosine distance was used to measure similarity. The identification of languages proved to be an easier task compared to the others. [6]

G. *Classification of lyrics based on rhyme and style*. Rudolf Mayer et. al. Experienced in rhyming and stylistic features for lyrics grading and processing. They used a word group along with lyrics tag and other statistical features to process the lyrics. A rhyme is actually two words that, when spelled, sound the same. This feature is often used for words at the end of a verse. Proper evaluation of the proposed method was not performed. [7]

H. *Automated Sega lyrics generation*. Bhaukauraly, Didorally and Pudaruth worked to provide a tool for lyricists to generate Sega lyrics in the Mauritian Creole language. 66 people were asked to rate 10 songs as either man-written or machine-generated. Out of these 10 songs, 5 are self-composed. Apparently, the generated lyrics are of pretty good quality as about 50% of the respondents actually didn't know if the lyrics already existed or were generated by a computer program. The main weakness of this work is the lack of information in the implementation about how sentences are actually made or produced. [8]

III. Architecture

Recurrent Neural Networks (RNN)

RNNs are a form of Neural Network that allows for the use of previous outputs as inputs. They employ the abstract idea of Sequential Memory, which states that a sequence of items provides greater information and efficiency in obtaining a model's output (think the Alphabet). Sequences have an intrinsic property of structure and information. RNNs can recognise patterns and sequences and utilise them to create predictions. They do this by establishing a feedback loop in the network that uses prior data to guide the next iteration of inputs. The hidden state is a vector representation of prior inputs in this feedback loop. This permits information to remain across the model, which is impossible with standard feed-forward networks. [9]

RNNs have a number of advantages, including the ability to analyse any length of input sequence since the model's size does not scale with the amount of the input, and the ability to take into account previous historical data while operating. RNNs are helpful for identifying patterns in data sequences such as text, audio, or numerical time series data because of their benefits. Our method has a narrower domain using RNNs, however the ***Vanishing Gradient Problem*** might impair text (lyric) production. [10]

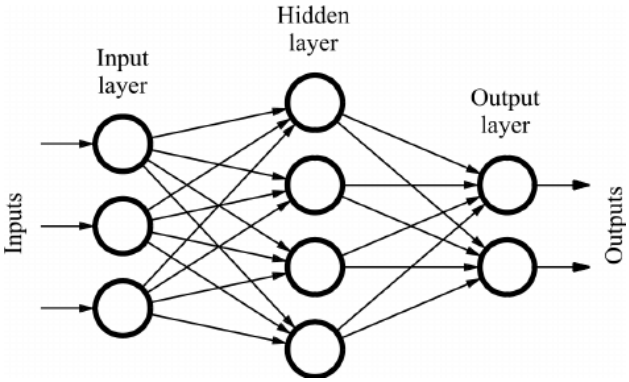


Figure 1: Conventional Feed-Forward Neural Network [11]

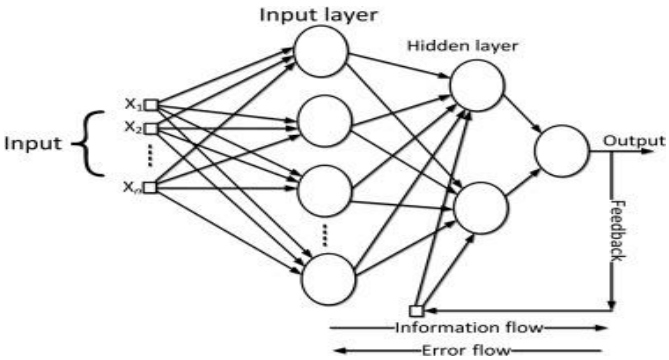
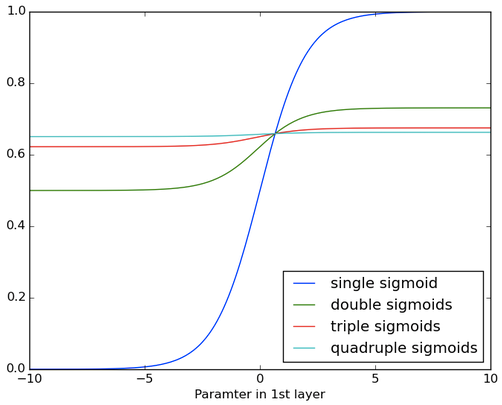


Figure 2: Recurrent Neural Network [12]

Long Short-Term Memory Networks (LSTM)

As previously stated, RNNs have difficulties learning long-term dependencies across time steps. RNNs have a short-term memory and can't access information from the past. The Vanishing Gradient Problem occurs when the gradient of the loss function (values used to update weights) diminishes rapidly during back-propagation and eventually disappears. A gradient that gets too thin (and finally zero) doesn't help you learn much. Because of the microscopic adjustments of the weights by exceedingly small gradients, the neural network's earlier layers are unable to learn. [13]



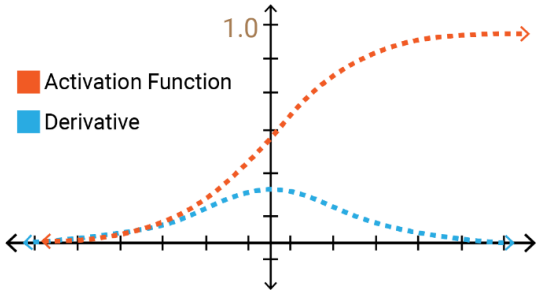


Figure 3 and 4: Vanishing Gradient Problem [14] [15]

One can utilise LSTMs to fix this issue. An LSTM is a form of RNN that can learn long-term dependencies. LSTMs are capable of preserving errors that can be transmitted backwards in time and layers. They improve RNNs by allowing them to learn across multiple time steps by keeping the error value constant.

LSTMs can do this by employing gates, which are tensor operations that can figure out what information to add or remove from the hidden state. The LSTM network has an input gate, a "forget" gate, and an output gate in each unit. The input gate has the capacity to evaluate the worth of provided data. The "forget gate" can select whether information should be discarded or saved for later use. The output gate is responsible for determining whether or not information is relevant at a given phase. The gates take inputs as sigmoid function parameters throughout each phase. The sigmoid returns a number between 0 (allow nothing through the gate) and 1 (permit everything through the gate) (let everything through the gate). This idea is used in back-propagation (updating layer values).

LSTMs may pick which information to take forward and which information to drop by employing these gates. These gates assist govern information flow inside the network and allow the error value to remain throughout the network, giving them a significant benefit over RNNs. We concluded that adopting LSTMs would be the best line of action for our lyric generation model. LSTM units were used to form a network because we're trying to produce lyrics based on an initial input sequence. [16]

Most linear algebra operations may be parallelized to enhance speed with GPU support, and vector operations like matrix multiplication and gradient descent can be applied to huge matrices that are processed in parallel. The Compute Unified Device Architecture (CUDA) interface enables vector operations to take use of GPU parallelism. CuDNN uses CUDA to implement large-matrix kernels on the GPU. [17]

.CuDNNLSTM is optimised for CUDA parallel processing and will not run without a GPU. LSTM, on the other hand, is intended for regular CPUs. Parallelism allows for faster execution times.

As a result, CuDNNLSTMs were used in the model instead of ordinary LSTMs. We were able to obtain a total training duration almost 5 times faster than conventional LSTM execution with this simple adjustment.

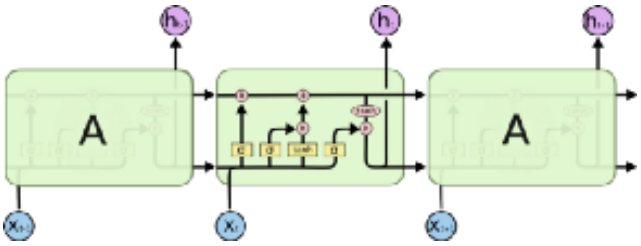


Figure 5: LSTM Network Diagram [18]

Input Lyrics Data

Finding a dataset to feed into the LSTM network was the first step in creating our lyric generator. Fortunately, I was able to find a somewhat robust dataset on *Kaggle* [19]. Lyrics from prominent musicians such as Taylor Swift, Coldplay, Linkin Park, Eminem, The Chain Smokers, and others were included in the dataset.

Our goal was to create a lyrics generator that closely matched a certain artist's song writing style. Because we ran into memory difficulties numerous times while creating our lyric generator, the difficulty with the dataset was extracting a portion of it that was sufficient but not too huge. To train our models using this information, we built unique CSV files for each artist.

Pre-processing: The Lyrics of a particular artist was being joined to form a single file, and standardized, by making all characters to lowercase and removing any non-alphanumeric characters. The sting was then broken into a word and rhyme index.

Model Building

The Neural Network has four 256-node bi-directional LSTM layers, one input layer with the 100 character sequences formed during data pre-processing, a flattened, dense, dropout, and lastly an activation layer with a 'soft-max' activation function. A call-back function is used to determine which character is most likely to appear given the preceding character. In our example, the *'checkpoint call-back'*, a call-back is a function that is invoked after each epoch. A checkpoint call-back saves the model's weights each time the model improves

Dropout is a regularization method where input and recurrent connections to LSTM units are probabilistically excluded from activation and weight updates while training a network. This has the effect of reducing overfitting and improving model performance. [20]

The training parameters include 30 epochs, a batch size of 128 and a validation split of 20% (4:1). Once the models were trained and fit to the data given, we were able to input a 100 or more-character sequence and watch the model output as many song lines as specified by the user.

The model predicts output one character at a time since it is character-based. To facilitate this operation, a tensor was generated by mapping unique characters to their frequency in the data. This was performed using Keras' *Tokenizer* function, which allowed us to compress and vectorize characters effectively based on their input frequency. Every character that appears in an artist's total song

lyrics is assigned to two different dictionaries. The data is divided into samples and labels. After that, the data is reshaped, the size is normalised, and one-hot encoding is applied. The generated tensor is then converted into the corresponding character and concatenated to the 'key' string. This new 'key' string is then inputted back into the model to generate the tensor of the subsequent character.

The loss function used is *'Categorical Cross Entropy'* because several articles prior to this study had also use the same loss function. The optimizer we used is *'Adam'* because it is computationally efficient and doesn't have too many memory requirements.

BERT Integration

BERT (Bidirectional Encoder Representations from Transformers) is a new NLP approach developed by Google researchers. BERT works on a high level by employing transformers (a NLP approach that uses Attention) to analyse and learn the relationship between words. It is a bidirectional approach that can comprehend the context or relationship of a word to its surrounding words in a corpus. The BERT model's most major benefit is that it allows for transfer learning, which substantially speeds up the training process while preserving accuracy.

Consequently, *'Bidirectional CuDNNLSTM layers'* were introduced into the neural network. [21] This integration allowed the network to generate more nuanced, human-like lyrics. Also, as mentioned in the previous sections, a shift was done from LSTM to CuDNNLSTM, because the former uses the CPU and the latter uses the GPU, that's why CuDNNLSTM is much faster than LSTM. Results were generated almost 15 times faster.

Batch Sizes

Initially, a default batch size of four was used, however this resulted in an extremely noisy stochastic gradient descent of the loss function. It was found that raising the batch size to 64 or 128 units resulted in a steady gradient descent and greatly reduced training time.

Unidirectional vs Bidirectional Neurons

The hidden state of an LSTM saves information from previous inputs. Because the only inputs it has seen are from the past, the information preserved by the unidirectional LSTM is limited to the past. Bidirectional processing runs your inputs in two directions, one from past to future and the other from future to past. This differs from unidirectional processing in that the LSTM that runs backwards preserves information from the future, whereas using the two hidden states combined, you can preserve information from both past and future at any point in time. Although determining what they are best suited for is a difficult task, bi-directional LSTMs produce excellent outcomes since they are better at understanding context. Let's say we try to predict the next word in a sentence, on a high level what a unidirectional LSTM will see is, [22]

The boys went to ....

And will try to predict the next word only by this context, with bidirectional LSTM you will be able to see information further down the road for example

Forward LSTM: *The boys went to ...* **Backward LSTM:** *... and then they got out of the pool*

You can see that using the information from the future it could be easier for the network to understand what the next word is.

IV. RESULTS AND DISCUSSIONS

Generated Lyrics

With dropout layer:

The use of a dropout layer in the model, was able to affect the overall overfitting of the testing results only very slightly. However, it's implementation brought out an interesting feature in the result. As evident from the attached results, oftentimes it was seen to

generate a set of words/phrase a number of time in succession. Although this effect in no way affected the overall appeal of the generated lyrics, it did add a slight poetic edge to the lyrics. Such use of phrases in literature is a poetic device called *'Repetition'*, which is employed to make a phrase/text more noticeable.

```
Seed :
" takeaway, yeah, yeah, yeah-yeah
your heart for takeaway
your heart for takeaway, yeah-yeah, yeah-yea "

h
your heart for takeaway, yeah, yeah, yeah-yeah
bafore i love you, na, na, na
i'm gonna leave you, na, na, na
before i'm someone you leave behind
i'll break your heart so you don't break mine
before i love you, na, na, na
i'm gonna leave you, na, na, na
even if i'm not here to stay
i still want your heart
your heart for takeaway, yeah, yeah, yeah-yeah
your heart for takeaway, yeah, yeah, yeah-yea
Done
```

OBSN: Since the given *seed/key* phrase, has a repetitive phrase, the model is able to pin on nuance and use the same phrase and device to give the lyric a more poetic edge.

```
Seed :
" m okay when thinkin' 'bout you
you were right when you said it, keep my cards to my chest
and i neve "

r let no one know me, i pushed you out
now i get that you needed just a little more love
and to just see me wide open and breaking down
when no one's there to listen, i will hear your pain
i know that i was less than, but you know people change
so tell me, what's the difference if we still feel the same?
it takes a little distance to make you see the way
i go from okay, i'm okay to i'm not okay
fr
Done
```

OBSN: Here the given *seed/key* phrase, has no repetitive phrase, yet the model is able to use the phrase "I'm okay" and use the *'repetitive'* device to

Without dropout layer

As compared to the former implementation with the dropout layers, here the 'repetitive' element only appears about 25% of the times, whereas in the former it was around 66.67%. The lyrics generated are otherwise indistinguishable from the former implementation, even with a higher overfitting and validation loss. Furthermore, on performing an opining poll with peers, this set of results proved to be a far more acceptable as song lyrics, with an edge of over 80%.

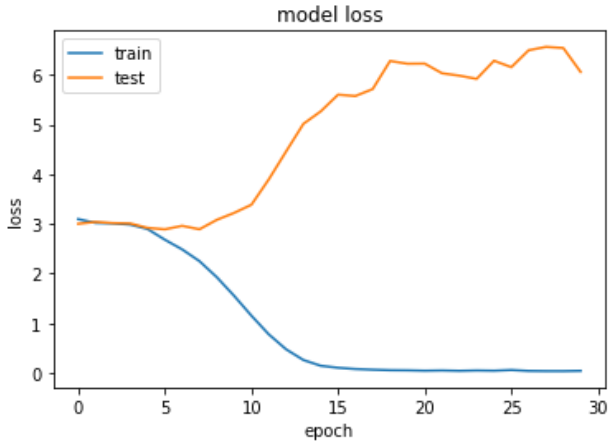
```
Seed :
" ause
he's the reason for the teardrops on my guitar
the only thing that keeps me wishing on a wishin "

g star
he's the song in the car
i keep singing, don't know why i do...
drew walks by me
can't he tell that i can't breathe?
and there he goes, so perfectly
the kind of flawless i wish i could be
she'd better hood him tight, give him all her love
look in those beautiful eyes and know she's lucky 'cau
Done
```

```
Seed :
" ause
he's the reason for the teardrops on my guitar
the only thing that keeps me wishing on a wishin "

g star
he's the song in the car
i keep singing, don't know why i do...
drew walks by me
can't he tell that i can't breathe?
and there he goes, so perfectly
the kind of flawless i wish i could be
she'd better hood him tight, give him all her love
look in those beautiful eyes and know she's lucky 'cau
Done
```

```
Seed :
" to me
i can't help but wish you took me with you
and this is when the feeling sinks in
i don't wanna "

 miss you like this
come back be here, come back be here
i guess you're in london today
i don't wanna need you this way
come back be here, come back be here
this is falling in love in the cruelest way
this is falling for you and you are worlds away
nawhh you're aooadu and thating inte me she break
t
Done
```

```
Seed :
"
and you said, "i never regretted the day that i called you mine"
can i call you mine? ooh, ah, ooh
 "


can i call you mine? ooh, ah, ooh
can i call you mine?
ooh, ah, ooh
can i call you mine?

do you mean, do you mean what you say?
what you said, now you can't take away
you're my gospel, but i'm losing faith, losing faith
do you mean, do you mean what you say?
take a minute, do you need to stop and think?
what we have, no, we can't throw away, throw away
show me that you mean it, ayy
show me that you mean it
do you really mean it?
everything happens for a reason
show me that you mean it do you, d
Done
```

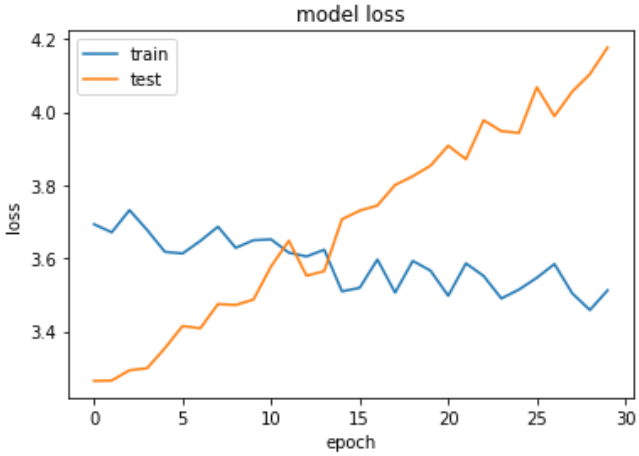
Inclusion/Exclusion of the Dropout layer(s)



Comparison of validation loss vs testing loss in bidirectional LSTM implementation

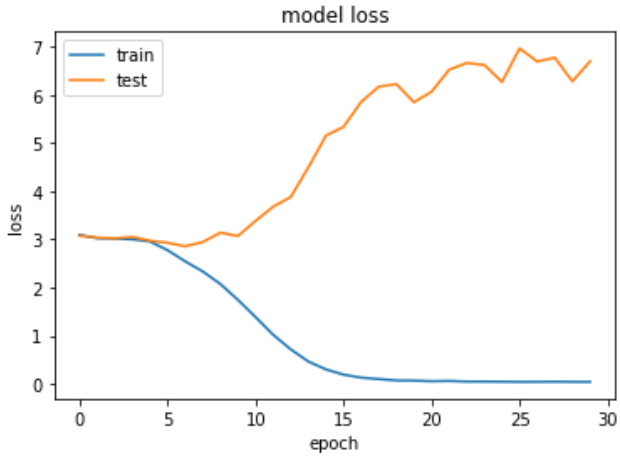
Loss - 0.0420 valalidation-loss6.0612

(Control)



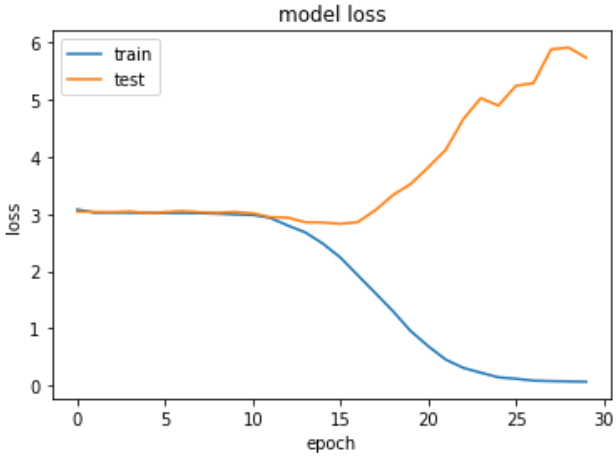
Ex1.
Comparison of validation loss vs testing loss in 1st dropout implementation (just after Dense layer)
This served as a control for the experiments with the dropouts.

Loss - 3.5126 validation-loss - 4.1771



Ex2.
Comparison of validation loss vs testing loss in 2st dropout implementation, after every hidden layer.

Loss - loss0.0655 validation-loss - 5.7330



Ex3.
Comparison of validation loss vs testing loss in 3st dropout implementation, after hidden layers, just before the output Dense layer.

Loss - 0.0427 validation-loss - 6.6893

As evident from the training vs testing loss values, our model appears to be overfitting the data. The conventional response, is to try to mitigate this by addition of dropout layers. A model trained with zero dropout layers served as the control for this test. Three possible configurations of the dropout layer(s) were tested:

- 1st dropout implementation (just after Dense layer) – *general implementation*
- 2st dropout implementation, after every hidden layer.
- 3rd dropout implementation, after hidden layers, just before the output Dense layer.

Through multiple runs, it was found that although the 3rd implementation did successfully reduce the overfitting to a certain extent, the difference between the testing and training loss was still worthwhile. But the difference arises in the lyrics generated.

The model with a dropout layer, often got caught in endless loops of certain words and phrases. The final lyrics generated were also not of sufficiently high quality. Whereas the model, without any dropout layer, having more distinct difference between training and testing loss, was able to generate higher quality lyrics at every iteration.

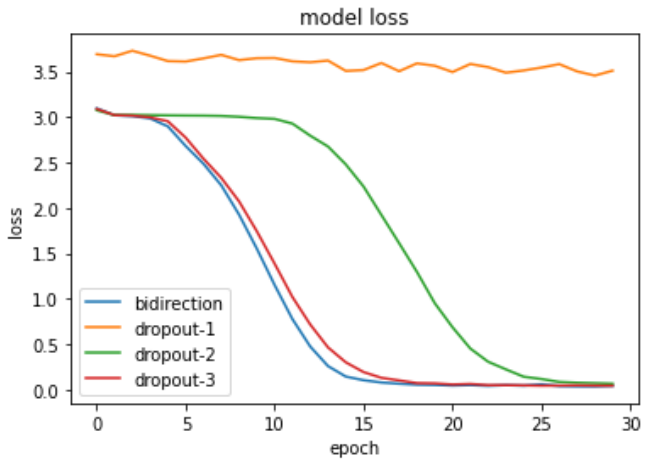
Further Comparisons

The traditional implementation of the dropout layers in a Neural Network, places it just after the Dense Layer. As seen in the following graphs, this becomes quite evident even with this system. Over subsequent change in number of epochs spent in training the model, the graphs of training and validation loss shows a fairly constant values for the traditional implementation. (*dropout 1*)

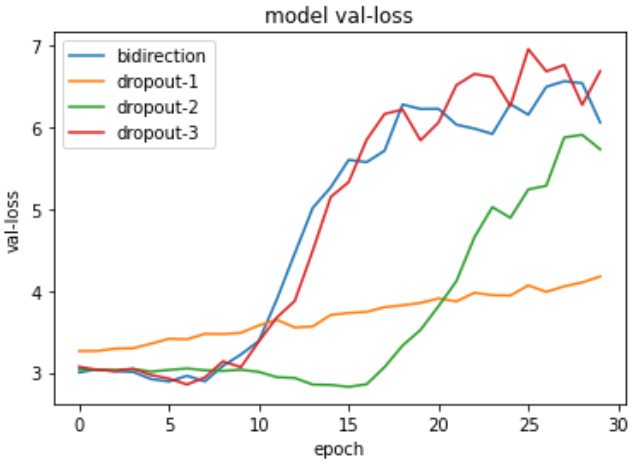
With the dropout layer positioned just before the Dense layer (*dropout-3*), produces training and validation loss closely resembling that of the control (*bidirectional*) curve. This shows that having the dropout layer at this position contributes next to nothing towards the goal of countering the overfitting issue.

When the dropout layer is positioned after every hidden layer (*dropout-2*), the approach worked best particularly for the lower values of epochs, up to around 12. Till this point, it was not only providing stable loss values for both training and validation sets but lower loss values. However, after the critical point of 12 epochs, the values showed a drastic change assuming a sigmoid structure just like the control (*bidirectional*) curve, but heavily skewed towards the right, i.e. higher epoch values, and finally merging with the control and dropout-3 curves at around epoch 25.

Thus it was evident that for an overall controlled state, the traditional approach proved to be the best, however at lower epoch values, positioning similar dropout layers between hidden layers, also serves as a viable alternative.



Comparison of loss in bidirectional, 1st 2nd and 3rd dropout implementation



Comparison of validation loss in bidirectional, 1st 2nd and 3rd dropout implementation

V. OBSERVATIONS AND INFERENCES

The phenomenon of overfitting (inferring bias), draws close parallels to human creativity. As social-being, we all have some likes or dislikes, and we always tend be bias towards the things we admire. This is reflected in artist's song writing as well, where degree of bias is shown towards certain emotions. The overfitting, in theory, is somewhat able to simulate that creative bias, resulting is far more 'human-like' lyrics being generated.

According to the over-fitted brain theory, organisms' brains face the same difficulty of overfitting to their daily stimulus distribution, resulting in poor generalisation and overfitting. The brain may recover the generalizability of its perceptual and cognitive capacities and improve task performance by imagining out-of-distribution sensory stimulus every night. [23]

Similarly, it is possible that the overfitting problem faced by our model, can be generalized using intermittent and finely tuned '*noise-injections'*, in the form of noisy or corrupted or faulty data. This is a common method implemented in handing overfitting in case of DNNs.

VI. CONCLUSION AND FUTURE WORK

We can now create our own mixtapes! The lyric generator we created is significant and interesting to us because we were able to dive deeper into NLP and neural networks to see what kind of abstract applications they have. In this day and age, when everything is becoming automated and enhanced due to machine learning, our study seems to align with the idea that the applications of NLP and data science can extend to any field imaginable. It's fascinating to see how interdisciplinary data science truly is and how it can bridge the gap between STEM and other distinct fields (in this case, music).

There is a plethora of ways we might enhance and build on this concept in the future. First and foremost, several of the difficulties highlighted in the preceding sections, such as the overfitting problem, may certainly be addressed, by noise injections. Despite the usage of GPU-accelerated neurons, the training time was fairly long, and this has to be addressed in the future too. Fine-tuning parameters is one course of action that may be taken to see if model performance can be improved, both from the LSTM and BERT sides of the application.

This concept might be extended to different literary styles and mediums, such as poetry, and with the correct dataset, the lyrics generator may evolve into a poetry generator. Adding further layers or establishing a concurrent LSTM to add pitch and rhythm to the lyrics is one avenue that can also be considered. As a result of which, we may be able to create a more comprehensive, healthy music generating engine.

VII. CITATIONS

Research Articles and Papers

[1] Gonçalo Oliveira, Hugo & Cardoso, Amílcar & Pereira, Francisco. (2007). Tra-la-Lyrics: An approach to generate text based on rhythm. Proceedings of the 4th International Joint Workshop on Computational Creativity. 47-55.

[2] Nguyen H. and Sa B., (2009). Rap Lyrics Generator. https://nlp.stanford.edu/courses/cs224n/2009/fp/5.pdf

[3] Gervás, P. (2013). Computational Modelling of Poetry Generation, Corpus ID: 35020911

[4] Logan, Beth & Kositsky, A. & Moreno, Pedro. (2004). Semantic analysis of song lyrics. 827 - 830 Vol.2. 10.1109/ICME.2004.1394328.

[5] Settles, Burr. (2010). Computational creativity tools for songwriters. 49-57.

[6] Mahedero, Jose & Martinez, Alvaro & Cano, Pedro & Koppenberger, Markus & Gouyon, Fabien. (2005). Natural language processing of lyrics. 475-478. 10.1145/1101149.1101255.

[7] Mayer, Rudolf & Neumayer, Robert & Rauber, Andreas. (2008). Rhyme and Style Features for Musical Genre Classification by Song Lyrics. ISMIR 2008 - 9th International Conference on Music Information Retrieval. 337-342.
[8] Pudaruth, Sameerchand & Amourdon, Sandiana & Anseline, Joey. (2014). Automated generation of song lyrics using CFGs. 613-616. 10.1109/IC3.2014.6897243.
[9] Santhoopa Jayawardhana, Sequence Models & Recurrent Neural Networks (RNNs), Jul 27, 2020. https://towardsdatascience.com/sequence-models-and-recurrent-neural-networks-rnns-62cadeb4f1e1
[10] SuperDataScience Team, Recurrent Neural Networks (RNN) - The Vanishing Gradient Problem, Thursday Aug 23, 2018. https://www.superdatascience.com/blogs/recurrent-neural-networks-rnn-the-vanishing-gradient-problem
[11] Md Tahmid Rahman Laskar, York UniversityUtilizing the Transformer Architecture for Question Answering. March 2021. DOI: 10.13140/RG.2.2.22446.23364
[12] Tarun Kumar Gupta, Khalid Raza, Chapter 7 - Optimization of ANN Architecture: A Review on Nature-Inspired Techniques, Nilanjan Dey, Surekha Borra, Amira S. Ashour, Fuqian Shi, Machine Learning in Bio-Signal Analysis and Diagnostic Imaging, Academic Press, 2019, Pages 159-182, ISBN 9780128160862. DOI: https://doi.org/10.1016/B978-0-12-816086-2.00007-2.
[13] Robert DiPietro, Gregory D. Hager, Chapter 21 - Deep learning: RNNs and LSTM, S. Kevin Zhou, Daniel Rueckert, Gabor Fichtinger, In The Elsevier and MICCAI Society Book Series, Handbook of Medical Image Computing and Computer Assisted Intervention, Academic Press, 2020, Pages 503-519, ISBN 9780128161760. DOI: https://doi.org/10.1016/B978-0-12-816176-0.00026-0.
[14] Chris Nicholson, A Beginner's Guide to LSTMs and Recurrent Neural Networks, 2020, https://wiki.pathmind.com/lstm
[15] Jay Singh, Activation function breakthrough, 2020, inblog.in/ACTIVATION-FUNCTION-BREAKTHROUGH-VOyvxhTELU
[16] Michael Phi, Illustrated Guide to LSTM's and GRU's: A step by step explanation, Sep 24, 2018. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
[17] Alejandro Saucedo, Beyond CUDA: GPU Accelerated Python for Machine Learning on Cross-Vendor Graphics Cards Made Simple, Nov 13, 2020. https://towardsdatascience.com/beyond-cuda-gpu-accelerated-python-for-machine-learning-in-cross-vendor-graphics-cards-made-simple-6cc828a45cc3
[18] Ramasamy, Lokeshkumar & Kaliappan, Jayakumar & Prem, Vishal & Nonghuloo, Mark. (2020). Analyses and Modeling of Deep Learning Neural Networks for Sequence-to-Sequence Translation. International Journal of Advanced Science and Technology. 3152-3159.
[19] Deep Shah, (2021). Song Lyrics Dataset, Version 5. Retrieved October 12, 2021, from https://www.kaggle.com/deepshah16/song-lyrics-dataset
[20] Jason Brownlee, A Gentle Introduction to Dropout for Regularizing Deep Neural Networks, December 3, 2018. https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/
[21] Rani Horev, BERT Explained: State of the art language model for NLP, Nov 10, 2018. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
[22] Raghav Aggarwal, Bi LSTM, Jul 4, 2019. https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0
[23] Erik Hoel, The overfitted brain: Dreams evolved to assist generalization, Patterns, Volume 2, Issue 5, 2021, 100244, ISSN 2666-3899. DOI: https://doi.org/10.1016/j.patter.2021.100244.

VIII. REFERENCES

[1] Yi Yu, Abhishek Srivastava, and Simon Canales. 2021. Conditional LSTM-GAN for Melody Generation from Lyrics. ACM Trans. Multimedia Comput. Commun. Appl. 17, 1, Article 35 (February 2021), 20 pages. DOI: https://doi.org/10.1145/3424116
[2] Jean-Pierre Briot and François Pachet. 2017. Music generation by deep learning—Challenges and directions. arxiv:1712.04371. Retrieved from http://arxiv.org/abs/1712.04371
[3] Jian Wu, Changran Hu, Yulong Wang, Xiaolin Hu, and Jun Zhu. 2017. A hierarchical recurrent neural network for symbolic melody generation. arxiv:1712.05274. Retrieved from https://arxiv.org/abs/1712.0527
[4] Potash, Peter & Romanov, Alexey & Rumshisky, Anna. (2015). GhostWriter: Using an LSTM for Automatic Rap Lyric Generation. 1919-1924. 10.18653/v1/D15-1221.

[5] Addanki, K., Wu, D.: Unsupervised rhyme scheme identification in hip hop lyrics using hidden Markov models. In: Dediu, A.-H., Martín-Vide, C., Mitkov, R., Truthe, B. (eds.) SLSP 2013. LNCS (LNAI), vol. 7978, pp. 39–50. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39593-2_3

[6] Enrique, A.: Word-level LSTM text generator. Creating automatic song lyrics with Neural Networks, June 2018. https://medium.com/coinmonks/word-level-lstm-text-generator-creating-automatic-song-lyrics-with-neural-networks-b8a1617104fb. Accessed 27 Feb 2019

[7] Enrique, A.: Automatic song lyrics generation with Word Embeddings, June 2018. https://medium.com/coinmonks/word-level-lstm-text-generator-creating-automatic-song-lyrics-with-neural-networks-b8a1617104fb. Accessed 27 Feb 2019

[8] Yongju Tong, YuLing Liu, Jie Wang, Guojiang Xin. Text steganography on RNN-Generated lyrics[J]. Mathematical Biosciences and Engineering, 2019, 16(5): 5451-5463. doi: 10.3934/mbe.2019271

[9] Naman Jain, Ankush Chauhan, Atharva Chewale, Ojas Mithbavkar, Ujjaval Shah, Mayank Singh. Bollyrics: Automatic Lyrics Generator for Romanised Hindi. https://arxiv.org/abs/2007.12916v1

[10] Gill, Harrison; Lee, Daniel; and Marwell, Nick (2020) "Deep Learning in Musical Lyric Generation: An LSTM-Based Approach," The Yale Undergraduate Research Journal: Vol. 1: Iss. 1, Article 1. Available at: https://elischolar.library.yale.edu/yurj/vol1/iss1/