Lyrics Generation using Recurrent Neural Network

Anant P. Srivastava[1]
Department of Computer Science
University of Texas at Dallas
Dallas, US
aps180006@utdallas.edu

Anshul Pardhi[2]
Department of Computer Science
University of Texas at Dallas
Dallas, US
arp180012@utdallas.edu

Ashwani K. Kashyap[3]
Department of Computer Science
University of Texas at Dallas
Dallas, US
axk190033@utdallas.edu

Ruchi Singh[4]
Department of Computer Science
University of Texas at Dallas
Dallas, US
rxs180057@utdallas.edu

***Abstract*—The influence of Machine Learning in the field of art has been ever-growing especially in the last decade or so. Composing lyrics using algorithmic programming is a great way to test the abilities and potential of a machine. Artificial Neural Network (ANN) is one of the most common buzzwords that one comes across in Machine Learning algorithms. It is modeled after the human brain and does carry a lot of potential to solve complex world problems. Recurrent Neural Network (RNN) adds a twist to a Neural Network. Unlike the feedforward Neural Network, RNN creates its internal state memory that helps in processing sequence of inputs. This is one of the key features of RNN that we realized, which could be used in generating or composing lyrics of a song. Since each sequence of input (words) is used to process the future sequence of inputs (words), thus generating lyrics using RNN made sense and gave persistence to generated lyrics. Using model evaluation techniques and a large dataset, the accuracy, and quality of songs were maintained and optimized.**

1 INTRODUCTION

In the project, we aim to generate new song lyrics based on the artist's previously released song's context and style. We have chosen a Kaggle dataset of over 57,000 songs, having over 650 artists. The dataset contains artist name, song name, a link of the song for reference & lyrics of that song. We tend to create an RNN character-level language model on the mentioned dataset. Using model evaluation techniques, the model is checked for its accuracy and is parameter optimized. The trained model will predict the next character based on the context of the previous sequence and will generate new lyrics based on an artist's style.

2 RELATED WORK

Before starting with this project, we were familiar with two types of learning in the field of Machine Learning:
1. Supervised Learning
2. Unsupervised Learning

As our interest lies more towards the former, we have chosen Supervised Learning for the scope of our project. With Supervised learning, we can see the actual/predicted label of a data instance after a complete training and can utilize the error function to calculate the difference between the actual class and the predicted class.

We came across applications of Deep Learning which is a subfield of machine learning, dealing with algorithms inspired by the structure and function of the brain called Artificial Neural Networks. Neural networks are made up of artificial neurons, similar in concept to neurons in the human brain. [10] These neurons are connected to each other, form huge connections, and the system works through the activation of these neurons. [10] We have studied the internal concepts of feed-forward neural networks. Feed-forward neural networks allow signals to travel one-way only - from input to output. There are no loops (feedback) i.e., the output of any layer doesn't affect the same layer. [11] Feed forward neural networks tend to be straight-forward, from input to output.

In lyrics generation, we must keep the context of the previous words to generate future words/characters. Since feed-forward neural networks don't allow the output of one layer to be used as the input for the same layer, we had to think of some algorithms which allow the feedback concepts. We came across the concepts of Recurrent Neural Networks. Feedback (Recurrent) networks can have signals traveling in both directions by

introducing loops in the network. [11] Computations derived from earlier inputs are fed back into the network, which gives them a kind of memory. [11] For our experiments, we have chosen over 57000 Song Lyrics Kaggle Dataset which contains an exhaustive list of artists and their songs. In order to be able to interpret the better results, we have built a python wrapper that allows the user to select the artist whom they want to generate the songs, or instead just simply to run the model on the entire global dataset.

3 RNN ALGORITHM

The main idea behind Recurrent Neural Networks (RNN) is to make use of sequential data. [5] The output of one layer is fed as the input to another layer and so on. The recurrent concept of RNN works such as the network performs the same tasks for each element of the sequence and the output depends on the previous computations. RNN keeps the memory that captures the information of all the previous calculations. [5] Unlike plain neural networks (which accepts the fixed-sized vector as an input and produce fixed-sized vector as an output), RNN accepts processing input of any length. We make use of the memory which it keeps storing previous calculations and feeding them into the next tasks of sequence and so on. RNN models work better on a large amount of sequential data.

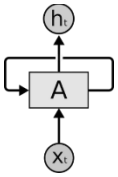

Figure 1: Figure showing the output of a layer goes as an input in RNN. [1]

The loop structure makes sure the output to be fed as an input to predict the next sequence. It allows the network to take the sequence of inputs.

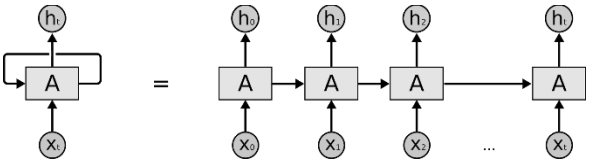

Figure 2: Figure showing the internal connection of RNN layers. [1]

According to the figure, first, it takes x_0 as input and feeds it to the first layer and then outputs h_0 which again is fed along with x_1, to the next layer. Similarly, the second layer generates h_1 as output; and h_1 and x_2 are the input for the next step. That's how it keeps remembering the context of the sequence while training.

3.1 The math behind the algorithm

Let's dive into the mathematical theory of RNN algorithm:

The RNN system contains a sequential input, a sequential output, multiple time steps and multiple hidden layers in the network. [2] It has a set of feed-forward neural networks because of time steps. The hidden layer values are calculated not only from input values but also from time step values and the hidden layer weights remain the same for time steps. [2] RNN algorithm accepts the input vector and output vector and the output vector values are calculated not only by the current input values but using the entire history of values we feed into the network. It keeps some internal memory that it updates at every time step.

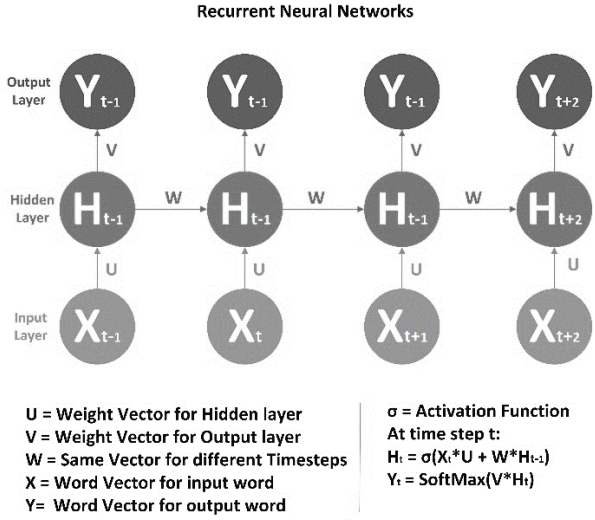

Figure 3: Figure showing the calculation of weight vectors at each layer in RNN. [2]

The above figure represents the basic concept of a recurrent neural network.

X - input vector
Y - output vector
U - Weight vector for the hidden layer
V - Weight vector for the output layer
W - Same vector for different time steps
We compute the three main matrices which work as the parameters of RNN class –

W_{xh} - matrix of input to the hidden state
W_{hh} - matrix of hidden state to the previous hidden state
W_{hy} - matrix of hidden state to output state

3.2 Forward Pass

According to the figure, we calculate the hidden layer timesteps values by the function:
H(t) = Activation Function (input*hidden layer weights (H) + W*H (t-1))

The output vector would be calculated by the function:

Y(t) = Soft-max Function (hidden layer weights (H) * H(t))

H(t-1) is the previous time step value. The activation function could be Sigmoid, Tanh or Relu. [2]

As we calculated for H(t), we can calculate for all other time steps.

U and V are weight vectors that are different for every time step. Initially, weight vectors are assigned randomly.

We feed-forward all the calculated values to the network. Once the feed forwarding is done, we calculate the error and backpropagate the error using the backpropagation algorithm. The cost function used in RNN is often the cross-entropy loss.

Cross-Entropy Loss: It measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverge from the actual label. So, predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

We can define the cross-entropy function as -

$$H = -y(i) * \log_p (y(i))$$

Where y(i) is the actual class label and p(y(i)) is the probability of predicted class label.

If we go deeper into the working of RNN, we have two separate RNNs, one is used to receive the input vectors and the second RNN is used to receive the output of first RNN as its input. It's all just the vectors that are coming and going out and some gradient flowing through each layer during backpropagation. [3]

3.3 Backpropagation through time

The backpropagation technique is the same as plain neural networks, but the current time step is calculated based on the previous time step, so we traverse the whole network back. We sum up the gradients for W at each time step.

We aim to calculate the following things –
1. The total error change with respect to the output
2. The output change with respect to the weight vectors

We calculate the gradient of the error with respect to the parameters U, W and V and then try to learn good parameters using gradient descent.

To calculate these gradients, we use the chain rule of differentiation, which is known as the backpropagation algorithm.

$$\frac{\partial \mathrm{J}t}{\partial \mathrm{V}} = \sum_t \frac{\partial \mathrm{J}t}{\partial \mathrm{Y}t} * \frac{\partial \mathrm{Y}t}{\partial \mathrm{V}}$$
$$\frac{\partial \mathrm{J}t}{\partial \mathrm{W}} = \sum_t \frac{\partial \mathrm{J}t}{\partial \mathrm{H}t} * \frac{\partial \mathrm{H}t}{\partial \mathrm{W}}$$
$$\frac{\partial \mathrm{J}t}{\partial \mathrm{U}} = \sum_t \frac{\partial \mathrm{j}t}{\partial \mathrm{H}t} * \frac{\partial \mathrm{H}t}{\partial \mathrm{U}}$$

Figure 4: RNN Backpropagation Chain Rule. [2]

Delta (Jt) is the gradient of the error with respect to the input to hidden vectors, hidden to hidden state vectors and hidden to output state vectors.

The chain rule expands more and more since W's are the same for all time steps. [2] We sum up the change in gradients at each time step for one training example.

4 IMPLEMENTATION

Keeping the concept of Recurrent Neural Networks in mind, we started the implementation of the model by defining certain initial configurations of the model to be followed throughout the training.

- Size of hidden layer nodes: In the model, the number of nodes in a hidden layer is taken as 100. Each node in a hidden layer takes a certain input and produces an output. After a certain activation, this output is served as an input for the next hidden state.
- Input sequence length: It is the length of the sequence to be fed as the input from the entire input string of lyrics during each cycle of forward-backward propagation (epoch).
- Pass: Pass is the number of iterations to be performed during training until the entire data string is covered as the input (separated as small input chunks).

- Chars to predict: Chars to predict is the number of the next sequential characters to predict after the complete training of the model.

In addition to it, a utility model is implemented to be used throughout the model. The utility model holds information of – the data, dimensions of the data, data size, tokenized dictionaries and is responsible for the following

- Tokenization: The method tokenizes each unique character in the data to a specific unique number and reverse-map the same number back to the original character.
- Preprocess: The method is responsible for taking input from the user given a list of artists and based on the user's selected artist, method preprocess the data from the entire dataset.
- Print Artist: The method prints the list of the available artist for which the songs are available.

The crux of the program resides in the module named 'RNN', which is required to be initialized using the configuration and utility modules. The RNN module is responsible for the following:

- Training: The method is responsible for the training on the dataset by executing the number of forward-backward propagation cycles defined in the configuration. With each iteration, the method update weights to minimize the loss computed during backpropagation.
- Forward Pass: The method runs a forward propagation given an input chunk of data and a certain hidden state at a time and computes a cross-entropy loss based on the predicted output.
- Backward Pass: Given a certain hidden state and predicted outputs, the backward pass is responsible for computing the change in weight matrices by back-propagating over the predicted output at every timestamp.
- Generate Sequence: Given a specific hidden state and a starting input character, the method generates the next sequence of a specific length using the trained weights updated after the training of the model.

5 RESULTS & ANALYSIS

We trained the RNN on the lyrics of the selected artist (here, an artist named Usher) and extended our model to predict sequences of predicted song lyrics based on previous songs of that artist. We trained our model with more than 57,000 songs, 100 hidden layers, 10 million iterations and predicting the next 200 characters of the sequence. After every 50th iteration, we check the cross-entropy loss. We investigated the cross-entropy loss and based on our data and model parameter configurations, observed how the cross-entropy loss decreases. The initial loss was 104.99467761 and it decreased to 40.42472864 after 10 million iterations. Some snapshots of the cross-entropy loss with iterations are captured below.

Iteration	Cross Entropy Loss
1	104.99467761
161900	69.86752385
1000000	64.81955973
2836750	56.60050261
4064350	50.17084192
5332750	47.04216629
8213200	44.68927431
9999850	40.42472864

Figure 5: Cross entropy loss with each iteration

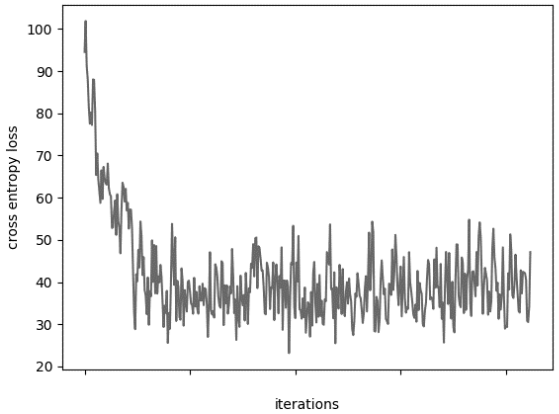

Figure 6: Cross entropy loss with each iteration

As we can see from the table and the graph above, the cross-entropy loss decreases with the number of iterations. As a result, the generated words make more sense and relate to their past words. The following song lyrics were generated on training the RNN.

Iteration 50 -
"oeem rn(tg nMserenk o iiTs iyet,ehs itoemitusii s fo yree esnh
eige hoy tyf a)esn uleetrnnoyhoe e yeeeem eiyseno
s
ne neseeit eeruey
yiemof e e e
el i eeeeat
oeenemneot s nt uM nglumirsoey eon(',sLh"

Iteration 2836750 -
"ten in your love alling as. I'm
Everyed oh

For to you're a-blamby and my my Myer to that a a dop not I high do
What Athe brirdyy while you with you all my on hamb thing, leacep on the rockay”

Iteration 9999850 -
“So goodbye
Don't die the even babaty whispel the parted snarsic
You said you

Sanessings that feel shown
Just pullilleh, this heat goad turna better nection the wordlde lid.....
A
Don't all”

6 CONCLUSION & FUTURE WORK

In the project, we looked into the concepts of RNN and after thorough analysis and consideration, we observed certain trends and concluded that though RNN keeps the context of the previous input sequence, in order to generate the next output sequence and is able to generate few sequences properly, RNN alone is not sufficient enough to automate lyrics generation. RNN suffers from a term called “exploding/vanishing gradient” in which with each iteration, the changes in the weight matrix are either significantly huge or just almost negligible. Such a problem results in inappropriate weight changes which lead to the poor prediction of the next sequence.

In order to overcome such a problem, a technique called Long-Short-Term-Memory (LSTM) is employed so that during backpropagation the change in weight remains in a considerable range. LSTM uses the concepts of an internal mechanism called gates, which helps in regulating the flow of information. Using these gates, LSTM knows which information to keep in a sequence, which to neglect. [7]

The key concept of LSTM is the cell state and its various gates. The cell state is responsible for transferring all the relevant information down to the sequence chain. [7] As the information is carried via a cell state, the new information gets added or removed using gates. The three gates in an LSTM network includes:

1. Forget Gate: Responsible for throwing irrelevant information.
2. Input Gate: Responsible to update the cell state with some input values.
3. Output Gate: Responsible for computing the next hidden state.

Though RNN’s are good for processing sequential data, they are prone to ‘Short Term Memory’. LSTM is an alternate way to get rid of such a problem. In 2018, Bill Gates called LSTM a “Huge Milestone in advancing artificial intelligence” when OpenAI bots were able to beat humans in DOTA 2 game. [8]

For more accurate results and a significant decrease in the cross-entropy loss, we can include LSTM implementation in our model to make it better.

7 REFERENCES

[1] Olah, C, (2015). “Understanding LSTM networks,” 2015, URL https://colah.github.io/posts/2015-08-Understanding-LSTMs/
[2] Danish, Irfan, “A brief summary of maths behind RNN”, URL https://medium.com/towards-artificial-intelligence/a-brief-summary-of-maths-behind-rnn-recurrent-neural-networks-b71bbc183ff
[3] Banerjee Survo, (2018). “An Introduction to Recurrent Neural Networks” 2018, URL https://medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912
[4] Cross Entropy. “ML Glossary”, URL https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html
[5] Britz, Denny (2015). Recurrent Neural Network Tutorial, Part 1- Introduction to RNNs, 2015. URL http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
[6] Raval, Siraj, (2017). “Artificial Intelligence Education”, URL, 2017 https://www.youtube.com/channel/UCWN3xxRkmTPmbKwht9FuE5A
[7] Raval, Siraj, (2017). “Recurrent Neural Network, The Math of Intelligence”, 2017 URL https://www.youtube.com/watch?v=BwmddtPFWtA
[8] Nguyen Michael, (2018). “Illustrated Guide to LSTM’s and GRU’s: A step by step explanation”. URL https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
[9] Rodriguez, Jesus (July 2, 2018). "The Science Behind OpenAI Five that just Produced One of the Greatest Breakthrough in the History of AI". Towards Data Science. Retrieved 2019-01-15.
[10] Olafenwa John, (2017). INTRODUCTION TO NEURAL NETWORKS, https://medium.com/@johnolafenwa/introduction-to-neural-networks-ca7eab1d27d7
[11] Dontas George, (2010). difference-between-feed-forward-and-recurrent-neural-networks. URL https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21