Adam Villarreal
Math 373

Text Generation using LSTMs

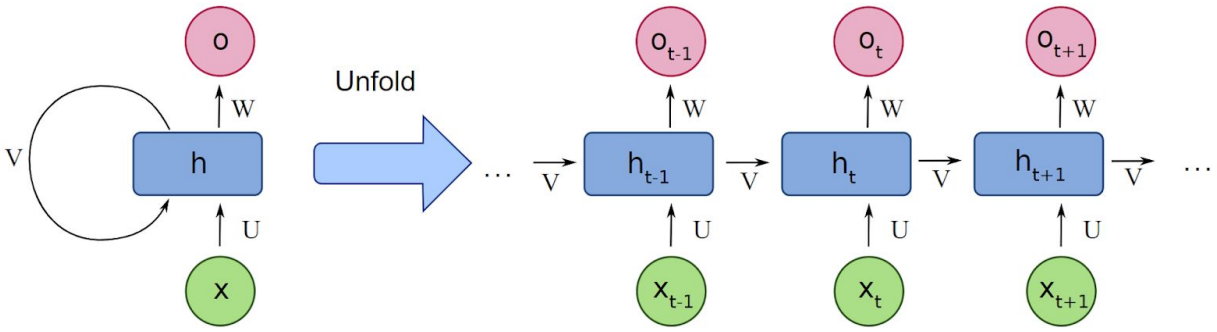
Project Goal:

In this course, we have learned how to create Statistical/ML models that use non-sequential data, such as images or variables for the use of regression tasks. These are considered non-sequential since the dataset presumes that each sample share no relationship in terms of time. Sequential data presumes that the samples are connected through time, and text is an example of this. An example is this sentence, “The doctor recommends to eat an apple a day.”, here we see that the verb ‘eat’ connects the subject ‘doctor’ and the object ‘apple’.

In this project, we explore sequential modeling more in-depth and how we can predict the next word from a sequence of words. For example, if the sequence is, “In this morning, I usually drink ____”, the model should predict ‘coffee’ or ‘tea’. I will gather great works from different authors and have the model learn from the text for the use of text generation.

Domain Background:

Recurrent Neural Networks:



The image is a Simple Recurrent Network or Elman Network

Definitions:

x_t: input vector at time-step t
h_t: hidden layer vector (hidden state) at time-step t
o_t: output vector at time-step t

b_h : bias used in the creation of the next hidden state
b_o : bias used for the output

U : weight matrix from input to the hidden state
V : weight matrix from hidden state to the next hidden state
W : weight matrix from hidden state to output

All weight matrices and biases stay the same throughout each time step.

Mathematical View:

$x_t \in R^n$
$h_t \in R^d$

$b_h \in R^d$
$b_o \in R^k$

$U \in R^{d \times n}$
$V \in R^{d \times d}$
$W \in R^{k \times d}$
(W can be the weight matrix that connects h_t to a fully connected layer)

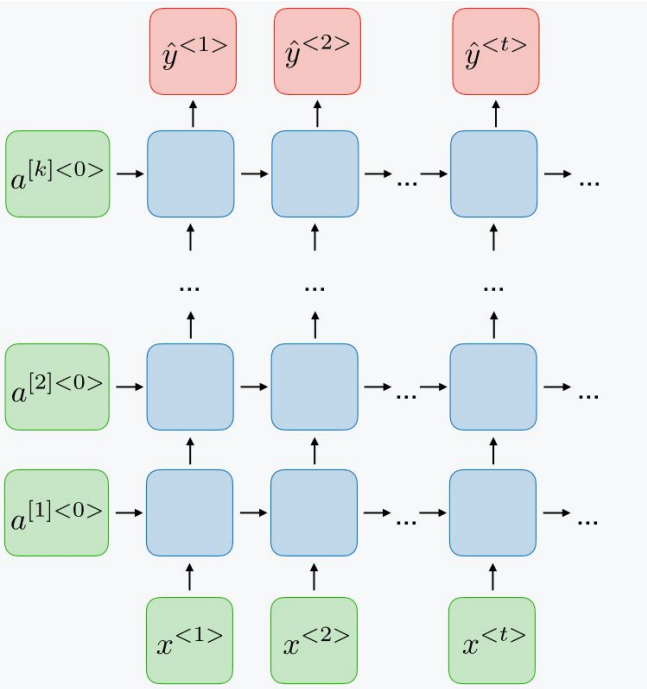
$\Phi : R \rightarrow R$
(Activation Function between layers: tanh, relu, or sigmoid)

1. Getting the hidden state at time-step *t*
   $h_t = \Phi(b + V h_{t-1} + U x_t)$

2. Applying a fully connected layer at each time step
   $o_t = \Phi(b_o + W h_t)$
   (Usually, the activation function here will be softmax if you are predicting k-classes at each time step)

3. After calculating the output, the network will repeat these steps for the next time-step

Deep RNNs:



The image is a Multiple-Layer RNN

Just as deep neural networks have multiple layers, RNNs can also have multiple layers. If the number of layers increases, the model can learn more interesting patterns. However, it becomes more computationally expensive, so you must balance your resources with the amount of time you would like to train the model. The intuition is the same, it just increases the number of hidden states per time-step.

Vanishing and Exploding Gradients

The backpropagation algorithm used in RNNs is different from feed-forward neural networks since each gradient of the loss has a dependency on all the inputs in the sequence through time. The method of backpropagation is called Backpropagation Through Time or BPTT for short. To explain this more in-depth, here is a lecture that was given at MIT that explains it quite well. I encourage you to watch the entire lecture if you have time, it is great at giving the fundamentals of RNNs.

Vanishing Gradients are when the gradients get close to 0 as they are calculated through time. Simple RNNs, such as the one we have looked at, have vanishing gradients over a certain number of time-steps in the sequence. Therefore Simple RNNs are not able to have Long-Term Dependencies after a certain number of time-steps, or rather it has difficulty remembering the data from the beginning of the sequence. Gated RNNs such as LSTMs and GRUs have risen to tackle the Vanishing Gradient problem, and LSTMs are used in this project.

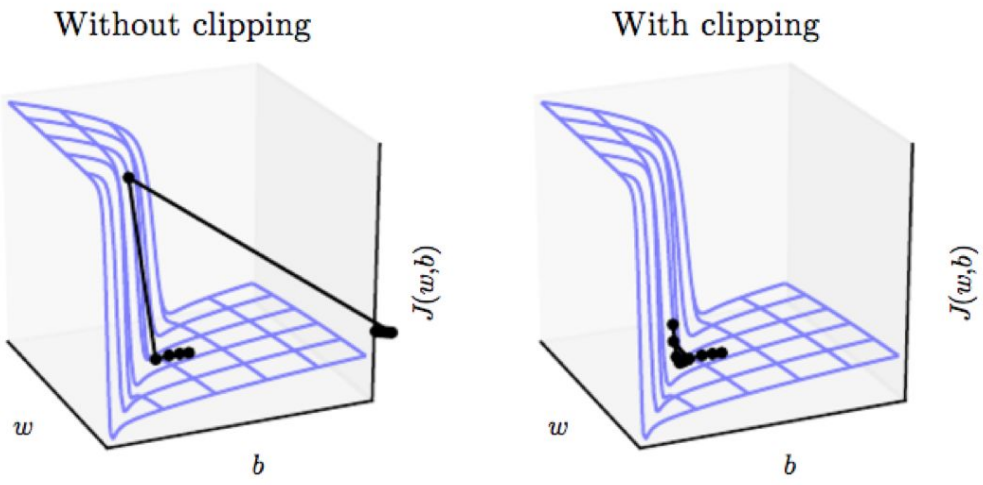
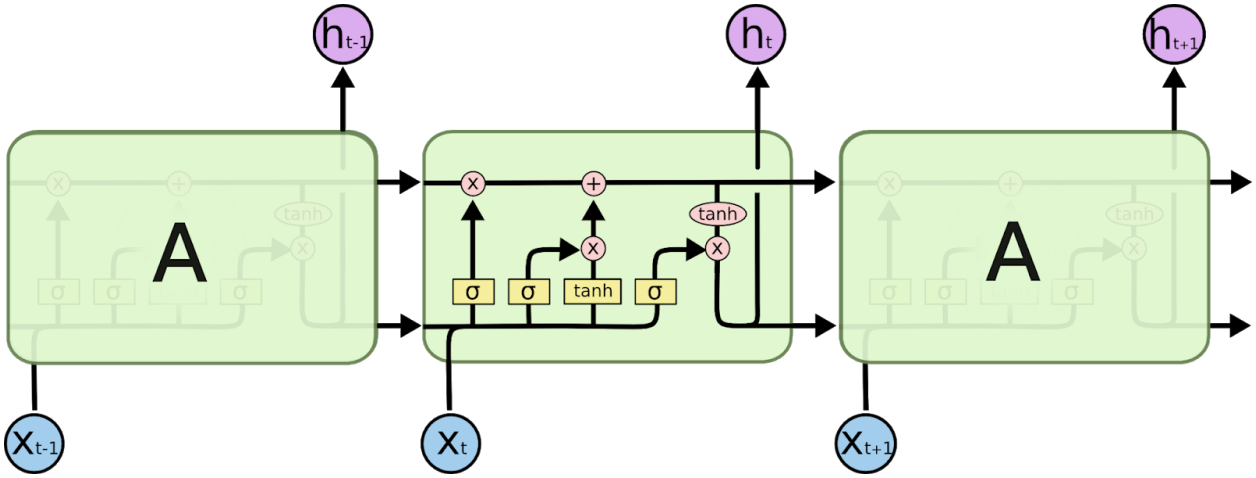


Figure 10.17 from Section 10.11.1 of Deep Learning Book

Exploding Gradients are when the gradients become too large, that when an optimization step occurs, it can step so far that the model will start to converge in a different region. The solution for Exploding Gradients is to clip the gradients to prevent it from exploding. One option is to clip the norm of the gradient, and you can read more about this in [Deep Learning Book section 10.11.1] by Ian Goodfellow.

Solution of LSTMs



A Look into LSTM cell and it's gates

As said before, gated RNNs are a solution to the Vanishing Gradient problem. LSTMs and GRUs are gated RNNs, and they attempt to retain information throughout a sequence. In each LSTM cell, it outputs both a cell state and a hidden state, the hidden state reacts as normally but the cell state only transfers through the cells. What LSTMs do is to learn what information needs to be passed through into the rest of the sequence and what information needs to be restricted.

There are four gates in the LSTM:

1. Forget Gate - Takes the input and previous hidden state and determines what to remember

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

2. Learn Gate - Decides what new information will be stored in the cell state

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$N_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

$$Output: \ N_t i_t$$

3. Remember Gate - Brings Learn Gate and Forget Gate to update the cell state

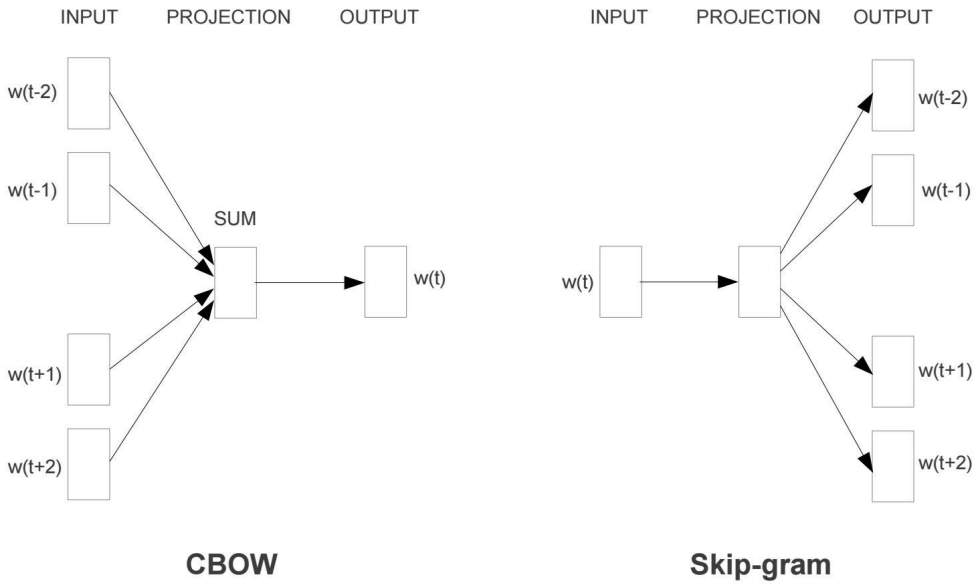
$$C_t = C_{t-1} * f_t + N_t * i_t$$

4. Use Gate - Outputs the hidden state

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$

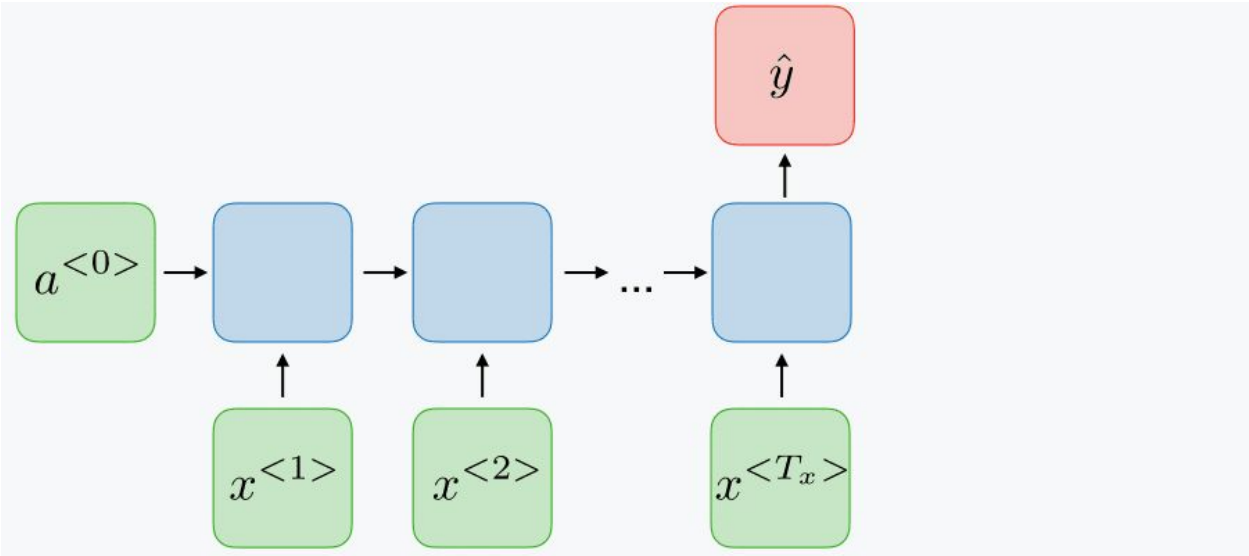
Skip-Gram Word2Vec



Skip-gram Word2Vec

One of the issues with early NLP problems were the dimensionality of representing individual words as a one-hot encoded vector. The problem is that they can hold a lot of unnecessary memory and can be computationally heavy when weights are connected to the vector. What Skip-gram Word2Vec does is that it outputs a unique vector for each word and decreases the dimensionality. It does this through an embedding layer or the weights of a feed-forward network. We can get a unique representation of a word through the rows in the weight matrix in a feed-forward network because the input is a one-hot encoded vector being multiplied against a weight matrix. This will just output the row of the weight matrix that corresponds with the 1 in the input. There is a great conceptual overview here and the original paper is here.

Creating my LSTM:



Many to One RNN

From the image above, my neural network will be structured like this. Each input for a time-step will be a word, and the output *y* will be the next word after time-step *t* in the sequence. Using the same example from the beginning of the paper, our sequence will be:

sequence = [In, this, morning, I, usually, drink]

Y = [coffee]

Using Cross-Entropy Loss, we will measure the difference between the predicted output with the correct output, in which the network will learn the patterns between the input and the output. There are outputs given at every time-step, however, we will only focus on the last time-step *t*, where we take the loss and backpropagate from there back into the network.

Data:

First Dataset

The first dataset includes these Great Works:

1. Karl Marx's Communist Manifesto
2. Friedrich Wilhelm Nietzsche's Beyond Good and Evil
3. Friedrich Wilhelm Nietzsche's Thus Spoke Zarathustra
4. Søren Kierkegaard's Selected Writings, including Fear and Trembling

After getting the texts online, I hand-cleaned it to remove repetitive new lines, titles of chapters, commentaries, footnotes, and anything that did not contain the writing of the authors. After cleaning the texts, I concatenated the texts together and then began preprocessing it for the model. Theoretically, since all of the data is used to train the network, the proportion of how the model will be influenced by the authors is equal to the proportion of the author's writings in the dataset. Also, the disadvantage of this dataset is that the style of writing between works is different so the generated text will not be as coherent.

Writings	Contribution to Dataset
Beyond Good and Evil	26%
Thus Spoke Zarathustra	39%
Kierkegaard's Writings	30%
Communist Manifesto	5%

All of these works are in the Public Domain

These texts are provided by Project Gutenberg

Second Dataset

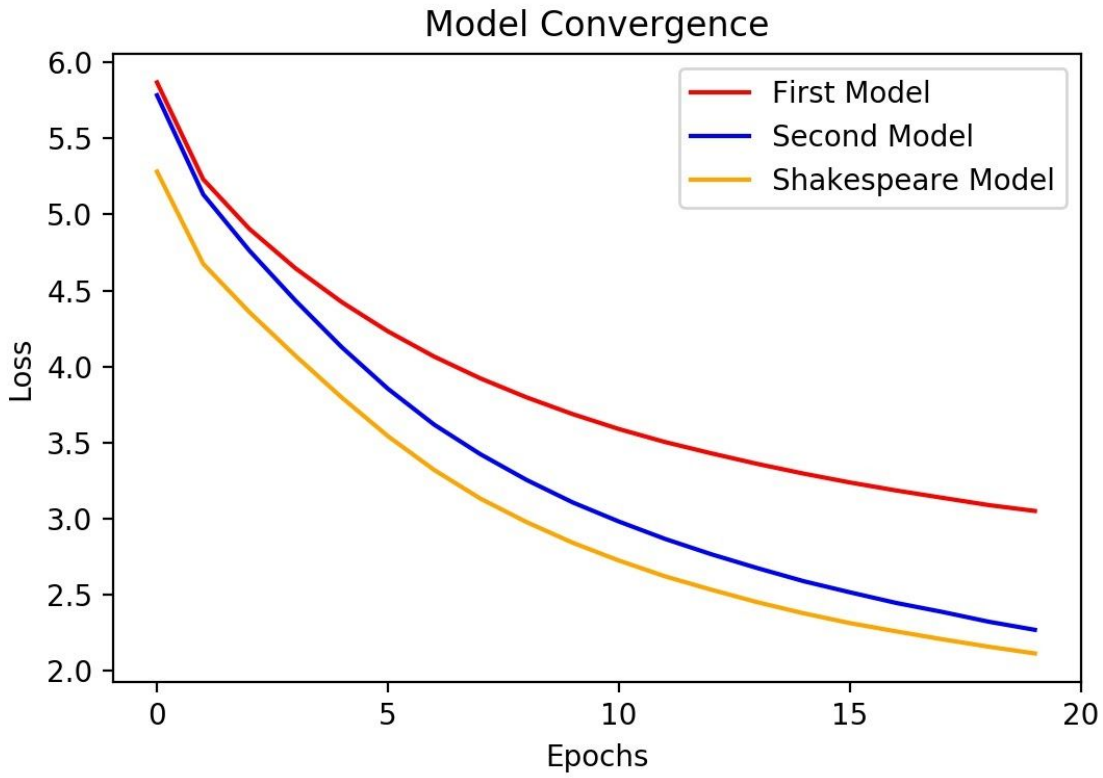
Andrej Kaparhty's famous article, The Unreasonable Effectiveness of Recurrent Neural Networks, gives an introduction and overview of the success of RNNs. One of the projects he highlights is using Shakespeare for text generation. He concatenated the works of Shakespeare into one file and uploaded it to this GitHub repo. Since it was already cleaned, I was able to preprocess the data right away. The advantage of this dataset over the previous one is that the text contains only one author with the same writing style. Therefore the generated text will seem to be more uniform in regards to the style of writing.

Results:

Hyperparameters shared between models:

- Batch Size - 128
- Sequence Length - 20
- Epochs - 20
- Number of Layers - 2
- Learning Rate - 0.001
- Embedding Dimension - 300

The only hyperparameter that changed was the number of hidden features that were outputted from the LSTM into the fully-connected layer.



Each Model's Convergence within 20 Epochs

Both the first model and second model were trained on the first dataset. The First Model was trained using 256 hidden features, and we can see that it performed the worst. The Second Model was trained using 512 hidden features and it performed significantly better. After seeing the beneficial effect of increasing the number of hidden features, the Shakespeare Model was trained with the same hyperparameters of the Second Model. However, since the Shakespeare Model was trained on the second dataset, the model's performance can be explained by the dataset itself. My understanding of why it performed better was either the smaller size of the dataset or because there was only one author.

Generated Text:

First Model

Truth exclusively on the pennies of the spirit-- the antithesis severity of production-- the tyrant, the sole survivors- organisation of a aristocratic mode of production and the grammatical instinct of an aristocratic cruelty. there is the object of the world, the unforeseen decreases. But the same nation has no country in his own image, for the participants beheld it desinteresse, and the crowding had been rub.

The man of the earth, however, became absorbed in the earth.

The ass, however, here brayed ye. but he was silent, the
surmounter of all keys, the murderer of gravity!

Thou hast reaped from the way, the bees, however, is the seats of
existence.

Verily, I recognise the forest. then did I ask you: I am not
my neighbour’s horse!

I am Zarathustra the godless.

And I answered: “I am I pray, and clambering apes; but the higher men have
always sunk!

Ye creating ones, the worst of all my poetisation and my love; for I should have
disavowed my soul.

Verily, I am asking unto you: a dog- sayer. but I am not the leading
of my charity and a tyrant, but the most courageous- day-- that
which renders the loftiest testament would be conducted.

In the same place the old testament I prefer to do so that you
expose yourself to assert:" come hither!"

Second Model

Truth! Hath it ever been the best and just?

O Zarathustra, I am not a butcher- reader? a haven weary, or
discord.

The ass, however, here brayed ye- a.

The destroyer of a thousand years glitter with me, but as sacred riddles. But it is probable that goethe thought differently. The same holds true of the sufferer, the mistake of the proletariat-insurrection and arrogance; or whether we know, in all innocence, and I feel rightly, and to be allowed to know. The effect in this inviter is the absolute course of the celebrated sex, as the leading interests of labour, the fearfulness class.

In political practice, therefore, in the form of the working class, who, in short, in proportion as the emotions of existence, in order to thwart itself into a social structure and cordiality of all the labour of production and of exchange, trade with the colonies, the increase in the conditions of bourgeois society, in short, in order that to be master and disclose(riddles which comes at all times to do wrong, or" national" will, that is, to transform her dress and to wash them completely sufficient perjured for the very opposite of life-- for it is really that it is not so paltry, that is not intentional, that is, in effect, a rich and magnificent gross common masquerade in the grand style, partly and commercially, and also in their criticism of german taste. In the middle ages, however, are more easier, and more condemned for power. presupposing always, historically considered, and epochs for action.

The communists disdain in favour of the world- market, and therewith hardly on earth obtained, without a realm of the forlorn hope, and that the foundation of the present- market- moral creatures, and the pirate after them, the beguiling and magnificent

Shakespeare Model

Romeo: glut turn: bitterest brick brick brick guiltless congeal'd guiltless bitterest congeal'd congeal'd congeal'd.
cursed be a lovely fellow old to draw;
which in the lethe of the barren rosaline
doth make the same up down, even so oft,
as I am galled and leave.

Camillo:
I do owe it.

Camillo:
I do not know the gods.

Coriolanus:
why, what is it?

Sicinius:
I would they were barbarians-- obstinate,
I will not do't, and teach him back with thee.

Romeo:
what is my crown?

Nurse:
ay, but not change a mile.

Mercutio:
o lamentable friar! o injurious wretch!
that gallant death hath closed so far a child?

John of Gaunt:
my father bids me well, i will not be.
where are you sewing, because you do me so.

Conclusion:

After exploring the usage of sequential modeling and the importance of sequential data, LSTMs can produce state-of-the-art results. We can see that in the process of text generation, it has the availability to capture relationships between words that output text that is coherent. However, from my investigation, it was not perfect, since it failed sometimes to continue coherency throughout the sentences. Also, we would like to see if any decisions can be made about the corpus that can influence the desired output. For future work, I would like to investigate what factors can lead to better coherency throughout the entire outputted text. Also, the use of RNNs can lead to music generation and I would like to investigate how an RNN can replicate melodies from the input data.

Useful References:

1. Christopher Olah: Understanding LSTM Networks
2. Ian Goodfellow, Yoshua Bengio and Aaron Courville: Deep Learning Book Chapter 10
3. MIT 6.S191 RNN Lecture: Recurrent Neural Networks
4. MIT 6.S094 RNN Lecture: Recurrent Neural Networks for Steering Through Time
5. Stanford CS231n RNN Lecture: Lecture 10
6. Andrej Kaparthy: The Unreasonable Effectiveness of Recurrent Neural Networks