**Report for Assignment 4:**

**Architecture:**

**Pre-processing:**

-First, we begin with loading the ‘tiny-shakespeare.txt’ document for preprocessing. We use **load\_doc** function for that.

-After that we ‘clean’ the document. We use ‘.split()’ to split the words into a list. Then we use the join function to obtain one long sequence of characters by joining those words in the previous list together.

-Creating input sequences: our input sequence will have 13 characters. We make an empty list (sequences) and iterate through all characters beginning with the 13th character (or the length of the input character). We then start appending those characters into the list (sequences) we created. The last character will be our output character.

-We use **save\_doc** to save the sequences into a text file (sequences.txt)

**Training:**

-First, we use the **load\_doc** function to load the sequences.txt and split the texts by new lines.

-Then we sort the text by unique characters and create a dictionary of values, equating each unique character to a number. (0-64). **There are 65 possible characters.** (new line and ‘space’ included)

-Then we convert each character in each line to a number (0-64) corresponding to its unique mapping and add it to our ‘sequences’ list.

-We then convert the list of sequences into an array and select all columns (except the last) as our input (X) and we select the last column as our output (y).

-Then converting the X array into a one-hot vector, we use the keras to\_categorical (for each row in X). Then we convert it back to an array. Now we have an array of the one-hot vectors. We also do the same for y.

**Model:**

-We use a layer-by layer model, Sequential(). We will have two layers, the first **an LSTM layer with 512 units and input shapes 13 and 65.**

-The second layer is a Dense layer with 65 units and softmax as the activation function.

- I added **Batch Normalization** between the LSTM and Dense layer but the training time was approximately the same with approximately the same amount of loss. So my results are without batch normalization, however, I left the line of code in.

-We will be using cross entropy loss and the optimizer ‘adam’.

-We run this model for **50 epochs** (which took a considerable amount of time on CPU) and with a **batch** **size of 128**.

-We then save the model and mapping to files.

**Text Generation:**

-The **generate\_seq** function takes a sequence of characters and produces a sequence of characters of fixed length. It encodes the characters into integers, which then become one-hot encoded (using to\_categorical). Then using model.predict\_classes() we are able to predict the next characters in the sequence. Then all that is decoded, and the sequence of fixed length is produced.

-We use the **generate\_seq** function with the saved model and mapping to generate the sequence, with specified starting characters, sequence length and projected character length

**Network Sample Prediction example starting with ‘First Citizen’:**

First Citizen: We have put you out: But would seem strange unto him and the king's son, For that I was wont to conquer others, Hath still and bearing by the part of Edward's; For so your doom in this remorse. MARCIUS: Hang all the mean before him, And be amout for Rome, as I am, royal every oritone: We two days make thee straight And happy moth'd upon the cheek of death to me that she has ever been, One of our souls had carried, so I did I mistake my heart; poor our friends are fled to watch Both are asles.

**Loss after 50 epochs:** 0.5937. (accuracy = 80.45%)

**BONUS-----------------------------------------------------------------------------------------:**

-The **architecture** is **similar to** the original except for a few implementations:

-We thoroughly clean the document, including removing punctuations and making it all lower case, using the **clean\_doc** function.

-There are 12670 possible words.

-The Tokenizer() function is used to convert the list of words into a list of integers, right before converting the sequences into an array.

-We use an ‘Embedding’ Layer with the first argument being the number of unique words (vocab\_size) and input length of 5 with 50 embedding vectors.

-We also have two **LSTM layers of units 256 and 100**.

-Lastly, there are two Dense layers of units 100 and ‘vocab\_size’ with activation functions relu and softmax respectively.

-We train for **120 epochs** with a **batch size of 128**. Most vector amounts are chosen through trial and error and due to training time.

-An obvious consequence of completely cleaning the text is that the resultant prediction has no punctuation.

**Network Sample Prediction example starting with ‘First’:**

citizen perhaps such time were in a thimble we kill thee severe whom i could speak a deed in every business as i presume that for because they shortly have we put out his eyes wherein thou darest to thee so looks in aught my breast have not only leave for sir to see your grace for their presumption but water lives by present tapestry in being froward arrived to him duchess of york aumerle hath court these braves of him when he shall bring her before confess was not many message as they have endured the king and i that have spoken do with thee even on this fellow gentlemen go get you gone to the drunkards which with absence menenius whos fetch out an gates his lives blemishd twain out and when they enter were my daughters son if ever thy husband be so noble for they are froward petitioners into your guests he might have made his health and two my kinsman norfolk get it this word the ruins ont makes not the greeks will have beheld brutus she straight clarence what offence coming petruchio signior attorneys is no houses to no more accident cannot build the ooze of the world than a wellgraced and full suck within one black land since i find a bold divine apricocks prospero maids montague darkly till these death cannot prove with barnardines coriolanus no remedy sir so how what whats a madness angelo gave me to marcius then about the advantage duke vincentio not grieved for you news biondello do you fear assurance and your love menenius so thrive ill hammer you was a roman by the way but rather but so become my sovereign and another is not thyself richard gentlemen canst thou aid it for leontes thou hast spoke before her excuse paris o shes within behind by beasts gloucester and gentle good morrow provost tranio grumio we are not unanswerd tis guilty to me i mean to shift your deathbed and to jest gentlemen too pompey i pray not your silence you sing your curse this teeming often revels with oaths here in your offence you shall have three obscene the land i am bound to see me too much able the extreme gate of this one thy shame make us to richmond and let him have a brothers friend if not your katharina do thee and that you do pompey he shall not showd us it is true pitiful couple sir in any knees which i have banishd draw so indirectly on this man i do beseech you do ever not be gone i will to part you as messenger this fellow is a woman and a miscreant built in his love a liberal people and thither lost his fearful mistress you have always armd with it provost here lie to gain angelo what said a piece we say to do this ear has sent by mine life and this begot having stayd nor longer Antonio

**Loss after 120 epochs: 1.0241**

Citations: <https://machinelearningmastery.com/develop-character-based-neural-language-model-keras/>

<https://towardsdatascience.com/simple-character-based-neural-language-model-for-poem-generator-using-keras-8295f52ff5c2>

<https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/>

<https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/>