**SPL-1 Project Report, 2019**

**Implementation of Neural Network**

**And Deep Neural Network**

**Course: Software Project Lab I**

**Course no: SE305**

Submitted by

**Sadikul Haque Sadi**

**BSSE Roll no: 1003**

**BSSE Session: 2017-2018**

Supervised by

**Prof. Dr. Md. Shoyaib**

**Designation: Professor**

**Institute of Information Technology**



**Institute of Information Technology**

**University of Dhaka**

**29-05-2019**

**Table of Contents**

1. Introduction 1

1.1background study 1

1.2Challenges 1

1. Project Overview 1
2. User Manual 1
3. Conclusion 1
4. Appendix 1

References 1

***I*ndex of Figures**

**Figure 1: Neural Network 1**

**Figure 2: Recurrent Neural Network(RNN) cell 2**

**Figure 3: Gated recurrent Unit (GRU)cell 3**

**Figure 4: Long short term memory(LSTM) cell 4**

**Figure 5: Sigmoid activation function 6**

**Figure 6: tan hyperbolic activation function 6**

**Figure 7: Rectified Linear Unit activation function 7**

**Figure 8: softmax activation function 7**

**Figure 9: preparing dataset 9**

**Figure 10: Forward-Propagation in RNN 9**

**Figure 11: Forward-Propagation in LSTM 10**

**Figure 12: Forward-Propagation in GRU 10**

**Figure 13: Backpropagation. 11**

**Figure 14: Sample input dataset. 11**

**Figure 15: Sample output. 11**

**Figure 16: text predicting. 11**

**1. Introduction**

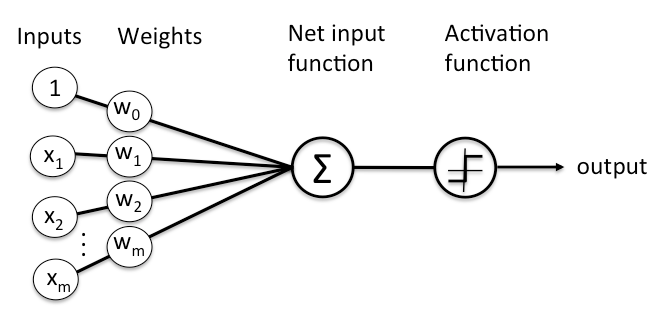
Neural networks are set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. i.e. they use the processing of the brain as a basis to develop algorithms that can be used to model complex patterns and prediction problems. They learn to map a set of inputs to a set of outputs from training data. Among different kinds of neural networks **recurrent neural network**, **gated recurrent unit** and **long short term memory** are mostly used and well known models.

NNs are composed of a large number of highly interconnected neurons working in unison to solve a specific problem. Like people, they learn by example. A neural network is configured for a specific application, such as pattern recognition or data classification or speech or character recognition or natural language processing etc.

**1.1 Background Study**

The architecture of neural network

Neural networks consist of input layer, interconnected hidden layers(neurons) and an output layer. The patterns they recognize are numerical, contained in vectors, into which all real world data i.e. images, texts etc. can be translated. There’s also some activation functions common to all kind of neural networks like- tanh, Relu, softmax, sigmoid function and many others.

 Figure 1: neural network architecture

**Different kinds of Neural Network**

My project is about text generating and there’s some neural networks which can do this job. Like-

* + - Recurrent Neural Network
    - Gated Recurrent Unit
    - Long short term memory

**Recurrent Neural Network**

The idea behind RNNs is to make use of sequential information. In a traditional neural network, we assume that all inputs (and outputs) are independent of each other. But for many tasks that’s a very bad idea. If you want to predict the next word in a sentence you better know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a “memory” which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps (more on this later). Here is what a typical RNN looks like:



# Figure 2: A recurrent neural network and the unfolding in time of the computation involved in its forward computation. Source: Nature

**Gated Recurrent Unit**

GRU is a gating mechanism in recurrent neural network, aims to solve vanishing gradient problem which comes with a standard RNN. GRU can also be considered as a variation on the LSTM because both are designed similarly and, in some cases, produce equally excellent results.

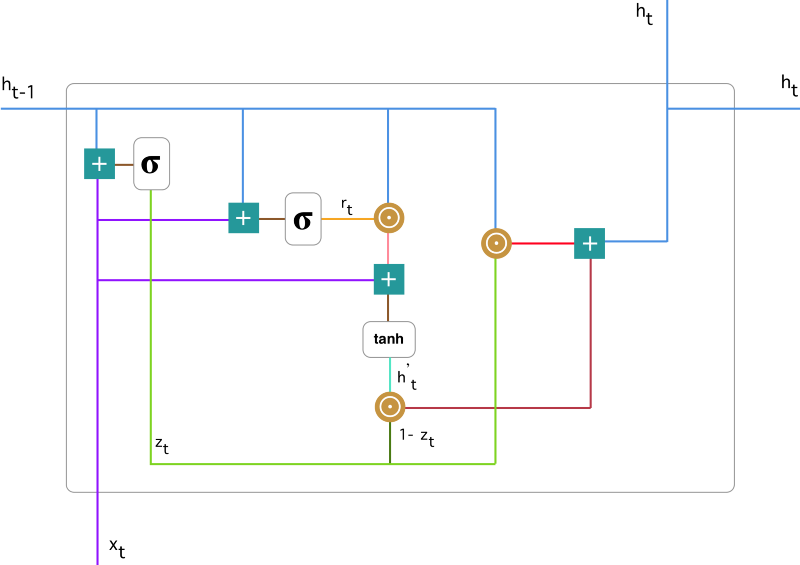
GRU uses, so called, **update gate(zt) and reset gate(rt)**. Basically, these are two vectors which decide what information should be passed to the output. The special thing about them is that they can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction. A single GRU cell looks like-

Figure 3: gated recurrent unit

**LSTM network**

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

A LSTM cell consists of a memory cell unit and some gates (input gate, forget gate, output gate).

The network takes three inputs. X\_t is the input of the current time step. h\_t-1 is the output from the previous LSTM unit and C\_t-1 is the “memory” of the previous unit, which I think is the most important input. As for outputs, h\_t is the output of the current network. C\_t is the memory of the current unit.

Therefore, this single unit makes decision by considering the current input, previous output and previous memory. And it generates a new output and alters its memory. A LSTM cell looks like-

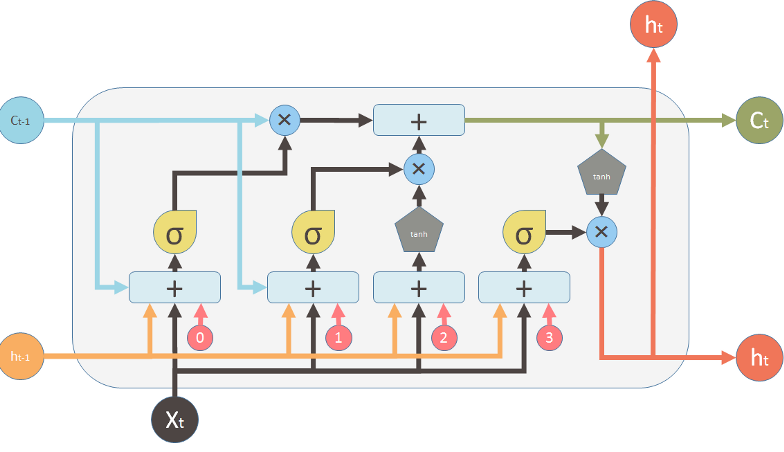


Figure 4: A LSTM cell

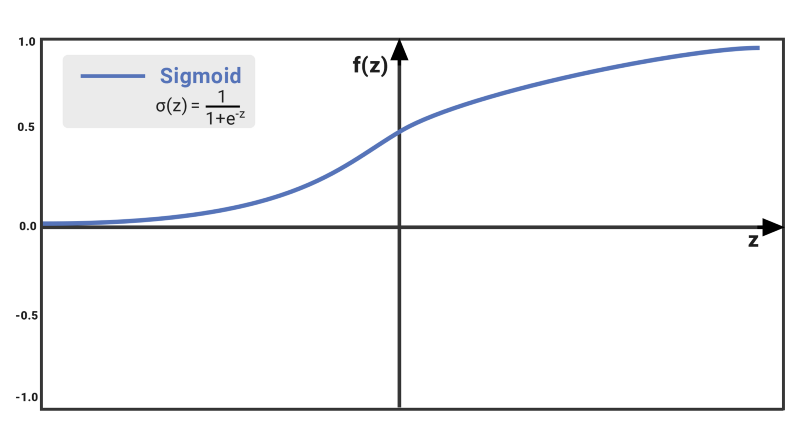
**Activation Functions**

Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be activated (“fired”) or not, based on whether each neuron’s input is relevant for the model’s prediction. Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1.

There’s linear and non-linear activation functions. Non-linear activation functions are capable of dealing with complex non-linear relations. Some functions are –

**Sigmoid function**

The Sigmoid function takes a value as input and outputs another value between 0 and 1. It is non-linear and easy to work with when constructing a neural network model. The good part about this function is that continuously differentiable over different values of z and has a fixed output range.

 Figure 5: sigmoid function

**Tanh or tan hyperbolic activation function**

tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped). The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.

The function is **differentiable**. The function is **monotonic** while its **derivative is not monotonic**. The tanh function is mainly used classification between two classes.

Figure 6: tanh and sigmoid comparison

#### **ReLU (Rectified Linear Unit) Activation Function**

The ReLU is the most used activation function in the world right now. Since, it is used in almost all the convolutional neural networks or deep learning. As you can see, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

**Range:** [ 0 to infinity)

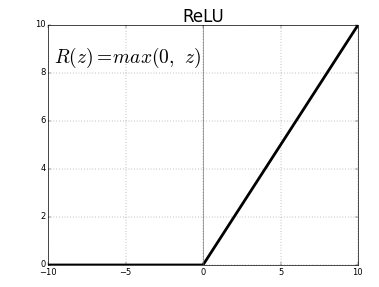
The function and its derivative **both are** **monotonic**.

Figure 7: ReLU activation function

**Softmax activation function**

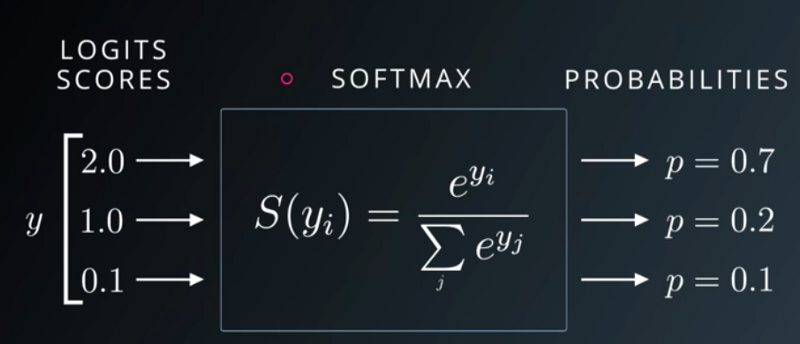
The Softmax regression is a form of logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sums up to 1. The output values are between the range [0,1] which is nice because we are able to avoid binary classification and accommodate as many classes or dimensions in our neural network model. This is why softmax is sometimes referred to as a multinomial logistic regression.

Figure 8: softmax activation function

**1.2 Challenges**

Implementing such a neural network model is really challenging.Sometimes it becomes difficult to know whether the model is giving correct output or not. Some challenges are-

* + - Understanding neural network architecture
    - Backpropagation through time
    - Performing frequent matrix operations
    - Learning python within a short time
    - Handling large code in python

**2.Project Overview**

My project is character level language modeling. It takes a chunk of data as input, then training starts on this data and after the training first character of the input dataset is passed to the trained model and it generates text with the trained model. I have done the same thing using three different models-

* + - RNN
    - GRU
    - LSTM

**2.1 dataset preparation**

To implement a RNN, at first input dataset need to be prepared. Neural network can only recognize numerical values, so text dataset needed to be converted into numerical vectors.

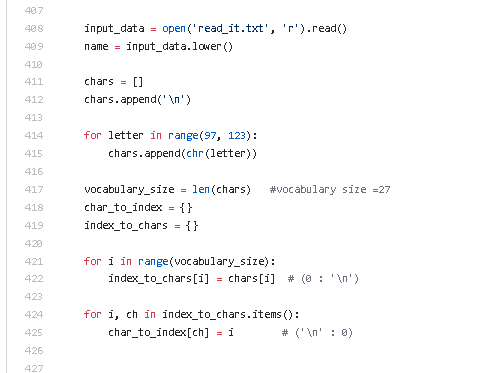
****

Figure 9: preparing dataset

In line 425, each character is assigned(mapped) to an integer value. Like ‘a’ to 1, ‘b’ to 2 and so on. Similarly, in line 422, the integers values are mapped to corresponding characters in ‘char\_to\_index’ dictionary. Like 0 to ‘\n’, 27 to ‘z’ and so on.

**2.2 forward propagation**

In a Feed-Forward neural network, the information only moves in one direction, from the input layer, through the hidden layers, to the output layer. The information moves straight through the network. Because of that, the information never touches a node twice.

Feed-Forward Neural Networks, have no memory of the input they received previously and are therefore bad in predicting what’s coming next. Because a feedforward network only considers the current input, it has no notion of order in time. They simply can’t remember anything about what happened in the past, except their training.

**Forward propagation in RNN**

a<0> = ᇹ (a vector of zeros)

a<1> = g (Waa a<0> + Wax X<1> + ba) ->>tanh/Relu

Ŷ<1> = g (Wya a<1> + by)

Generalize form,

a<t> = g (Waa a<t-1> + Wax X<t> + ba)

Ŷ<t> = g (Wya a<t> + by)

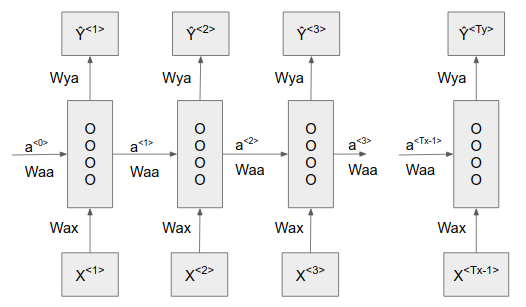
****

Figure 10: RNN cell forward propagation

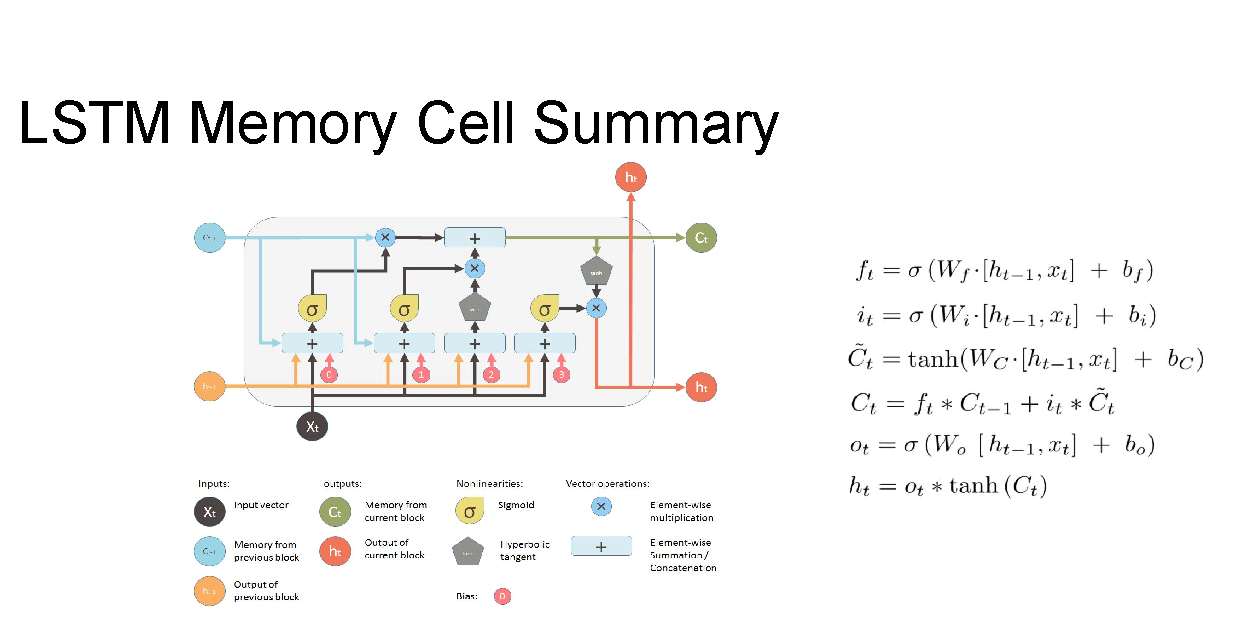


Figure 11: LSTM forward propagation

**GRU forward propagation**

Figure 12: GRU forward propagation

* GRU Combines the forget and input gates into a single “update gate.”
* Merges the cell state and hidden state

**2.3 Loss Function:** Logistic regression

Loss is calculated for the sake of calculating backpropagation.

L<t>(Ŷ<t>,Y<t>) = - Y<t> logŶ<t> - (1- Y<t>) log(1-logŶ<t>)

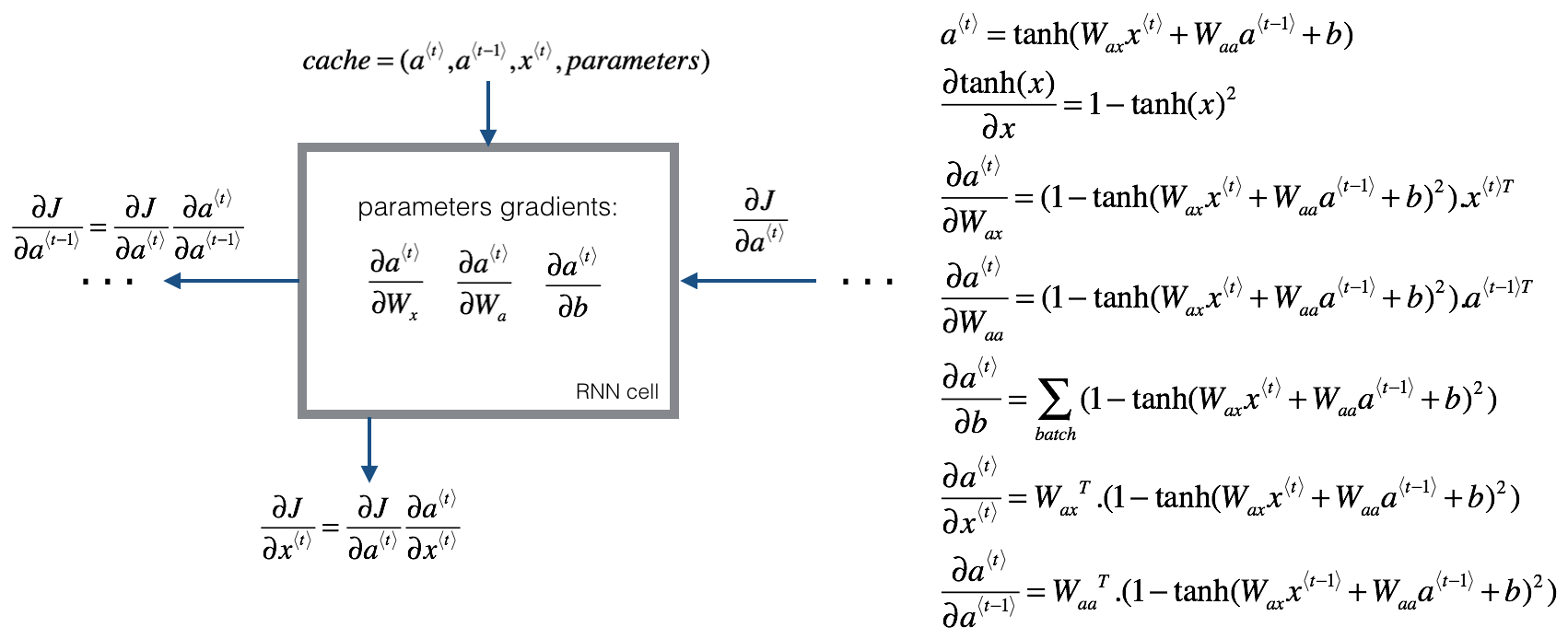
Total Loss:

L(Ŷ,Y) = ∑Tyt=1 L<t>(Ŷ<t>,Y<t>)

Backpropagation Through Time

\frac{\partial E}{\partial W} = \sum\limits_{t} \frac{\partial E_t}{\partial W}We calculate the gradients of the error with respect to our parameters U, Vand W and then l learn good parameters using Stochastic Gradient Descent. Just like we sum up the errors, we also sum up the gradients at each time step for one training example:

To calculate these gradients, we use the chain rule of differentiation. That’s the backpropagation algorithm when applied backwards starting from the error.

figure 13: backpropagation through time

**2.4 Update Parameter**

After backpropagation it’s time to update parameters. There’s several methods for updating parameters. I used AdaGrad update method.

**AdaGrad**

AdaGrad (also known as adaptive gradient algorithm) is a modified stochastic gradient descent with per-parameter learning rate (η), it increases the learning rate (lager updates) for sparser parameters and decreases the learning rate (smaller updates) for less sparse ones. This strategy often improves convergence performance over standard stochastic gradient descent for dealing with sparse data. In practice, AdaGrad greatly improved the robustness of SGD and it is often used for training large-scale neural nets.

Here, θ = parameter(weight)

g = gradient

1. **User Manual**

Here, input is a chunk of dataset which is plain text can be names or poems or essays.

Sample input dataset that is fed forward to RNN/LSTM model–

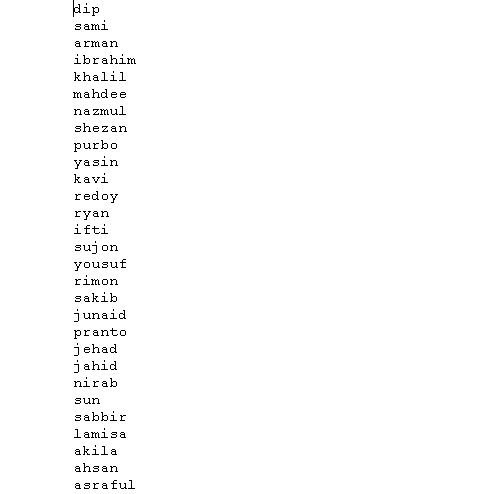


Figure 14: sample input passed to RNN forward model

The dataset is passed through the RNN/LSTM model and training continues. After training I just passed the first character of the dataset. Here- ‘d’ and then the rest of the sequence is generated by the trained model.

Sample output but not for these names-

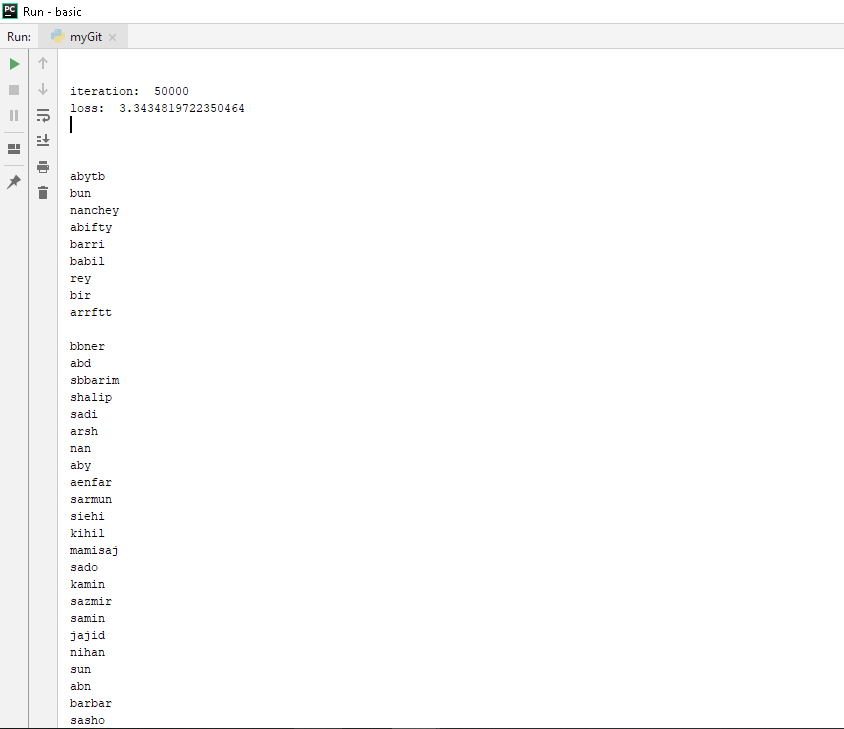


Figure: sample output after 50000 iteration

After training finishes, now we can check our model what has it learnt. This time we can give it one, two or more characters to see what sequence it predicts. Hope, it will not predict what we expect.



Figure: text generating using RNN

**4. Conclusion**

‘Neural networks do not work miracles. But if they used sensibly they produce amazing results.’

At first I was not feeling comfortable with the neural networks. They were seemed to be very hard to realize.

But now it’s a bit clearer to me. Thanks to my supervisor for encouraging me to make this done.

It’s application fields are very interesting. Further, I want to implement this in word level and also interested in image captioning and attention model seeking.

**5.Appendix**

Sampling is left. It’s done after the network learns the pattern of a chunk of dataset. In sampling we don’t send the input dataset anymore. Rather, we just for first time send a input and then every next input is the previous cell’s output. It’s just a feed forward method. Where no loss is calculated and nothing updates.

**6.References**

1.[colah's lstm](https://www.google.com/url?client=internal-uds-cse&cx=partner-pub-6638247779433690:1054494704&q=https://colah.github.io/posts/2015-08-Understanding-LSTMs/&sa=U&ved=2ahUKEwjw8-y-0b3iAhV34nMBHZoMBYAQFjAAegQIABAC&usg=AOvVaw26t7e7Mpt5pXoEFGi27cBU) [25/3/2019]

2.[https://dl.howtocode.com.bd/](https://dl.howtocode.com.bd/multi-layer-nn/multi-layer-nn-check) [15/2/2019]

3. <https://skymind.ai/wiki/lstm> [5/4/2019]

4.<https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be> [8/4/2019]

5. <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/> [20/2/2019]

6.<https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6> [24/2/2019]