NEXT WORD PREDICTION USING RECURRENT NEURAL NETWORK

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of,

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

SHASHI KIRAN M	321810304002
DILIP KUMAR	321810304005
JEEVAN REDDY	321810304022

Under the esteemed guidance of

Dr. SATHISH KUMAR MANI

Associate Professor

Department of Computer Science & Engineering,

GITAM SCHOOL OF TECHNOLOGY

GANDHI INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(Deemed to be University)

Bengaluru Campus.

April 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM SCHOOL OF TECHNOLOGY

GITAM
(Deemed to be University)

CERTIFICATE

This is to certify that the project report entitled **"NEXT WORD PREDICTION USING RECURRENT NEURAL NETWORK"** is a bonafide record of work carried out by **SHASHI KIRAN M (321810304002)**, **DILIP KUMAR(321810304005), JEEVAN REDDY(321810304022)** submitted in partial fulfillment of requirement for the award of degree of **Bachelors of Technology in Computer Science and Engineering.**

Project Guide.	**Head of the Department.**
SIGNATURE OF THE GUIDE	**SIGNATURE OF THE HoD**
Dr. Sathish Kumar Mani	**Dr. Vamsidhar Yendapalli,**
Associate professor.	**Professor.**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM SCHOOL OF TECHNOLOGY

GITAM

(Deemed to be University)

DECLARATION

We, hereby declare that the project report entitled **"NEXT WORD PREDICTION USING RECURRENT NEURAL NETWORK"** is an original work done in the Department of Computer Science and Engineering, **GITAM School of Technology, GITAM (Deemed to be University)** submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree.

Date:

Registration No(s).	Name(s)	Signature(s)
1. 321810304002	**SHASHI KIRAN M**	
2. 321810304005	**DILIP KUMAR**	
3. 321810304022	**JEEVAN REDDY**	

ACKNOWLEDGEMENT

The Satisfaction and Euphoria that accompany the successful completion of any Major Project would be incomplete without mentioning the people who made it possible, whose constant guidance and encouragement crowned our efforts with success. We take this opportunity to express the deepest gratitude and appreciation to all those who held us directly or indirectly towards the successful completion of the Major Project.

I would like to thank **Dr. S DINESH**, Ph.D., Director, GITAM School of Technology for all the facilities provided.

I would like to thank **Dr. VAMSIDHAR**, Ph.D., Professor-HOD, Department of Computer Science and Engineering for his support and encouragement that went a long way in the successful completion of this Major Project.

I consider that as a privilege to express our heartfelt gratitude and respect to **Dr. Sathish Kumar Mani**, Associate professor, department of Computer Science and Engineering for being our internal guide and **Dr. Hem Kumar** for being our Project coordinator for his integral and incessant support offered to us throughout the course of this project and for a constant source of inspiration throughout the Major Project.

I would like to thank our parents and our friends for their support and encouragement in the completion of the project in time. Last but not least I would like to thank all the teaching and non-teaching staff members of the Computer Science and Engineering

ABSTRACT

Next Word Prediction is also known as **Language Modeling**. It is the undertaking of predicting what word comes straightaway. It is one of the major assignments of NLP and has numerous applications. Attempting to make a model utilizing Nietzsche default text record which will foresee clients sentence after the clients composed 40 letters, the model will comprehend 40 letters and anticipate impending top 10 words utilizing RNN neural organization which will be executed utilizing Tensorflow. Our Aim of creating this model to predict 10 or more then 10 words as fast as possible utilizing minimum time. As RNN is Long Short Time Memory it will understand past text and predict the words which may be helpful for the user to frame sentences and this technique uses letter to letter prediction means it predicts letter after letter to create a word.

TABLE OF CONTENTS

1. INTRODUCTION

Natural Language Processing (NLP) is a significant part of artificial Intelligence, which incorporates AI, which contributes to finding productive approaches to speak with people and gain from the associations with them. One such commitment is to give portable clients anticipated ”next words,” as they type along within applications, with an end goal to assist message conveyance by having the client select a proposed word as opposed to composing it. As LSTM is Long Short Time Memory it will understand the past text and predict the words which may be helpful for the user to frame sentences and this technique uses a letter to letter prediction means it predicts a character to create a word. As writing an essay and framing a big paragraph are time-consuming it will help end-users to frame important parts of the paragraph and help users to focus on the topic instead of wasting time on what to type next. We expect to create or mimic auto-complete features using LSTM. Most of the software uses different methods like NLP and normal neural networks to do this task. We will be experimenting with this problem using LSTM by using the Default Nietzsche text file also known as our training data to train a model. Next Word Prediction is also called Language Modeling that is the task of predicting what word comes next. It is one of the fundamental tasks of NLP and has many applications.

2. LITERATURE SURVEY

This section describes the ideas that are taken in for the literature review. This paper represents an exploration of the contributions that have already been made in the academic field.

[1] Multi-window convolution(MRNN) algorithm is implemented. They have created a residual-connected minimal gated unit(MGU) which is a short version of LSTM. In this CNN, they attempt to go past a few layers when the training result is lesser than expected. They have good accuracy by far using multiple layers of neural networks.

[2] This paper uses the RNN algorithm and also they have used GRU, another form of RNN for code completion problems as RNN helps to predict the next code syntax for users. Authors claim that their method is more accurate compared to existing methods. They have separated next word prediction into two components: within-vocabulary words and identifier prediction. They have used the LSTM neural language model to predict vocabulary words. A pointer network model is proposed to identify prediction.

[3] Authors worked on Bangla Language. They have proposed a novel method for word prediction and word completion. They have proposed an N-gram-based language model which predicts a set of words. Using this they have accomplished agreeable outcomes.

[4] Authors have used LSTM for the next word prediction for the Assamese language. They have stored transcripted language according to the International Phonetic Association (IPA) chart and fed it to their model. They created a model for physically challenged people. This model uses Unigram, Bigram, Trigram, based Approach for next word predic- tion and was found out on average to predict the word but accuracy was around 30-40 percent.

[5] In this paper, they created an auto-next-keyword for the Bengali language which was challenging and It was found that it is hard to get good accuracy by using the RNN algorithm due to its vanishing gradients and heavy recurrent NN taking more time to train and test.

[6] They predicted the next character highlighter(PNCH) for the Indian language. It was more of text correction and less about next word prediction but was quite good to understand. The Method is called hit and miss but accuracy is less and the model was not efficient for problem statements of this type.

[7]This was the first approach to tackle problems of this type. The paper discusses LM and the perplexity algorithm which is the basis of natural language processing. This helps us to make a 3D input data for our model.

[8] 1-degree feature patterns algorithm is used to solve the problem of Vanishing but provides less accuracy as basic sentences were used in training and testing data like ’A man cries’, ’A baby cries’, ’A dog barks’, and ’A cat mews’.

[9] As GPT is quite a huge and costly model for this type of task as word prediction is a simple project, GPT will only act like wasting useful resources for simple tasks.

3.SOFTWARE AND HARDWARE SPECIFICATION

3.1 Software Requirements:

Python

Jupyter notebook or Visual Studio Code

Numpy, RNN, LSTM.

3.1.1.Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aims to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

3.1.2.Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. 5 Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots, and rich media, usually ending with the ".ipynb" extension.

3.2 Hardware Requirements :

Processor (>1.8 GHz)

Ram (>4GB)

Monitor

4. PROBLEM STATEMENT

This paper is helpful in creating a model to predict the next possible text. Some of the paper is helpful in predicting the next code using SVM and RNN. There is a new algorithm ex: LSTM which helps in getting good results for this problem statement. But there are some limitations of this system.

This process of predicting the next Word is quite complex because we have to predict words which the user thinks so it is predicting the thoughts of the user so the accuracy is quite low compared to other ML projects.

Many algorithms like SVM, Decision Tree, etc are not providing good results and take more time to predict the results as the task is quite complex.

To be able to make useful predictions, a text predictor needs much knowledge about language so we have to keep training the neural network continuously on new languages and new data.

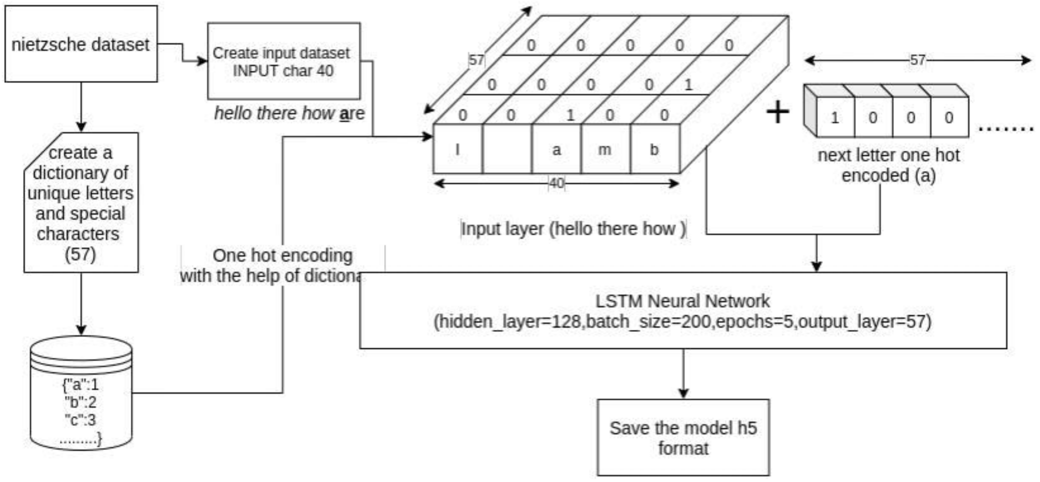
5. METHODOLOGY

5.1 DATA PRE-PROCESSING:

This proposed work in Figure 1. is an illustration to create a flexible model that can help users to detect the next word while understanding user vocable in a fast and effective way which needs the user needs to provide 40 letters then it passes this letter to LSTM NN and predicts N number of letters

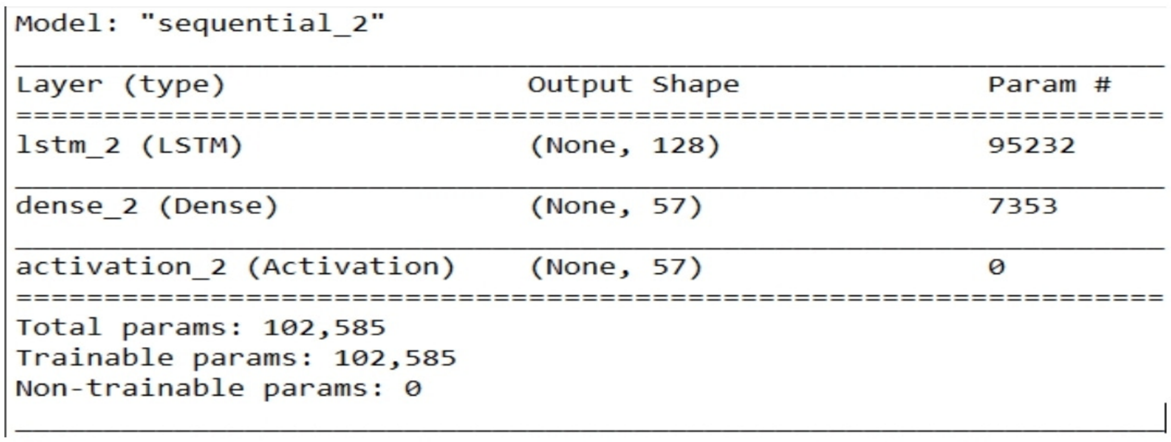
We can see in figure 1. the input is given up to 40 letters. After this, this sentence is made to go through LSTM Neural Network

Now LSTM understands and learns every letter, and after this it creates a score for the next letter.



nietzsche dataset
Create input dataset INPUT char 40
hello there how are
create a dictionary of unique letters and special characters (57)
{"a":1 "b":2 "c":3 ....}
One hot encoding with the help of dictiona
57
40
Input layer (hello there how)
+
57
next letter one hot encoded (a)
LSTM Neural Network (hidden_layer=128,batch_size=200,epochs=5,output_layer=57)
Save the model h5 format

Figure 1. Proposed Work



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 128)               95232
_________________________________________________________________
dense_2 (Dense)              (None, 57)                7353
_________________________________________________________________
activation_2 (Activation)    (None, 57)                0
=================================================================
Total params: 102,585
Trainable params: 102,585
Non-trainable params: 0
_________________________________________________________________
```

Figure 2. Neural network Architecture

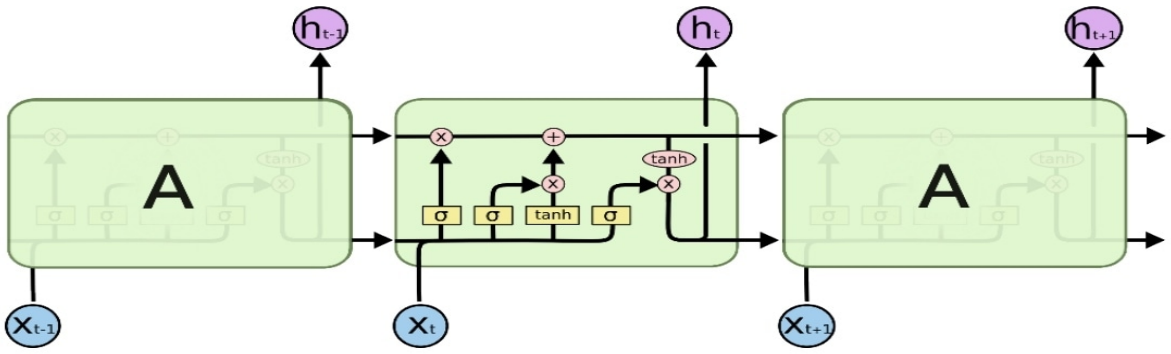


Figure 3. Recurrent Neural network

This process is repeated and this score will again pass through the same LSTM and later it will predict a word letter by letter.

Below is our neural network architecture plus our implementation methodology using the Tensorflow library.

Letter to bits, As computers don't understand words so converting words to bits or

array of bits using NumPy software.

Now creating a 3D array of all words it's like one-hot encoding for all letter a unique characters (200285, 40, 57) this was our training data

Later passing this X features to our model with input Neural node 40 and hidden node 128 then this will have an output layer with node equal to the input node.

5.2 LSTM Networks:

Step 1: first import our helpful model Numpy pandas and other modules later importing Nietzsche default txt which is our dataset.

Step 2: The way to LSTMs is the cell express, the even line going through the highest point of the outline.

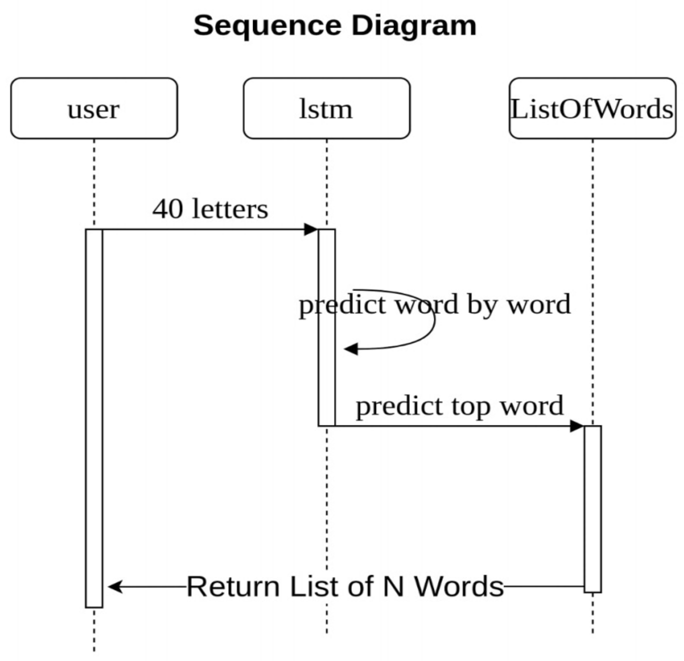


Figure 4. sequence diagram of application

The cell state is similar to a transport line. It runs straight down the whole chain, with just some minor direct communications. It’s easy for data to stream along with it without any alterations.

The LSTM can eliminate or add data to the phone state, painstakingly managed by structures called entryways.

Gates are an approach to alternatively let data through. They are made out of a sigmoid neural net layer and a pointwise duplication activity.

The sigmoid layer yields numbers somewhere in the range of nothing and one, depicting the amount of every segment ought to be let through. A worth of zero signifies "let nothing through," while a worth of one signifies "let everything through!"

A LSTM has three of these gates, to secure and control the cell state.

5.3. Data Description and Data cleaning:

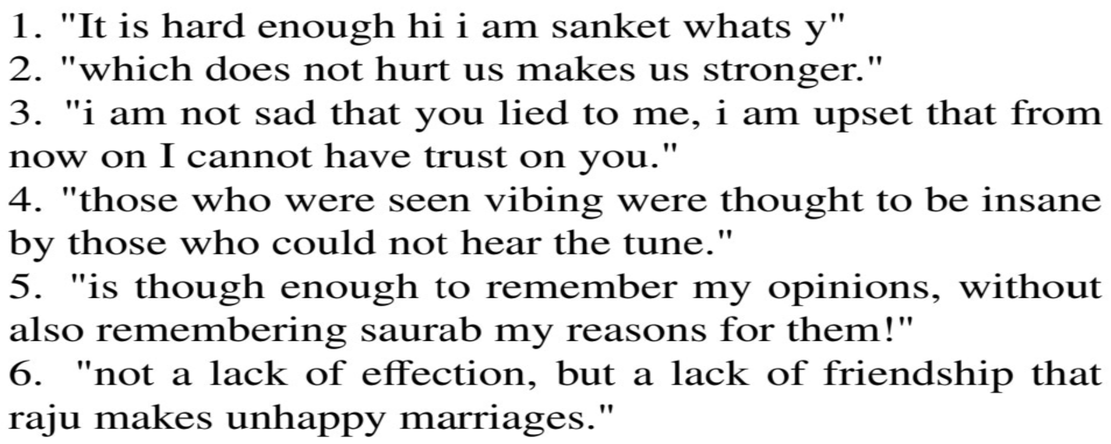
The dataset contains 25,107 words from ebook author Franz Kafka. The Datasets for text data are easy to find and we can consider Project Gutenberg which is a volunteer effort to digitize and archive cultural works, to “encourage the creation and distribution of eBooks”. From here we can get many stories, documentations, and text data which are necessary for our problem statement.

6. Implementation

User: Here the user's work is to input words for the LSTM Neural Network like in "Figure. 4"

LSTM: LSTM is trained on the default next word prediction text file; It will calculate its weight and predict letter by letter for a top word.

List of word: This will hold all the words and if the user wants N number of top works it will count N-words and return the list of words to the user.



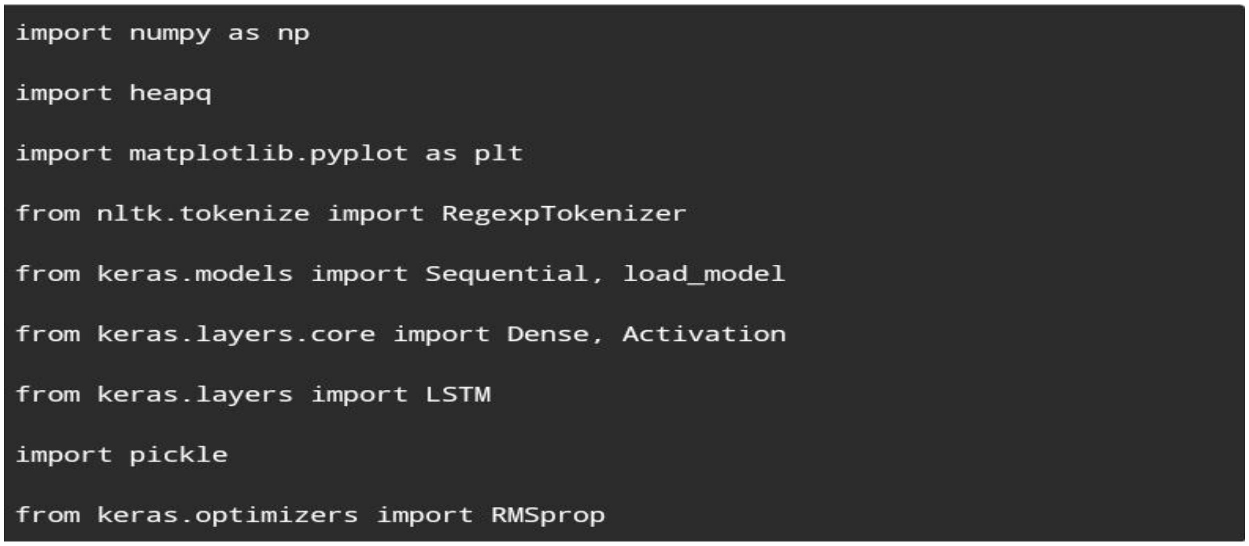
1. "It is hard enough hi i am sanket whats y"
2. "which does not hurt us makes us stronger."
3. "i am not sad that you lied to me, i am upset that from now on I cannot have trust on you."
4. "those who were seen vibing were thought to be insane by those who could not hear the tune."
5. "is though enough to remember my opinions, without also remembering saurab my reasons for them!"
6. "not a lack of effection, but a lack of friendship that raju makes unhappy marriages."

Figure 5. Input test cases

Steps to Implement:-

6.1 Import the required libraries :

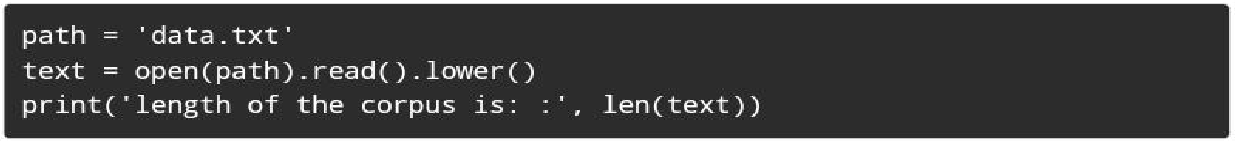
We could use TensorFlow with Keras for our model building. We could then import the LSTM model from Keras and use it. For different NLP tasks, we could then use the NLTK library.



```
import numpy as np

import heapq

import matplotlib.pyplot as plt

from nltk.tokenize import RegexpTokenizer

from keras.models import Sequential, load_model

from keras.layers.core import Dense, Activation

from keras.layers import LSTM

import pickle

from keras.optimizers import RMSprop
```

6.2 Read the dataset :

We could check the length of the corpus by using the len function on text after reading and converting everything to lower case to avoid duplication of words.



```
path = 'data.txt'
text = open(path).read().lower()
print('length of the corpus is: :', len(text))
```

```
length of the corpus is: 581887
```

6.3 Using Tokenizers:

We require tokenizers to split it into separate words and store them.

```
tokenizer = RegexpTokenizer(r'w+')
words = tokenizer.tokenize(text)
```

6.4 Getting Unique words:

To get all the unique words, we require a dictionary with each word in the data within the list of unique words as the key and its significant portions as value.

```
unique_words = np.unique(words)
unique_word_index = dict((c, i) for i, c in enumerate(unique_words))
```

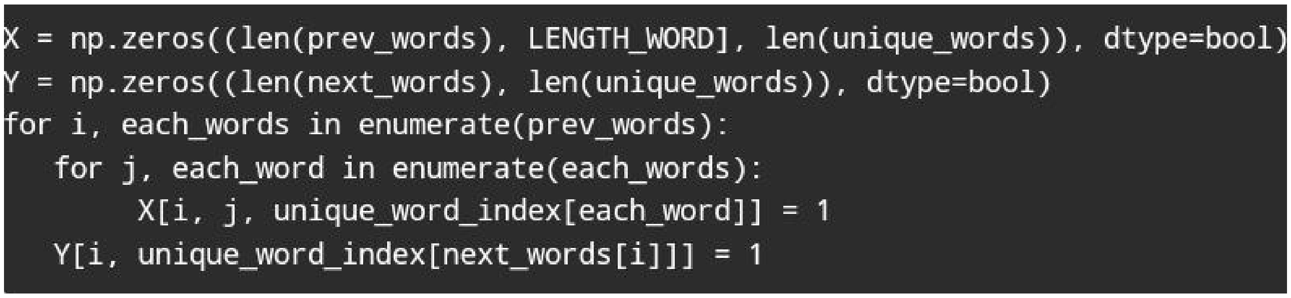
6.5 Feature Engineering :

Feature engineering will make the words into numerical representation so that it is easy to process them.

```
LENGTH_WORD = 5
next_words = []
prev_words = []
for j in range(len(words) - LENGTH_WORD):
     prev_words.append(words[j:j + LENGTH_WORD])
     next_words.append(words[j + LENGTH_WORD])
print(prev_words[0])
print(next_words[0])
```

6.6 Storing features and labels:

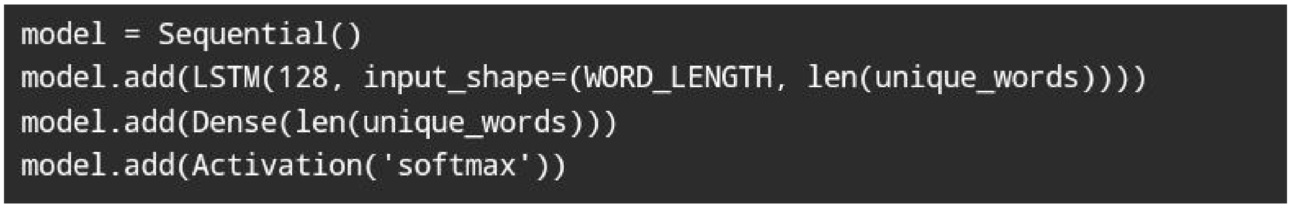
X will be used to get the features and Y to get the labels associated with them.



```
X = np.zeros((len(prev_words), LENGTH_WORD], len(unique_words)), dtype=bool)
Y = np.zeros((len(next_words), len(unique_words)), dtype=bool)
for i, each_words in enumerate(prev_words):
   for j, each_word in enumerate(each_words):
        X[i, j, unique_word_index[each_word]] = 1
   Y[i, unique_word_index[next_words[i]]] = 1
```

6.7 Building our model:

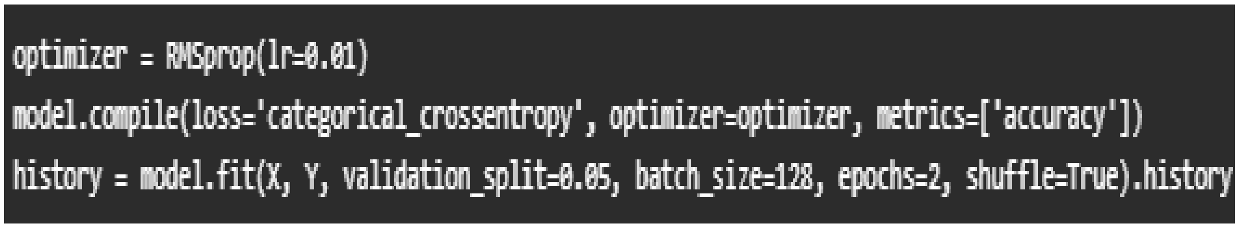
We could observe that we have built an LSTM model and used a softmax activation at the end to get a few specific words predicted by the model.



```
model = Sequential()
model.add(LSTM(128, input_shape=(WORD_LENGTH, len(unique_words))))
model.add(Dense(len(unique_words)))
model.add(Activation('softmax'))
```

6.8 Model training:

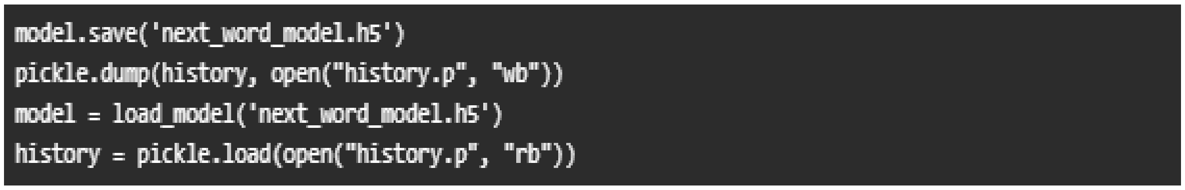
The model training uses RMSprop as the optimizer with a learning rate of 0.02 and uses categorical cross-entropy for loss function. With a batch size of 128 and a split of 0.5, we train our model.



```
optimizer = RMSprop(lr=0.01)
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
history = model.fit(X, Y, validation_split=0.05, batch_size=128, epochs=2, shuffle=True).history
```

6.9 Saving model :

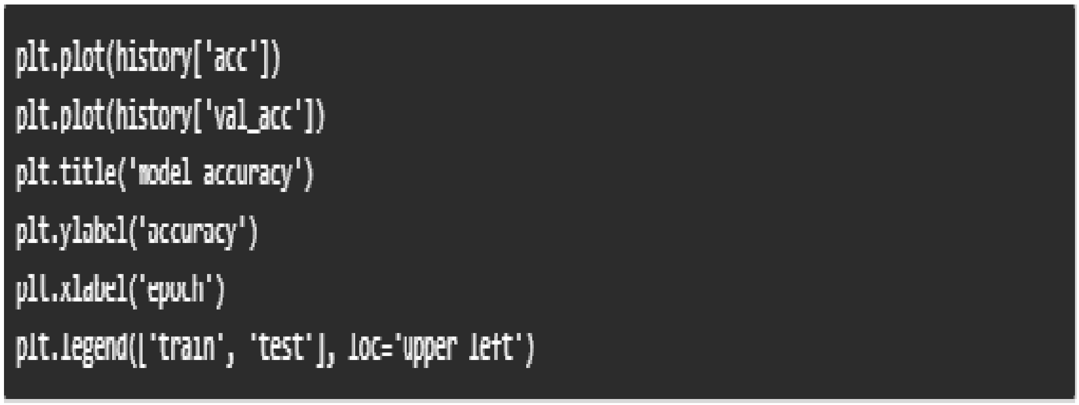
The model is saved using the save function and loaded.



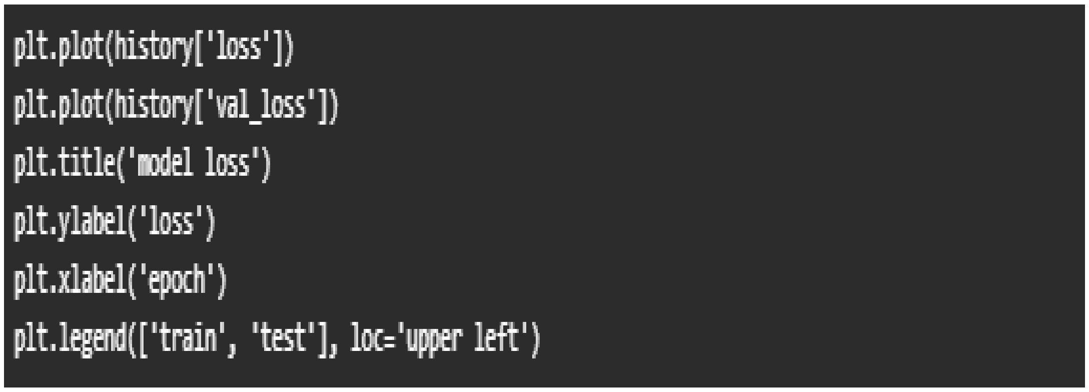
```
model.save('next_word_model.h5')
pickle.dump(history, open("history.p", "wb"))
model = load_model('next_word_model.h5')
history = pickle.load(open("history.p", "rb"))
```

6.10 Evaluating the model:

Now we could observe the results of the models by evaluating them.



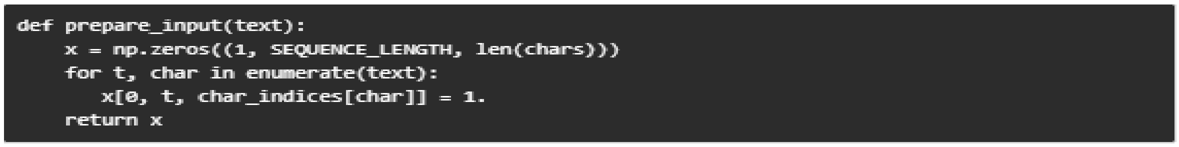
```
plt.plot(history['acc'])
plt.plot(history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
```



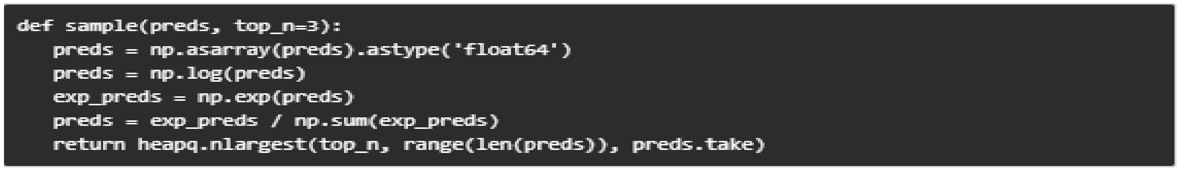
```
plt.plot(history['loss'])
plt.plot(history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
```

6.11 Testing next word:

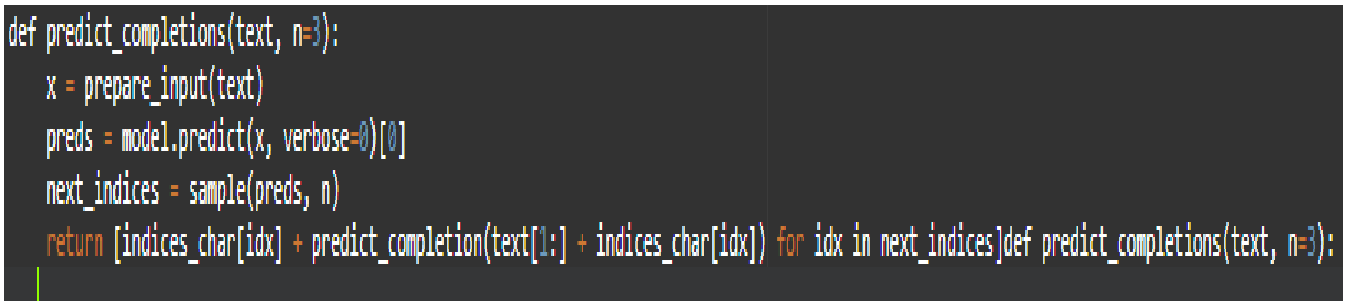
These functions will help us to predict the next few words when we provide a sentence.



```
def prepare_input(text):
    x = np.zeros((1, SEQUENCE_LENGTH, len(chars)))
    for t, char in enumerate(text):
       x[0, t, char_indices[char]] = 1.
    return x
```



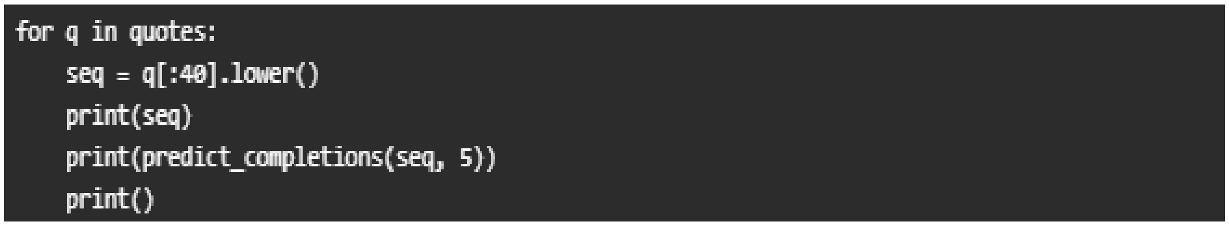
```
def sample(preds, top_n=3):
   preds = np.asarray(preds).astype('float64')
   preds = np.log(preds)
   exp_preds = np.exp(preds)
   preds = exp_preds / np.sum(exp_preds)
   return heapq.nlargest(top_n, range(len(preds)), preds.take)
```



```
def predict_completions(text, n=3):
    x = prepare_input(text)
    preds = model.predict(x, verbose=0)[0]
    next_indices = sample(preds, n)
    return [indices_char[idx] + predict_completion(text[1:] + indices_char[idx]) for idx in next_indices]def predict_completions(text, n=3):
```

6.12 Predict the next word:

Using this code we get the next word.



```
for q in quotes:
    seq = q[:40].lower()
    print(seq)
    print(predict_completions(seq, 5))
    print()
```

7. Experimental Result

From ”Figure. 7” we can observe that the n-grams approach is inferior to the LSTM approach as LSTMs have the memory to review the setting further, harking back to the substance corpus. While starting another endeavor, you ought to consider one of the current pre-arranged designs by looking on the web for open-source executions. Thus, you will not have to start without any planning and don’t need to worry about the arrangement cycle or hyperparameters.

Trying to create a model using Nietzsche's default text file which will predict users sentence after the users typed 40 letters, the model will understand 40 letters and predict upcoming letter/words using LSTM neural network which will be implemented using Tensorflow.

This product has more scope on social media for syntax analysis and semantic analysis in natural language processing in Artificial intelligence.

After this we can try to adapt some input layers for different input layers. We could observe that with every variable input layer, the size of the output prediction may vary and it has accuracy of 54% to 55%.

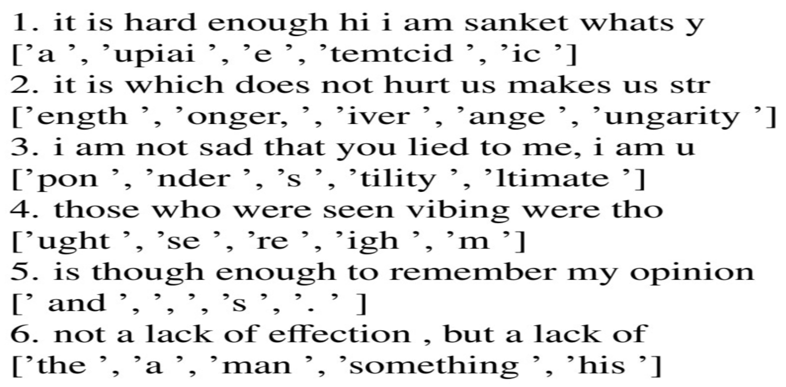
For 10 input nodes the training accuracy is around 56% but the testing accuracy is around 54%.

For 20 input nodes the training accuracy is around 56% but the testing accuracy is around 55%.

For 30 input nodes the training accuracy is around 56% but the testing accuracy is around 55.5% same as 20.

For 40 input nodes the training accuracy is around 56.3% but the testing accuracy is around 54.9% .

Based on this example in " Figure 5 ". We can understand that by taking 6 test cases as input and then passing 5 of them in params, We could observe that the neural networks did a great job in predicting the possible result. It is observed that in the second case in " Figure 6 " the model predicted the string which comes after "str" (i.e stronger, strength).
From "Figure 7 " With an accuracy of around 56% our model did a very good job in real test case scenarios predicting these input test cases.



```
1. it is hard enough hi i am sanket whats y
['a ', 'upiai ', 'e ', 'temtcid ', 'ic ']
2. it is which does not hurt us makes us str
['ength ', 'onger, ', 'iver ', 'ange ', 'ungarity ']
3. i am not sad that you lied to me, i am u
['pon ', 'nder ', 's ', 'tility ', 'ltimate ']
4. those who were seen vibing were tho
['ught ', 'se ', 're ', 'igh ', 'm ']
5. is though enough to remember my opinion
[' and ', ', ', 's ', '. ' ]
6. not a lack of effection , but a lack of
['the ', 'a ', 'man ', 'something ', 'his ']
```

Figure 6. N(5) number of outputs

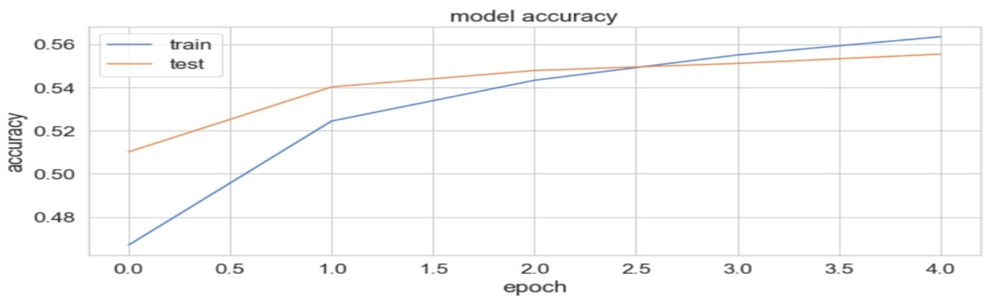


Figure 7. Accuracy on training and testing data

8. Conclusion

Standard RNNs and other language models become less exact when the hole between the specific circumstance and the word to be anticipated increments. Here’s when LSTM is used to handle the drawn-out reliance issue since it has memory cells to recall the past setting. You can study LSTM Neural Net. Our task in this project is to train and try an algorithm that best fits this task. Since the problem is quite complex, we will implement it through an LSTM. We created a 3d vector layer of input and a 2d vector layer for output and fed it through to the LSTM layer having 128 hidden layers, and managed to get accuracy to around 56% during five epochs. This paper presents how the system is predicting and correcting the next/target words using other mechanisms and using the TensorFlow closed-loop system, the scalability of a trained system can be increased using the perplexity concept of the system will decide that the sentence has more misspelled words. The performance of the system can be increased.

9. Future work

Understanding the paragraph using machine learning algorithms like RNN can help soon to understand and frame paragraphs and stories on their own.

Creating lyrics and songs can be a major field in which this algorithm can help the end-users to predict the next phrase in songs considering the model is trained on a music lyrics data set.

With more data we can train the model which will reevaluate the weights to understand the core features of paragraphs/sentences to predict good results.

Paraphrasing means formulating someone else's ideas in your own words. To paraphrase a source, you have to rewrite a passage without changing the meaning of the original text, so our algorithms can predict more number words considering a single sentence and help users to frame n number of sentences.

10. Reference

[1] J. Yang, H. Wang and K. Guo, "Natural Language Word Prediction Model Based on Multi-Window Convolution and Residual Network," in IEEE Access, vol. 8, pp. 188036-188043, 2020, doi: 10.1109/ACCESS.2020.3031200.

[2] K. Terada and Y. Watanabe, "Code Completion for Programming Education based on Recurrent Neural Network," 2019 IEEE 11th International Workshop on Computational Intelligence and Applications (IWCIA), Hiroshima, Japan, 2019, pp. 109-114, doi: 10.1109/IWCIA47330.2019.8955090.

[3] Habib, Md AL-Mamun, Abdullah Rahman, Md Siddiquee, Shah Ahmed, Farruk. (2018). An Exploratory Approach to Find a Novel Metric Based Optimum Language Model for Automatic Bangla Word Prediction. International Journal of Intelligent Systems and Applications. 2. 47-54. 10.5815/ijisa.2018.02.05.

[4] Partha Pratim Barman, Abhijit Boruah, A RNN based Approach for next word prediction in Assamese Phonetic Transcription,Procedia Computer Science,Volume143,2018,Pages117-123,ISSN1877-0509,https://doi.org/10.1016/j.procs.2018.10.359.

[5] S. M. Sarwar and Abdullah-Al-Mamun, "Next word prediction for phonetic typing by grouping language models," 2016 2nd International Conference on Information Management (ICIM), London, 2016, pp. 73-76, doi: 10.1109/INFOMAN.2016.7477536.

[6] M. K. Sharma, S. Sarcar, P. K. Saha and D. Samanta, "Visual clue: An approach to predict and highlight next character," 2012 4th International Conference on Intelligent Human Computer Interaction (IHCI), Kharagpur, 2012, pp. 1-7, doi: 10.1109/IHCI.2012.6481820.

[7] Minghui Wang, Wenquan Liu and Yixion Zhong, "Simple recurrent network for Chinese word prediction," Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan), Nagoya, Japan, 1993, pp. 263-266 vol.1, doi: 10.1109/IJCNN.1993.713907.

[8] Y. Ajioka and Y. Anzai, "Prediction of next alphabets and words of four sentences by the adaptive junction," IJCNN-91-Seattle International Joint Conference on Neural Networks, Seattle, WA, USA, 1991, pp. 897 vol.2-, doi: 10.1109/IJCNN.1991.155477.

[9] Joel Stremmel, Arjun Singh. (2020). Pretraining Federated Text Models for Next, Word Prediction using GPT2.