PREDICTIVE TEXT GENERATION USING DEEP LEARNING APPROACHES

Project submitted to
Shri Ramdeobaba College of Engineering & Management, Nagpur in partial fulfillment of requirement for the award of degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

By

Himani Dighorikar(03)
Ishika Bajaj (104)
Shivam Gupta (76)
Shridhar Ashtikar (77)

Guide

Prof. D. A. Borikar

Department of Computer Science and Engineering

Shri Ramdeobaba College of Engineering and Management, Nagpur 440013

(An Autonomous Institute affiliated to Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur)

May 2022

PREDICTIVE TEXT GENERATION USING DEEP LEARNING APPROACHES

Project submitted to
Shri Ramdeobaba College of Engineering & Management, Nagpur in partial fulfillment of requirement for the award of degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

By

Himani Dighorikar(03)
Ishika Bajaj (104)
Shivam Gupta (76)
Shridhar Ashtikar (77)

Guide

Prof. D. A. Borikar

Department of Computer Science and Engineering

Shri Ramdeobaba College of Engineering and Management, Nagpur 440013

(An Autonomous Institute affiliated to Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur)

May 2022

SHRI RAMDEOBABA COLLEGE OF ENGINEERING & MANAGEMENT,

NAGPUR

(An Autonomous Institute affiliated to Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Thesis on **“PREDICTIVE TEXT GENERATION USING DEEP LEARNING APPROACHES”** is a bonafide work of Himani Dighorikar, Ishika Bajaj, Shivam Gupta and Shridhar Ashtikar submitted to the Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur in partial fulfillment of the award of a Degree of Bachelor of Engineering. It has been carried out at the Department of Computer Science and Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur during the academic year 2021-22.

Date: 14 May, 2022

Place: Nagpur

Prof. D. A. Borikar
(Project Guide)
Department of Computer
Science Engineering

Dr Avinash Agrawal
(H.O.D)
Department of Computer
Science and Engineering

Dr. R. S. Pande
(Principal)

DECLARATION

I, hereby declare that the thesis "**PREDICTIVE TEXT GENERATION USING DEEP LEARNING APPROACHES**" submitted herein, has been carried out in the Department of Computer Science and Engineering of Shri Ramdeobaba College of Engineering & Management, Nagpur. The work is original and has not been submitted earlier as a whole or part for the award of any degree / diploma at this or any other Institute / University.

Date: 14 May, 2022

Place: Nagpur

Name of the Student	Roll No.	Signature
Himani Manoj Dighorikar	03	
Ishika Bajaj	104	
Shivam Gupta	76	
Shridhar Ashtikar	77	

APPROVAL SHEET

This report entitled

“PREDICTIVE TEXT GENERATION USING DEEP LEARNING APPROACHES”

By

Himani Dighorikar

Ishika Bajaj

Shivam Gupta

Shridhar Ashtikar

Is approved for the degree of Bachelor and Engineering.

Name & Signature of Supervisor(s)   Name & Signature of External Examiner(s)

Name & Signature of HOD

Date: 14 May, 2022
Place: Nagpur

ACKNOWLEDGEMENT

The project is a combined effort of a group of individuals who synergize to contribute towards the desired objectives. Apart from the efforts by us, the success of the project shares an equal proportion on the engagement and guidance of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We would like to show our greatest appreciation towards **Prof. D.A Borikar** for providing us with the facilities for completing this project and for his constant guidance.

We would like to express our deepest gratitude to **Dr. Avinash Agrawal,** Head, Department of Computer Science and Engineering, RCOEM, Nagpur for providing us the opportunity to embark on this project.

We are thankful to the AL and DL of RCOEM, Nagpur hosted in CSE department for NVIDIA DGX DL Workstation of center of Excellence during extensive training of our model.

We would like to thank and express sincere gratitude towards our Principal **Dr. R.S. Pande** for being our source of inspiration throughout this project.

Finally, extend our gratefulness to all the faculty members of CSE department who have always been so supportive and providing resources needed in our project development. We are very grateful to all of them who have unconditionally supported us throughout the project.

Date: 14 May, 2022

Projectee
Himani Dighorikar
Ishika Bajaj
Shivam Gupta
Shridhar Ashtikar

ABSTRACT

Predicting the next word has been one of the most important subjects of discussion in Natural Language Processing. Nowadays, instead of wasting time on writing everything and then proof reading it, we simply use Auto-complete. We use it every day but don’t give it time of day to understand how it happens. Natural language generation (NLG) focuses on generation of natural, human interpretable language. NLG is a methodology that allows us to predict the next word in a sentence most likely to be used.

Sequence prediction problems have been a major problem for a long time. Recurrent Neural Network (RNN) has been a good solution for sequential prediction problems. This work aims to create a generative model for text. Even though, RNN has its own limitations such as vanishing and exploding gradient descent problems and inefficiency to keep track of long-term dependencies. To overcome these drawbacks, Long Short Term Memory (LSTM) has been a path-breaking solution to deal with sequential data and text data in particular.

The project aims to develop a model that predicts words based on users typing and suggests users some words that can possibly come next. The LSTM model and BiLSTM model comparative analysis is done to study how the model behaves differently. The project also aims to review all the studies and efforts made by researchers in predictive system development and their methodologies.

Keywords: Text-prediction, Deep learning, Natural Language Processing, Recurrent Neural Network (RNN), Long short-term memory (LSTM), Natural language generation, Bi-LSTM.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ ABBREVIATIONS OR NOMENCLATURE

Abbreviation	**Expansion**
LSTM	Long short term memory
Bi-LSTM	Bidirectional Long Short term memory
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
NLP	Natural Language Processing

CHAPTER 1

INTRODUCTION

Nowadays technology has become a crucial part of our life, every day we read about new inventions which solves a problem which we never knew existed and that can make our lives even easier. Machines are trained to think and perform tasks that humans do. With that communication system also plays a vital role. Communication system in this advancing world is enhancing day-by-day. And there are many ways through which we are progressing towards it.

1.1 Problem Definition

Design and develop a NLP system that will use deep learning approaches to predict the next word based on the trained corpus.

The existing systems like Gmail system, Google search provides the predictive functionality. Gmail has a system of suggesting a set of words to users which are most commonly used. The Google search engine suggests words to users based on their history of searches.

The problem definition is aligned to the target stated above that will be built using deep learning approaches.

1.2 Motivation

The main motivation drawn from the model will be to help people who are not efficient in English to write complex sentences with ease. It will eliminate spelling errors and will boost typing speed. It will also help users in learning the language by suggesting appropriate words to them, possibly even expanding their vocabulary. The system will also be useful for users with learning disabilities to quickly frame sentences without grammatical errors since it automatically suggests words to them.

1.3 Overview

Many complicated and more efficient algorithms have been created in recent years to abate some of the tedious tasks in many real world problems. One of the major issues for specially-abled people is having communication issues. In order to overcome problems faced by such people, word prediction tools were developed which can help to communicate and also to help the people with less speed typing. The research on word prediction has been performing well. Word prediction technique does the task of guessing the preceding word which will follow few initial text fragments often. Existing systems work on a word prediction model, which suggests the next immediate word based on the current available word. These systems work using machine learning algorithms which have limitations to create accurate sentence structure. Developing technologies have been producing more accurate outcomes than the existing system technologies, models developed using deep learning concepts are capable of handling more data efficiently and predict better than ML algorithms.

The proposed model is based on Deep learning approaches. There is no need for any human interactions as the computer, once trained, learns from its experience and produces intended results in a quicker and efficient manner.

Sequential data in particular is very volatile in nature and is often expected to have many dependencies. To handle such data, special types of neural networks have been deployed and are proven effective in dealing with such expectations. Neural networks work on the basic principle of the human brain neurons. One such type of neural network known as RNN is widely used solving data prediction problems. RNN has its own limitations such as vanishing and exploding gradient descent problems, and inefficiency to keep track of long-term dependencies. To overcome these drawbacks, Long Short Term Memory (LSTM) has been a better solution to deal with sequential data and text data in particular.

1.4 Objectives

- To create a software prototype which will be able to predict the next word while typing in order to increase rate of communication.
- To design a system that will reduce number of keystrokes while typing.
- To review the previous studies and efforts put on to build the predictive models

1.5 Outcomes

The following are the project outcomes:

1. To increase users rate of communication

   This system will suggest users some words based on their text written previously which would allow them to directly use the words instead of typing those words one after another and even can suggest some words that possibly not clicked at that time. This will save a lot of time and contribute to increasing the rate of communication.
2. To reduce the number of keystrokes

   The main goal of these systems is to increase the keystroke saving (KSS), which is the percentage of keystrokes that the user saves by using word prediction systems, besides ensuring a good quality of the produced text.
3. To help physically challenged people

   It provides facilities to impaired people to use computer programs, mainly for social interaction.
4. To help improving particular language skills of users and new word prediction

   The project also ensures users get new words to learn and help them to try and recall words when they are stuck at a point while writing. This would increase their language skills.

CHAPTER 2
LITERATURE SURVEY

There exist studies based on how the word prediction model can be built on unigram, bigram and trigram. Authors have described the prediction model to predict the next word in Assamese language using the higher order n-grams [1].

N-gram Approach

The traditional NLP approach for text prediction is the n-grams approach. Each gram is a word in another set of words and training is performed in order to extract information and create a set of features to categorize the data **[2].** This mainly uses joint probability tables to increase the rate of understanding but also recursively requires lots of information to get trained which makes it an unfeasible approach. However, the Naive Bayes overcomes the issue of recursive training but is less accurate. In the context of predicting in an applicable time, Latent semantic analysis (LSA) was created, which analyzes the relationship between the text and even after being an accurate technique for prediction, its inference highly depends on the frequency of words and not on the sequence of words. In this paper, the author developed the word predictive model using the hybrid technology of Naive Bayes and LSA and efforts were made to improve the prediction precision through Gradient Descent **[3][4].**

Machine learning Approaches

When the predictive system was advancing, NLP techniques employed machine learning approaches which were relatively time consuming. Later, various statistical approaches for text generation such as bag of words, word2vec are applied which generates a bag of vocabulary for text generation. These approaches could generate text at character level or word level without ensuring the generation of meaningful sentences because they lack understanding the context of the sentence. In order to ensure grammatical errors, methods like Lemmatization, POS Tagging, etc. are adapted. Word embedding features in the neural networks gained popularity over the traditional system for its correctness in generation of text **[5]**. This introduced various models like RNN, LSTM, Bi-LSTM.

RNN-based models view text as a sequence of words, and are intended to capture word dependencies and text structures but due to limitation of vanishing gradient, more featured models are used such as LSTM and Bi-LSTM **[12].** Bi-LSTM accuracy as compared to LSTM is examined more accurate since input flows in forward and backward direction and it learns quickly as compared to LSTM. **[6]**

Word Classification

Fazly A, et al. **[7]** developed a word prediction model using Machine learning and new feature extraction and selection techniques which was adapted from Mutual Information (MI) and Chi Square ($\chi 2$). They casted this problem as a word classification task and a bunch of words were classified to determine the most correct word in a given context.

Dipti Pawade, et, al. have worked in the field of text generation in which they generated a new story by combining two different stories. They have given multiple input files and then generated an output story aligned to the stories of input files to some extent. **[8]**

Multi-Window Convolution and Residual Network

Yang, et al. have proposed a MCNN-ReMGU model based on Multi-Window Convolution and Residual Network for natural language word prediction. The proposed model solves the problem of redundant network layers which lead to poor network performance by adopting residual connection to the original MGU and also the activation function is modified to ReLU to train deeper networks. The shown work reflects 65.2% validation accuracy and 63.7% testing accuracy for MCNN-ReMGU model, 85.6% validation & 84.1% testing accuracy when without batch comparison and 70.2% validation and 67.8% testing when without L2 norm **[9].**

Gated Recurrent Unit (GRU)

O. F. Rakib, et, al. have proposed next word prediction for Bangla Language using GRU (Gated Recurrent Unit) based RNN (Recurrent Neural Network) on n-gram dataset to develop language models that can predict the word(s) from the input sequence supplied is the recommended technique. On average, the accuracy of the 5-gram model is 99.70

percent, the 4-gram model is 99.24 percent, the Tri-gram model is 95.84 percent, the Bi-gram model is 78.15 percent, and the Unigram model is 32.17 percent. **[10]**

Linear interpolation and geometric interpolation models

Cavalieri, et al. presented an interpolation model to develop a word prediction model by merging the part of speech based model and n-gram language model using three different languages namely Spanish, Portuguese and English. They have built different POS-based models for each language using linear interpolation and geometric interpolation models. For the word prediction task, they have defined a partial derivative function to derive the interpolation model which was used to improve the prediction. Although, the model can be further extended to add information like semantic checks and more enhanced interpolation models. **[11]**

Habib, et al. proposed a word prediction model in Bangla language using a stochastic model. A novel approaches Unigram, Bigram, Trigram, Back off and Linear Interpolation is proposed in order to find the optimum language model based on performance metric. In addition, for finding out better performance, a large Bangla corpus of different word types is used. **[13]**

CNN and Bi-LSTM model

BaekCheol Jang, et al. proposed the sequence classification problems in the Word prediction System using CNN and Bi-LSTM model. This paper solves the problem of extracting meaningful information from big data, classifying it into different categories, and predicting end-user behavior or emotions. The problems are improved by extending the bidirectional LSTMs to enhance model performance. **[14]**

Harsha Vardhana, et al. developed a word level predictive model using LSTM and RNN that simply generates text based on the history of sentences. The long term dependency property helps to predict the words more accurately and the work shows how the LSTM model can be used to generate text and analysis of the performance of the model with other models like WD-LSTM, Baseline- Seq2Seq and Random. **[15]**

CHAPTER 3

METHODOLOGY

3.1 Dataset

The dataset for the training is a story "Metamorphosis", written by Shakespeare which is easy to predict text especially for a specific domain of knowledge. This approximately contains 2000 unique word sentences which in turn generates around 20,000 sequences.

3.2 Proposed Methodology

The figure 3.2.1 describes an overview of the proposed methodology. The process starts from the data extraction which follows preprocessing on the data and data transformation to get the tokens (words) from the text. The model is built using Bi-LSTM and LSTM layers and finally the model is trained. Each iteration evaluates the model results of training in terms of its accuracy and the model is saved. The saved model is then used to predict words based on user input to evaluate the model's performance.

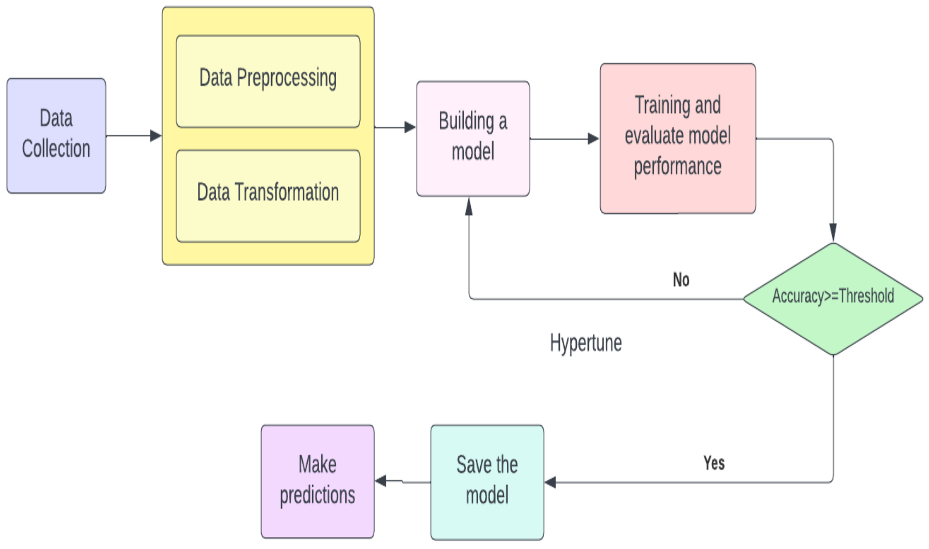
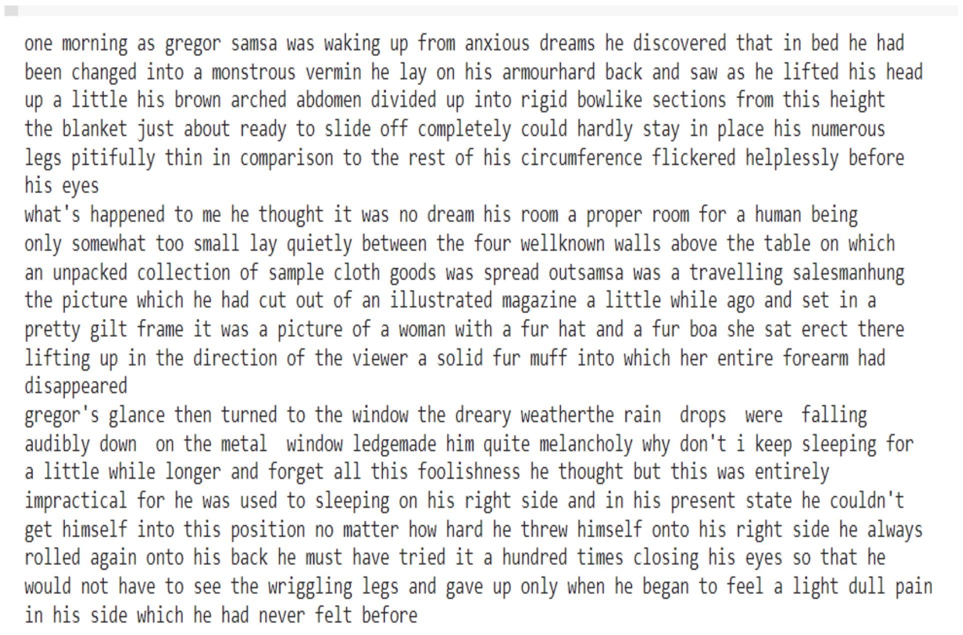


Fig. 3.1 Systems Overview

3.3 Data Preprocessing

Preprocessing must be done to remove noise, uncertainty from the data and to clean the data to make it ready for further analysis. As Data preprocessing is concerned, all the punctuation marks are removed, the proper spacing is ensured and the uniformity of data is maintained.



```
one morning as gregor samsa was waking up from anxious dreams he discovered that in bed he had
been changed into a monstrous vermin he lay on his armourhard back and saw as he lifted his head
up a little his brown arched abdomen divided up into rigid bowlike sections from this height
the blanket just about ready to slide off completely could hardly stay in place his numerous
legs pitifully thin in comparison to the rest of his circumference flickered helplessly before
his eyes
what's happened to me he thought it was no dream his room a proper room for a human being
only somewhat too small lay quietly between the four wellknown walls above the table on which
an unpacked collection of sample cloth goods was spread outsamsa was a travelling salesmanhung
the picture which he had cut out of an illustrated magazine a little while ago and set in a
pretty gilt frame it was a picture of a woman with a fur hat and a fur boa she sat erect there
lifting up in the direction of the viewer a solid fur muff into which her entire forearm had
disappeared
gregor's glance then turned to the window the dreary weatherthe rain  drops  were  falling
audibly down  on the metal  window ledgemade him quite melancholy why don't i keep sleeping for
a little while longer and forget all this foolishness he thought but this was entirely
impractical for he was used to sleeping on his right side and in his present state he couldn't
get himself into this position no matter how hard he threw himself onto his right side he always
rolled again onto his back he must have tried it a hundred times closing his eyes so that he
would not have to see the wriggling legs and gave up only when he began to feel a light dull pain
in his side which he had never felt before
```

Fig. 3.2 Data Preprocessed

To achieve uniformity in the data and split the sentences into words, a standard set of preprocessing is performed using the Tokenizer library from Keras. The Tokenizer class is trained on entire data and all the unique words are separated. This will generate a dictionary where each token/word will be assigned a particular place i.e., they get indexed.

For example,

Suppose a sentence given as: “**friendly laugh that made her unable to speak straight away**”

```
Dictionary = {“friendly” : 112, “her” :14,  “laugh” : 89,  “made” : 90,
“speak” : 55, “that” : 12, “to” : 10, “unable” : 15 .…}
```

In this way the tokens are indexed and will be used to generate input sequences.

3.4 Modeling Approach

Text prediction has numerous implementations and in our work we have proposed Long short term memory RNN which overcomes the drawback of simple RNN.

3.4.1 Long Short term Memory (LSTM)

The LSTM recurrent unit tries to “remember” all the past knowledge that the network has seen so far and to “forget” irrelevant data. LSTM has three layers as a forget gate, input gate and output gate which makes it capable of learning long term dependencies.

Forget Gate:

Ex: Aman is a nice person. Vishal is evil.

In the first sentence our subject is Aman and the second sentence's subject is Vishal. System first inputs the first sentence and processes it. After accepting the second sentence, the system should realize that now we are no longer talking about Aman so no need to store it and the forget gate will remove it from the memory cell.

Input Gate:

Ex: Vishal knows swimming and he is gold medalist in it.

Now in both sentences we have a common subject which is Vishal, so now the system should realize the context is matching and store the previous one in memory for further processing.

Output Gate:

Ex: Vishal is very brave and he wants to work in ______.

During this process, we have to complete the sentence. Now, when the system sees “work in” they relate it to Vishal and based on the current expectation we have to give relevant words to fill in the blank. This is the function of Output gate.

The vital part when it comes to predicting the next words is CONTEXT and for that processing from one side is not enough. So for that purpose Bidirectional LSTM is used.

3.4.2 Bi-Directional LSTM (Bi-LSTM)

Bidirectional LSTM is a recurrent neural network used primarily on natural language processing. Unlike standard LSTM, the input flows in both directions, and it’s capable of processing data from both aspects backward and forward. It’s also an effective tool for modeling the sequential dependencies between words and phrases in both directions of the sequence.

Bi-LSTM adds one more LSTM layer, which reverses the direction of information flow. It means that the input sequence flows backward in the additional LSTM layer and then we combine the outputs from both LSTM layers. This functionality of Bi-LSTMs adds the advantage of complete and faster processing and learning on the input sequence.

For example:

Apple is something that …

It might be about an apple as a fruit or an Apple, a company. Thus, LSTM doesn’t know what “Apple” means, since it doesn’t know the context from the future.

It can be interpreted as Apple is something that competitors simply cannot reproduce. Or it can be, Apple is something that I like to eat.

In both sentences the context of Apple is different and it depends on previous data and it only happens with backward propagation.

As LSTM only recognizes the relationship between values at the beginning and end of a sequence. LSTM has no concern with the current word and previous sequence. Hence LSTM is not able to generate more meaningful output. However Bi-LSTM will have a different output for every word in a sequence. So, basically we have trained three models in order to draw a comparative analysis on the performance of LSTM and Bi-LSTM models.

3.3 System Specification

3.3.1 Hardware Requirements

- RAM: 8 GB or more

3.3.2 Software Requirements

- Operating system: Windows 10 or higher
- Technology used: Python (Tensorflow, Pandas) , Flask
- Tools: Jupyter Notebook | Google Colab
- Cloud for deployment

3.5 Technology Stack

The project is divided into two parts, one is training a model to predict words and another one is to develop a deliverable for the predictive system. For the user's sake, an editor is designed in which they can efficiently write their articles, text, etc. by making use of a predictive system.

Table 3.5.1 Predictive model development Tools

Logo	Language and Libraries used
python™	**Python:** We used python as a programming language. Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation.
	Tensorflow: TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.
K Keras	**Keras:** Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It is an open-source software library.
Pandas	**Pandas:** Pandas is a fast, flexible and easy to use data analysis and manipulation tool built on top of Python programming language.
NVIDIA.	**GPU:** Server used for faster processing.

Table 3.5.2 Website development Tools

Logo	Language and Libraries used
python™	**Python:** Python programming language is used as a development language for the backend in the website.
Flask web development, one drop at a time	**Flask:** Backend development is done using Flask, a Python web framework. It has a small and easy-to-extend core: it's a micro framework and is easy for small web applications development.
HTML 5 CSS 3	**HTML/CSS:** For the frontend portion, mainly styling and structuring HTML and CSS is used which is easy and
JS	**JavaScript:** JavaScript programming language is used in the frontend to make the web application dynamic in nature.

CHAPTER 4

IMPLEMENTATION

4.1 Training the data

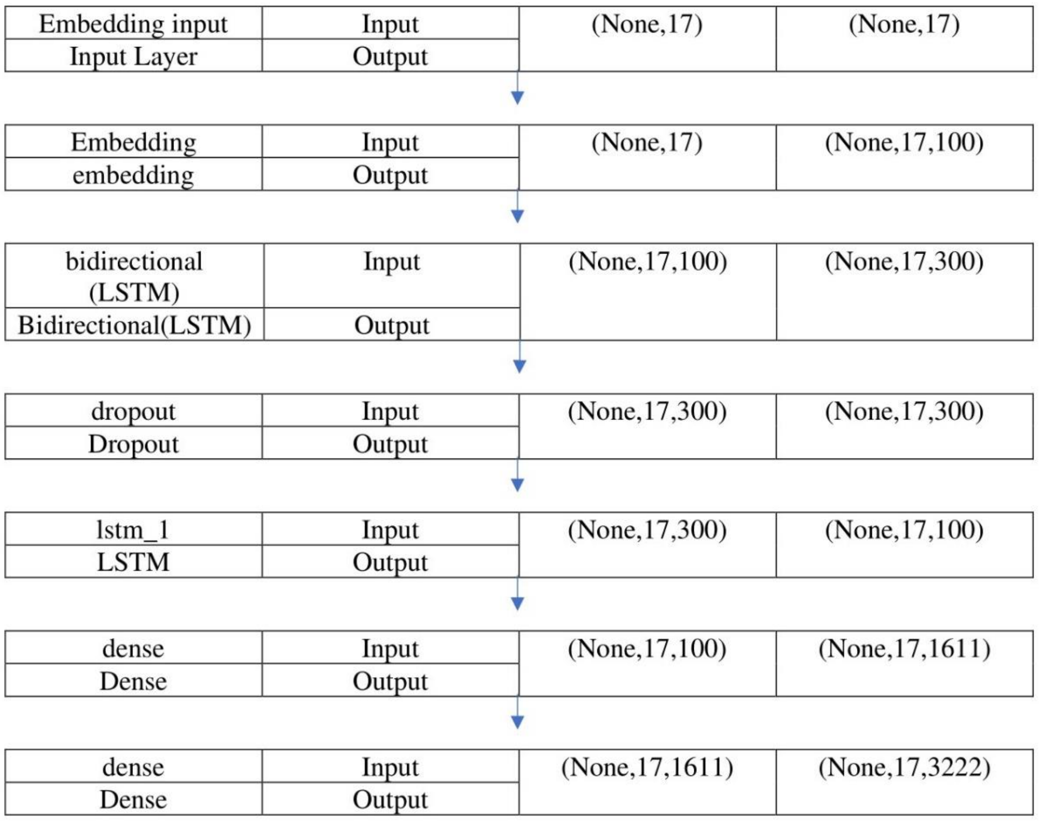
Using the dictionary of unique words, the input sequence for each sentence would be generated as follows:

> Friendly - [112], Friendly laugh - [112, 89], Friendly laugh that - [112, 89, 12], Friendly laugh that made - [112, 89, 12, 14], Friendly laugh that made her - [112, 89, 12, 90, 14, 15], etc.

These input sequences help to find dependency of words and which words would be most likely to come after a word. Based on the previous inputs, the next word will be predicted. Then padding sequences are used which would normalize the input sequences and ensure all sequences have the same length.

4.2 Model Construction

The architecture of the deep learning model utilized for the task of predictive system generation. The class labels are converted to a one-hot encoded vector using Numpy arrays which converts the categorical data into a better format. A sequential model is built using an embedding layer which enables it to convert the word into a fixed length vector. The next layer is a Bi-LSTM layer of batch size = 150 with a dropout layer followed by a LSTM layer with activation functions like RELU and SIGMOID. The model’s structure and layers are shown in Figure 4.2.



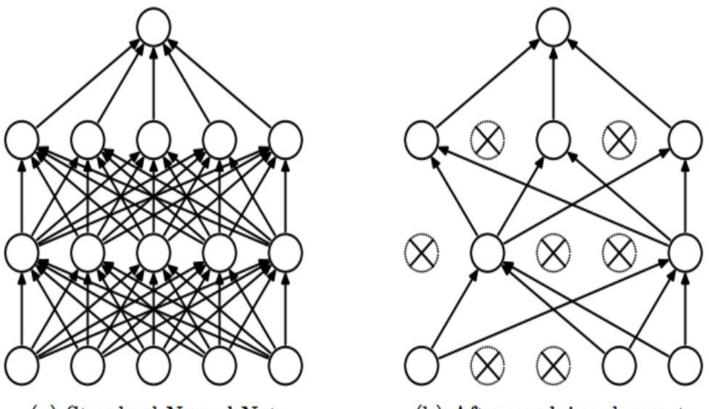
Embedding input	Input	(None,17)	(None,17)
Input Layer	Output		
Embedding	Input	(None,17)	(None,17,100)
embedding	Output		
bidirectional (LSTM)	Input	(None,17,100)	(None,17,300)
Bidirectional(LSTM)	Output		
dropout	Input	(None,17,300)	(None,17,300)
Dropout	Output		
lstm_1	Input	(None,17,300)	(None,17,100)
LSTM	Output		
dense	Input	(None,17,100)	(None,17,1611)
Dense	Output		
dense	Input	(None,17,1611)	(None,17,3222)
Dense	Output		

Fig. 4.1: Architecture of predictive model

Each layer that is used in this model and their functions are described in brief below:

- **Dropout Layer:**

  Dropout is a technique that drops neurons from the neural network or 'ignores' them during training, in other words, different neurons are removed from the network on a temporary basis. In our case we have added a Dropout layer of 0.2 in order to prevent any kind of over fitting.


(a) Standard Neural Net (b) After applying dropout.

- **Dense Layer**

  Dense Layer is a simple layer of neurons in which each neuron receives input from all the neurons of the previous layer, thus called dense.

- **Activation Function**

  i) **Softmax Activation function:** It is mainly used to normalize neural networks output to fit between zero and one. It is used to represent the certainty “probability” in the network output.

  ii) **ReLU Activation function:** The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

- **Categorical_crossentropy**: Used as a loss function for multi-class classification models where there are two or more output labels. The output label is assigned a one-hot category encoding value in the form of 0s and 1. The output label, if present in integer form, is converted into categorical encoding using Keras.

- **Optimizers** are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce the losses. Optimizers are used to solve optimization problems by minimizing the function.

- **Adam** is a replacement optimization algorithm for stochastic gradient descent for training deep learning models.

The whole layered structure will scan the sequences of text from left to right and right to left and will understand the text with its context. Finally at each iteration, it will save the results and that saved model will be used to predict the words based on the input given to it.

4.3 THE PREDICTION PROCESS

After a model is trained an input sequence of seed words is given as an input. These seed words help the network to initiate the process of prediction. A random line is considered from the input text and the model returns the words with the highest probability and this predicted word is then appended back to the input seed sentence and is passed on to the next layer. This will occur iteratively until the desired length is reached.

The process of prediction using API call is shown in the following figure

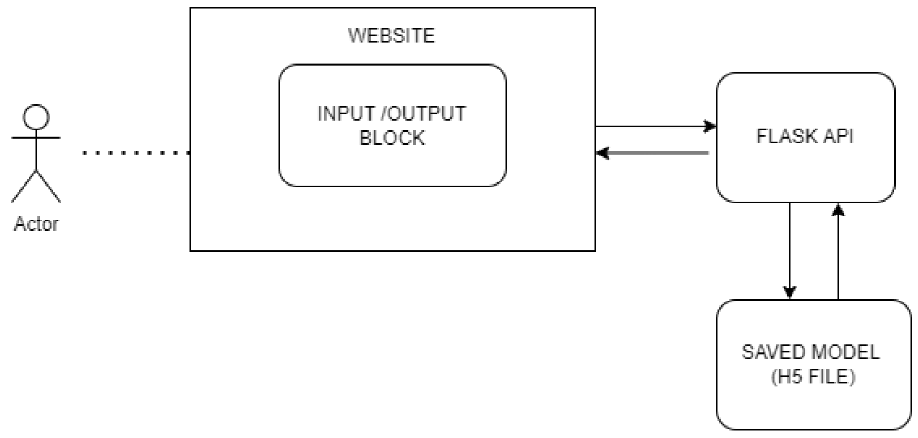
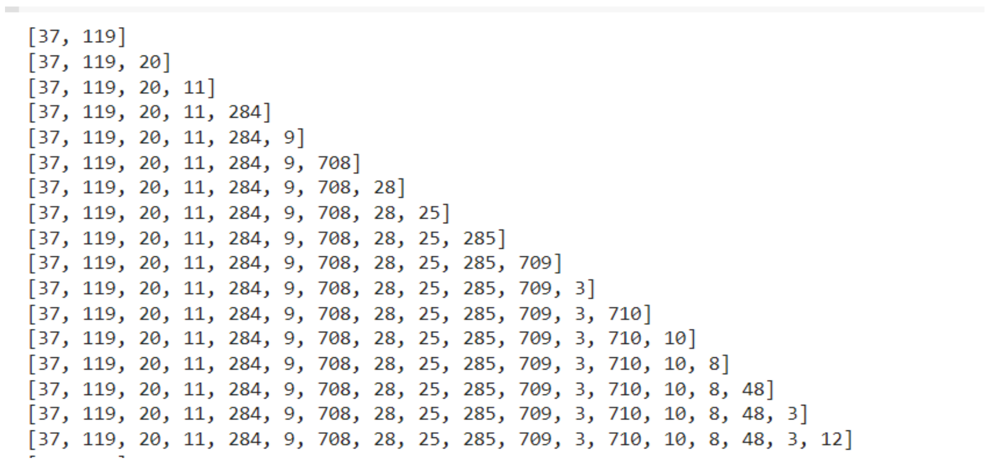


Fig. 4.2 The block diagram depicting the process of prediction using API calls.

CHAPTER 5

RESULTS AND DISCUSSION

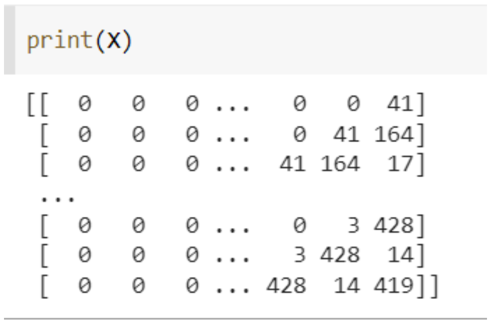
The given data is pre-processed before training the model as mentioned in detail in Section 3.2. Input sequences that are generated from the input data are shown in figure 5.1.



```
[37, 119]
[37, 119, 20]
[37, 119, 20, 11]
[37, 119, 20, 11, 284]
[37, 119, 20, 11, 284, 9]
[37, 119, 20, 11, 284, 9, 708]
[37, 119, 20, 11, 284, 9, 708, 28]
[37, 119, 20, 11, 284, 9, 708, 28, 25]
[37, 119, 20, 11, 284, 9, 708, 28, 25, 285]
[37, 119, 20, 11, 284, 9, 708, 28, 25, 285, 709]
[37, 119, 20, 11, 284, 9, 708, 28, 25, 285, 709, 3]
[37, 119, 20, 11, 284, 9, 708, 28, 25, 285, 709, 3, 710]
[37, 119, 20, 11, 284, 9, 708, 28, 25, 285, 709, 3, 710, 10]
[37, 119, 20, 11, 284, 9, 708, 28, 25, 285, 709, 3, 710, 10, 8]
[37, 119, 20, 11, 284, 9, 708, 28, 25, 285, 709, 3, 710, 10, 8, 48]
[37, 119, 20, 11, 284, 9, 708, 28, 25, 285, 709, 3, 710, 10, 8, 48, 3]
[37, 119, 20, 11, 284, 9, 708, 28, 25, 285, 709, 3, 710, 10, 8, 48, 3, 12]
```

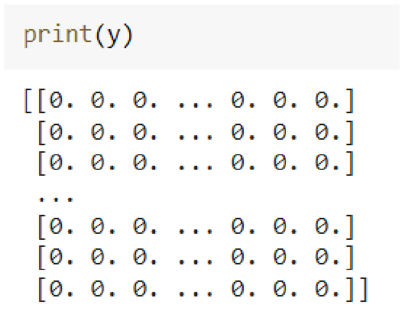
Fig. 5.1: Input sequences generated from text

From the input sequences, the required labels and predictor classes are generated and required one-hot encoding is done on predictor classes.



```
print(X)
```

```
[[  0   0   0 ...   0   0  41]
 [  0   0   0 ...   0  41 164]
 [  0   0   0 ...  41 164  17]
 ...
 [  0   0   0 ...   0   3 428]
 [  0   0   0 ...   3 428  14]
 [  0   0   0 ... 428  14 419]]
```

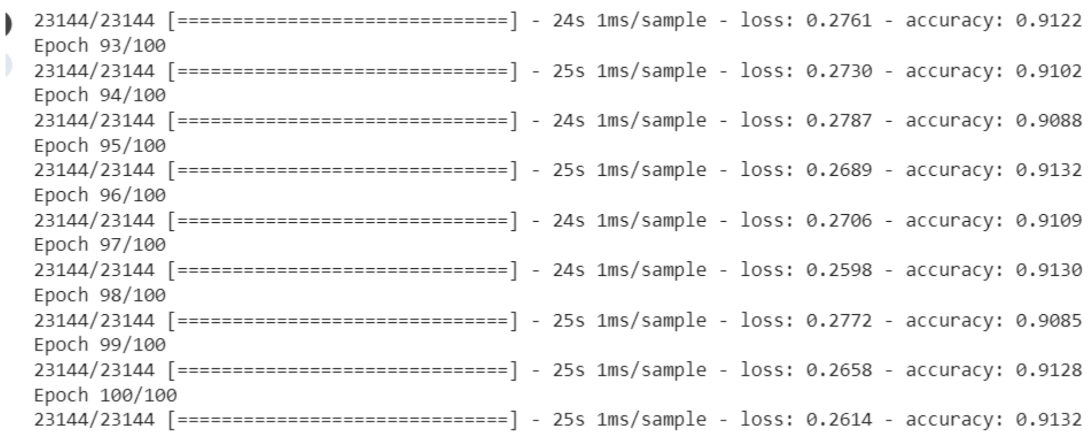


```
print(y)
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

Fig. 5.2 Labels and predictors classes

Model takes total 23144 samples of input sequences and run for 100 epochs results in the accuracy of 91%.



```
23144/23144 [==============================] - 24s 1ms/sample - loss: 0.2761 - accuracy: 0.9122
Epoch 93/100
23144/23144 [==============================] - 25s 1ms/sample - loss: 0.2730 - accuracy: 0.9102
Epoch 94/100
23144/23144 [==============================] - 24s 1ms/sample - loss: 0.2787 - accuracy: 0.9088
Epoch 95/100
23144/23144 [==============================] - 25s 1ms/sample - loss: 0.2689 - accuracy: 0.9132
Epoch 96/100
23144/23144 [==============================] - 24s 1ms/sample - loss: 0.2706 - accuracy: 0.9109
Epoch 97/100
23144/23144 [==============================] - 24s 1ms/sample - loss: 0.2598 - accuracy: 0.9130
Epoch 98/100
23144/23144 [==============================] - 25s 1ms/sample - loss: 0.2772 - accuracy: 0.9085
Epoch 99/100
23144/23144 [==============================] - 25s 1ms/sample - loss: 0.2658 - accuracy: 0.9128
Epoch 100/100
23144/23144 [==============================] - 25s 1ms/sample - loss: 0.2614 - accuracy: 0.9132
```

Fig. 5.3 Training accuracy and model performance

Comparison of accuracy of LSTM and Bi-LSTM is done in which it is observed that after 25 epochs the LSTM gives an accuracy of 74% and Bi-LSTM gives an accuracy of 91%.

Table 5.1 Model accuracy comparison of LSTM and Bi-LSTM

Model	**Epochs**	**Accuracy**
LSTM model	25	74%
Bi-LSTM model	100	91%

With all the different layers, the model is trained for 100 epochs with training samples of around 20,000 sequences. The accuracy curve is shown in figure 5.2 and Loss curve is shown in figure 5.3.

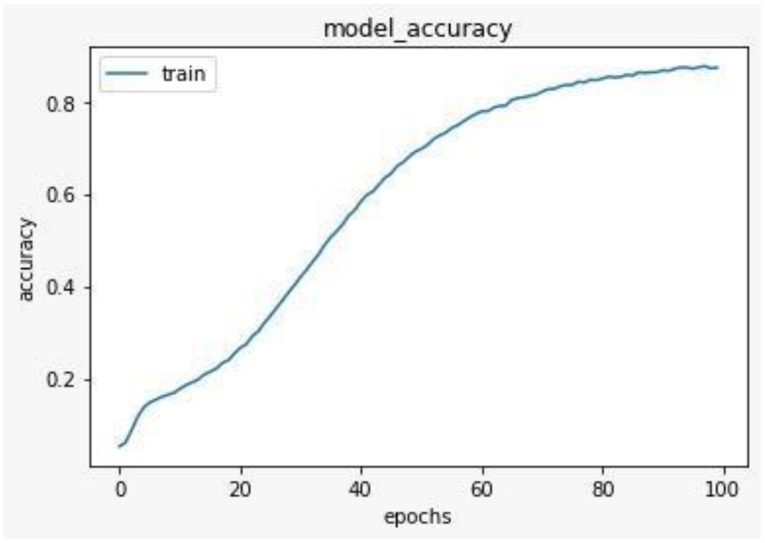


Fig. 5.4: Model Accuracy graph

```
: <matplotlib.legend.Legend at 0x7fb96bbb1e48>
```

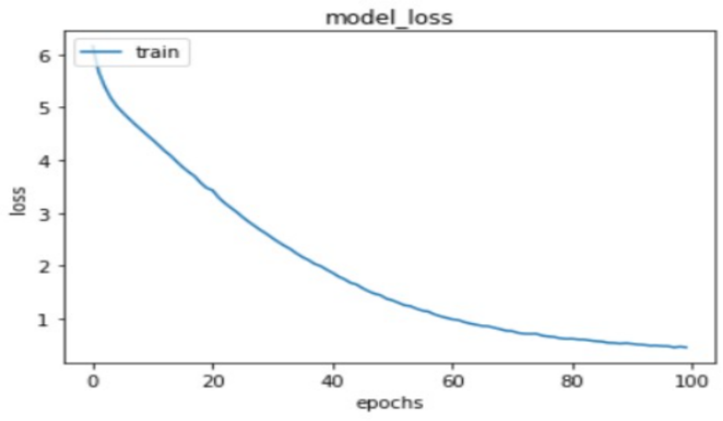
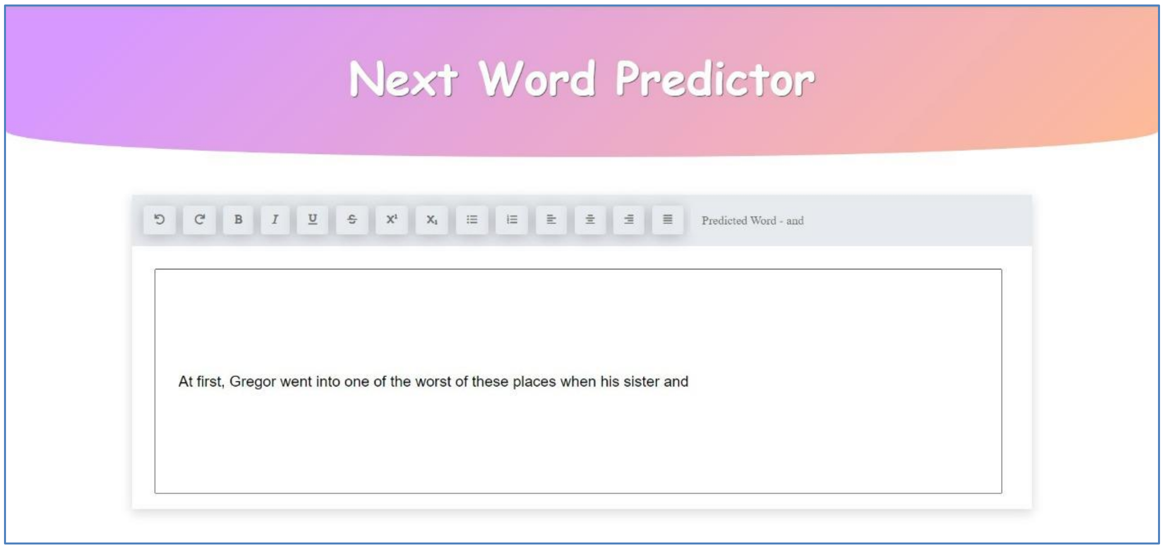


Fig. 5.5: Model Loss graph

The demonstration for this predictive model is shown using a website which is an editor where users can write text, blogs, articles, etc. and can get the predictions. The figure 5.4 shows the editor and how it predicts words when a user requests for a word or when it types text.

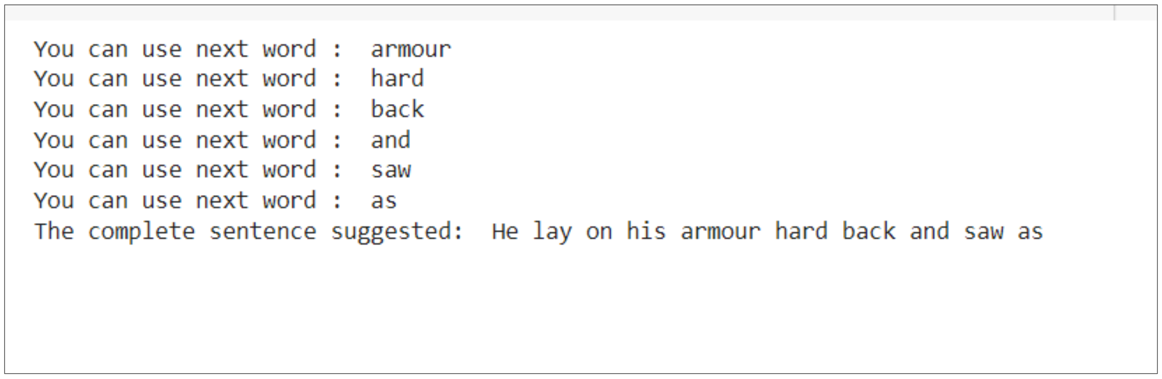


At first, Gregor went into one of the worst of these places when his sister and

Fig 5.6: Shows Home page of the word prediction website

As mentioned in Section 4.3, how the prediction is carried out, the Fig. 5.5 shows the output by taking some set of words.

The seed text: “He lay on his”



```
You can use next word :  armour
You can use next word :  hard
You can use next word :  back
You can use next word :  and
You can use next word :  saw
You can use next word :  as
The complete sentence suggested:  He lay on his armour hard back and saw as
```

Figure 5.7: Words predicted after seeding some text

Following Snippets shows the designed website UI.

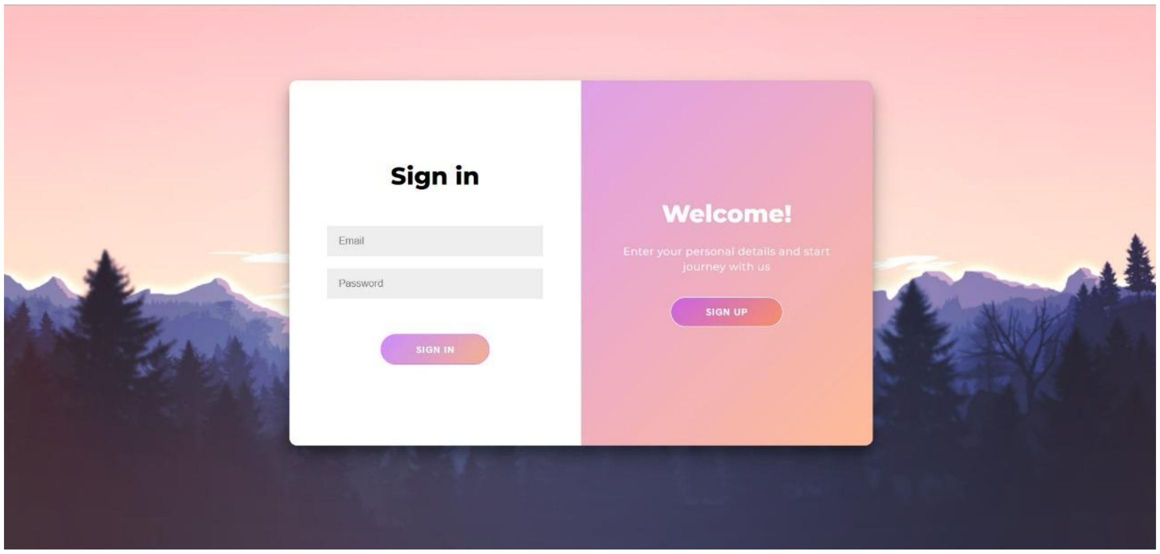


Fig. 5.8: User sign in and authentication

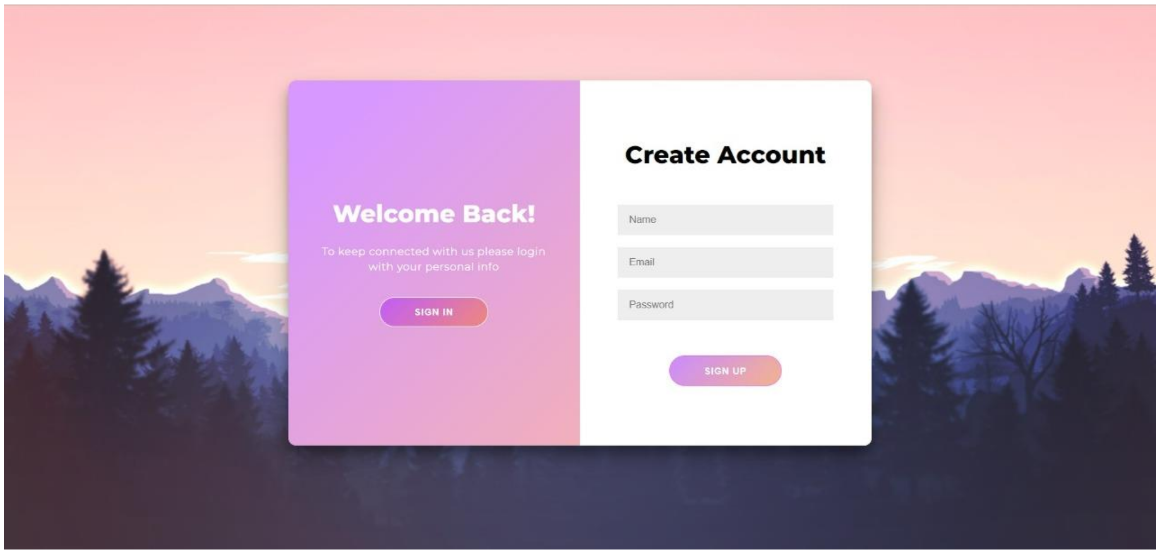

Welcome Back!
To keep connected with us please login with your personal info
SIGN IN
Create Account
Name
Email
Password
SIGN UP

Figure 5.9: User signup and authentication

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

Text generation is the task of generating text with the goal of appearing indistinguishable to human-written text. Text generation is a subfield of natural language processing. It leverages knowledge in computational linguistics and artificial intelligence to automatically generate natural language texts, which can satisfy certain communicative requirements.

Our model demonstrates how to generate text using a word-based RNN. We have taken a dataset of Shakespeare's famous word "Metamorphosis" and also one of the Harry Potter's editions to train the model. Given a sequence of words from this dataset, train a model to predict the next word in the sequence. Longer sequences of text can be generated by calling the model repeatedly.

The Report mainly focused on deep learning techniques like RNN, LSTM and Bi-LSTM which gives different results on different epochs giving varied accuracy. Large number of datasets was used for proper prediction. The research shows text sequence prediction can be implemented through deep learning techniques which can change the scenario of typing whole sentences. The study of various papers gives a clear edge stating deep learning might provide better results when compared with other techniques. Previously, machine learning and natural language processing were used in prediction but deep learning models produced better accuracy.

We introduced a brief design and working of recurrent neural network language models with LSTM units. LSTMs and Bi-LSTM are the most widely used networks for language modeling. The main objective of this work is to train an LSTM model to generate text on a given dataset and analyze its performance. When it comes to implementing an LSTM in Keras, the process is similar to implementing other neural networks created with the sequential model. We start by declaring the type of model structure we are going to use, and then add layers to the model one at a time. LSTM layers are readily accessible to us in Keras, we just have to import the layers and then add them. Embedding Layer is the primary layer. In between the primary layers of the LSTM, we will use layers of dropout, which helps prevent the issue of over fitting. Finally, the last layer in the network will be a densely connected layer that will use a Softmax and ReLU activation function and output probabilities.

An accuracy of 88% has been achieved. This accuracy can further be increased by further reducing the training loss by tweaking the hyper parameters of the model. Experimental results show that the proposed method outperforms existing traditional text generation systems.

Future Scope

In the future, the system can be extended for other natural language generation tasks like story auto-completion, poem auto-completion, etc. We can elaborate the model by changing the dataset or limiting it to a specific dataset. The system can be adapted to new words that are not a part of its vocabulary. This adaptation will be done when the model encounters a new word and adds the word to the vocabulary. This way the model becomes more generalized. The system can be personalized to predict words based on the user's history.

REFERENCES

[1] M.P. Bhuyan, S.K. Sarma - "A Higher-Order N-gram Model to enhance automatic Word Prediction for Assamese sentences containing ambiguous Words", Volume-8, Issue-6, August, 2019, DOI: 10.35940/ijeat.F8706.088619

[2] W.B.Cavnar, et.al., "N-gram-based text categorization," Ann Arbor MI, vol. 48113, no. 2, pp. 161–175, 1994.

[3] Foltz, P.W. Latent semantic analysis for text-based research. Behavior Research Methods, Instruments, & Computers 28, 197-202, 1996. https://doi.org/10.3758/BF03204765

[4] Henrique X. Goulart, Mauro D. L. Tosi, Daniel Soares-Gonc ̧alves, Rodrigo F. Maia and Guilherme Wachs-Lopes: "Hybrid Model For Word Prediction Using Naive Bayes and Latent Information", 2018.

[5] C. Aliprandi, N. Carmignani, N. Deha, P. Mancarella, and M. Rubino, "Advances in nip applied to word prediction", J. Mol. BioI., vol. 147, pp. 195- 197,2008.

[6] Radhika Sharma, Nishtha Gael, Nishita Aggarwal, Prajyot Kaur and Chandra Prakash, - "Next word prediction in Hindi using Deep Learning techniques", 2019, DOI: 10.1109/ICDSE47409.2019.8971796.

[7] Fazly A., "The Use of Syntax in Word Completion Utilities," Master Thesis, University of Toronto, Canada, 2002.

[8] Dipti Pawade, Avani Sakhapara, Mansi Jain, Neha Jain, Krushi Gada, "Story Scrambler - Automatic Text Generation Using Word Level RNN-LSTM", International Journal of Information Technology and Computer Science(IJITCS), Vol.10, No.6, pp.44-53, 2018, DOI:10.5815/ijitcs.2018.06.05

[9] O. F. Rakib, S. Akter, M. A. Khan, A. K. Das and K. M. Habibullah, "Bangla Word Prediction and Sentence Completion Using GRU: An Extended Version of RNN on N-gram Language Model," 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), 2019, pp. 1-6, DOI: 10.1109/STI47673.2019.9068063.

[10] Daniel C. Cavalieri, Sira E. Palazuelos-Cagigas, Teodiano F. Bastos-Filho, and Mario Sarcinelli-Filho, "Combination of Language Models for Word Prediction: An Exponential Approach", IEEE Transactions on audio, speech, and language processing, VOL. 0, February 2015, DOI: 10.1109/TASLP.2016.2547743.

[11] Sourabh Ambulgekar1, Sanket Malewadikar2, Raju Garande, and Dr. Bharti Joshi, "Next Words Prediction Using Recurrent NeuralNetworks", 2021, DOI: 10.1051/itmconf/20214003034.

[12] Habib, Md AL-Mamun, Adbullah Rahman, Md Siddiquee, Shah Ahmed, Farruk, "An Exploratory Approach to Find a Novel Metric Based Optimum Language Model for Automatic Bangla Word Prediction", International Journal of Intelligence Systems and Applications, 2018, DOI: 10.5815/ijisa.2018.02.05.

[13] BaekCheol Jang, Myeonghwi Kim, Gaspard Harerimana, Sang-ug Kang and Jong Wook Kim- "Bi-LSTM Model to Increase Accuracy in Text Classification: Combining Word2vec CNN and Attention Mechanism"- 24 August 2020, DOI: https://doi.org/10.3390/app10175841

[14] Harsha Vardhana Krishna Sai Buddana, PVS. Manogna, Surampudi Sai Kaushik, Shijin Kumar P.S, "Word Level LSTM and Recurrent Neural Network for Automatic Text Generation" International Conference on Communication and Informatics, 2021.

[15] https://blog.keras.io/keras-as-a-simplified-interface-to-tensorflow-tutorial.html - Keras tutorials. (Accessed : 12 January, 2022)

[16] https://flask.palletsprojects.com/en/2.1.x/ - Flask documentation (Accessed : 21 March, 2022)

[17] https://keras.io/api/layers/recurrent_layers/lstm/-Keras LSTM documentation, (Accessed : 13 February, 2022)

[18] https://towardsdatascience.com/text-preprocessing-in-natural-language-processing-using-python-6113ff5decd8 - Text preprocessing. (Accessed : 13 February, 2022)

ANNEXURE A: NVIDIA DGX DL WORKSTATION

The Centre of Excellence has been helpful for faster processing in the project. RCOEM with association NVIDIA has set-up the **1st Centre of Excellence in Artificial Intelligence and Deep learning (AI & DL)** in center India.

Throughout the project, the facility provided by NVIDIA DGX DL Workstation of Centre of Excellence for AI and DL of RCOEM has been useful and the development process was faster and effective. This helped us to train model faster and focus on many other aspects of training a ML/DL model. The system has increased the training where our normal system used to take only few samples from the input samples seeded, the NVIDIA server helps to train on large data samples.