Department of Computer Science and Engineering

Project Report

on

Next Word Prediction Model Using Deep Learning

Introduction To Deep Learning
20-mark component

by

M Raj Karthik **1MS20CS066**

Under the guidance of

Brunda G

Lecturer, Dept. of Computer Science and Engineering

M S RAMAIAH INSTITUTE OF TECHNOLOGY

(Autonomous Institute, Affiliated to VTU)

BANGALORE-560054

www.msrit.edu

Abstract

The project aims to develop a text generation model leveraging Long Short-Term Memory (LSTM) neural networks. The primary objective of this project is to predict the next word in a given text sequence, making it a valuable tool for applications like autocomplete suggestions and natural language processing. To accomplish this, the project employs TensorFlow and Keras, key libraries in the deep learning and neural network space. The LSTM architecture is utilized for its ability to capture sequential patterns and dependencies in textual data. The project involves tokenizing and preprocessing text data from "IndiaUS.txt," training an LSTM-based model, and enabling users to input text prompts for generating coherent and contextually relevant next-word predictions.

Table of Contents

Chapter 1 – Introduction

With the advent of artificial intelligence and deep learning, the capability to predict and automate tasks has grown exponentially. Among these tasks, natural language processing (NLP) has taken a central role.

Next word prediction can be applied in various domains such as chatbots, virtual assistants, text editors, and more, making interactions smoother and more intuitive.This project aims to design a deep learning model that can predict the next word in a sequence, providing a foundation for numerous applications in the domain of NLP.

The project begins by importing essential Python libraries, such as NumPy, TensorFlow, and Keras. It then sources its training data from the "IndiaUS.txt" file, which presumably contains the corpus for the model. This file is read and tokenized using the Tokenizer module from Keras. The text data is transformed into sequences, and sequences are padded to ensure uniform input dimensions. The project's core lies in training an LSTM-based neural network, as LSTM is well-suited for understanding and predicting sequential patterns in language. The design and implementation steps involve configuring the model architecture, compiling it with appropriate settings, and training it on the processed text data to enhance its predictive capabilities.

Chapter 2 - Literature Review

1. This author includes the main algorithm and their accuracies which performed on predicting Next Word Prediction.In this paper, author proposedNgram language model.Language models assign probabilities to a series of words or a sentence or the probability of the next word given a preceding group of words.In the Stupid Back Off algorithm, the attempt to build language models by distributing true probabilities.These models can be useful in a variety of fields, such as spell correction, speech recognition, machinelearning, etc.N-gram modelling is utilized to suggest text accurately.The Ngram model has been used for next word prediction to reduce the amount of time.The model is 96.3% accurate.

2. Multi-window convolution and residual-connected minimal gated unit (MGU) network for the natural language word prediction.The convolution kernels with different sizes are used to extract the local feature information of different graininess between the word sequences.CNN extracts sequence feature information with different granularity by using different window sizes of convolution kernels.The overall experimental results show that the proposed MCNN-ReMGU significantly improves the performance of the word prediction task over the traditional methods.N-gram language model has been used to improve the text input rate in work.It is revealed that 33.36% reduction in typing time and 73.53% reduction in keystroke.The designed system reduced the time of typing free text which might be an approach for EHRs improvement in terms of documentation.

3. In this paper, author proposed LSTM-RNN language model for the Next Word Prediction.Federated Averaging Algorithm is used which averages the model parameters. After applying the gradient to all models of database.Federated training on Stack Overflow without any pretraining which yields the two learning curves that exhibit the lowest levels of train and validation accuracy respectively.Dimensionality reduction approach is useful for federated training in which we are constrained by model size.As this paper has achieved the level of accuracy around 22.1%, and

22.2%.Federated learning is a decentralized approach for training models on distributed devices, by summarizing local changes and sending aggregate parameters from local models to the cloud rather than the data itself.

4. This paper has used memory-based learning for next word prediction using Recurrent Neural Network and Long short-term memory.The goal of NLP is to learn and analyze the difficulties of automated generation and comprehending of languages of human beings.RNN has the capability of learning to utilize the previous information when the space between appropriate information and the position that is necessary is tiny. LSTM suffers from a large number of parameters, but it resolves the problem of memory.This model has reached up to the accuracy of 44.2%.Using hybrid-based technique such as Naive Bayes and Latent Semantic Analysis (LSA) model.The probabilistic method Naive Bayes is utilized in NLP like N gram.This model has produced accuracy of 88.2%.
5. Next word prediction is a highly discussed topic in current domain of Natural Language Processing research.Recurrent Neural Network based language model to improve prediction of next word in sequential data.LSTM to generate complex long-range structured sequences.The Next Word using LSTM with an accuracy of 88.02% for Assamese text and 72.10% for phonetically transcript Assamese language
6. Gated Recurrent Unit based Recurrent Neural Network on n-gram dataset.Change in earlier layers will effect output which is shallow. (inefficient).The loss analysis is high using in GRU when compare with Recurrent Neural Network.Average accuracy of 99.70% for 5-gram model, 99.24% for 4-gram model, 95.84% for Tri-gram model, 78.15% and 32.17% respectively for Bi-gram and Uni-gram models on average.
7. RNN help to predict next code syntax for users. The used LSTM neural language model to predict within vocabulary words.RNN is reduced no of layers which leads to low accuracy.The short version of LSTM in this CNN try to skip few layers while training result in less training time and they have good accuracy.This model is accuracy of around 56%.
8. In this research a Recurrent Neural Network has been trained to build a word predictor for Sinhala language. LSTM layers were built and analyzed to obtain the

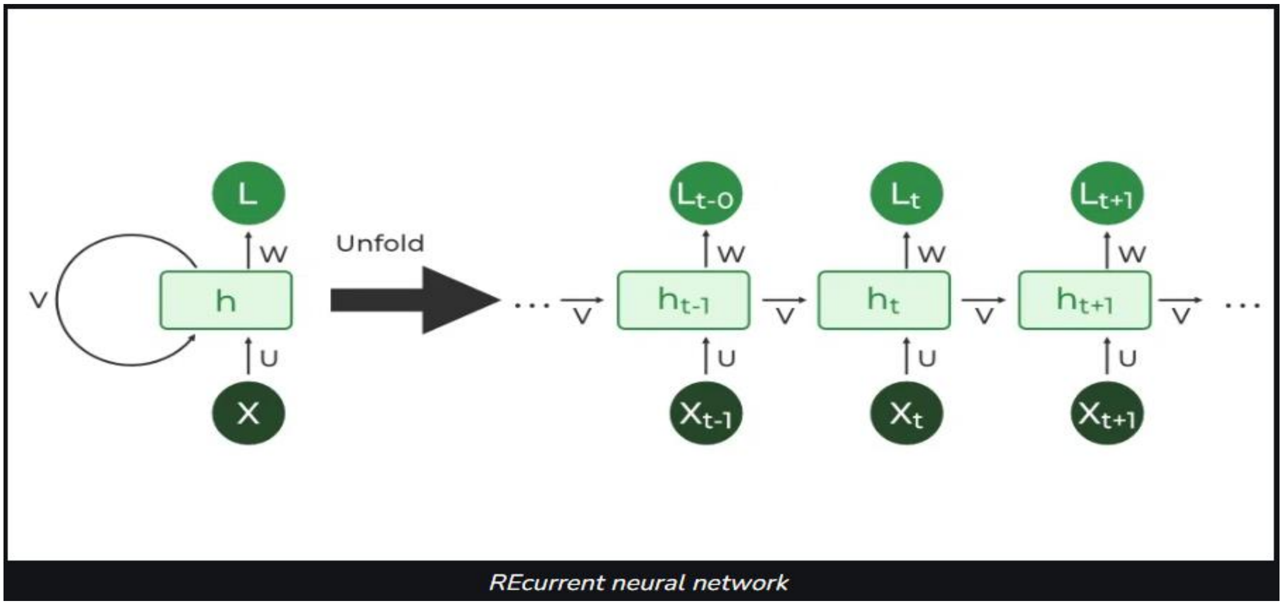
Keras model that displays the highest accuracy and minimum loss.It was observed that more the layers resulted in a decrease of the training and the validation accuracy.It is found that word predictive technologies have several benefits, and one is reducing the keystrokes from 50% to 60% which helps in typing that saves user time and effort.Accurate predictions with a percentage accuracy of 72%.

9. Long short-term memory (LSTM) with a Coupled Input and trained on the server and baseline n-grammodel was compared to the federated learning model trained from scratch. Recurrent NeuralNetworks are a collection of algorithms that is aimed at recognizing patternsThe loss analysis demonstrates that the least lost is in bidirectional RNN.Word prediction is helpful for users because it can boost typing speed and help to omit errors.Accuracy predictions with a percentage accuracy of 75%.
10. Federated Averaging Algorithm is used which averages the model parameters. After applying the gradient to all models of databaseFederated training on Stack Overflow without any pretraining which yields the two learning curves that exhibit the lowest levels of train and validation accuracy respectively.Dimensionality reduction approach is useful for federated training in which we are constrained by model size.As this paper has achieved the level of accuracy around 22.1%, and 22.2%.

Chapter 3 - Existing Techniques

Recurrent Neural Network:

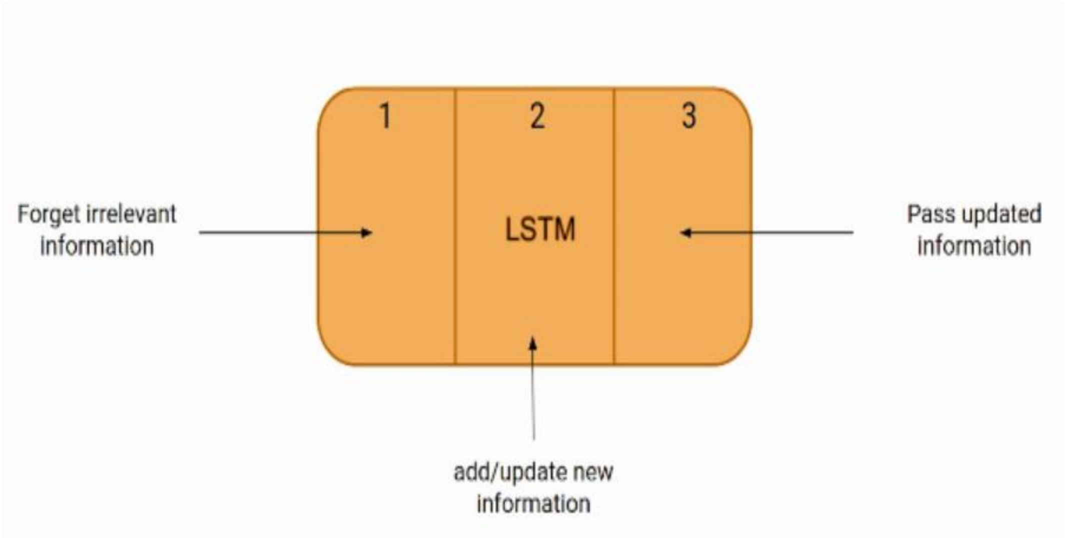
Recurrent Neural Network(RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is its Hidden state, which remembers some information about a sequence. The state is also referred to as Memory State since it remembers the previous input to the network. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.



REcurrent neural network

Long Short-Term Memory:

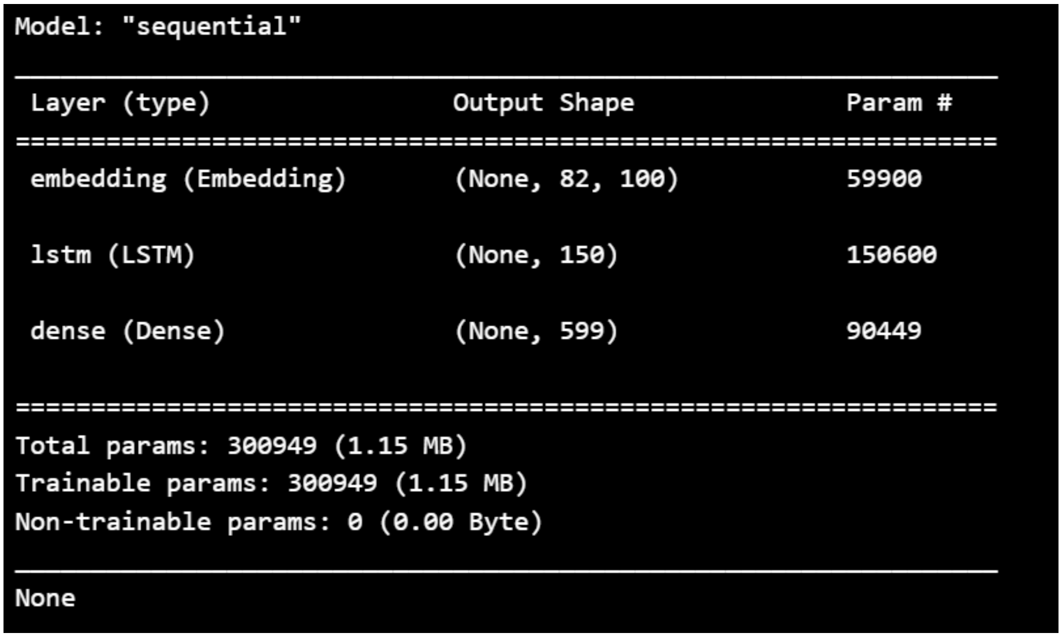
LSTM (Long Short-Term Memory) is a recurrent neural network (RNN) architecture widely used in Deep Learning. It excels at capturing long-term dependencies, making it ideal for sequence prediction tasks. Unlike traditional neural networks, LSTM incorporates feedback connections, allowing it to process entire sequences of data, not just individual data points. This makes it highly effective in understanding and predicting patterns in sequential data like time series, text, and speech. The LSTM network architecture consists of three parts, as shown in the image below, and each part performs an individual function.



Chapter 4 – Design and Implementation

The design and implementation of the "Next Word Prediction using LSTM" project consist of several key components. It begins with text tokenization, where the Tokenizer is employed to create a word index, allowing the project to map words to numerical indices and determine the total number of unique words. These tokenized sequences are used to construct the input data for the LSTM model. The model's architecture is defined with an embedding layer, LSTM layer, and a dense layer. The embedding layer helps convert words into dense vector representations, while the LSTM layer captures complex sequential patterns within the text data. The dense layer, equipped with a softmax activation, outputs the predicted next word.

After architecture definition, the model is compiled, specifying the loss function, optimizer, and evaluation metric. It employs categorical cross-entropy as the loss function, the Adam optimizer for gradient descent, and accuracy as the evaluation metric. The training phase iterates for 100 epochs, enabling the model to learn and improve its predictions. The evaluation results, though not explicitly provided in the code, would typically involve assessing the quality and relevance of the text generated by the model. Human evaluations and metrics like perplexity could be used to gauge the effectiveness of the text generation. The project's success is measured by the model's ability to generate meaningful and contextually appropriate next words based on user input, making it a valuable tool for applications in natural language processing.

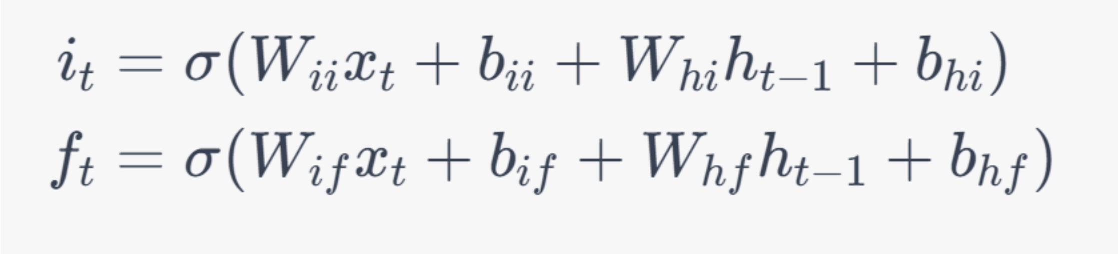


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 82, 100)           59900

 lstm (LSTM)                 (None, 150)               150600

 dense (Dense)               (None, 599)               90449

=================================================================
Total params: 300949 (1.15 MB)
Trainable params: 300949 (1.15 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

The Long Short-Term Memory (LSTM) network is a type of recurrent neural network (RNN) designed to overcome the vanishing gradient problem in traditional RNNs. LSTMs are particularly effective in learning and remembering patterns over long sequences of data. Let's break down the mathematical workings of an LSTM:

1. **Input and Forget Gates:**
    - **Input Gate (i_t):** Determines which information from the input should be added to the cell state.
    - **Forget Gate (f_t):** Determines what information should be discarded from the cell state.

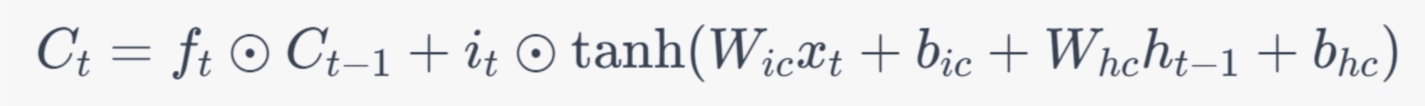
Mathematically:



$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi})$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf})$$

2. **Cell State Update:**
    - A new cell state (*Ct*) is computed based on the input and forget gates.

Mathematically:

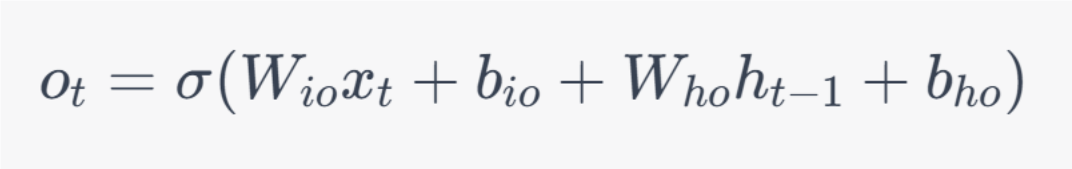


$$C_t = f_t \odot C_{t-1} + i_t \odot \tanh(W_{ic}x_t + b_{ic} + W_{hc}h_{t-1} + b_{hc})$$

⊙ represents element-wise multiplication.

3. **Output Gate:**
    - Determines the next hidden state (*ht*) based on the updated cell state.

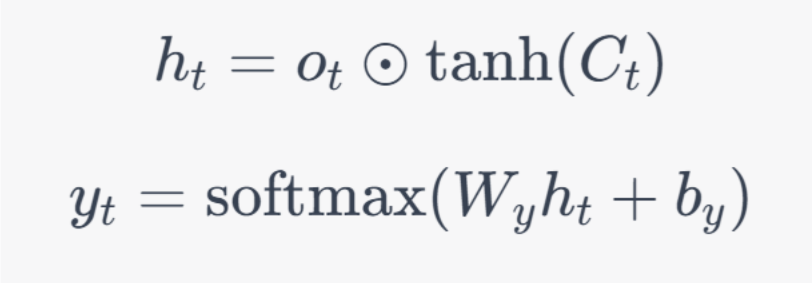
Mathematically:



$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho})$$

4. **Final Output:**
    - The final output (*yt*) is computed using the hidden state.

Mathematically:



$$h_t = o_t \odot \tanh(C_t)$$

$$y_t = \text{softmax}(W_y h_t + b_y)$$

Here:

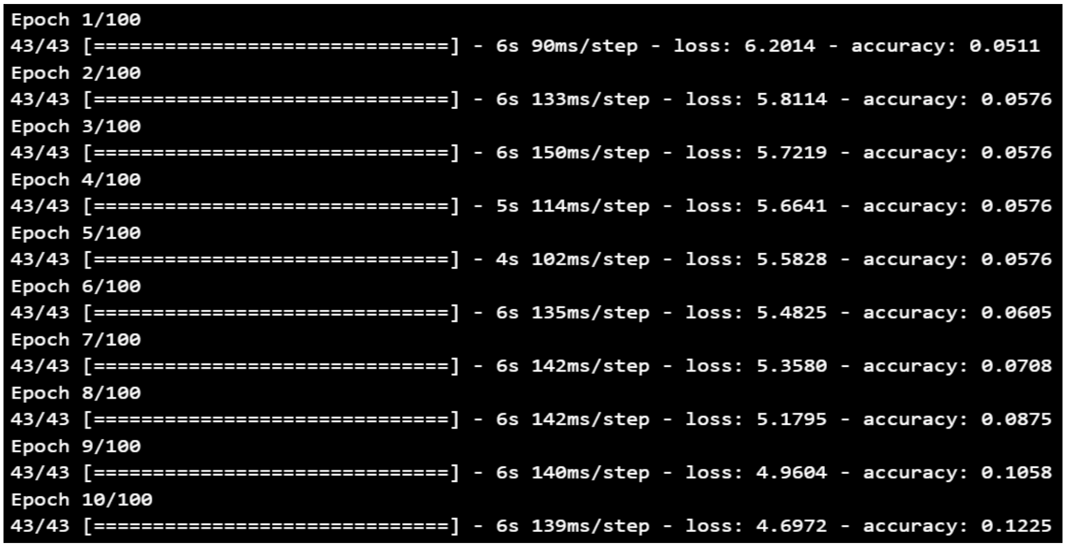
- W and b are weight matrices and bias vectors.
- σ is the sigmoid activation function.
- tanh is the hyperbolic tangent activation function.
- softmaxsoftmax is used for multi-class classification problems.

This mathematical structure allows LSTMs to selectively learn, forget, and update information over long sequences, making them powerful for tasks involving sequential data.

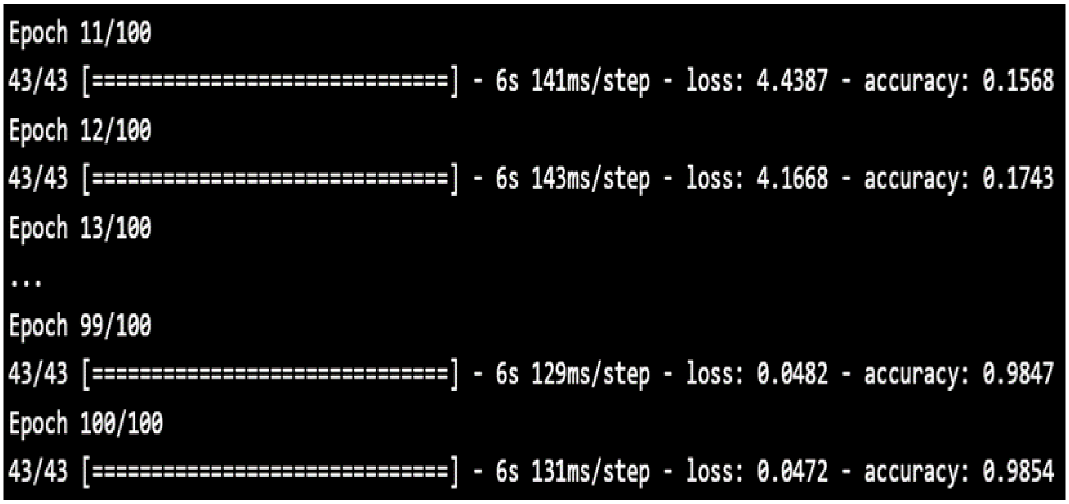
Chapter 5 – Evaluation Results

The training logs show the progress of the model during the training phase. The project utilizes a text generation model based on an LSTM architecture to predict the next word in a given text sequence. In this context, the "loss" is a measure of how well the model's predictions match the actual next words in the training data. The "accuracy" indicates the percentage of correct predictions during training.

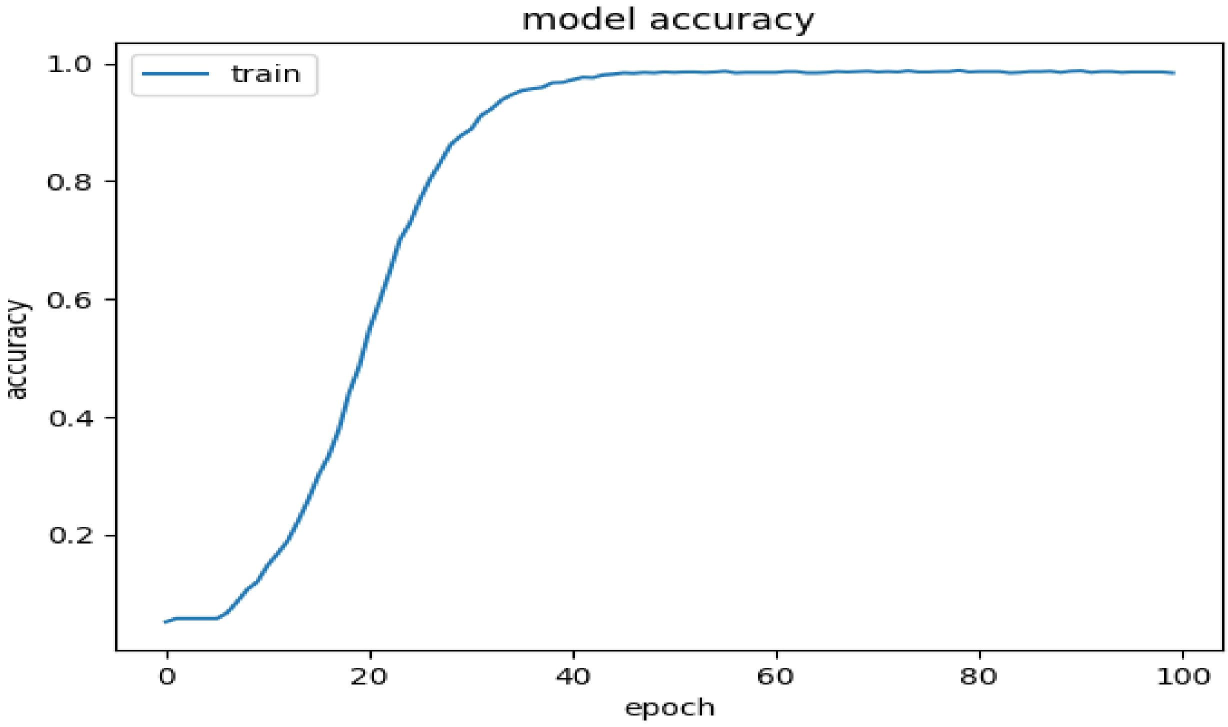
Throughout the training, we can observe that both the loss and accuracy values evolve. In the early epochs, the model starts with a high loss and low accuracy, which is expected as the model's predictions are random at the beginning. As the epochs progress, the loss decreases, indicating that the model is improving in making more accurate predictions. The accuracy, too, increases over time, showcasing the model's ability to predict the next word more precisely.

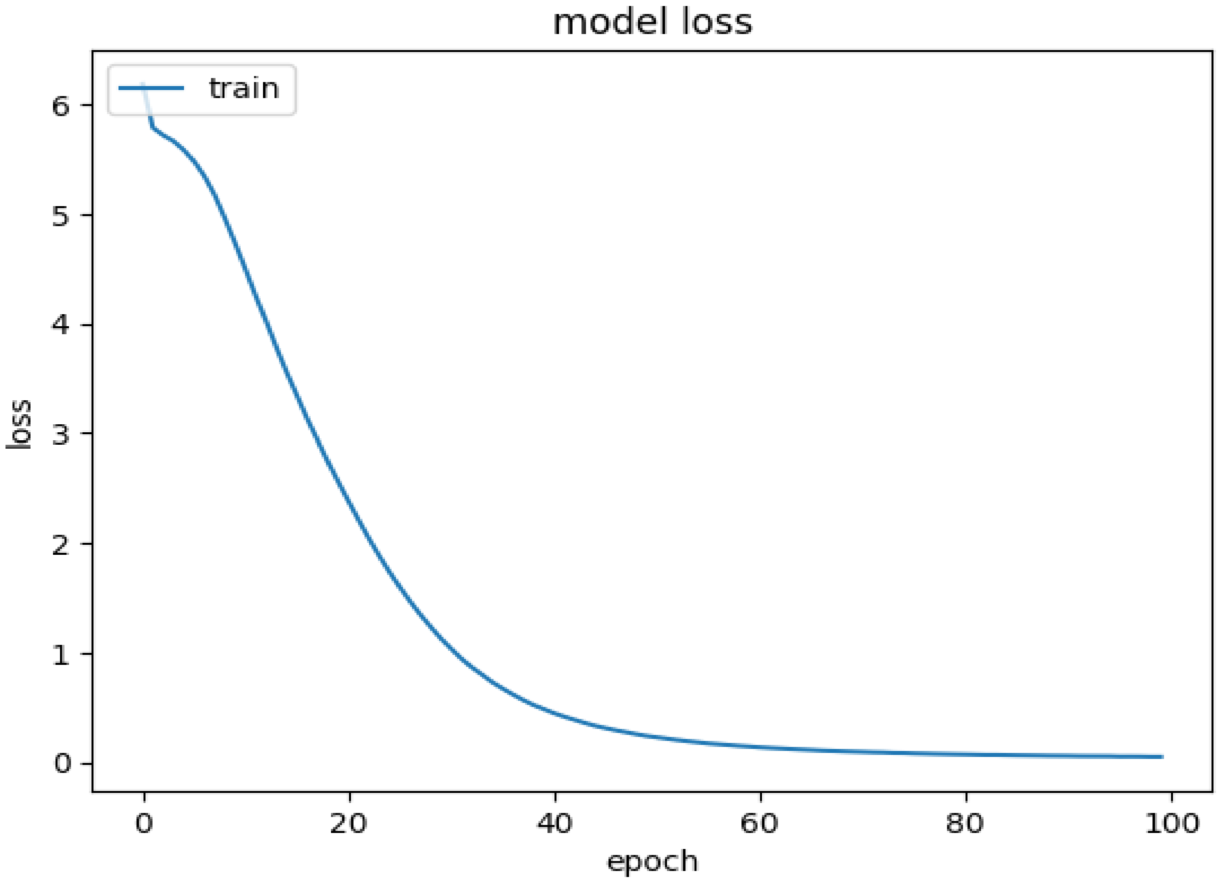


```
Epoch 1/100
43/43 [==============================] - 6s 90ms/step - loss: 6.2014 - accuracy: 0.0511
Epoch 2/100
43/43 [==============================] - 6s 133ms/step - loss: 5.8114 - accuracy: 0.0576
Epoch 3/100
43/43 [==============================] - 6s 150ms/step - loss: 5.7219 - accuracy: 0.0576
Epoch 4/100
43/43 [==============================] - 5s 114ms/step - loss: 5.6641 - accuracy: 0.0576
Epoch 5/100
43/43 [==============================] - 4s 102ms/step - loss: 5.5828 - accuracy: 0.0576
Epoch 6/100
43/43 [==============================] - 6s 135ms/step - loss: 5.4825 - accuracy: 0.0605
Epoch 7/100
43/43 [==============================] - 6s 142ms/step - loss: 5.3580 - accuracy: 0.0708
Epoch 8/100
43/43 [==============================] - 6s 142ms/step - loss: 5.1795 - accuracy: 0.0875
Epoch 9/100
43/43 [==============================] - 6s 140ms/step - loss: 4.9604 - accuracy: 0.1058
Epoch 10/100
43/43 [==============================] - 6s 139ms/step - loss: 4.6972 - accuracy: 0.1225
```



```
Epoch 11/100
43/43 [==============================] - 6s 141ms/step - loss: 4.4387 - accuracy: 0.1568
Epoch 12/100
43/43 [==============================] - 6s 143ms/step - loss: 4.1668 - accuracy: 0.1743
Epoch 13/100
...
Epoch 99/100
43/43 [==============================] - 6s 129ms/step - loss: 0.0482 - accuracy: 0.9847
Epoch 100/100
43/43 [==============================] - 6s 131ms/step - loss: 0.0472 - accuracy: 0.9854
```


model accuracy
train
1.0
0.8
0.6
0.4
0.2
accuracy
0
20
40
60
80
100
epoch


model loss
train
6
5
4
3
2
1
0
loss
0
20
40
60
80
100
epoch

Chapter 6 – Conclusion and Future work

The Next Word Prediction Model using deep learning demonstrates promising results in predicting subsequent words in a sequence. With advancements in deep learning architectures and computational power, there's potential for further improvement. Such models pave the way for more interactive and intuitive applications in NLP.

The training results suggest that the "Next Word Prediction using LSTM" project successfully trained the text generation model. The decreasing loss and increasing accuracy demonstrate the model's learning capability, as it refines its predictions with each epoch. By the end of 100 epochs, the model achieves a low loss and high accuracy, indicating that it can effectively predict the next word in a given text sequence. However, to fully assess the model's performance, it's essential to evaluate it on an independent dataset or validate its generated text for coherence and relevance.

In conclusion, the project has accomplished its objective of training a text generation model using LSTM and has demonstrated its learning and predictive capabilities. Further steps might include fine-tuning the model and conducting more extensive evaluations to ensure it generates contextually relevant and coherent text.

References

1) Hamarashid, H. K., Saeed, S. A., & Rashid, T. A. (2021). Next word prediction based on the N-gram model for Kurdish Sorani and Kurmanji. Neural Computing and Applications, 33(9), 4547-4566.

2) Stremmel, J., & Singh, A. (2021, April). Pretraining federated text models for next word prediction. In Future of Information and Communication Conference (pp. 477-488). Springer, Cham.

3) Hamarashid, H. K., Saeed, S. A., & Rashid, T. A. (2022). A comprehensive review and evaluation on text predictive and entertainment systems. Soft Computing, 1-22.

4) Barman, P. P., & Boruah, A. (2020). A RNN based Approach for next word prediction in Assamese Phonetic Transcription. Procedia computer science, 143, 117-123.

5) Rakib, O. F., Akter, S., Khan, M. A., Das, A. K., & Habibullah, K. M. (2019, December). Bangla word prediction and sentence completion using GRU: an extended version of RNN on N-gram language model. In 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI) (pp. 1-6). IEEE.

6) Ambulgekar, S., Malewadikar, S., Garande, R., & Joshi, B. (2021). Next Words Prediction Using Recurrent NeuralNetworks. In ITM Web of Conferences (Vol. 40, p. 03034). EDP Sciences.

7) Naulla, N. T. K., & Fernando, T. G. I. (2022, February). Predicting the Next Word of a Sinhala Word Series Using Recurrent Neural Networks. In 2022 2nd International Conference on Advanced Research in Computing (ICARC) (pp. 13-18). IEEE.

8) Shakhovska, K., Dumyn, I., Kryvinska, N., & Kagita, M. K. (2021). An Approach for a Next-Word Prediction for Ukrainian Language. Wireless Communications and Mobile Computing, 2021.

9) Stremmel, J., & Singh, A. (2021, April). Pretraining federated text models for next word prediction. In Future of Information and Communication Conference (pp. 477-488). Springer, Cham.

10) Yazdani, A., Safdari, R., Golkar, A., & R Niakan Kalhori, S. (2019). Words prediction based on N-gram model for free-text entry in electronic health records. Health information science and systems, 7(1), 1-7.