Next Word Prediction using LSTM RNN

A PROJECT REPORT

Submitted by

Jeevan EG

in partial fulfillment for the award of the degree

of

BTech (Hons)

in

BRANCH OF STUDY

School of Computer Science and Engineering
RV University
RV Vidyaniketan,8th Mile, Mysuru Road, Bengaluru, Karnataka, India - 562112

July-August 2024

DECLARATION

I, **Jeevan EG (1RVU22CSE069),** student fifth semester B.Tech in **Computer Science & Engineering,** at School of Computer Science and Engineering, **RV University,** hereby declare that the project work titled “Next Word Prediction using LSTM RNN” has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Computer Science & Engineering** during the academic year **2023-2024**. Further, the matter presented in the project has not been submitted previously by anybody for the award of any degree or any diploma to any other University, to the best of our knowledge and faith.

Name: Jeevan EG
USN: 1RVU22CSE69

Signature

Place: Bangalore

Date: 20/12/2024

School of Computer Science and Engineering

RV University
RV Vidyaniketan,8th Mile, Mysuru Road, Bengaluru, Karnataka, India - 562112

CERTIFICATE

This is to certify that the project work titled “Next Word Prediction using LSTM RNN '' is performed by Jeevan EG **(1RVU22CSE069) ,** a bonafide students of Bachelor of Technology at the School of Computer Science and Engineering, RV university, Bangaluru in partial fulfillment for the award of degree Bachelor of Technology in Computer Science & Engineering , during the Academic year **2020-2021**.

Prof. Santhosh S Nair	**Dr. Sudhakar KN**	**Dr. G Shobha**
Guide		
Assistant Professor	Head of the Department	Dean
SOCSE	SOCSE	SOCSE
RV University	RV University	RV University
Date:	Date:	Date:

Name of the Examiner	Signature of Examiner
1.	
2.	

TABLE OF CONTENTS

ABSTRACT

This project focuses on implementing a text generation system using Long Short-Term Memory (LSTM), a specialized Recurrent Neural Network (RNN) architecture, aimed at predicting the next word in a sequence by leveraging the power of sequential data modelling. The dataset was sourced from Kaggle's Fake News Detection corpus, containing diverse and challenging text data to test the model's capabilities. The study involved extensive data preprocessing steps such as text cleaning, tokenization, and embedding generation to convert textual input into a format suitable for neural network training.

The model architecture includes an embedding layer, two LSTM layers with 150 units each, and a dense output layer with softmax activation for word prediction. The model was trained over 100 epochs using the Adam optimizer and categorical cross-entropy loss function. Performance was evaluated using perplexity and accuracy metrics, demonstrating the efficacy of LSTMs in capturing sequential patterns in text. Key results showed consistent improvement in prediction accuracy over epochs, with the model generating coherent and contextually relevant text sequences. Challenges such as handling rare words and ambiguous contexts were noted.

This work highlights the potential of LSTM models for next-word prediction tasks and sets the groundwork for future improvements, such as incorporating attention mechanisms or expanding to larger datasets, to further enhance performance and scalability.

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol	**Explanation**
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
NLP	Natural Language Processing
GPU	Graphics Processing Unit
Epoch	One complete iteration through the dataset during training
Loss	Measure of model prediction error

1. INTRODUCTION

1.1 Background

Natural Language Processing (NLP) has revolutionized how machines interact with and understand human language, enabling them to perform tasks that were once thought to be exclusively within the realm of human intelligence. A significant advancement within NLP is next-word prediction, a technology that empowers applications such as predictive text on smartphones, conversational AI in chatbots, and the sophisticated language models that power many of the tools we use today. This project leverages the power of Long Short-Term Memory (LSTM) networks, a specialized type of Recurrent Neural Network (RNN) renowned for its ability to effectively model and capture long-range dependencies within sequential data. By harnessing the strengths of LSTM, this project aims to develop and evaluate a robust next-word prediction model. The project encompasses critical stages, including meticulous data preprocessing, the development and training of the LSTM model, and rigorous performance evaluation using metrics such as accuracy and perplexity.

2.2 Problem Statement

Next-word prediction is complex due to the diverse syntactic structures and contextual dependencies in human language. Ambiguities like polysemy and the rarity of certain words make it a challenging task. This project aims to address these challenges by developing an LSTM-based model that can generate accurate and contextually relevant predictions. The model will leverage the capabilities of LSTM networks to better understand and anticipate language patterns, improving its ability to predict the next word in a sequence. The goal is to enhance the model’s performance, making it more effective at capturing and responding to the nuances of different language contexts.

2.Related work

2.1 Literature Review

The advent of deep learning has revolutionized Natural Language Processing (NLP), with Long Short-Term Memory (LSTM) networks emerging as a preferred choice for sequence modeling tasks. In 1997, Hochreiter and Schmidhuber introduced the LSTM architecture, which addressed the limitations of traditional Recurrent Neural Networks (RNNs) by incorporating mechanisms to manage long-term dependencies in sequences. LSTMs are specifically designed to remember information over time, making them well-suited for tasks such as language modeling, where the relationships between words in a sentence are crucial for accurate predictions. Subsequent research has demonstrated the effectiveness of LSTMs in a variety of NLP applications, including text generation, translation, and sentiment analysis. By incorporating gating mechanisms—like the input, forget, and output gates—LSTMs can selectively store and forget information, allowing them to capture complex, long-range dependencies in text. This ability is a key reason why LSTMs have become a standard tool for models aiming to generate coherent and contextually appropriate language.

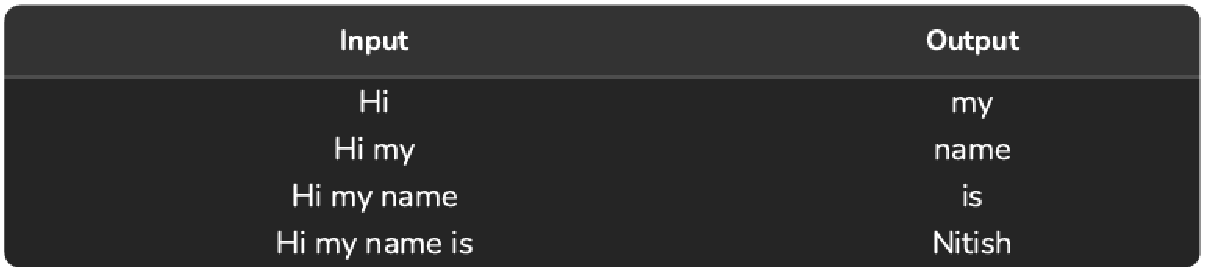
2.2 Comparative Analysis

Traditional approaches such as n-gram models and Markov Chains have been historically used for text generation tasks. These models rely on fixed-length context windows to predict the next word based on the immediate preceding words, which often limits their ability to capture complex linguistic patterns. N-gram models, for example, are constrained by the length of the context they can consider, typically only a few words. Markov Chains, which model text based on state transitions, also struggle with long-term dependencies and the fluid nature of language. In contrast, LSTMs offer a significant advantage due to their ability to maintain and process contextual information over much longer sequences. The architecture of LSTMs allows them to learn patterns from entire passages, taking into account dependencies across a much wider context window. This capability makes LSTMs more adept at producing coherent, contextually meaningful predictions that align with the natural flow of language. Their design enables them to handle the nuances of idiomatic expressions, polysemy, and rare words, making them a preferred choice for tasks that require deep contextual understanding.

3.METHODOLOGY

3.1 Data Preprocessing

- **Dataset**: The text corpus was derived from the Kaggle Fake News Detection Dataset, containing thousands of news articles.
  - The next word predictor works by converting the text generation task into a supervised learning task. The main challenge is to create a dataset from the text where there is a mapping between input and output.

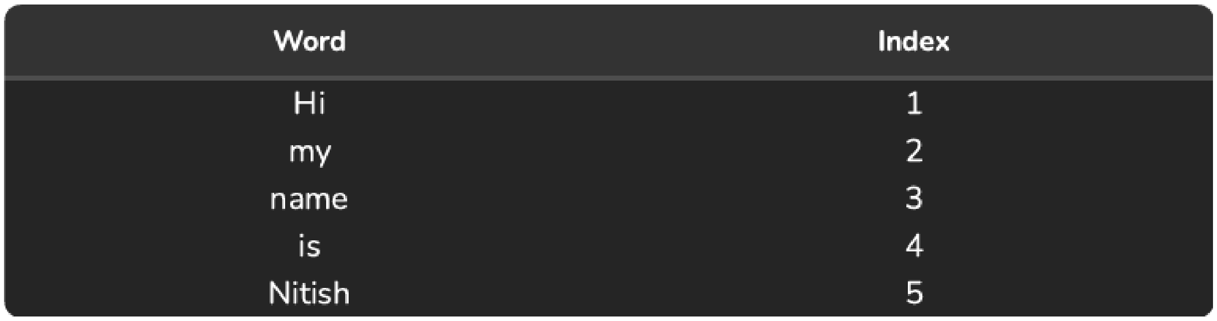


Input	Output
Hi	my
Hi my	name
Hi my name	is
Hi my name is	Nitish

Figure 1: Conversion of data for supervised learning task

- **Preprocessing Steps**: The dataset underwent several preprocessing stages, including:
  - Converting text to lowercase.
  - Removing special characters and punctuation.
  - Tokenization to break text into individual words.
  - Generating word embeddings to numerically represent words for model input.

  - The Tokenizer class is used to assign an index or number to each word in the dataset. It has several methods, including fit_on_texts, which is used to pass the text into the tokenizer.



Word	Index
Hi	1
my	2
name	3
is	4
Nitish	5

Figure 2: Tokenized words

3.2 Model Design

- **LSTM RNN Architecture:**

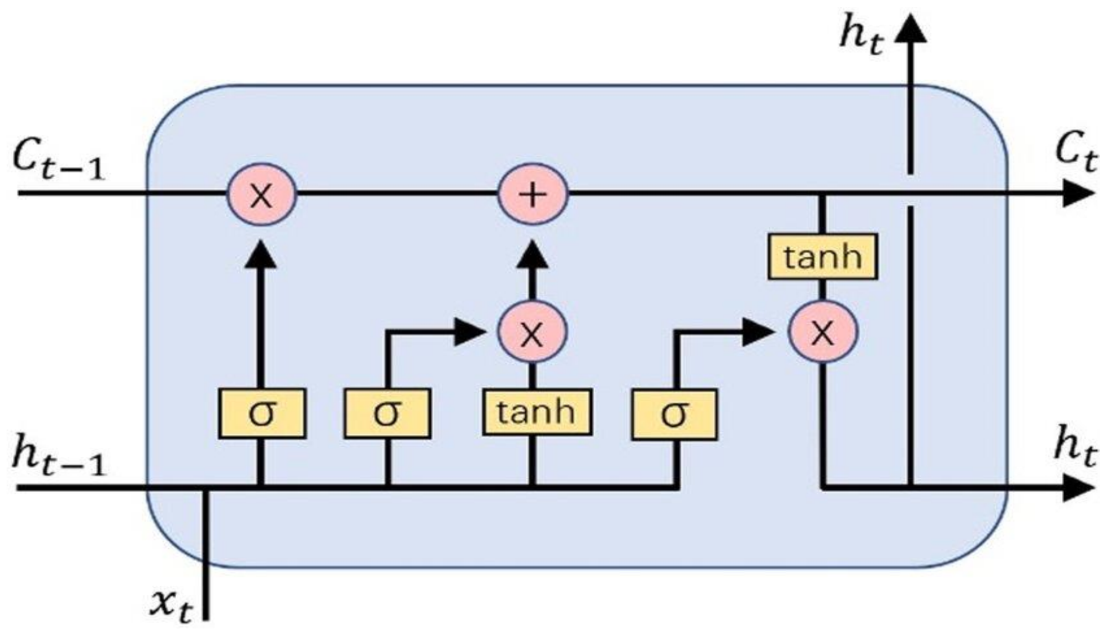


Figure 3: LSTM Architecture

- **Model Architecture**:
  - Embedding Layer: Maps each word to a dense vector of fixed size.
  - Two LSTM Layers: Each with 150 units, designed to capture temporal patterns and long-term dependencies.
  - Dense Layer: Applies softmax activation to predict probabilities for each word in the vocabulary.

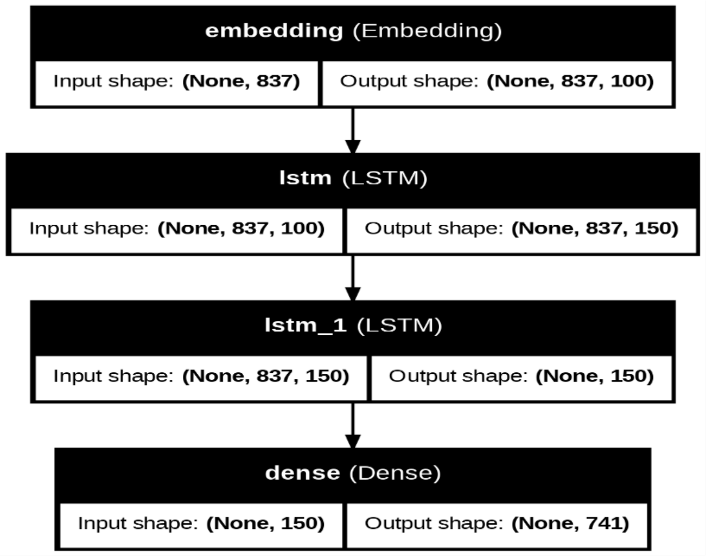

embedding (Embedding)
Input shape: **(None, 837)**
Output shape: **(None, 837, 100)**
lstm (LSTM)
Input shape: **(None, 837, 100)**
Output shape: **(None, 837, 150)**
lstm_1 (LSTM)
Input shape: **(None, 837, 150)**
Output shape: **(None, 150)**
dense (Dense)
Input shape: **(None, 150)**
Output shape: **(None, 741)**

Figure 4: Model Architecture

4. IMPLEMENTATION

4.1 Tools and Frameworks

- **Programming Language**: Python is chosen for its extensive libraries and support for machine learning, especially with deep learning frameworks like TensorFlow and Keras.
- **Libraries**:
  - **TensorFlow/Keras**: Utilized for developing and training the LSTM model due to its ease of use and flexibility in managing neural network architectures.
  - **NumPy**: Essential for numerical operations, enabling efficient manipulation and processing of large datasets.
  - **Matplotlib**: Employed for visualization purposes, allowing the monitoring of model training progress and visualization of results.

4.2 Training and Evaluation

- **Tokenization and Padding**:
  - **Tokenization**: The input text was tokenized into individual words or subwords to convert text data into a format suitable for LSTM processing. Tokenization helps break down sentences into numerical indices that the model can understand.
  - **Padding**: To handle variable-length sequences, each sequence was padded to ensure uniform input length. This is crucial for maintaining a consistent input shape required by the LSTM model.

Model Training:

- **Architecture**: The LSTM model architecture includes multiple LSTM layers with hidden states, allowing it to capture long-range dependencies in the data. Each LSTM layer is followed by a dense layer for output.
- **Training Process**:
  - **Epochs**: The model was trained over 100 epochs to allow sufficient learning time for the model to converge. Each epoch represents one complete pass through the entire dataset.
  - **Optimizer**: The Adam optimizer was chosen for its ability to efficiently update model parameters by adapting the learning rate during training, leading to faster convergence and improved generalization.

  - **Loss Function**: Categorical cross-entropy was used as the loss function. This is suitable for multi-class classification tasks and measures the error between predicted and actual output probabilities.

Performance Metrics:

- **Accuracy**: Monitored during training to evaluate how well the model is learning. It measures the proportion of correct predictions made by the model compared to the total number of predictions.
- **Perplexity**: Calculated to assess the model's ability to predict the next word. Lower perplexity indicates better model performance, as it suggests more accurate predictions.

5. RESULT AND DISCUSSION

5.1 Evaluation Metrics

- **Loss**: The training loss consistently decreased over epochs, indicating effective learning.

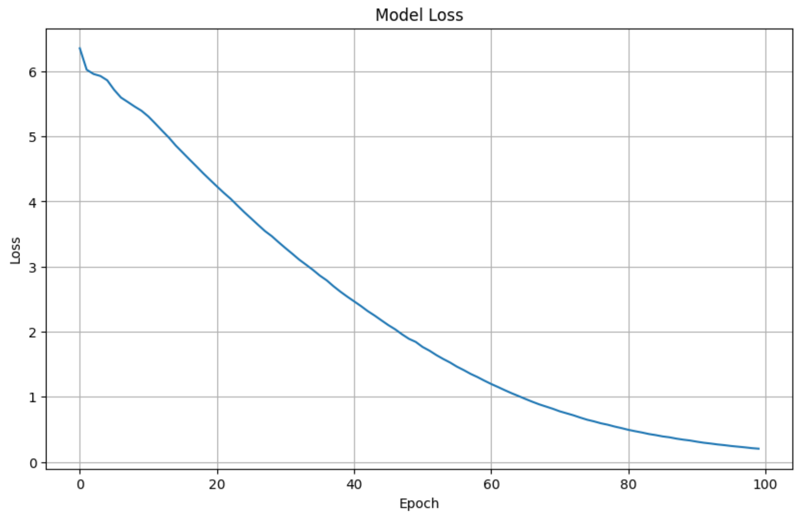


Figure 5: Training Data Loss

- **Accuracy**: The model achieved a peak accuracy of approximately 99% by the final epoch, demonstrating its capability to generalize patterns in the data.

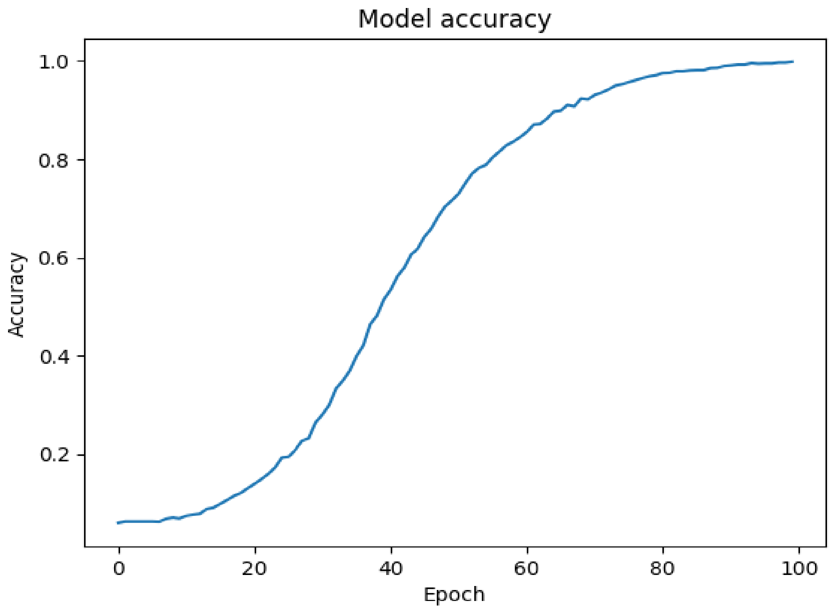
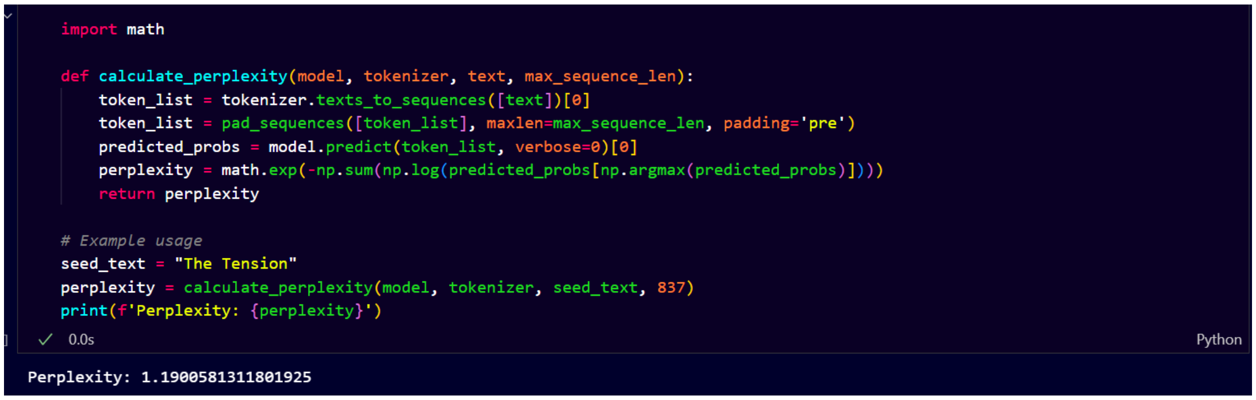


Figure 6: Training Data Accuracy

- **Perplexity**: The calculated perplexity score highlighted the model's ability to generate coherent text sequences with low uncertainty.



```python
import math

def calculate_perplexity(model, tokenizer, text, max_sequence_len):
    token_list = tokenizer.texts_to_sequences([text])[0]
    token_list = pad_sequences([token_list], maxlen=max_sequence_len, padding='pre')
    predicted_probs = model.predict(token_list, verbose=0)[0]
    perplexity = math.exp(-np.sum(np.log(predicted_probs[np.argmax(predicted_probs)])))
    return perplexity

# Example usage
seed_text = "The Tension"
perplexity = calculate_perplexity(model, tokenizer, seed_text, 837)
print(f'Perplexity: {perplexity}')
```

```
Perplexity: 1.1900581311801925
```

Figure 7: Perplexity Score of sample Input

5.2 Observations and Limitations

- **Strengths**: The LSTM model effectively captured contextual information and generated relevant text predictions.
- **Limitations**:
  - The absence of a validation split limited the ability to monitor overfitting.
  - Rare words and ambiguous contexts posed challenges for accurate prediction.
  - Computation time was significant due to the complexity of the model and dataset size.

6. CONCLUSION

This project has demonstrated the potential of LSTM Recurrent Neural Networks (RNNs) in next-word prediction tasks. The model effectively learned sequential patterns within text data, achieving high accuracy and generating contextually meaningful predictions. These results highlight the viability of LSTMs for language modeling tasks, showcasing their ability to capture long-term dependencies and handle complex linguistic structures. However, there remains room for improvement, particularly in dealing with rare words and enhancing computational efficiency. Future work could focus on exploring alternative architectures, such as transformer models, which may offer better performance for next-word prediction. Additionally, improving the model's ability to handle infrequent words through more sophisticated techniques, like domain-specific embeddings or rare word detection mechanisms, could further enhance its accuracy. Overall, this project contributes valuable insights into the capabilities of LSTM models and sets the stage for future advancements in language modeling and text generation tasks.

7.FUTURE SCOPE

- **Attention Mechanisms**: Incorporating attention layers to enhance the model's focus on relevant parts of the sequence.
- **Dataset Expansion**: Training on larger and more diverse text corpora to improve generalization.
- **Real-Time Deployment**: Integrating the model into applications like predictive keyboards or chatbots for practical use.

REFERENCES

1. YOUTUBE: https://www.youtube.com/playlist?list=PLKnIA16_RmvZo7fp5kkIth6nRTeQQsjfX

2. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780. This seminal paper introduced LSTM networks and demonstrated their ability to handle long-term dependencies better than traditional RNNs.

APPENDIX

GitHub Link: https://github.com/JeevanEG/Summer-Internship