ISSN (Online): 3139-7689

Available online: www.ijsrmes.com

International Journal for Scientific Research in Modern Engineering and Science

Published in Conference Special Issue: Vol. 1, Special Issue 1, pp. 177–182, 2026

COMPARATIVE ANALYSIS OF VANILLA RNN AND LSTM FOR NEXT-WORD PREDICTION IN NATURAL LANGUAGE PROCESSING

Akshat Dixit[1*], Shankar Sharan Tripathi [2]

[1]*Student, Computer Science and Engineering, Shri Shankaracharya Technical Campus (SSTC), Bhilai*

[2]*Assistant Professor, Computer Science and Engineering, Shri Shankaracharya Technical Campus*

Abstract

Natural Language Processing (NLP) is growing fast, and predicting the next word is a core part of it. To do this well, neural networks need to remember the context of a sentence. In this paper, we compare two popular models: Vanilla Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks. We trained both models on a custom dataset of Educational FAQs to test how well they predict the next word in a sequence. To keep the comparison fair, both models used the exact same embedding and output layers. Our practical results showed a massive difference in how they learn. The Vanilla RNN memorized the training data well, getting a 95.06% accuracy, but it completely failed on the validation set with just 26.59% accuracy. This happened because of the vanishing gradient problem, which causes standard RNNs to forget earlier words. On the other hand, the LSTM model solved this issue using its built-in memory gates. It achieved a much higher validation accuracy of 90.75% and had a much lower error rate. This experiment clearly proves that LSTMs are structurally better than standard RNNs for handling text and building language models.

Keywords: *Natural Language Processing; Next-Word Prediction; Recurrent Neural Network; Long Short-Term Memory; Vanishing Gradient Problem; Text Generation*

1. Introduction

1.1 Background

Natural Language Processing (NLP) has become one of the most influential domains of Artificial Intelligence. Applications such as predictive typing, intelligent search suggestions, chatbots, and virtual assistants rely heavily on language modeling techniques. Among these tasks, next-word prediction plays a central role. Although predicting the next word may appear straightforward, it requires understanding sentence structure and maintaining contextual continuity. For example:

"Artificial intelligence is changing the future of _____"

To make an accurate prediction, a model must retain information from earlier words in the sequence. Traditional feedforward neural networks cannot effectively process sequential data because they treat each input independently. To address this limitation, Recurrent Neural Networks (RNNs) were introduced.

1.2 Problem Statement

Vanilla RNNs maintain a hidden state that propagates information across time steps. However, during training on long sequences, they suffer from the **vanishing gradient problem**, where gradients shrink during backpropagation. As a result, the model struggles to learn long-term dependencies and often fails to generalize.
This limitation makes Vanilla RNNs unreliable for practical language modeling tasks involving longer context.

1.3 Objective of the Study

The primary objective of this study is to conduct a structured and fair comparison between Vanilla RNN and LSTM architectures for next-word prediction using the same dataset and training setup.
The goal is to analyze whether LSTM's gated memory mechanism significantly improves generalization performance.

2. Related Work

Early language modeling approaches relied on statistical methods such as n-gram models. These methods estimated word probabilities using a fixed window of previous words, which limited their ability to capture long-range dependencies.
With the advancement of deep learning, RNNs became popular for sequence modeling tasks including speech recognition and text generation. However, researchers observed that training RNNs on long sequences led to unstable gradients and poor long-term memory retention.
LSTM networks were introduced to overcome this issue. Their gated architecture enables selective retention and forgetting of information. As a result, LSTMs became widely adopted in NLP tasks before the emergence of Transformer-based models.
.

3. Dataset And Preprocessing

3.1 Dataset Description

A custom dataset of Educational FAQs was used for this experiment. The dataset contains structured academic question-answer pairs, providing context-rich text for next-word prediction.
After preprocessing:

1) Vocabulary size: 283 unique words
2) Total training sequences: 863
3) Uniform padded sequence length

3.2 Preprocessing Steps:

The following preprocessing operations were performed:

1) Conversion to lowercase
2) Removal of punctuation

3) Tokenization into numerical indices

4) Sequence generation for next-word mapping

5) Padding sequences to fixed length

6) One-hot encoding of target words

Example:

"Deep learning models are powerful"

Generated pairs:

Deep → learning

Deep learning → models

Deep learning models → are

Deep learning models are → powerful

This incremental learning approach allows the model to understand how context evolves in a sentence.

4. MODEL ARCHITECTURE

To ensure fairness, both models used identical embedding layers, output layers, optimizer (Adam), and loss function (Categorical Cross-Entropy). Only the recurrent layer differed.

4.1 Vanilla RNN Model

Architecture

Embedding Layer (283, 100)

SimpleRNN (150 units)

Dense (Softmax activation)

Mathematical Formulation

Hidden state update:

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b)$$

Output layer:

$$yt = softmax(Wyht + by)$$

Loss function:

$L = -i = 1\sum N\ yi\ \ log(y\ hati\ )$

Repeated multiplication of gradients during backpropagation leads to gradient shrinkage, causing difficulty in learning long-term dependencies.

4.2 LSTM Model

Architecture

Embedding Layer

LSTM (150 units)

Dense (Softmax)

Mathematical Formulation

Forget Gate:

$$ft = \sigma(Wf\ [ht - 1, xt\ ] + bf)$$

Input Gate:

$$it = \sigma(Wi\ [ht - 1, xt\ ] + bi)$$

Candidate Memory:

$$C\ t = tanh(Wc\ [ht - 1, xt\ ] + bc)$$

Cell State Update:

$$Ct = ftCt - 1 + itC\ t$$

Output Gate:

$$ot = \sigma(Wo\ [ht - 1, xt\ ] + bo)$$

Hidden State:

$$ht = ot\ tanh(Ct)$$

The gated memory mechanism enables controlled information flow across time steps.

5. Results and Comparative Analysis

5.1 Training Performance

Vanilla RNN: 95.06%

LSTM: ~95%

Both models achieved similar training accuracy.

5.2 Validation Performance

Vanilla RNN: 26.59%

LSTM: 90.75%

The validation results reveal a significant performance gap.

Table 1: Performance Comparison

Model	Training Accuracy	Validation Accuracy	Generalization
Vanilla RNN	95.06%	26.59%	Poor
LSTM	~95%	90.75%	Strong

5.3 Validation Curve Analysis

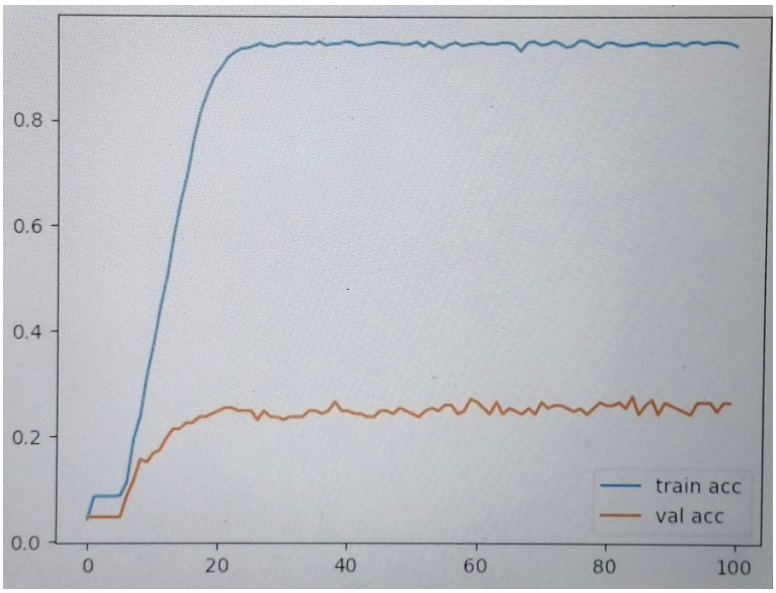

Figure 1. Training accuracy vs validation accuracy graph for RNN

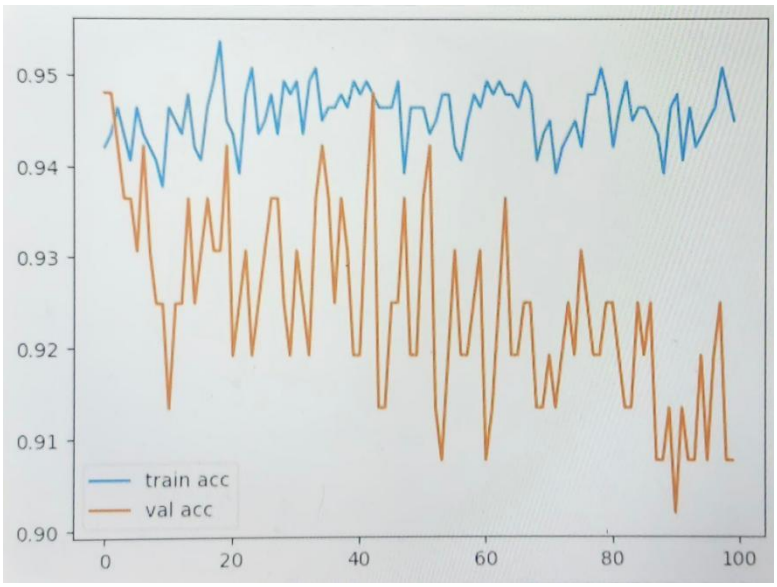

Figure 2. Training accuracy vs validation accuracy graph for LSTM

The Vanilla RNN exhibited a large gap between training and validation accuracy, indicating severe overfitting. In contrast, the LSTM model showed closely aligned training and validation curves, reflecting stable learning behavior.

6. Discussion and Conclusion

The experimental results clearly demonstrate that the limitations of Vanilla RNN are practically observable. Although the Vanilla RNN achieved high training accuracy, it failed to generalize on unseen data due to the vanishing gradient problem, which restricts long-term contextual learning.

In contrast, the LSTM model addressed this issue through gated memory mechanisms, allowing it to preserve relevant information and significantly improve validation accuracy and training stability.

The comparison confirms that while Vanilla RNN may memorize patterns, it struggles with contextual continuity. LSTM demonstrates superior generalization in sequential text modeling tasks.

Overall, this study validates that LSTM architectures are more suitable for next-word prediction and language modeling applications. Future work may explore GRU models, larger datasets, and comparisons with Transformer-based architectures.

References

[1] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[5] T. Mikolov et al., "Recurrent neural network-based language model," *INTERSPEECH*, 2010.

D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd Edition (Draft), 2019.