IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, April 2026
Impact Factor: 8.2

Typen: A Real-Time LSTM-Based System for Next-Word Prediction

Dr. T. Seshu Chakravarthy[1], M. Pallavi[2], P. Mohan Siva Arun Kumar[3], M. Venkatesh[4], P. Renu Nivas Sarma[5]
Associate Professor, Department of Computer Science and Engineering[1]
UG Students, Department of Computer Science and Engineering[2,3]
Vasireddy Venkatadri Institute of Technology, Nambur, Guntur Dt., Andhra Pradesh.
tschakravarthy@vvit.net

Abstract: *This paper presents **Typen**, a web-based intelligent writing assistant designed to enhance typing efficiency through real-time next-word prediction. The system leverages a Long Short-Term Memory (LSTM) network to model sequential dependencies in text and generate context-aware suggestions. A large corpus sourced from public domain literature is preprocessed through text cleaning, tokenization, and sequence generation to train the predictive model. The architecture consists of an embedding layer followed by stacked LSTM layers and dropout regularization to improve generalization. The trained model is deployed using a Flask-based backend API, while a React-based frontend interface enables user interaction with dynamic suggestions. The system processes user input asynchronously and returns the most probable next words with minimal latency. Experimental evaluation demonstrates a Top-1 accuracy of approximately 21% and a perplexity score near 100, indicating the complexity of open-domain language prediction while validating the effectiveness of the proposed approach. The results show that the model captures meaningful linguistic patterns and provides useful assistance during text composition. Overall, Typen offers a practical integration of deep learning and web technologies, providing an efficient tool to reduce cognitive effort and improve writing flow.*

Keywords: Natural Language Processing, LSTM, Deep Learning, Next-Word Prediction, Text Generation, Writing Assistant, Sequence Modeling

I. INTRODUCTION

Continuous writing often becomes challenging when users struggle to decide the next appropriate word, leading to interruptions in thought flow. While basic predictive typing is available in mobile keyboards, advanced context-aware assistance in desktop environments remains limited.

This work introduces **Typen**, a machine learning-based system designed to assist users by predicting the next word in real time. The core of the system is a Long Short-Term Memory (LSTM) network, which is well-suited for capturing sequential dependencies in text data.

The system integrates a deep learning model with a web-based interface, enabling users to receive suggestions dynamically as they type. The frontend provides a structured writing environment, while the backend processes input and generates predictions efficiently. This paper outlines the complete pipeline, including data preprocessing, model design, training, and deployment.

II. LITERATURE SURVEY

Melis et al. [1] showed that LSTM models can perform as well as transformer models on smaller datasets when properly tuned with regularization techniques. Their work taught us that older architectures like LSTM should not be

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, April 2026
Impact Factor: 8.2

ignored simply because new models exist, and that careful optimization often matters more than choosing the latest architecture. This finding is important for applications where computational resources are limited.

Nandakumar et al. [2] explained why neural language models work well for next-word prediction from a mathematical viewpoint. They showed that these models succeed because they capture predictable patterns in language, such as words that commonly appear together and grammatical rules. This theoretical understanding helps researchers design better prediction systems by knowing what makes models effective.

Nugaliyadde et al. [3] improved LSTM models by adding extra memory mechanisms to better handle long text sequences. Their approach reduced prediction uncertainty by 5 to 10 percent compared to standard LSTM models. This taught us that giving models explicit storage for important information helps them maintain context over longer passages, though it requires more computational resources.

Lan et al. [4] discovered that training LSTM models with additional tasks alongside next-word prediction helps them learn better word representations. The extra objectives force the model to extract more meaningful features from text, leading to faster training and more accurate predictions. This approach improves model performance without changing the underlying architecture.

Ozaki et al. [5] investigated whether LSTM models understand complex grammar, specifically looking at how they handle sentences where words are separated from their grammatical positions. They found that LSTM models learn sophisticated grammatical rules simply by reading lots of text, without being explicitly taught grammar. This research confirmed that these models develop human-like understanding of language structure through exposure alone.

III. PROPOSED SYSTEM

The development of the Typen writing assistant requires a highly structured pipeline, transitioning from raw text ingestion to real-time predictive deployment.

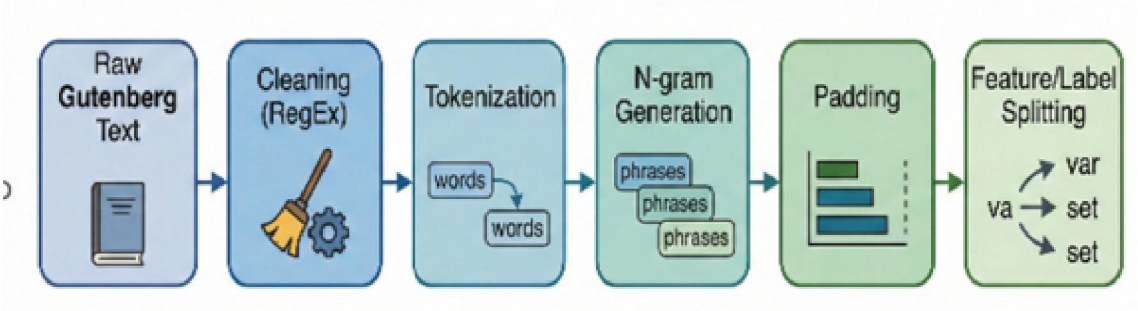

Fig - 1 Data Preprocessing pipeline for transforming raw corpus into training tensors

A. Dataset Acquisition and Preprocessing

The predictive capability of any language model is entirely dependent on the quality and scale of its training corpus. For this research, the primary dataset was sourced from Project Gutenberg, a digital library of public domain books.

The raw textual data contained significant noise, requiring rigorous cleaning and preprocessing. The preprocessing pipeline consisted of the following sequential steps:

Text Sanitization: All metadata, copyright headers, and non-narrative text were systematically stripped. Special characters, punctuation, and numerical digits were removed using regular expressions to ensure the model focuses exclusively on alphabetical linguistic patterns. The entire corpus was converted to lowercase to maintain uniformity.

Tokenization: The continuous sanitized text was divided into discrete computational units (tokens). A specialized Tokenizer was fitted on the corpus with a restricted vocabulary size of 10,000 words to optimize memory consumption and model complexity. Out-of-vocabulary (OOV) words were mapped to a generic <OOV> token.

Sequence Generation: To train the model to predict the next word, the text was structured into input-output pairs. Continuous token sequences were broken down into overlapping N-grams.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, April 2026
Impact Factor: 8.2

Padding and Truncation: Because neural networks require fixed-dimension tensor inputs, the generated sequences were pre-padded to a uniform maximum sequence length of 30-time steps. Sequences exceeding this length were truncated.

Feature and Label Separation: The final matrix was sliced, designating the first 29 tokens as the input feature vector (X) and the final 30th token as the target label (y). The target labels were kept in sparse categorical format to interface with the objective loss function.

B. STACKED LSTM ARCHITECTURE

The core predictive engine is a deep Sequential neural network built using TensorFlow and Keras. The architecture is specifically optimized for sequence modeling and is constructed as follows:

Embedding Layer: The input layer receives a sequence of integers (representing words) and maps them into a dense, continuous vector space. The embedding dimensionality is set to 100, allowing the network to learn semantic relationships between the 10,000 vocabulary words.

Stacked LSTM Layers: The network employs a stacked configuration of two LSTM layers, each containing 192 hidden units. Stacking LSTMs allows the network to learn hierarchical temporal representations; the first LSTM outputs a full sequence of hidden states to the second LSTM, which distills this into a final contextual representation.

Dropout Regularization: To prevent the model from overfitting to the training corpus and simply memorizing specific book phrases, a Dropout layer with a rate of 0.3 is applied after the recurrent layers. This randomly zeroes out 30% of the network connections during each training step, forcing the model to learn generalized grammatical rules.

Dense Output Layer: The final layer is a fully connected Dense network containing 10,000 neurons (equal to the vocabulary size). It utilizes a SoftMax activation function to convert the raw output logits into a normalized probability distribution, where the sum of all predicted word probabilities equals 1.0.

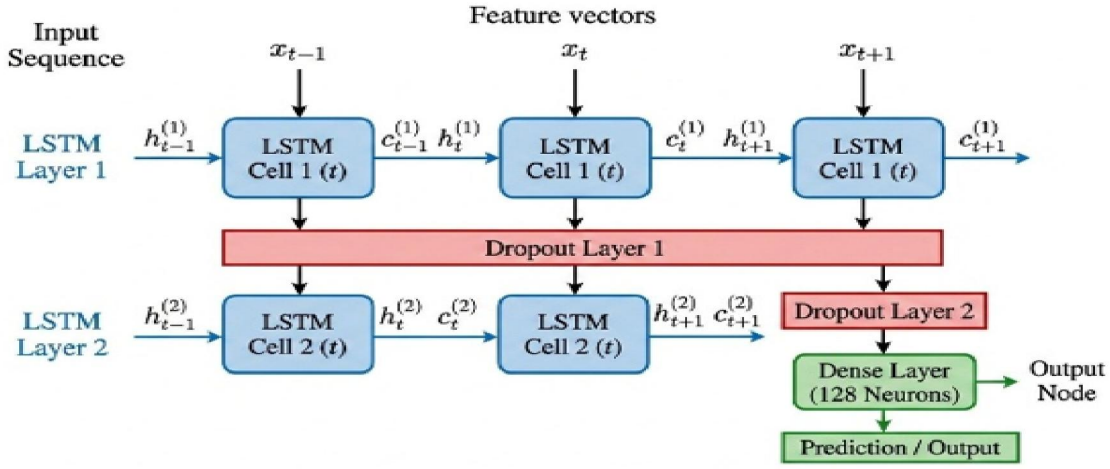

Fig - 2 Stacked LSTM Neural Network Architecture for Sequence Modelling

C. SYSTEM ARCHITECTURE AND INTEGRATION

The project demonstrates the integration of Deep Learning with a scalable, full-stack web application. The system workflow operates on a strict client-server model designed for real-time responsiveness.

Frontend Interface (React.js): The client-side application is developed using React.js to provide a highly responsive, writer-focused interface. It features a Rich Text Editor where the user types, and a Dynamic Prediction Panel positioned as a sidebar.

Backend API (Flask): The trained LSTM model is saved as an HDF5 file and hosted on a Python Flask web server. The server exposes a RESTful API endpoint dedicated to prediction handling.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, April 2026 Impact Factor: 8.2

System Workflow: As the user types in the React editor, the current text string is captured via an onChange event listener and transmitted to the Flask backend via an asynchronous HTTP POST request. The backend immediately preprocesses the input string using the saved Tokenizer, pads it to the required 29-word length, and feeds it into the LSTM model.

Real-Time Inference: The model computes a forward pass and returns the probability distribution. The backend extracts the top 5 predicted words with the highest probabilities and returns them as a JSON response. The React frontend parses this response and dynamically renders the ranked suggestions in the sidebar, which the user can insert into the text with a single click.

IV. RESULTS AND DISCUSSION

A. TRAINING CONFIGURATION AND CONVERGENCE

The model was compiled using the Adam optimizer, configured with a learning rate of 0.0007 and a gradient clipping value of 1.0 to ensure stable weight updates. The objective function utilized was Sparse Categorical Cross-Entropy, appropriate for mutually exclusive multi-class classification over the vocabulary space.

The dataset, comprising 3,000,000 formatted sequences, was split utilizing a 90/10 ratio for training and validation. The network was trained over 12 epochs with a batch size of 128. During the training phase, early stopping was implemented to monitor validation loss with a patience of 2 epochs, ensuring the restoration of optimal weights and preventing catastrophic overfitting. The training behavior indicated a steady decrease in loss, stabilizing at a final training loss of 4.6311 and a validation loss of 4.7660.

B. EVALUATION METRICS

Evaluating open-domain language models strictly by absolute accuracy can be misleading due to the high variance of natural language. A single context sequence can be followed by dozens of grammatically and semantically correct words.

On the unseen testing dataset (10% holdout), the model achieved:

Loss: 4.6

Top-1 Accuracy: 21.42%

To standardly measure the performance of the language model, we calculate its Perplexity. Perplexity measures how well a probability distribution predicts a sample, essentially quantifying the model's "uncertainty." It is mathematically defined as the exponentiation of the categorical cross-entropy loss:

$$\text{Perplexity} = \exp(\text{Loss})$$

$$\text{Perplexity} = e^{4.6} \approx 99.4$$

A perplexity of 99.4 indicates that, on average, the model is as confused as if it had to guess uniformly among 99 possible next words. In the context of standard LSTM architectures trained on limited vocabularies, a Top-1 Accuracy approaching 21% demonstrates that the network has successfully learned strong syntactical patterns and local contextual dependencies, providing highly viable suggestions for general writing assistance.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, April 2026
Impact Factor: 8.2

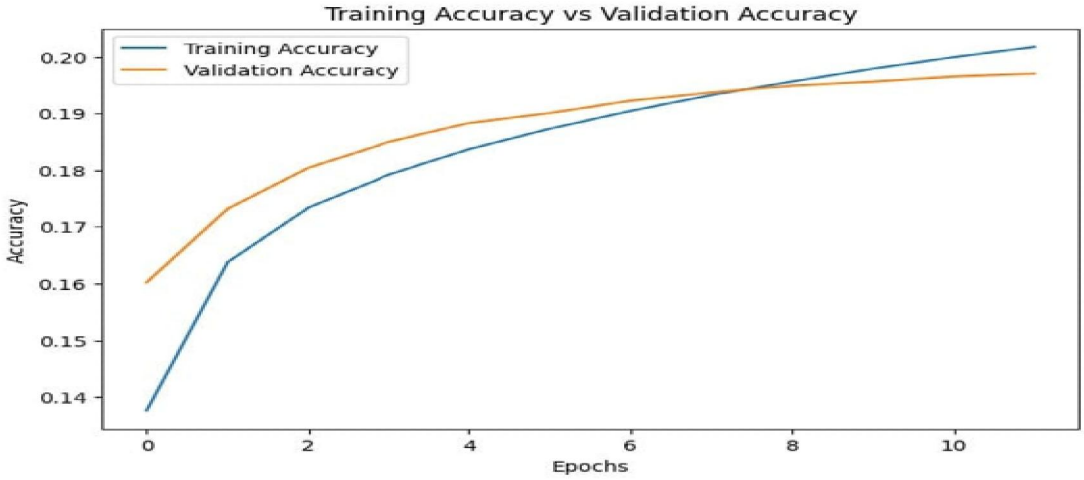

Chart - 1: Training Accuracy Vs Validation Accuracy

The above graph shows a line graph comparing training accuracy and validation accuracy over 10 epochs. The training accuracy (blue line) steadily increases and remains high, while the validation accuracy (orange line) rises initially but then plateaus and slightly declines, indicating potential overfitting.

The graph below illustrates training and validation loss over epochs, with the blue line representing training loss and the orange line representing validation loss. Both curves decrease sharply initially, with validation loss consistently higher than training loss, and they begin to plateau with a slight divergence in later epochs. This pattern suggests the model is learning effectively but may be beginning to overfit as validation loss stops decreasing while training loss continues to drop.

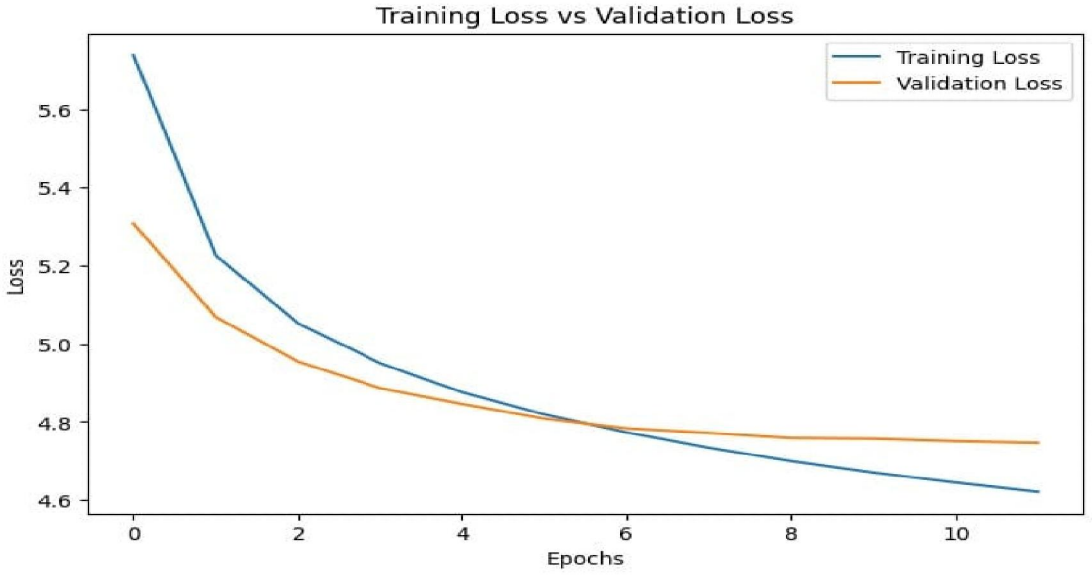

Chart - 2: Training Loss Vs Validation Loss

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, April 2026 Impact Factor: 8.2

C. INTERFACE UTILITY AND DEPLOYMENT DISCUSSION

The true value of the Typen system lies in its architectural deployment. By offloading the heavy tensor computations to a backend REST API, the client-side React application remains lightweight and highly responsive.

The screen design successfully isolates the mechanical act of typing from the cognitive task of word selection. By presenting ranked suggestions and creative vocabulary alternatives in a designated sidebar, the system ensures that automated predictive interventions do not abruptly interrupt the writer's flow unless explicitly utilized. As the user types more words, the predictions accurately mutate in real-time, reflecting a strong integration of NLP preprocessing and real-time API communication.

V. CONCLUSION

This research project successfully designed, trained, and deployed "Typen," a web-based Machine Learning application tailored for intelligent writing assistance. By leveraging a stacked Long Short-Term Memory (LSTM) architecture, the system effectively captures sequential linguistic dependencies to generate accurate next-word probability distributions. The integration of the deep learning model within a decoupled full-stack environment—utilizing a React.js frontend and a Flask backend—demonstrates a strong understanding of scalable ML architectures and real-time API integration. The model's evaluation, yielding a perplexity of 99.4 and a Top-1 accuracy of over 21%, validates its capability to provide ranked, context-aware vocabulary suggestions. Ultimately, this project serves as a comprehensive framework combining Natural Language Processing with interactive software design, delivering a highly functional tool that alleviates cognitive load and enhances typing speed for content creators.

REFERENCES

[1] G. Melis, T. Kočiský, and P. Blunsom, "Circling Back to Recurrent Models of Language," in Proceedings of the International Conference on Learning Representations, Virtual Conference, 2023, pp. 1-15.

[2] V. Nandakumar, P. Mi, and T. Liu (2023). "Why can neural language models solve next-word prediction? A mathematical perspective." arXiv.org. [Online]. Available: https://arxiv.org/abs/2305.11462

[3] A. Nugaliyadde, F. Sohel, K. W. Wong, and H. Xie, "Language modeling through long-term memory network," in Proceedings of the International Joint Conference on Neural Networks, Budapest, Hungary, 2019, pp. 1-6.

[4] Q. Lan, L. Kumar, and M. White, "Predictive Representation Learning for Language Modeling," in Proceedings of the AAAI Conference on Artificial Intelligence, Virtual Conference, 2021, pp. 1-9.

[5] S. Ozaki, D. Yurovsky, and L. Levin, "How well do LSTM language models learn filler-gap dependencies?," in Proceedings of the Annual Meeting of the Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 1-12.

[6] J. L. Elman, "Distributed representations, simple recurrent networks, and grammatical structure," Mach. Learn., vol. 7, no. 2-3, pp. 195–225, 1991.

[7] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," Journal of Machine Learning Research, vol. 3, pp. 1137-1155, 2003.

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

[9] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in Proc. 11th Annual Conference of the International Speech Communication Association (INTERSPEECH), 2010, pp. 1045-1048.

[10] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in Advances in Neural Information Processing Systems (NIPS), 2014, pp. 3104-3112.