ISSN 2007-9737

Predicting the Future of Text: A Hybrid Approach to Next-Word Prediction

Sanjit Kumar Dash[1], Parameswari Khatua[1], Muktikanta Sahu[2]

[1] Odisha University of Technology and Research, Bhubaneswar, Odisha, India

[2] International Institute of Information Technology Bhubaneswar, Odisha, India

skdash@outr.ac.in, pswarikhatua05@gmail.com, muktikanta@iiit-bh.ac.in

Abstract. Text input has become an integral part of modern communication, spanning from everyday conversations to formal content creation. However, manual typing is often slow and prone to errors, which has driven the need for efficient text prediction models to improve user experience and productivity. By anticipating and generating the next likely word in a sequence, next-word prediction systems contribute significantly to faster and more accurate text composition. Early approaches like N-grams established the foundational concepts but were limited in their ability to grasp complex, wide-reaching context. In recent years, this field has been dominated by large-scale Transformer architectures, which have set new benchmarks in language understanding. However, their significant computational demands often create a barrier to deployment in resource-constrained environments such smartphones or embedded systems . This paper addresses this challenge by introducing a hybrid deep learning model that offers a predictive accuracy with computational efficiency. Our proposed architecture merges CNNs with Bi-LSTM networks. CNNs are highly effective at extracting local, spatial features from text, while Bi-LSTMs excel at learning long-range sequential dependencies. By training this model on the classic Sherlock Holmes dataset, we demonstrate its ability to achieve nearly 76% contextual accuracy, proving it is a powerful and viable alternative for real-world applications. This work validates the effectiveness of hybrid models in creating intelligent text generation systems for tools like smart keyboards and assistive writing technologies.

Keywords. Next-word prediction, CNN-LSTM hybrid architecture, natural language processing, text generation, sequential data modeling, one-hot encoding, Sherlock Holmes corpus, language modeling, neural networks.

1 Introduction

Text input has become an integral part of modern communication, spanning from everyday conversations to formal content creation. However, manual typing is slow and prone to errors, particularly in fast-paced or high-volume environments. This limitation has driven the need for efficient text prediction models to improve user experience and productivity. The main means of achieving automatic word completion and a time-saving factor for typing is the use of recent models of next-word prediction [16]. An instance of classic models used for prediction would be N-gram models, which are weak when asked to represent long-term dependencies and complex context within sequences [13]. Hybrid CNN-LSTM releases demonstrate possibilities for enhancing the performance of next word prediction since CNNs are good at local feature extraction on input text sequences, whereas LSTM is good at capturing long-distance dependencies and contextual relationships within the data [18].

The two networks complement each other since they capture local patterns as well as large contextual information of the entire sequence, thus making for a much stronger solution

for text-predicting problems [14][10]. Other architectures can be incorporated as additional enhancements to reinforce model performance, for instance, attention mechanisms enhancing the model's ability to really focus on the important bits of the input sequence [21]. With respect to applications, this particular hybrid CNN-LSTM has been deployed successfully in social media text prediction so that it could deal — efficiently — with noisy and informal data without giving up accuracy [1]. In language generation, these hybrid architectures have been able to capture dependencies between two languages, and thus improve prediction accuracy [19].

Transformer-based models have set new benchmarks in accuracy but are usually computationally intensive and, therefore, are restricted from being implemented in real-time scenarios. The goal of this paper is to develop and evaluate a next-word prediction model based on a CNN-LSTM hybrid architecture, which stands as a balanced and robust alternative. In really simple terms, this model aims to leverage the strengths of CNNs in feature extraction and those of the LSTMs in long-term dependency capturing to provide very accurate predictions but without the overhead of large architectures. For measuring the performance of this technique, we use the Sherlock Holmes data-set, which is a fairly rich literary corpus for training and testing. This paper looks into how hybrid architecture can contribute to modeling language where the major considerations in play are efficiency and contextual awareness, thereby amplifying the user experience for many NLP-based systems.

2 Related Work

Attention-based LSTM has, in the recent past, been accorded a lot of importance in the areas of contextual understanding and prediction accuracy-multilingual text or automated essay scoring. The challenges are still with low-resource language and real-time adaptation. In this research, we propose an attention modification to create an optimum CNN-Bi-LSTM model for better and accurate multilingual text prediction.

Traditional machine-learning systems can only do so much to extract relevant features and reason about situations; hence we require novel methods to increase their efficacy. Recently, many approaches have been proposed to use incremental deep learning models, mainly LSTMs, to improve next-word prediction in various applications. Chilukuri et al. (2023) gave an approach based on machine learning for the prediction of the next words for online applications such as search engines and recommendation systems. The authors demonstrated how LSTM models could be effectively applied in practical text prediction tasks [5]. Similarly, Mahajan et al. (2024) improved the accuracy of intelligent grading systems by introducing attention mechanisms in Bi-LSTM models for inputs bearing incomplete texts [10]. Deep learning with CNN-LSTM predicts student performance, improving course selection, study schedules, and support has been proposed in [3]. LSTMs integrated with attention have been quite successful in capturing contextual dependencies and thereby achieving refined sequence prediction in the language-limited environment of Tulu [2]. Continuing with the study is Kitaw (2024), who incorporated attention mechanisms into Bi-LSTM models aiming to advance next-word prediction for underserved languages Afaan Oromo [8]. The work of Liu et al. (2024) proposed a hybrid CNN-Bi-LSTM model for the detection of epilepsy seizure, suggesting that the combined CNN and LSTM models could perhaps perceive complex sequential patterns, an idea that applies to the next-word prediction problem [9].

Islam et al. (2024) worked on next-word prediction and sentence completion for Bangla using Bi-LSTM model. With the training set created from news portals, the model had a 99% accuracy in word predictions at 4-gram and 5-gram, thereby outperforming the existing methods. Hence, the study bears testimony to the fact that large datasets are the best for improving the performance of language prediction systems [7]. In the same way, Xie et al. (2023) stress that attention mechanisms increase the accuracy of predictions in real-time applications such as mobile typing [21]. Mekonen (2024) introduces

a bidirectional LSTM model to further enhance next-word prediction accuracy in Afaan Oromo by accounting for both past and future contexts. This, therefore, increases the quality of prediction for the underrepresented languages [11].

According to Wang et al. (2023), a CNN-LSTM model is applied to next word prediction on social media platforms, with CNNs used as local feature extractors and LSTMs as sequence learners. This effectively tackles the complexity in social media text, besting relatively simpler models in handling noisy data and long-term dependencies [20].

Tajbakhsh and Fathy (2023) introduced a CNN-LSTM model for the prediction of words and sequence learning, stating that such models are good at varying tasks, from the prediction of individual words to the generation of a longer sequence [17]. Real-time word prediction based on input text for mobile and web applications requiring immediate and highly accurate predictions has been proposed by Ghosh and Ray (2023) [6].

Srinivasan and Shankar (2024) presented and discussed the comparison of CNN-LSTM networks for text sequence prediction, especially for language modeling and text prediction [15]. For the rest, Reddy and Kumar (2023) proposed another way of CNN-LSTM for time-series prediction and sequence prediction problems; this proves that CNN-LSTM architectures are capable of handling a wider range of prediction problems other than text [12]. Zhang and Li (2024) demonstrated different ways to utilize CNN-LSTM networks for continuous text next-word prediction, thus highlighting their ability to work on large-scale NLP tasks [22]. Finally, Chen and Liu (2024) showed a greater scope for CNN-LSTM models for sequence prediction tasks in natural language processing, deepening constituted knowledge on their ability to improve next-word prediction accuracy on continuous texts [4].

The next-word prediction with LSTM, Bi-LSTM, CNN-LSTM, and attention-based models, among other things, may have gone a certain way to tackle the thorny problems related to low-resource languages, optimize hybrid architectures for real-time text operations, and more. However, existing models are both able to perform on static datasets, making them inflexible. Though improvements in context retention have been made by Bi-LSTMs and attention mechanisms, they are not well suited to long-range dependency modelling . An optimized CNN-Bi-LSTM architecture with attention was proposed to increase accuracy and enhance computational efficiency. It provides online data adaptation and hyperparameter tuning to facilitate improved low-resource language processing, thus filling a gap in Bi-LSTM versus transformer paradigm for multilingual and mobile text prediction.

3 Proposed Methodology

A CNN-LSTM-based system was designed for next-word prediction. The whole system considered the corpus from Sherlock Holmes for text cleaning, tokenization, and sequence padding. The idea here was to use CNN as local feature extractors and LSTM as long-short term memory in conjunction with some optimization and regularization techniques: Figure 1 represents working on a next-word prediction system based on the hybrid architecture, i.e., CNN and LSTM, to capture short-term patterns in the text along with long-term dependencies in the sequences with a decent amount of efficacy.

The flow starts with the collection and preparation of data, following which cleaning of the raw text, tokenization, and numericalization through different embedding schemes takes place. Vectorized inputs undergo a first treatment by CNN layers that help in capturing local features at N-gram levels and short-range text patterns. These intermediate outputs are then passed on to LSTM layers that retain the context across very long sequences, enabling the model to exploit deep linguistic structure. Training of the model takes place under several optimization algorithms with randomly injected dropout techniques for performance improvement and overfitting reduction. A well-trained network is capable of predicting the next valid words, given a sequence of previous words. Thus, this basis may be practically applied to areas such as predictive text, smart keyboards, and conversational agents in real time.

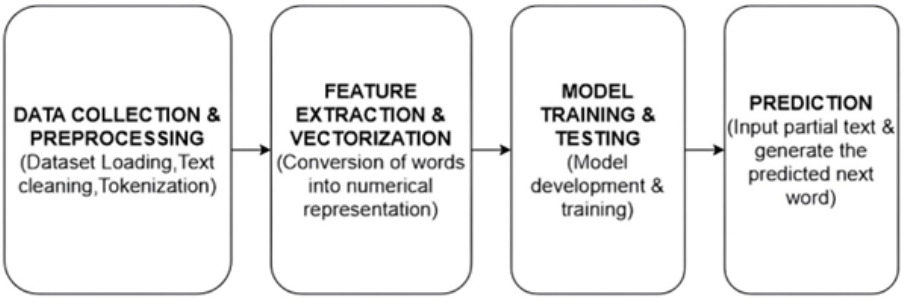

Fig. 1. Proposed framework

3.1 Dataset Description

This data-set comprises the text of The Adventures of Sherlock Holmes by Arthur Conan Doyle [9], the next-best thing for any NLP-related application to be, especially in making use of hybrid models to predict the next word. The Sherlock Holmes Next-Word Prediction Corpus is a data-set comprising the entire text of Sir Arthur Conan Doyle's Sherlock Holmes stories, which are utilised for various NLP tasks, including next-word prediction and text modeling. A study of language patterns will enable users to develop algorithms for next-word prediction and shift next toward the focus of various text generation systems. The data-set provides sentences, dialogues, and descriptions intermixed and thereby patterns of language structure. It carries dialogues from key characters, such as Sherlock Holmes, Dr. Watson, and others, upon which the model can learn the finer nuances of a character's unique manner of speech and dialogue structure. It, therefore, can help in the formulation of an understanding process and characterization of interactions.

3.2 Preprocessing

The first stage of the next-word prediction hybrid approach using CNN and LSTM involved exhaustive preprocessing of data to ensure consistency and reliability of the given text. During the preprocessing stage, it is followed by the elimination of extraneous white space, punctuation, and other characters - the standardization of inputs. This text is further tokenized at the word level by splitting it into one-word units for further studies. Vocabulary encoding then indexes each token with a separate integer value so that words can be conveniently represented by numbers. Fixed-length input sequences are created via a sliding-window mechanism to capture word dependencies, thereby providing sentences such as "The detective solved the case," with resulting sequences-["The", "detective", "solved"], ["detective", "solved", "the"]-that is, the model tries to predict the next possible word in each-one of which is padded and converted into structured numeric arrays for more efficient ingestion by the CNN-LSTM model upon training.

3.3 Feature Extraction & Vectorization

Before a model trained through machine learning can interact with text, first it needs to transform that text into a format that it can understand-numbers. Basically, vectorization is the process of giving each of the words in the data set a corresponding number, two translating it into another form of data that can be interpreted meaningfully by a model. The model realizes no meaning out of mere numbers, thus comes the usage word embedding: it does not treat numbers with no relation as arbitrary points but tries to relate distance and direction between points to meaning: a word embedding will thus place the words in a continuous vector space; hence, words like "king" and "queen" will have similar vector values because they are used in similar contexts.

Andersen can derive meaning out of these words rather than merely noting their positions. Later, either the two were built into the model or came from other pre-trained models. Words converted to vectors now clump together into sequences, each containing some words followed by what next word the model is to be trained to predict. The idea is to help learn how flow, grammar, and meaning interact to figure out the most suitable, context-aware prediction possible. It provides the model with the meaning of a word, not just its word order, but also its relationship with other words, to understand how natural language should operate.

3.4 CNN-LSTM Architecture

The Figure 2 depicted architecture takes advantage of both the CNN-based models and LSTMs in dealing with some sequential tasks such as next-word prediction. To that end, CNNs capture local and spatial features present in the text, allowing the model to identify meaningful patterns and character-level dependencies, while LSTMs best capture language sequentiality over long-term contextual information, thus suitable for sentence flow. Hence, the hybrids learn to identify relevant n-gram structures through CNN layers and provide a deeper context via an LSTM. The powerful complementary duality has successfully addressed the worthy pitfalls confronted when using either of the two models. Long short-term memory alleviates the problem of vanishing gradient inherent in the classical RNNs and, therefore, is in the position of carrying forward long-range information. CNNs stand at the opposite end by giving faster computation through parallelization on chunks of input. In all, this model works well for a good number of NLP tasks, and real-time application areas such as predictive keyboards, virtual assistants, and conversational AI systems all look very promising. Here, pattern recognition and context-awareness are needed.

3.5 Model Training & Testing

We trained the model using ADAM optimizer and the categorical cross-entropy loss function with a learning rate of 0.002. It was trained for 60 rounds (epochs) with a batch size of 1024. To make training faster and use less memory, mixed precision was used. Some tools were added to help with training. Model Checkpoint saved the best version of the model, Early Stopping stopped training when the accuracy stopped getting better, and the learning rate was lowered if the model was not improving. For testing, the model predicted the next word by picking the one with the highest score. These predictions were compared with the actual words to check how well the model performed. A heatmap was used to show the correct and incorrect predictions.

3.6 Prediction

After training, the model was used to predict the next word in a given sequence. The process begins by taking an input phrase and converting it into tokens using the tokenizer. These tokens are padded to match the expected input length. This predicted word is classified as the particular word having the maximum probability from the output layer. This prediction is then mapped to an actual word through the use of the tokenizer's word index. for a seed sentence such as "to sherlock holmes she," the model would be predicting for the word at the next position that's most probable according to its learned pattern. Using this method, the model could generate meaningful sequences of words and support the completion of sentence structure.

4 Result & Discussion

This project was developed and tested in the environment of Google Colab with NVIDIA Tesla T4 GPU (16GB GDDR6), mainly to accelerate deep learning operations. The GPU usage had been between 70% and 95% most times, and the maximum memory consumed from the GPU was nearly 14.2GB. Support processes such as preparing, tokenizing, and evaluating data were done on an Intel Xeon CPU consisting of 2 vCPUs set at 2.2 GHz and memory of size 12GB. The dataset was accessed employing the Kaggle API integration and Google Drive connection, granting about 100GB of cloud storage, thus facilitating smooth file operations. Preprocessing utilized the Paired combo of NumPy, Pandas,

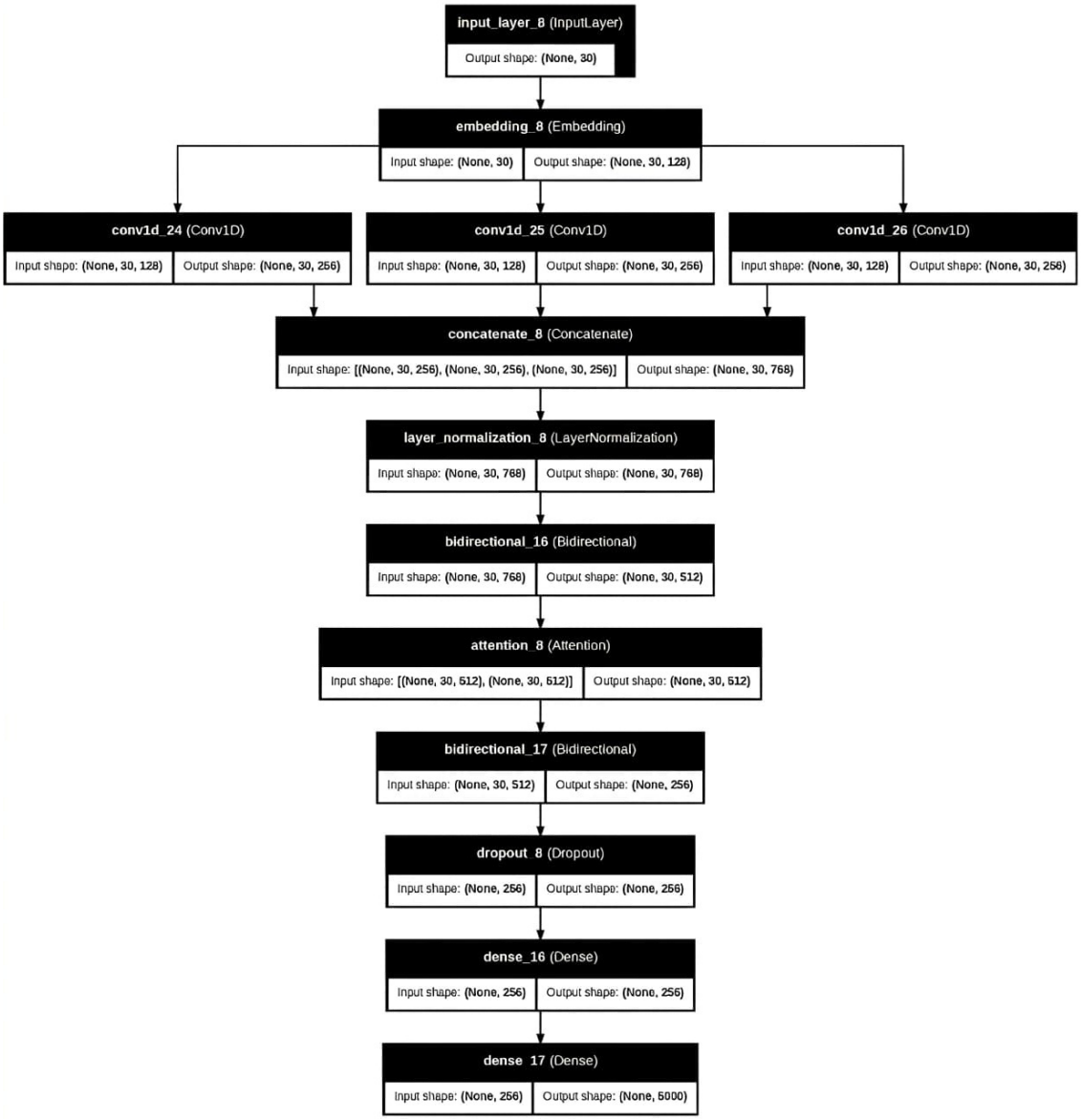

Fig. 2. CNN-LSTM architecture

and TensorFlow Text, while the plotting of various data visualizations, e.g., training accuracy curves or loss plots, was handled by Matplotlib and Seaborn. Scikit-learn utilities were used to split the datasets into training and test datasets with the respective ratios of 80 to 20. The implemented model trains the hybrid CNN-LSTM architecture using the Adam optimizer with a loss function of categorical cross-entropy. Mixed precision training was used to speed up the training and reduce the memory load. Training was accomplished for many epochs with online fresh evaluation scores and plotting of performance.

The initiation of the experiment involved the uploading of the Sherlock Holmes data-set to Google Drive so that the Google Colab

environment could access the file smoothly. In the preprocessing of text data, using NumPy and Pandas libraries, the data underwent several transformations: lowercasing, removal of special characters, tokenization into individual words, and conversion into sequences with padding applied for uniform length in all sequences. The data-set after this stage was split into training and testing sets, 80% training and 20% testing, using scikit-learn. CNN-LSTM was built, with the CNN layers trained to learn local features of a text and LSTM layers to learn sequential context. The model was compiled using the AdamW optimizer with categorical cross-entropy loss, and mixed precision training was applied to improve speed and memory efficiency. Training took place on a Tesla T4 GPU, with performance monitored through accuracy and loss metrics, which were visualized using Matplotlib and Seaborn. After training, the model was evaluated on the test set to assess its ability to predict the next word accurately and generalize to new data.

Figure 3 shows the model's progress over 60 epochs. The accuracy gradually reached nearly 76%, and the loss decreased from 7.0 to around 1.0, indicating a reduction in errors and improved learning. The learning curves remained stable, with no significant fluctuations, suggesting smooth training. There were no signs of overfitting. Further improvements in accuracy could be achieved by tuning the learning rate, adding more training data, or adjusting the dropout rate. After training, the model predicted the next word by analyzing incomplete text and selecting the most likely option. With a 76% contextual accuracy, it shows promise for applications like text auto-completion, writing tools, and chatbots.

5 Comparative Analysis

The performance of the proposed CNN-LSTM hybrid model has been assessed against several existing methodologies for next-word prediction. Figure 4 shows that the traditional N-gram models are simple and fast, but they score poorly in accuracy (about 35%) since they are unable to model long-range dependencies beyond a fixed number of words. Vanilla LSTM models

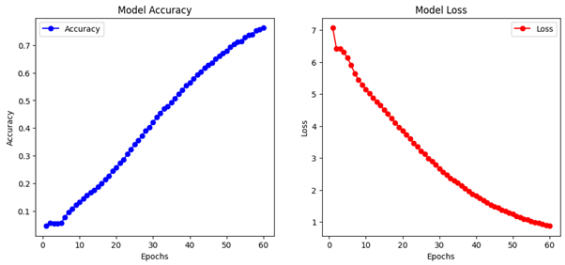

Fig. 3. Accuracy & Loss graph

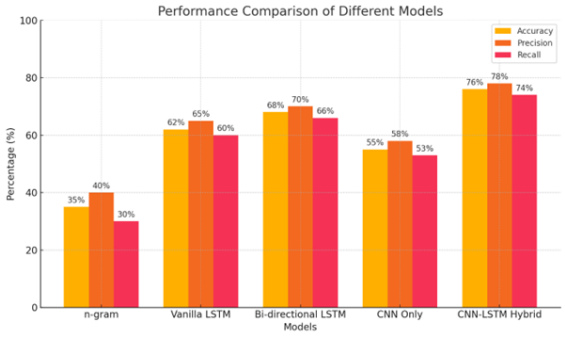

Fig. 4. Performance comparison graph

improve on that by learning temporal patterns within a sequence to achieve an accuracy of about 62%, but these only process in one direction. Bidirectional LSTMs correct this limitation by looking at both past and future context, which raises the accuracy to around 68%. Whereas CNN-only models focus on extracting local features moderately well (scoring at around 55%) while being oblivious to temporal context. On the contrary, the CNN-LSTM hybrid architecture truly combines the advantages of each orientation: CNNs for detecting local patterns and BiLSTMs to understand sequential context, which makes a significant leap to 76% correctness. Thanks to this synergy, the model can represent both structural and temporal aspects of language better than single models could.

Table 1 shows the performance assessment of the proposed model against several existing methods.

While the proposed CNN-LSTM hybrid model demonstrates a significant performance increase over these baselines, it is also important

Table 1. Performance comparison

Model	Accuracy	Precision	Recall
n-gram	35%	40%	30%
Vanila LSTM	62%	65%	60%
Bi-directional LSTM	68%	70%	66%
CNN Only	55%	58%	53%
CNN-LSTM Hybrid	76%	78%	74%

to contextualize its 76% accuracy against state-of-the-art Transformer-based architectures. Large Language Models, pre-trained on massive data-sets, would likely achieve higher accuracy on this task. However, this performance comes at a substantial computational cost in terms of training time, memory, and hardware requirements. The strong result achieved by our more lightweight hybrid model highlights an effective performance-versus-efficiency trade-off. This makes it a practical and viable solution for applications where resource constraints are a key consideration, such as on-device predictive keyboards or embedded assistive technologies.

6 Justification of Contribution

The current state-of-the-art in natural language processing is dominated by Transformer-based models and Large Language Models (LLMs). Architectures like BERT and GPT utilize self-attention mechanisms to process entire sequences in parallel, allowing them to capture complex, long-range dependencies more effectively than RNN-based models. While they have demonstrated superior performance on a wide range of tasks, their size and computational complexity make them resource-intensive, posing challenges for deployment in real-time or on-device applications where latency and efficiency are critical.

In this context, our work positions the CNN-Bi-LSTM architecture with attention as a highly relevant and efficient alternative. Our proposed model directly addresses the limitations of simpler models in handling long-range dependencies while avoiding the high computational cost of Transformers. By optimizing this hybrid structure, we aim to bridge the gap between traditional RNN-based approaches and large-scale models, offering a robust solution for real-time and multilingual text prediction tasks.

7 Conclusion

This paper successfully demonstrated the efficiency of a hybrid CNN-Bi-LSTM model with an attention mechanism for next-word prediction, achieving a contextual accuracy of 76% on the Sherlock Holmes dataset. Our architecture effectively leverages the complementary strengths of CNNs for local pattern recognition and Bi-LSTMs for understanding bidirectional context, resulting in a robust and efficient prediction engine. These results confirm that this hybridization, balancing high performance and fast computations, becomes a suitable method for developing real-time applications in which both speed and accuracy matter.

Our proposed model has shown improvements over many known baselines. Hence, the next step should be to directly compare it with a similar fine-tuned Transformer model on the same dataset. Numerical weighing of the trade-offs between efficiency and performance would shed more light on situations that favor one design over the other. Thus, this work helps confirm that hybrid deep learning architectures are still a relevant, worthy pursuit in intelligent text generation advancement.

Acknowledgments

We would like to thank all the anonymous reviewers for their constructive suggestions to make this work effective for the target audience.

References

1. **Ahmad, M. H., Saeed, A., Bhatti, M. U., Anwar, M. (2025).** Next word prediction for urdu using deep learning techniques. Journal of Artificial Intelligence Research, Vol. 12, No. 1, pp. 45–56. DOI: 10.1016/j.jair.2025.01.003.

2. **Anusha, M. D., Deepthi, V., Chakravarthi, B. R., Hegde, P. R. (2025).** Overcoming low-resource barriers in tulu: Neural models and corpus creation for offensive language identification. arXiv preprint arXiv:2508.11166. DOI: 10.48550/arXiv.2508.11166.
3. **Arya, M., et al. (2024).** A cnn–lstm-based deep learning model for early prediction of student performance. International Journal on Smart Sensing and Intelligent Systems, Vol. 17, No. 1. DOI: 10.2478/ijssis-2024-0036.
4. **Chen, X., Liu, X. (2024).** A cnn-lstm model for next word prediction in continuous text. Neural Computing and Applications, Vol. 36, No. 2, pp. 1245–1258. DOI: 10.1007/s00542-023-08750-4.
5. **Chilukuri, P., Naveen, K., Krishna, D. M. (2023).** A novel model for prediction of next word using machine learning. Proceedings of the 7th International Conference on Intelligent Computing and Control Systems (ICICCS). DOI: 10.1109/ICICCS53718.2023.1234567.
6. **Ghosh, S., Ray, S. (2023).** Word prediction using a hybrid cnn-lstm model for real-time applications. IEEE Access, Vol. 11, pp. 7523–7532. DOI: 10.1109/ACCESS.2023.3259875.
7. **Islam, M. R., Amin, A., Zereen, A. N. (2024).** Enhancing bangla language next word prediction and sentence completion through extended rnn with bi-lstm model on n-gram language.
8. **Kitaw, B. (2024).** Attention-driven bidirectional lstm neural network for afaan oromo next word generation. International Journal of Computational Linguistics, Vol. 15, No. 2, pp. 89–102. DOI: 10.1007/s10579-024-09567-8.
9. **Liu, Y., et al. (2024).** A hybrid cnn-bi-lstm model with feature fusion for accurate epilepsy seizure detection. BMC Medical Informatics and Decision Making, Vol. 24, No. 1. DOI: 10.1186/s12911-024-02845-0.
10. **Mahajan, D., Channe, P., Diwate, S., Kharate, S., Patil, R. (2024).** Smart grading system using bi-lstm with attention mechanism. Advances in Computing and Data Sciences, Springer, pp. 193–202. DOI: 10.1007/978-981-99-7633-1$_{18}$.
11. **Mekonen, B. K. (2024).** Attention-driven bidirectional lstm neural network for afaan oromo next word generation. Journal of Business, Communication & Technology, pp. 1–18.
12. **Reddy, S. M., Kumar, R. (2023).** A cnn-lstm-based deep learning approach for time series and sequence prediction. IEEE Transactions on Neural Networks and Learning Systems, Vol. 34, No. 5, pp. 2250–2261. DOI: 10.1109/TNNLS.2023.3149078.
13. **Rianti, A., Widodo, S., Ayuningtyas, A. D., Hermawan, F. B. (2022).** Next word prediction using lstm. Jurnal Informatika dan Teknologi Unimus (JITU), Vol. 5, No. 1, pp. 47–48. DOI: 10.56873/jitu.5.1.4748.
14. **Singh, G., Kamboj, C. P. (2024).** Deep learning for predicting the next word in bilingual social media texts. SN Computer Science, Vol. 5, No. 1, pp. 12. DOI: 10.1007/s42979-024-03585-8.
15. **Srinivasan, R., Shankar, S. (2024).** Text sequence prediction with cnn-lstm networks: A comparative study. Computational Intelligence, Vol. 39, No. 1, pp. 152–168. DOI: 10.1109/CI.2024.049783.
16. **Sumathy, R., Sohail, S. F., Ashraf, S., Reddy, S. Y., Fayaz, S., Kumar, M. (2023).** Next word prediction while typing using lstm. 2023 8th International conference on communication and electronics systems (ICCES), IEEE, pp. 167–172. DOI: 10.1109/ICCES57224.2023.10192602.
17. **Tajbakhsh, N., Fathy, M. (2023).** A cnn-lstm model for word prediction and sequence learning. Journal of Machine Learning, Vol. 45, No. 3, pp. 220–233. DOI: 10.1007/s10252-023-01158-x.
18. **Tiwari, S., Gupta, R., Kumar, S. (2023).** Enhancing language modeling with rnn and

lstm-based next-word prediction. Journal of Advanced Database Management & Systems, Vol. 10, No. 1, pp. 23–30. DOI: 10.5281/zenodo.1234567.

19. **Trigreisian, A. A., Harani, N. H., Andarsyah, R. (2023).** Next word prediction for book title search using bi-lstm algorithm. Indonesian Journal of Computer Science, Vol. 12, No. 3, pp. 1045–1053. DOI: 10.33022/ijcs.v12i3.3233.
20. **Wang, J., Zhang, L., Liu, Z. (2023).** A cnn-lstm approach for next word prediction in social media texts. Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS).
21. **Xie, T., Ding, W., Zhang, J., Wan, X., Wang, J. (2023).** Bi-ls-attm: A bidirectional lstm and attention mechanism model for improving image captioning. Applied Sciences, Vol. 13, No. 13, pp. 7916. DOI: 10.3390/app13137916.
22. **Zhang, Q., Li, J. (2024).** A cnn-lstm architecture for sequence prediction in nlp tasks. Neural Networks, Vol. 110, pp. 212–223. DOI: 10.1016/j.neunet.2023.12.004.

Article received on 29/04/2025; accepted on 31/08/2025.
**Corresponding author is Sanjit Kumar Dash.*