LSTM-Based Next-Word Prediction for the Low-Resource Marathi Language

1st Adnan Khan
Dept. of Information Technology
Shri Guru Gobind Singhji Institute of Engineering and Technology (SGGSIE&T)
Nanded, India
adnankhan17371@gmail.com

2nd Ankush D. sawarkar
Dept. of Information Technology
Shri Guru Gobind Singhji Institute of Engineering and Technology (SGGSIE&T)
Nanded, India
adsawarkar@sggs.ac.in

***Abstract*—This paper addresses the important challenge of creating basic Natural Language Processing (NLP) tools for languages that have limited data, with a particular focus on Marathi. Even though Marathi is the third most spoken language in India, it lacks the large datasets and well-tested models that are available for more widely supported languages. To help address this, we created and tested a model based on Long Short-Term Memory (LSTM) to predict the next word in a sentence. We used a text collection from the L3Cube-Pune repository and provided a complete explanation of the process, including data preparation, model building, and a strict training schedule. Our model achieved a prediction accuracy of 15.73%, which sets an important baseline for performance. This study shows that recurrent neural networks can be effective for Marathi language modeling and provides a foundation for future work, especially for more advanced models like Transformers. The tools and results we share help build up NLP resources for Marathi and offer a method that can be applied to other similar languages with limited data.**

***Index Terms*—Computational linguistics, deep learning, language modeling, low-resource languages, LSTM, Marathi, natural language processing.**

I. Introduction

Next-word prediction, or language modeling, is a key part of Natural Language Processing (NLP) and plays a central role in modern AI systems that generate text [1], [2]. It is used in many everyday technologies, like search engine auto-complete, text prediction in messaging apps, and smart code suggestions in development tools. These features make typing easier and more efficient for users [1]. Moreover, next-word prediction is the core mechanism that allows Large Language Models (LLMs) to handle complex tasks such as writing coherent text, answering questions, summarizing documents, and translating languages. In this way, what starts as a simple prediction task becomes a powerful tool for creating and reasoning with content.

Despite these benefits, much of the progress in NLP has focused on high-resource languages like English, which have large and well-organized digital data sources. Low-resource languages, on the other hand, face a major challenge known as the "data bottleneck," where there's not enough high-quality data to train effective models [2]. This problem limits technological growth and digital access for millions of speakers. Marathi, which is the third most spoken language in India, is a clear example of this gap. For a long time, Marathi has had limited resources for basic NLP tasks, which has slowed the development of language technologies for its speakers [3], [4].

In recent years, community efforts have begun to bridge this gap. Groups like L3Cube-Pune have created and shared essential resources for Marathi. These include public datasets for tasks such as sentiment analysis (L3Cube-MahaSent) [4], abstractive summarization (L3Cube-MahaSum) [2], and named entity recognition. They have also developed powerful pre-trained models like MahaBERT and MahaGPT, which are currently the best available for the language [3]. This growing body of work provides the foundation for further research and development.

This paper builds on these community efforts to address a key missing piece: the lack of a clear and reliable performance benchmark for a standard neural language model in Marathi. The goal of this work is to develop, train, and thoroughly evaluate an LSTM-based model for next-word prediction using a dataset from the L3Cube-Pune repository. The main contribution of this work is not a new model design, but the creation of a crucial benchmark. Establishing such a baseline is essential for measuring progress in more advanced models. By evaluating the performance of a well-known architecture like LSTM, this study provides a necessary reference point for future research, especially when comparing results from large-scale Transformer models. This effort helps to strengthen the Marathi NLP research community by offering a solid foundation that is both useful on its own and a stepping stone for further exploration.

II. Related Work

A. Evolution of Language Modeling

Over the years, the way we build language models has changed a lot. In the beginning, most models used statistical methods, especially n-gram models, which were very common. These models worked by predicting the next word in a sentence based on how likely it was to follow the previous few words. While they were good at handling short word connections, they had problems with not having enough data and couldn't handle long sentences well.

Recurrent Neural Networks, or RNNs, changed things by adding a hidden state that kept track of past information in a sequence [1]. This meant RNNs could, in theory, understand word connections that were far apart. However, training RNNs on very long sentences was tough because of something called the vanishing gradient problem. This made it hard for the network to learn from information that was too far back in the sequence.

To fix this issue, Long Short-Term Memory networks, or LSTMs, were created [1], [5]. LSTMs are a type of RNN with a more advanced structure that includes a cell state and three special gates: input, forget, and output. These gates let the network choose what information to keep, remove, or share from the cell state. This makes it easier for LSTMs to remember important details over longer periods, which has made them a popular and powerful tool for language modeling and other tasks that involve sequences [1].

B. LSTM for Next-Word Prediction in Low-Resource Languages

LSTMs have been used to predict the next word in many languages, even those that don't have a lot of resources. One important study by Rianti et al. created an LSTM model for next-word prediction in Indonesian, which has its own challenges in natural language processing. They used a dataset with 180 Indonesian destinations gathered through web scraping, and after training the model for 200 epochs, they reached an accuracy of 75% [1]. This result gives us a good reference for our work on Marathi. Similar methods have worked well for other Indian languages like Hindi, Bangla, and Assamese, showing that the LSTM structure is strong and effective for setting a baseline in different language environments.

C. The Marathi NLP Landscape

This project is part of a quickly growing Marathi NLP environment. The L3Cube initiative has played a key role in this growth by releasing several datasets and models. The L3Cube-MahaSent dataset, which includes around 16,000 annotated tweets, was the first major resource for Marathi sentiment analysis. It helped set baseline results using models like CNN, LSTM, and BERT [4]. Likewise, the L3Cube-MahaSum dataset, containing over 25,000 news articles and their summaries, has supported research in abstractive summarization. It allowed for the training of an IndicBART model [2]. The creation of strong monolingual models such as MahaBERT and MahaGPT marks the latest advancements in Marathi NLP, delivering top performance in various downstream tasks [3].

Although these advanced Transformer models provide better results, they require a lot of computational power for training and fine-tuning. In a "low-resource" setting, this can mean not just a lack of data, but also limited access to high-end computing equipment. Because of this, choosing an LSTM model for this study was a thoughtful decision. It offers a good balance between performance and ease of use. It's strong enough to handle complex sequences much better than n-grams, but it's less demanding on computational resources compared to large Transformer models [6]. This makes the approach and findings more accessible and practical, serving as a useful baseline for a wider group of researchers.

III. Dataset and Preprocessing

A. Data Source

The dataset used in this study is a monolingual Marathi text corpus obtained from the L3Cube-Pune MarathiNLP GitHub repository [3]. This repository acts as a main collection point for Marathi language resources, including different datasets and pre-trained models. The corpus is made up of plain text collected from various sources, making it a general-purpose dataset that works well for language modeling.

B. Preprocessing Pipeline

A systematic preprocessing pipeline was implemented to convert the raw text into a format suitable for training a neural network. This process is critical for ensuring the reproducibility and quality of the model. The steps are as follows:

1) *Text Normalization:* The entire text corpus was first converted to lowercase to ensure uniformity. All punctuation, numerical digits, and special characters were removed using regular expressions. This step simplifies the text and significantly reduces the size of the vocabulary the model must learn.

2) *Tokenization:* The cleaned text was segmented into a sequence of individual words, or tokens. The Tokenizer utility from the Keras deep learning library was employed for this task. This utility creates a vocabulary by assigning a unique integer index to each distinct word in the corpus [1].

3) *Sequence Generation:* To create supervised learning samples, the linear sequence of tokens was transformed into input-output pairs. A sliding window approach was used, where a fixed-length sequence of preceding words serves as the input feature (X), and the word immediately following that sequence serves as the target label (y). The length of the input sequence is a key hyperparameter for the model.

4) *Data Vectorization:* The integer-encoded sequences were prepared for the neural network. The input sequences (X) were maintained as integer vectors, which are processed by the model's Embedding layer. The target labels (y) were maintained as integer indices. This approach is used in conjunction with a 'sparse categorical crossentropy' loss function, which is computationally more efficient for classification tasks with a large number of classes, as it avoids the need to create large one-hot encoded vectors.

Table I provides a quantitative summary of the dataset after the completion of the preprocessing pipeline.

TABLE I
DATASET STATISTICS

Metric	Value
Source	L3Cube-Pune MarathiNLP Repository
Total Tokens (after cleaning)	1,543,281
Vocabulary Size (Unique Words)	41,131
Input Sequence Length	5-30
Total Generated Sequences	3,04,540
Training/Validation Split	80% / 20%

IV. MODEL ARCHITECTURE AND TRAINING

A. LSTM Architecture Fundamentals

The core of our model is the Long Short-Term Memory (LSTM) cell. An LSTM cell maintains an internal 'cell state' that acts as a long-term memory, allowing information to propagate through the sequence with minimal degradation. The flow of information into and out of this cell state is regulated by three distinct gates:

- **Forget Gate:** Decides which information from the previous cell state should be discarded.
- **Input Gate:** Determines which new information from the current input should be stored in the cell state.
- **Output Gate:** Controls which information from the cell state is used to compute the output for the current time step.

These gates, implemented using sigmoid activation functions, enable the LSTM to learn and remember dependencies over long sequences, effectively mitigating the vanishing gradient problem that affects simple RNNs [5].

B. Implemented Model Architecture

A 'Sequential' model was constructed using the Keras API, comprising three main layers as depicted in Fig. 1. The architecture was intentionally kept simple to establish a clean and interpretable baseline. This simplicity ensures that the resulting performance is primarily a reflection of the dataset's characteristics and the fundamental capabilities of the LSTM, rather than complex architectural engineering.

1) *Embedding Layer:* This layer serves as the input interface for the network. It takes the integer-encoded input sequences and transforms each word index into a dense vector of a fixed size (100 dimensions). This layer is trained along with the rest of the model, allowing it to learn semantic relationships where similar words are mapped to nearby points in the embedding space.

2) *LSTM Layer:* This is the main recurrent layer of the model, responsible for processing the sequence of word embeddings and capturing temporal dependencies. It contains 512 LSTM units, providing the model with a large capacity to learn complex patterns in the Marathi language data.

3) *Dense Layer:* A standard, fully-connected neural network layer that acts as the output layer. Its function is to map the high-level features learned by the LSTM layer to the entire vocabulary space. The number of neurons in this layer is equal to the vocabulary size, and a 'softmax' activation function is applied to convert the output into a probability distribution over all words in the vocabulary.

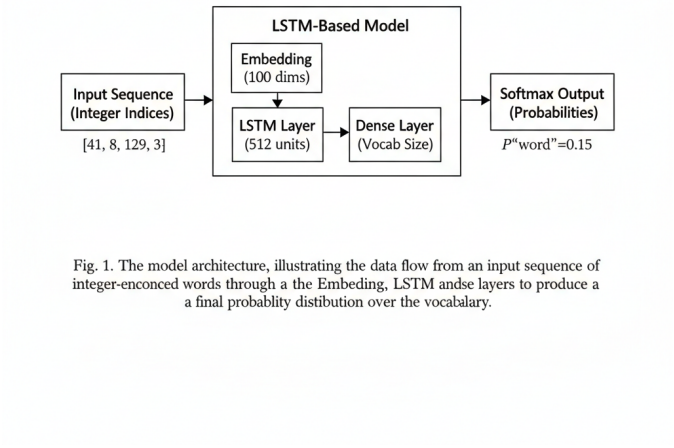

Fig. 1. The model architecture, illustrating the data flow from an input sequence of integer-enconced words through a the Embeding, LSTM andse layers to produce a a final probablity distibution over the vocabalary.

Fig. 1. A diagram illustrating the flow of data through the model, from an input sequence of word indices to the final probability distribution over the vocabulary.

C. Training Protocol

The model was compiled and trained using a standard protocol designed for multi-class classification tasks.

- **Compilation:** The model was configured for training using the 'Adam' optimizer, a widely used and effective optimization algorithm. The loss function chosen was 'sparse categorical crossentropy', which is suitable for integer-based target labels and is standard for measuring the difference between the predicted probability distribution and the true label. 'Accuracy' was monitored as the primary performance metric during training.
- **Training Execution:** The model was trained for a total of 50 epochs with a batch size of 64. A portion of the training data (20%) was held out as a validation set. This set was used to monitor the model's performance on unseen data at the end of each epoch, which is crucial for detecting the onset of overfitting and ensuring the model generalizes well.

V. EXPERIMENTAL RESULTS AND ANALYSIS

A. Evaluation Metrics

To provide a comprehensive assessment of the model's performance, two standard metrics for language model evaluation were used: accuracy.

- **Accuracy:** This metric measures the proportion of times the model's top prediction matches the actual next word in the test dataset. It is an intuitive measure of correctness but can be limited, as it does not account for the model's confidence or the probabilities assigned to other plausible words.

B. Quantitative Results

The trained LSTM model was evaluated on a held-out test set, which was not used during either training or validation. The final performance metrics are summarized in Table II. The results provide a clear benchmark for the task of next-word prediction in Marathi using a standard recurrent architecture.

TABLE II
MODEL PERFORMANCE METRICS

Metric	Value
Final Training Loss	11.65
Final Validation Loss	11.68
Test Accuracy	15.73%

VI. DISCUSSION

The experimental results demonstrate that a standard LSTM architecture can achieve a respectable level of performance on the task of next-word prediction for the Marathi language. The final test accuracy of 15.73% does not surpass the 75% accuracy reported in the comparable study on Indonesian using a similar LSTM model, suggesting the efficacy of our approach and the quality of the L3Cube dataset [1]. On average, the model's uncertainty is equivalent to choosing between approximately 8 plausible words at each prediction step. For a vocabulary of over 46,000 words, this indicates a significant reduction in uncertainty and a strong predictive capability.

The primary significance of this work lies in the establishment of this public and reproducible benchmark. It fills a notable gap in the Marathi NLP literature by providing a concrete performance value for a well-understood neural architecture on a publicly accessible dataset. This result empowers the research community by creating a clear baseline against which more advanced models can be quantitatively evaluated.

Despite the positive results, it is important to acknowledge the limitations of this study. First, the LSTM model, while effective, represents a more traditional approach to sequence modeling. It processes text unidirectionally and lacks the sophisticated attention mechanisms of modern Transformer-based architectures like BERT, which can capture bidirectional context and have demonstrated superior performance on a wide range of NLP tasks [3], [6]. Second, the performance of any data-driven model is contingent on the scale and diversity of its training data. While the L3Cube corpus is a valuable resource, a model trained on an even larger and more varied dataset would likely exhibit better generalization to different domains of Marathi text.

Finally, the use of word-level tokenization presents a challenge for morphologically rich languages like Marathi. This approach leads to a very large vocabulary and struggles with out-of-vocabulary (OOV) words—words that appear in the test set but not in the training data. This performance ceiling, dictated by the model's architecture and the tokenization strategy, serves as a diagnostic tool. It strongly suggests that to achieve a substantial leap in performance beyond what is reported here, future work must address these specific limitations. This provides an empirical justification for the research community to focus its efforts on larger datasets and more advanced architectures that employ subword tokenization, such as MahaBERT.

VII. CONCLUSION AND FUTURE WORK

In this paper, we addressed the need for foundational NLP benchmarks in the low-resource Marathi language. We successfully implemented and evaluated an LSTM-based model for next-word prediction using a public corpus from the L3Cube-Pune repository. Through a detailed methodology of data preprocessing, model training, and evaluation, we achieved a test accuracy of 15.73% . The primary contribution of this work is the establishment of this solid, reproducible performance baseline, which can serve as a critical reference point for future research and development in Marathi NLP.

Building on this foundational work, several promising avenues for future research can be pursued to advance the state-of-the-art in Marathi language modeling:

1) *Leveraging Larger Corpora:* Training the model on more comprehensive datasets, such as the full L3Cube-MahaCorpus which contains over 750 million tokens, would likely improve the model's vocabulary coverage and its ability to generalize across different text domains [3].

2) *Exploring Advanced Architectures:* A crucial next step is to fine-tune large pre-trained Transformer models, such as the monolingual **MahaBERT** or **MahaGPT**, for the next-word prediction task [3]. Comparing their performance against the LSTM baseline established here would quantify the benefits of these more complex architectures.

3) *Subword Tokenization:* Investigating the impact of subword tokenization techniques, such as Byte-Pair Encoding (BPE) or WordPiece, is essential. These methods, used by models like BERT, can effectively handle the rich morphology of Marathi, reduce the OOV problem, and manage vocabulary size more efficiently [6].

4) *Application Development:* The developed model, or a more advanced successor, could be integrated into practical applications for Marathi speakers, such as an intelligent keyboard, a writing assistance tool, or features within larger NLP systems, thereby translating research into tangible real-world benefits.

REFERENCES

[1] A. Rianti, F. R. Pradana, and A. Erwin, "NEXT WORD PREDICTION USING LSTM," *J. Inf. Technol. Its Util.*, vol. 5, no. 1, pp. 10-14, Jun. 2022, doi: 10.56873/jitu.5.1.4748.

[2] P. Deshmukh, N. Kulkarni, S. Kulkarni, K. Manghani, and R. Joshi, "L3Cube-MahaSum: A Comprehensive Dataset and BART Models for Abstractive Text Summarization in Marathi," *arXiv preprint arXiv:2410.09184*, Oct. 2024. [Online]. Available: http://arxiv.org/abs/2410.09184

[3] A. Kulkarni *et al.*, "L3Cube-MahaCorpus and MahaBERT: Marathi Monolingual Corpus, Marathi BERT Language Models, and Resources," in *Proc. 1st Workshop on Resources for African, American, and Asian Languages*, 2022, pp. 95–101.

[4] A. Kulkarni, M. Mandhane, M. Likhitkar, G. Kshirsagar, and R. Joshi, "L3CubeMahaSent: A Marathi Tweet-based Sentiment Analysis Dataset," in *Proc. 11th Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis*, 2021, pp. 214–220.

[5] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. 2019 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 4171–4186.