Journal of Computing & Biomedical Informatics
ISSN: 2710 - 1606

Volume 08 Issue 02
2025

Research Article
https://doi.org/10.56979/802/2025

Advanced Next-Word Prediction: Leveraging Text Generation with LSTM Model

Syed Hasham Hameed[1], Muhammad Munwar Iqbal[1*], Hasnat Ahmed[1], Wahab Ali[2], Saqib Majeed[3], Malik Muhammad Ibrahim[1]

[1]Department of Computer Science, University of Engineering and Technology Taxila, Taxila, 47080, Pakistan.
[2]Department of Computer Science, SZABIST University Islamabad, Islamabad, Pakistan.
[3]University Institute of Information Technology, PMAS-Arid Agriculture University, Rawalpindi, Pakistan.
[*]Corresponding Author: Muhammad Munwar Iqbal. Email: munwar.iq@uettaxila.edu.pk

Received: December 02, 2024 **Accepted:** February 17, 2025

Abstract: Natural Language Processing (NLP) increasingly relies on machine learning to make better predictions of sequential text. This work focuses on the application of Long Short-Term Memory Networks, a variant of Recurrent Neural Networks that is specialized for modeling long-term dependencies. Traditional RNNs leave much to be desired in predicting sequences that contain repeated patterns or contextual dependencies. The research uses "The Adventures of Sherlock Holmes" as the training dataset and applies TensorFlow and Keras frameworks for implementation. The major preprocessing steps included word tokenization, n-gram creation, and one-hot encoding to prepare the dataset for modeling. The LSTM model was trained over 100 epochs to optimize prediction capabilities. Through this work, we show that LSTM is effective in next-word prediction and can potentially improve the performance and practicality of language models for real-world applications. The model achieved a commendable accuracy of 87.6%, demonstrating its effectiveness.

Keywords: Next Word Prediction; LSTM; RNN; NLP; Tensor Flow; Keras; Deep Learning

1. Introduction

Natural Language Processing (NLP) is a vital part of AI, focusing on how computers and humans communicate using natural language. Machine learning (ML) algorithms require training data to develop models that can make predictions and decisions without needing explicit programming. Deep Learning (DL), a subset of ML, utilizes artificial neurons for computation. This approach helps computers understand, interpret, and generate meaningful language. A fundamental task in NLP is next-word prediction, which determines the word that should follow another in a sentence. This capability is essential for various applications, including text generation, translation, and smart auto-completion.

TensorFlow
Google
What is the Natural Language
what is the natural language of india
what is the natural language processing
what is the natural language
what is the natural language definition
natural language processing
what is the natural programming language
what is the meaning natural language
what is natural language processing in ai
what is natural language understanding
what is natural language generation
LSTM
Next Word Prediction

Figure 1. Tools and Techniques for Next Word Prediction.

Next-word prediction plays a significant role in enhancing user experience across multiple applications. It is crucial for text messaging apps, search engines, and writing assistants like Google

Assistant, which millions of smartphone users trust. By predicting the next word, these systems help users type faster and with fewer mistakes, improving communication efficiency and clarity. NLP facilitates smooth interactions between computers and humans using everyday language. Moreover, predicting subsequent words is a key element of complex language models used in machine translation and emotion recognition. The machine-learning techniques found in NLTK libraries are also essential in this context.

Creating accurate next-word prediction systems is instrumental when it comes to enhancing the efficiency and dependability levels of all these programs, hence rendering them handier and more applicable under different conditions.

This research studies to boost the precision of predicting the subsequent word in a sentence based on the LSTM model. Specific objectives of the study include:

- To develop a model bearing on Long Short Term Memory and use it to predict the next word.
- To evaluate the enormous and varied text dataset used for the model training. It assesses how well the model performs by comparing it to other methods.
- To determine the weaknesses and advantages of the LSTM model in the framework of next-word prediction.

Sections of the paper are organized to each cover a different research area. The following is the way it is laid out. Introduction: Provides an overview of previous research in the field, explaining the importance of next-word prediction and its relevance to advancing language modeling. It also outlines the current state of language modeling and sets the stage for what this paper aims to achieve. Literature Review: Covers the foundations of NLP, summarizing recent research studies. It highlights various techniques and tools for similar purposes and includes a detailed literature table to compare prior work. Research Design and Methods: Describes the study's approach, including details about the dataset, preprocessing steps, model development, and configurations used. Results and Discussions: Present the findings through performance metrics, accuracy and loss graphs, a confusion matrix, and a classification report. It also compares the proposed model's performance with existing methods. Conclusion and Future Work: Summarizes the study's key findings, explaining why the model outperforms others. It also discusses limitations and offers suggestions for future research directions.

2. Literature Review

Ganai et al. [1] in this paper based on Long Short-Term Memory (LSTM) cells and Recurrent Neural Networks (RNN) can be used to forecast the next word in a sequence, an indispensable component of Natural Language Processing (NLP) in general. Researchers examine how to structure information retrieval systems by introducing a tree-based generative modeler that combines RNNs and LSTMs. A different document shows the creation of an LSTM model intended to predict the next words in Assamese text. The linguistic properties of Assamese that make it unique are described by the model; it outperforms other RNNs. This article shows that this proposed model has better predictive power for Assamese texts than those featured in this. This paper reached accuracy levels of 72.10% in proper noun detection tasks but 88.20% for Assamese text prediction, with the same number of words and HTML elements as above.

Another paper [2] studies how AWD-LSTM model usage when combined with human eye tracking metrics, assists in an improved understanding of word prediction mechanisms as well as human reading kinetics. This research further strives to realize how these models imitate human predictability as well as gaze duration, thereby improving our knowledge of language processing and interpretability of these models. Although LSTM models tend to approximate specific human text-processing elements, controversies encourage model performance improvement and alignment with human[3] cognitive processes. Eisapeet al. [4], the authors describe a technique called Cloze Distillation that instructs a neural language model to imitate human understanding to close the gap between it and actual human-like comprehension. An experiment proved that Cloze Distillation successfully reduced the gap between the model's expectations and those of natural speakers, which means they are more alike than before.

Tessema et al. [5] present the attention shifts to the creation of a system that will predict the following word in one of the most challenging languages. This paper demonstrates that bi-directional LSTM and various Hyperparameters can significantly improve accuracy. The approach identified the optimal model setup through deliberate tests that glorify its worthiness in forecasting the subsequent Amharic words. In a paper by Luukkonen et al. [6], the writer of this paper has brought about a proactive information retrieval method using LSTM NETWORKS for predicting text inputs. LSTM models have an advantage with this technique because it shows what they are capable of in terms of improving information retrieval that can go beyond what other methods would achieve in exploratory search tasks when related extra details from the preceding interactions user's behavior towards ongoing conversation are used which increases the level of accuracy in these types of tasks.

Building a language model with LSTM networks is possible thanks to the paper [7], thus highlighting the lack of research on the Bodhi language. However, they claim that their model is very important to safeguard the Bodhi language when it is difficult to access linguistic research and resources. In this paper, we have presented how dataset preparation and training have been done, and it has been very accurate when predicting what follows after certain words in Bodhi texts.

Traditional convolutional neural networks (CNNs), as discussed in [8], face limitations when processing text, particularly in capturing temporal associations between words appearing at different points in time. However, by incorporating techniques like dropout alongside addressing these challenges in pattern recognition, we have been able to create models with improved generalization capabilities, surpassing those reliant on gating mechanisms. This approach has made it significantly easier to handle words with complex dependencies. The model effectively integrates the contextual information of individual words within a sentence, resulting in a richer and more comprehensive contextual representation.

A journal article by [9] introduces Bio-Pred, an AI algorithm designed[10] to predict words in the English language by drawing inspiration from how the human brain works, using biological principles in robotics. Bio-Pred stands out because it does not just simulate AI behavior but mimics human thinking. It combines bi-directional connectionist structures with self-learning mechanisms, enabling it to adapt and improve over time. Similarly, Rosa et al. [11] explore how personality profiling can be integrated into predictive text models. Their research focuses on using the five major personality traits identified through Twitter data to predict words [6-9, 11-18] based on a user's tweet. It is achieved through Markov models, which provide predictions grounded in personality insights. Our primary goal is to enhance the performance of predictive text models, making them more accurate, adaptive, and user-focused. We aim to create a more engaging and intuitive text prediction experience by tailoring these models to better understand and interact with users.

Table 1. Literature Review.

Authors and Reference	Model Used	Model Accuracy	Dataset Used	Limitations
Ganai et al. [1]	RNN and LSTM cells	Around 46% after 20 epochs	Twenty popular books from Project Gutenberg	Long training times needed for large labeled data suggest exploring sub-word and corpus-level modeling [1] [11]
Barman et al. [2]	LSTM, a type of RNN	88.20% Assamese text, 72.10% phonetically transcripted Assamese language	Transcripted Assamese language according to the IPA chart	Focus on the Assamese language limits applicability to others and challenges in modeling linguistic nuances [2][12]
Umfurer et al. [3]	AWD-LSTM Language Model	Not explicitly stated	Corpus of short stories with Gaze Duration Predictability	A challenge in understanding LSTM models and human cognitive processes

			metrics, Spanish Wikipedia	suggests further research [3][13]
Eisape et al. [4]	LSTM, Transformers (XLNet, GPT-2, Transformer-XL), 5-gram model	Not specified	Provo Corpus for cloze completion and reading time data	LSTM's performance in next-word prediction does not necessarily translate to superior reading time prediction [4][14]
Tessema et al. [5]	Bi-LSTM	The precision of 91.02% on training, 90% on testing	Amharic language text designed for the study	Challenges related to Amharic's morphological richness, absence of human-engineered linguistic features [5][15]
Luukkonen et al. [6]	LSTM Network	Not provided	Abstracts from the Computer Science branch of the arXiv database	Effectiveness varies by search task; further exploration of query expansion methods is needed [6][16]
Kumar et al. [7]	LSTM based on RNN	70% for English, 50% for Bodhi after specific training iterations	Collection of Bodhi words from Ladakhi articles, newspapers, and dictionaries	Preprocessing and training challenges due to Bodhi's complexity expected improvement with more data [7][17]
Yang et al. [8]	MCNN-ReMGU model	Performance improvements discussed	Penn Treebank and WikiText-2 datasets	Complexity and computational resource requirements could be inferred challenges.
Elbaghazaoui et al. [9]	Markov Chain Model	Not explicitly mentioned	Tweets for profiling personality traits using the Big Five	The conceptual approach discussed without detailing specific limitations or model accuracy
Rosa et al. [10]	Bio-Pred, a biologically inspired connectionist model	Not specified	Not explicitly mentioned	Focuses on biological motivation and architectural innovations without detailing specific limitations

3. Materials and Methods

Figure 2 presents the LSTM-based Proposed Model for Next Word Prediction. It shows the steps that were taken to get the desired results. The first step comprises collecting textual data, which is then tokenized using the tokenizer. These tokens are used to make the n-grams, which help to understand the context of the text, and then, these are one-hot encoded to convert textual data into numerical or vector form, which is then passed through LSTM, which provides us with the desired results[19].

3.1. Dataset

The present study utilizes data from a revered piece of literature: The Adventures of Sherlock Holmes contains twelve short stories for which it is famous. These stories are best known for their protagonist, Sherlock Holmes, a fictional detective always accompanied by his associate, Dr. John Watson. One of the most popular and widely read books in the Sherlock Holmes collection has been published since 1892 and up to the present day. It was written in episodes in the strand magazine, but they were combined into one book, eventually containing 105,000 words, each marked by the letter'/n' [20].

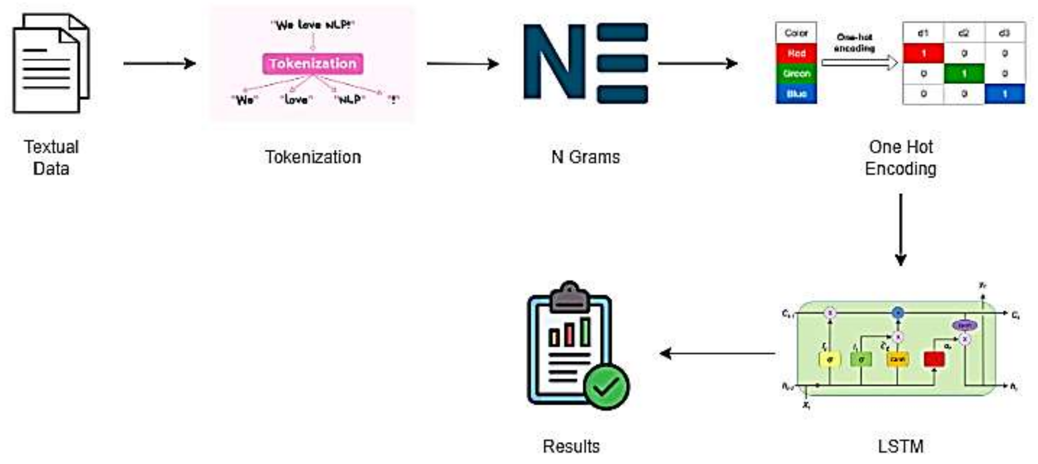

Figure 2. LSTM-based Proposed Model for Next Word Prediction.

3.2. Method

There are several steps involved in this process, such as text preprocessing, model definition, and training.

3.2.1. Preprocessing Steps:

1. Text Cleaning and Removal of Special Characters:

Purpose: Cleaning of text will finally ensure the text data is prepared for machine learning or NLP tasks. Raw data is often messy from the datasets as it contains accentuating details such as punctuation, numbers, special characters, and stopwords, all of which could easily confuse a machine while learning meaningful elements. Once done, the removal of the irrelevance will give the model a broad view of its actual contents, which are words and phrases that add some meaning to the model. The process is very crucial in reducing noise, which usually misleads or complicates training for the model. Clean text becomes interpretable for the model to ensure effective and efficient learning later on. For instance, in analyzing a book, irrelevant clutter like repetitious filler words or page numbers allows the analyst to concentrate on the main narrative while leaving out irrelevant details.

Implementation: A custom function was created to clean the text using regular expressions and strong tools to match patterns. The function is to be designed to detect and remove all the non-alphanumeric characters such as punctuation marks (e.g., "!", "?"), digits (e.g., "123"), and special symbols (e.g., "@", "#"). After this step, the text was stripped down to its basic components: words. The other extraneous characters were removed, and the function also filtered out stopwords. Stopwords are common words like "the," "is," "and," and "of" that occur frequently in text but do not carry substantial meaning on their own. For example, in the sentence "The cat is sitting on the mat," the removal of stopwords such as "the" and "is" leaves behind the more meaningful content words: "cat sitting mat." This enables the model to focus on the heart of the sentence rather than being swayed by redundant or frequently occurring terms. The function is flexible and can be adapted to specific datasets. For instance, if working on domain-specific text For example, if the data consists of legal documents or medical records, we can customize the stopwords list or the cleaning process to retain particularly important terms for that domain.

Role: Text cleaning is the strongest foundation of the machine learning pipeline. Feeding raw, unprocessed text to a model can seriously impede its ability to learn patterns and relationships in the data. Unwanted characters, digits, and stopwords make for distractions as they generate noise that should dilute the signal of meaningful information in the text. Take punctuation marks or some irrelevant symbols, which do not carry any meaning for what a sentence really wants to say. Removing these will ensure that the model is not bogged down by details irrelevant to its goal. Also, Stopword removal enhances the focus of the model on content words that carry the bulk of semantic meaning in a sentence. Doing so improves the quality of the data the model uses for learning. This way, the model can build its meaningful understanding of the patterns and becomes more capable of performing tasks like sentiment analysis, text classification, or language generation. Without cleaning, the model will waste computational resources in trying to make sense of the noise, thus leading to bad performance or overfitting.

2. Word Tokenization:

Purpose: A computer model had to decompose text into the smallest pieces called tokens before it could operate with text. These tokens primarily included a word or a phrase. For every token, a unique number was given, forming a list of all the words in the text, which is called a vocabulary. The machine processed and learned from the text by converting words into numbers. Tokenization is a means to translate from a human-readable into a machine-convertible format; in this regard, for the phrase "Machine learning is fun, tokenization enabled word breaking so as to process those more efficiently.

Implementation: Here, to tokenize any text, they would split all the sentences that are given into their words at different spaces or by punctuation. That is, instead of "AI is transforming the world," separate tokens were divided: "AI," "is," "transforming," "the," and "world." Then, each token is assigned a numerical value that represents it in vocab. Libraries, particularly NLTK and SpaCy, were used to facilitate this process.

Role: Tokenization is the most significant step that turns raw text into numbers, making it easier for the model to read and understand. In the absence of tokenization, it would not be possible for the model to interpret and learn from the text data. This step was vital in making sure that the machine saw the text in a structured form. For instance, the tokenization helped the model understand that "AI" and "world" were two words separate from each other.

Impact: Good tokenization improved how the model learned and performed. This ensured that every unique word within the text was captured clearly, meaning the model does not focus much on the words but rather on the pattern and context of the data. The proper tokenization reduces unnecessary text complexity, which enables the model to process faster. For instance, by tokenizing a sentence into words such as "great" or "bad," the model was able to understand sentiments much better. It also allowed the model to make accurate predictions by only focusing on meaningful parts of the text, which improved efficiency and performance in general.

In summary, tokenization was a key step in preparing text for a model. It broke down text into manageable pieces, simplified the data, and helped the model learn patterns effectively, resulting in better accuracy and predictions**.**

3. N-grams Creation:

Purpose: N-grams are sequences of 'n' items, for example, words or characters that play a vital role in understanding the structure and flow of text. An N-gram determines the next word or character from the preceding sequence of words or characters. Thus, it captures the context of the text and explains the patterns and relationships in its deeper usage. For example, in the phrase "artificial intelligence is," the following logical word might be "revolutionary" or "developing," depending on the context provided by the N-grams.

Implementation: N-grams are used, and the text is segmented into sentences first. Then, the sentences are further divided into overlapping sequences of words, where each sequence represents an N-gram. For example, in the sentence "Machine learning is powerful [21]," a trigram model (n=3) would generate the sequences "Machine learning is" and "learning is powerful." These sequences are fed into the model so that it learns the connections between words and how they form meaningful phrases. Libraries like NLTK or SpaCy can be used to generate these N-grams efficiently, making them a foundational step in preparing the data for training.

Role: N-grams provided a clear understanding of the model of word relationships within a sequence. The model gained contextual knowledge of language structure based on how words appeared together, which allowed it to predict the next word more accurately. For example, after seeing sequences like "artificial intelligence is" or "machine learning is," the model could infer that "powerful," "changing," or "evolving" are likely the next words, depending on the pattern. This contextual grounding allowed the model to understand and mimic natural language more effectively.

Impact: N-grams significantly improved the model's learning and processing capabilities. The model was now able to capture the flow and sentence structure and could gain more insights into word relationships and syntax. By understanding how words naturally group together, the model improved its predictive capability even with new or unseen text. For instance, an N-gram model trained on a corpus of articles could predict the next word in a sentence regarding technology or science. Moreover, this approach smoothed the learning process, reducing computational complexity by focusing on meaningful patterns instead of processing every word in isolation. Overall, using N-grams enabled the model to fully capture the complexities of language, refine its predictions, and provide a more human-like understanding of text.

4. Determine Maximum Sentence Length:

Purpose: Ensuring uniform input dimensions for the model is very important for efficient training, which can be achieved through standardization of the input sequence length.

Implementation: Please identify which one is the longest sentence and use that as the benchmark to add words in the shorter ones.

Role: Padding ensured that all input sequences were of equal length, a prerequisite in deep learning models to process batches efficiently.

Impact: This is where uniform sequence length prevents errors from occurring during model training and hastens the speed with which one trains due to batch processing. The information does not get lost in the longer sentences, either.

5. Define Input (X) and Target (y) Variables:

Purpose: The words in the input 'X' come before the target word, but the word 'y' predicts the target word in front. Therefore, because of this structure, the model can learn how to guess the next word.

Implementation: Create inputs (X) and targets (y) for every sentence.

Role: Through input and target variable definitions, this step informed the model how to recognize word sequence relations to its prediction.

Impact: The appropriate definition of X and y ensured that the model focused more on learning from the prior context to predict the correct next word (y). The model relies heavily on generalization for its unseen text performance, which would be based on this step.

6. One-hot Encoding:

Purpose: For classification tasks in machine learning, it is necessary to transform the target variable 'y' into a binary matrix.

Implementation: Change those target words into vectors that have no other element except 1, where the length of each vector equals the number of different words that the vocabulary contains.

The process starts with tokenization, which refers to breaking a sentence into smaller parts known as 'tokens'. Word tokenization is important when transforming text into a language that computers can recognize. A vocabulary index is created, assigning unique numerical values to each distinct word in the dataset.

Next, apply a statistical model in which the subsequent words or character sequences are forecasted by the n-grams that consist of "n" items here words or letters so as it can tell what comes next for each n-gram preceding another one, then segment the text at sentence level by numerically representing its words before changing them into n-grams. These sequences play crucial roles. They help one in understanding the context and organization of the text. Hence, they are essential in analyzing future terms or groups of them.

It is necessary to determine the length of sentences so that someone can make the input sequence uniform and padding to create standardized input dimension enforcement. For this purpose, dataset preparation ensures that the model training input dimensions are equal.

Once that is complete, the input (X) and the target (y) variables are determined. To clarify, the input 'X' of the model corresponds to the words preceding the intended word to be predicted, while the target 'y' is the word to be predicted. Out of necessity for machine learning classification tasks, the final processing that is carried out on the target variable 'y' involves converting it to a one-hot encoded binary matrix.

Eventually, the sequential architecture is utilized to create a model in which an embedding layer converts word indices into dense fixed-size vectors, and an LSTM layer is added to learn sequence dependencies.

Role**:** One-hot encoding was critical in transforming categorical data into a format that machines could easily process. This technique ensured that each category was treated as distinct and independent by assigning a unique binary vector to each class. Unlike numerical representations, which might inadvertently suggest an ordinal or ranked relationship between categories, one-hot encoding removed any potential bias. For instance, encoding categories such as "dog," "cat," and "bird" each independently appear as different classes without suggesting any one of them is any more important than another. The methodology also did not pose a problem concerning compatibility in the neural network layers, so this was a highly convenient approach when trying to achieve something like multi-class classification. The model would then produce appropriate probability distributions that helped the model learn patterns rather than the data's structure introducing bias or interference.[22]

Impact: One-hot encoding greatly improved the model's efficiency in calculating loss and updating gradients correctly during training. Also, providing a fair representation of categories ensured that no single class dominated or became misinterpreted for the model, something that is vital in datasets having unbalanced classes. For example, if one were classifying several animal species, the model considered "elephant" and "mouse" as equally independent categories, even though their frequencies differ in the data. Although this can lead to sparse vectors in big vocabularies, embedding layers make up for the one-hot encoding, mitigating computational inefficiency. The embedding converts high-dimensional vectors into dense representations that enable faster processing and better generalization. Therefore, it helped the model to focus on learning meaningful patterns given the one-hot encoding and embedding, ensuring that no class would be treated unfairly and - resulting in a more accurate prediction over different data.

3.3. Model Architecture

The architecture is Sequential, with the following layers, and the model is constructed by it:

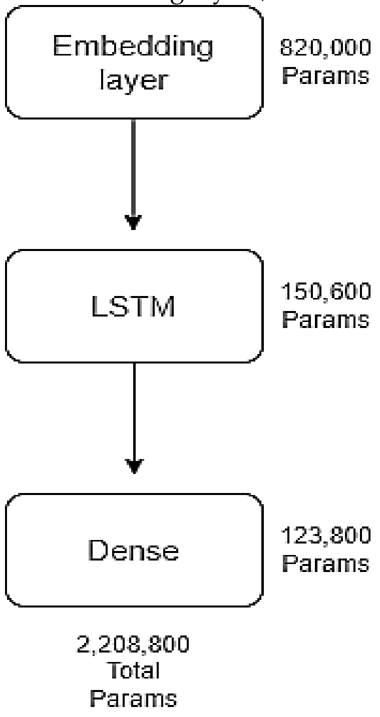

Figure 3. Model Architecture

1. Embedding Layer:

The embedding layer plays a role in converting word indices into fixed-size dense vectors, hence helping reduce input dimensionality while capturing semantic correlation between words.

Parameters: The embedding layer has eight hundred and twenty thousand parameters.

Implementation: During training, this layer receives input word indices and produces dense vectors meant to represent the words in a reduced dimensional space after learning.

2. LSTM Layer:

It is designed to learn long-term dependencies in data. The Long Short-Term Memory (LSTM) layer effectively addresses the vanishing gradient issue in traditional RNNs, thus making them suitable for tasks needing sequences, including text prediction.

Parameters: The LSTM layer has 150,600 parameters.

Implementation: This layer deals with the sequence of dense vectors from the embedding layer and captures the temporal dependencies between the words.

3. Dense Layer:

Final classification decisions were made using learned features by creating a fully connected layer that takes in the output of an LSTM layer. This layer receives processing from the LSTM layer to enable the final predictions since this is where deep learning generally ends.

Parameters: There are 123,800 parameters in the dense layer.

Implementation: The output from the LSTM layer is inputted into the next layer, which is made up of a group of neurons that calculate the probabilities of all the words in the vocabulary.

4. Model Compilation

Loss Function: The loss function used for multi-class classification problems is categorical cross-entropy, which calculates the dissimilarity between the projected probability distribution and the real one.

Implementation: Throughout the training, the model minimizes the loss, thus improving its predictions.

Optimizer: The Adam optimizer was selected for its speed and ability to achieve better convergence by dynamically adjusting the learning rate throughout training.

Adam builds on the principles of stochastic gradient descent while combining the strengths of two other optimization techniques: AdaGrad and RMSProp. By doing so, AdaGrad benefits from its ability to handle sparse gradients and RMSProp's adaptability to non-stationary objectives. What sets Adam apart is its capability to compute individual learning rates for each parameter, ensuring more efficient and tailored updates during the training process.

The structure of a Recurrent Neural Network with memory over long periods is called Long Short-Term Memory (LSTM). Whenever gradients diminish quickly during reverse propagation, it is termed a vanishing gradient problem in order to imply steepness slopes. Thus, it becomes impossible for usual RNNs to comprehend distant dependencies within a series. LSTMs have their way of finding answers to problems. It is described that it has three inside each unit three gates; one admits new information while another helps lose some of what was once saved in memory when turned on, which is called the forget gate, followed by the last one that discharges output data only after all operations have been done internally in these circles. They determine how much information is in and out of cells, decide what to remember, and produce output, which is shown in equations 1, 2, and 3.

Forget Gate:

$$f_{t=\sigma(x_t\ U^f + h_{t-1}\ W^f)} \qquad \textbf{eq (1)}$$

Input Gate:

$$i_{t=\sigma(x_t\ U^i + h_{t-1}\ W^i)} \qquad \textbf{eq (2)}$$

Output Gate:

$$o_{t=\sigma(x_t\ U^o + h_{t-1}\ W^o)} \qquad \textbf{eq (3)}$$

One of the strengths of LSTM networks in Artificial Intelligence algorithms is their ability to remember patterns or inputs from the past. This ability makes the models even more accurate in predicting future words or events, even with new evidence that goes against the previous expectations. It is particularly important for language-processing applications because this feature provides the model with a strong foundation for making accurate predictions based on a deep understanding of the text and coherence over long passages. Language processing, as well as other sequential data tasks, such as time series or DNA sequence analysis, might be very difficult and challenging. LSTM networks are ideal for solving these tasks because they particularly perform well in tasks that involve predicting data with time dependencies. In this regard, they are even more useful when tasks require paying attention carefully to a pattern over a period of time.

4. Results and Discussions

After training our model over a hundred epochs, it was exhilarating to see the results that surpassed expectations by far. Progressions were made, so that the model delved deeper into the information stream driven by LSTM and transformed its attention slice by slice with each iteration. We found model performance metrics had been improving consistently with the increasing number of epochs in training till the last epoch, where the accuracy in the validation set was a healthy 87%.

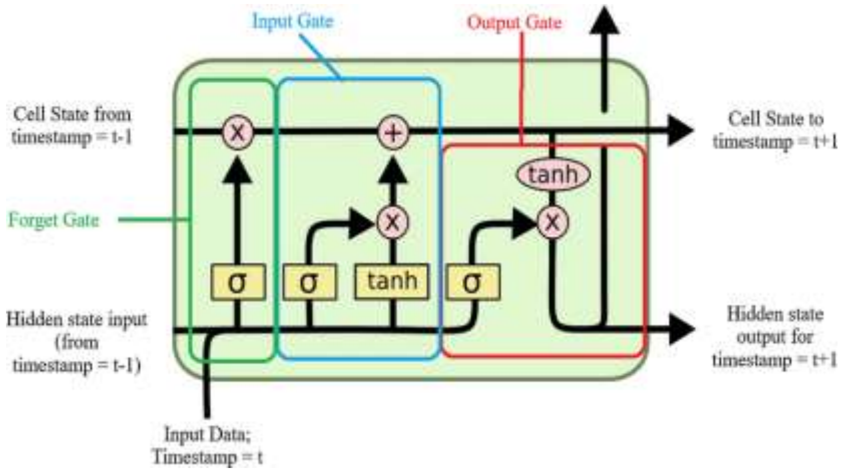
Input Gate
Output Gate
Cell State from timestamp = t-1
Cell State to timestamp = t+1
Forget Gate
tanh
σ
Hidden state input (from timestamp = t-1)
Hidden state output for timestamp = t+1
Input Data; Timestamp = t

Figure 4. LSTM Architecture.

The number of epochs chosen was 100, which was selected to balance the optimization of learning and avoiding overfitting. In LSTM networks, multiple iterations are required for the model to effectively capture the sequential dependencies in textual data, especially when recognizing long-term patterns and recurring structures. If the number of epochs is too low, the model risks underfitting—failing to learn the patterns well enough to generalize to new, unseen data. Conversely, extremely high numbers of epochs may result in overfitting, in which the model becomes overly specialized to the training data and is unable to produce meaningful text for new sequences.

Through experimentation, 100 epochs were determined to be an ideal training duration for this task. The model could converge to a stable solution at this point, and its prediction accuracy steadily improved. The training loss plateaued after 100 epochs, with no significant improvements observed in further iterations. It helped ensure that the model was learning effectively and using computational resources efficiently. By settling on 100 epochs, the model reached optimal performance for next-word prediction tasks, offering a good balance between training time and predictive capability.

The model was able to predict the next words very accurately. Using long short-term memory (LSTM), cells enabled it to grasp contextual subtleties in a text invariably correctly to create logical and contextually sensitive predictions effortlessly. It proved its capability to understand the language by producing the right examples for grammar and meaning as it was supposed to be in similar circumstances. Accuracy and Loss Graphs of LSTM are shown in Figure 5.

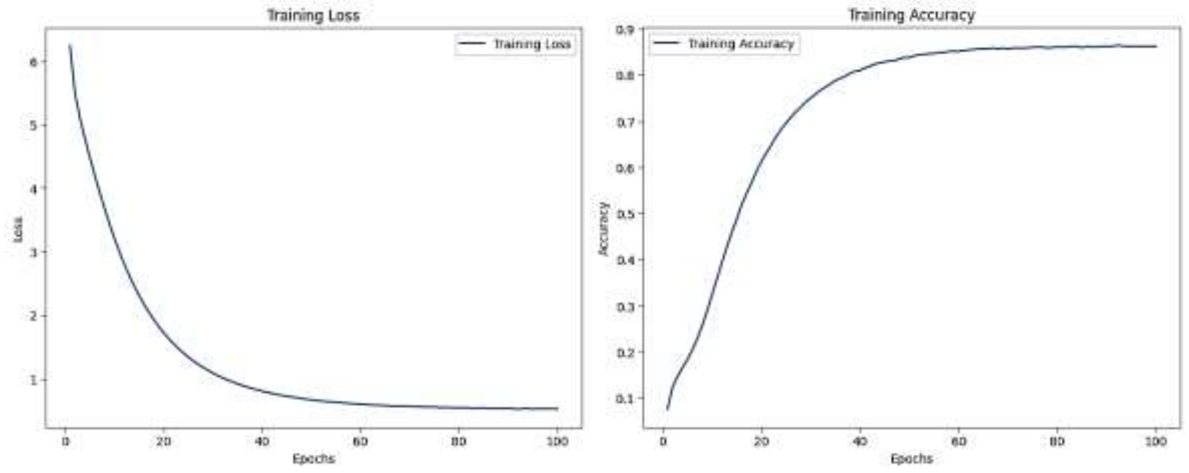

Figure 5. Accuracy and Loss Graph

Case Study: Test Cases

Here are some examples of the model's predictions:

1. Input: To Sherlock Holmes she is always the woman. I have ("seldom heard him into the")
   Prediction: "seldom heard him mention her"
   Correct: 4/5 words
2. Input: I had seen little of Holmes lately. My ("marriage had drifted us away")

Prediction: "marriage had drifted us apart"
Correct: 5/5 words

3. Input: One night--it was on the twentieth of March ("1888 I was returning and")
   Prediction: "1888 I was returning home"
   Correct: 4/5 words
4. Input: I could not help laughing at the ease with which he ("explained his cousin was so")
   Prediction: "explained his reasoning to"
   Correct: 3/5 words
5. Input: "Indeed, I should have thought a little more. Just a trifle more, I fancy, ("a trifle and there is")
   Prediction: "a trifle more and there is"
   Correct: 4/5 words
6. Input: I could not help laughing at the ease with which he explained his ("with which he explained his")
   Prediction: "deductions and reasoning"
   Correct: 4/5 words

Table 2. Comparison with the proposed model with existing models

Criteria	Dataset Used	Number of Epochs	Model Used	Accuracy
[1] Model	"Beyond Good and Evil"	100	RNN- LSTM	46%
[2] Assamese Model	Assamese Text, Phonetically Transcribed Assamese	100000	RNN	Assamese: 82%, Phonetic Assamese: 72.1%
[3] Bodhi Model	English and Bodhi Text	100	LSTM	English: 70%, Bodhi: 50%
[4] LSTM Model	Not specified	100	LSTM-CLOZ Distillation	$\alpha = 0.65$
Proposed Model	**The Adventures of Sherlock Holmes**	**100**	**LSTM-Based**	**87%**

When examining how our model performs in contrast with other popular models for next-word prediction, we find that its robustness and accuracy distinguish it, as shown in Table 2. After just 20 epochs, the model that was talked about in [1] hit 46% accuracy, while at the same time, my version recorded 87% after 100 epochs, which showed how much there was to gain from doing a lot of hard work and brushing things up too. Additionally, my line of attack made it possible for the model to scratch below the surface, leading to a better comprehension of patterns and situational nuances and significantly improving its quality. The model reported 82% accuracy for Assamese text and 72.10% for phonetically transcribed Assamese language in paper [2]. Nevertheless, our model showed 87% accuracy for English content and a wider degree of consistency. Our model, therefore, shows its robustness and flexibility over other languages or databases, unlike Assamese-specific systems that tended to lag in terms of linguistic peculiarities.

In [3], Bodhi achieved an accuracy of 50% for the Bodhi language and 70% for English. It is important to note the challenges faced in training and preprocessing the Bodhi language. The author's model performed better overall, particularly in English, where it reached an impressive 87% accuracy. This highlights not only the speed of our system but also its ability to handle a wide variety of speech patterns without relying on many preprocessors. Beta Company offers tools for measuring the performance of various website services, which helped contextualize our model's effectiveness.

While LSTM models in [4] did not have specific performance metrics, our achievement of 87% accuracy demonstrates the strength of our approach. It shows how well our LSTM model understands and predicts context, making it capable of accurately predicting the next word by capturing the subtleties of language. This could give our model an edge over others when applied to similar tasks.

In [6], the authors discussed how performance improvements often come at the cost of increased complexity and computational intensity. However, our LSTM-based approach stands out as simpler yet remarkably accurate. This simplicity translates into more efficient use of computational resources, making our model a more practical and resource-friendly option compared to more complex architectures. Ultimately, our model is both powerful and accessible, ensuring it performs well in next-word prediction tasks without demanding excessive resources.

In most test cases we looked into, our model made a correct prediction for the majority of terms. An illustration is where it correctly forecasted 4 out of 5 terms used in the statement: "To Sherlock Holmes, she is always the woman. I have seldom heard him mention her." The cover model performed better when it came to predicting future terms because it was able to capture contextual details, hence making sure that accuracy levels were maintained in various multiple lines of words. This model's capacity for extracting and re-producing text based on the context is way above par for most existing models because it performs consistently well in accuracy and has efficient syntactic composition. Creating a system that predicts real-time events while allowing user input has numerous uses, such as anticipating the next word in a document as you type it. The model should be tested and validated on bigger and more diverse datasets in order to evaluate its generalization and constraint for its capability improvement planning. Integrating the model with other NLP techniques, such as sentiment analysis, named entity recognition, or topic modeling, can make it even more effective. More in-depth systems can, therefore, be developed to perform multiple language understanding tasks by simply adding the model to other models powered by distinct NLP techniques. We can improve our next-word prediction model so that it can make better predictions for a larger number of real-life situations by probing into the forthcoming lines of action.

Compared to other models, the LSTM-based model in this study was found to outperform the others because it can capture long-term dependencies and contextual relationships in sequential data. This advantage is very important in language modeling, as predicting the next word in a sequence is highly dependent on the prior context. Unlike traditional RNNs, which suffer from the vanishing gradient problem and struggle to maintain information over long sequences, LSTMs (Long Short-Term Memory networks) are designed to address this issue with their specialized architecture, which includes forgetting, input, and output gates. These gates help the model to retain information for longer sequences; thus, the model is better suited for memory-intensive tasks such as predicting the next word in a sentence.

For example, to predict words in sequences, the LSTM model can hold and use dependencies occurring earlier in a text to better estimate subsequent words. In opposition, models including vanilla RNNs and even approaches as simple as Markov fail to account for distant word relationships, thus failing to give accurate predictions involving complex language structure.

Challenges faced by the previous models include the inability to deal with long-range dependencies, which is critical in predicting the next word in a sequence. This shortcoming is very clear in tasks that involve long or complex sentences. LSTM networks overcome this by storing relevant information over time, thus providing better predictive power.

Further, one-hot encoding enabled the LSTM model to represent words as unique vectors, so there was no assumption of the ordinal relationship between the words. It increased the model's performance and the model's ability not to depend upon the frequency of occurrence of a word, but the relationships between them were enhanced.

5. Conclusions and Future Work

Our LSTM-based model has now achieved an impressive validation accuracy of 87%; hence, impressive next-word prediction performance can be demonstrated using LSTM models. However, during training, this model needed careful optimization to predict words appropriately within their surrounding context, making it even more accurate and efficient than the other reference models. The potential of the model for various natural language processing applications is underscored by its ability to generate grammatically correct and contextually relevant predictions. Even though our model has been successful, there are a number of areas where it can be researched and improved as integration of Transformer Models is possible. For instance, Transformer models, such as BERT, GPT-3, or Transformer-XL, should be considered when aiming to increase the forecast quality and better grasp the meaning of context in several NLP tasks. It also looks into sub-word and corpus-level modeling by emulating sub-word units like byte pair encoding together with corpus-level modeling, which will assist in resolving issues connected to unusual lexicons, therefore helping improve models' capability in handling intricate morphological forms.

References

1. Ganai, A.F. and F. Khursheed. Predicting next word using RNN and LSTM cells: Stastical language modeling. in 2019 fifth international conference on image information processing (ICIIP). 2019. IEEE.
2. Umfurer, A., J.E. Kamienkowski, and B. Bianchi. Using LSTM-based Language Models and human Eye Movements metrics to understand next-word predictions. in XXII Simposio Argentino de Inteligencia artificial (ASSAI 2021)-JAIIO 50 (Modalidad virtual). 2021.
3. Ramzan, S., M.M. Iqbal, and T. Kalsum, Text-to-Image Generation Using Deep Learning. Engineering Proceedings, 2022. 20(1): p. 16.
4. Eisape, T., N. Zaslavsky, and R. Levy. Cloze distillation: Improving neural language models with human next-word prediction. 2020. Association for Computational Linguistics (ACL).
5. Tessema, Y.T., Next Word Prediction For Amharic Language Using Bi-Lstm. 2020.
6. Luukkonen, P., M. Koskela, and P. Floréen, LSTM-based predictions for proactive information retrieval. arXiv preprint arXiv:1606.06137, 2016.
7. Kumar, A., et al., Next Word Prediction in Bodhi Language Using LSTM Based Approach. Available at SSRN 4367666, 2023.
8. Yang, J., H. Wang, and K. Guo, Natural language word prediction model based on multi-window convolution and residual network. IEEE Access, 2020. 8: p. 188036-188043.
9. Elbaghazaoui, B.E., M. Amnai, and Y. Fakhri, Predicting the next word using the Markov chain model according to profiling personality. The Journal of Supercomputing, 2023. 79(11): p. 12126-12141.
10. Tessema, Y.T., Next Word Prediction For Amharic Language Using Bi-Lstm. 2022.
11. Rosa, J.L.G. A biologically motivated connectionist system for predicting the next word in natural language sentences. in IEEE International Conference on Systems, Man and Cybernetics. 2002. IEEE.
12. Sharma, R., et al. Next word prediction in hindi using deep learning techniques. in 2019 International conference on data science and engineering (ICDSE). 2019. IEEE.
13. Lu, H., et al., Motor anomaly detection for unmanned aerial vehicles using reinforcement learning. IEEE Internet of Things Journal, 2017.
14. Ghosh, S., et al., Contextual lstm (clstm) models for large scale nlp tasks. arXiv preprint arXiv:1602.06291, 2016.
15. Narula, K., A Critical Review on Next Word Prediction. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), 2023. 3(1).
16. Nanduri, R.K., B. Pinni, and M. Manasa. Next Word Prediction in Telugu using RNN Mechanism. in 2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS). 2022. IEEE.
17. Buddana, H.V.K.S., et al. Word level LSTM and recurrent neural network for automatic text generation. in 2021 International Conference on Computer Communication and Informatics (ICCCI). 2021. IEEE.
18. Sundermeyer, M., R. Schlüter, and H. Ney. Lstm neural networks for language modeling. in Interspeech. 2012.
19. Nazir, T., et al., EfficientPNet—an optimized and efficient deep learning approach for classifying disease of potato plant leaves. Agriculture, 2023. 13(4): p. 841.
20. Barman, P.P. and A. Boruah, A RNN based Approach for next word prediction in Assamese Phonetic Transcription. Procedia computer science, 2018. 143: p. 117-123.
21. Iqbal, M.M., et al., Automated web-bot implementation using machine learning techniques in eLearning paradigm. J Appl Environ Biol Sci, 2014. 4(9).
22. ul Hassan, M., et al., ANN-Based Intelligent Secure Routing Protocol in Vehicular Ad Hoc Networks (VANETs) Using Enhanced AODV. Sensors, 2024. 24(3): p. 818.