LSTM BASED NEXT WORD PREDICTION

Sai Soundharya Lakshmi B, Paavendhan K S

Department of Computer Technology
Madras Institute of Technology, Anna University

***Abstract*—Language prediction is a Natural Language Processing (NLP) application concerned with predicting the text given in the preceding text. Auto-complete or suggested responses are popular types of language prediction. Next word prediction, also known as language modelling, is a Natural Language Processing tool that can help forecast the next word. Earlier, some studies built the model using various approaches such as N-gram models, Federated text models etc., To make the predictions, each study employed their own model. Our objective is to develop the model using Long short-term memory (LSTM) to predict 5 keywords or the upcoming characters in the fastest possible time. We try to create a model using default text data that will predict the sentence that the user has typed. Once the user has entered 40 characters, the model will comprehend those 40 characters and anticipate the next top 5 words employing RNN architecture, which will be implemented using Python libraries Tensorflow and Keras. LSTM can interpret previous learning and predict words that can assist the user frame sentences.**

I. INTRODUCTION

Natural language processing (NLP) is a branch of artificial intelligence (AI) that focuses on natural language interaction between computers and humans. It involves teaching computers to comprehend, interpret, and generate human language, which includes both spoken and written communication. Next word prediction is a type of language modelling in which machine learning techniques are used to predict the most likely next word in a string of words. Natural language processing applications such as virtual assistants, chatbots, and text messaging make extensive use of this technology. There are various techniques used for next word prediction, including Markov models, n-gram models, and recurrent neural networks (RNNs). RNNs are particularly effective for next word prediction because they can take into account the entire sequence of words leading up to the current position in the text.

In this project, to build the model we specifically employ LSTM which is a type of recurrent neural network (RNN) that is particularly effective at processing and predicting sequences of data. Unlike traditional RNNs, which suffer from the problem of vanishing gradients, LSTM networks are able to retain information over longer periods of time, making them well-suited for tasks such as speech recognition, language modeling, and machine translation. LSTMs achieve this by incorporating specialized memory cells that are capable of selectively retaining or discarding information based on the input and context. The cells are connected through gates, which control the flow of information into and out of the cells. This gating mechanism allows LSTMs to selectively process and remember relevant information while filtering out noise and irrelevant data.

LSTMs have proven to be a powerful tool in natural language processing, image and speech recognition, and other applications where sequences of data need to be processed and analyzed. They have contributed to significant advances in the field of deep learning and continue to be an important area of research and development.

Rest of this paper is organized as follows. In Section II, the problem statement is described briefly. Existing problems in the current models are highlighted. In Section III, the objectives of our next word prediction model to overcome the current challenges is listed. Next, in the Section IV algorithm flow of the model is clearly explained. Section VI presents the implementation of the next word prediction model. It includes the dataset details, language and the tool that we used to build the model. Section VII presents the results achieved and the inferences made building the model. Future scope of the next word prediction model is listed in the Section VIII.

II. PROBLEM STATEMENT

The goal of a next word prediction model is to predict the most likely next word in a text sequence based on the context and preceding words. This is a challenging area to handle since human language is complicated and dynamic, with many distinct elements influencing the next word choice. The model must be able to comprehend the context of the spoken or written content, including the topic, tone, and intended audience. It also needs to be able to identify textual trends, such as the repetition of particular phrases and the likelihood of specific word combinations.

Hence the overall purpose of a next word prediction model is to produce accurate and contextually relevant predictions that can increase the efficiency and efficacy of various natural language processing applications.

Many deep learning algorithms had previously been used in creating the prediction model, however typical RNNs and other language models become less accurate when the distance between the context and the word to be predicted grows.

So we aim to develop the model based on LSTM which is used to tackle the long-term dependency problem because it has memory cells to remember the previous context. LSTM model uses Deep Learning with a network of artificial “cells” that manage memory, making them better suited for text prediction than traditional neural networks and other models.

III. OBJECTIVES

The primary objective of a next word prediction model is to improve the user experience of natural language processing applications, such as chatbots, virtual assistants, and mobile keyboards, by providing faster and more accurate predictions of the next word. By predicting the next word, users can type or speak more quickly and efficiently, allowing them to be more productive in various tasks, such as writing emails, composing documents, or texting. The model also enhances accuracy of natural language processing applications by reducing the number of mistakes and errors made by users. Predicting the next word can help reduce the cognitive load on users, particularly those with cognitive or motor impairments, by minimizing the effort required to type or speak. Next word prediction can enable new applications, such as language learning tools, content creation tools, and speech recognition systems, by providing contextually relevant suggestions for the next word. Overall, the objectives of a next word prediction model are to make natural language processing applications more efficient, effective, and user-friendly, while also enabling new applications that were previously not possible. Now we are building a model with some specific objectives mentioned as follows:

- Our main objective is to develop an LSTM based word prediction model that predict the most likely next word/string in a text sequence based on the context and preceding five words (40 characters).
- Another goal is to generate precise and contextually appropriate predictions that can boost the performance of different natural language processing applications.
- To avoid long-term dependency problem using LSTM since it has feedback connections, capable of processing the entire sequence of data, apart from single data points.

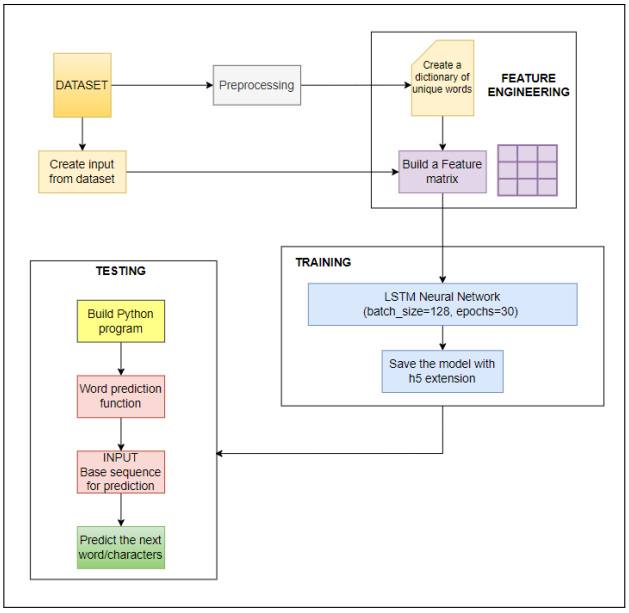

Fig. 1. Overall flow of algorithm

IV. ALGORITHM

The first step is to load the dataset and do the preprocessing by splitting the dataset into each word in order but without the presence of some special characters. Now the next process will be performing the feature engineering in our data. Feature Engineering means taking whatever information we have about our problem and turning it into numbers that we can use to build our feature matrix. For this purpose, we will require a dictionary with each word in the data within the list of unique words (Figure. 2) as the key, and it's significant portions as value.

```
chars = sorted(list(set(text)))
char_indices = dict((c, i) for i, c in enumerate(chars))
indices_char = dict((i, c) for i, c in enumerate(chars))

print ("unique chars: ",len(chars))
```

```
unique chars:  73
```

Fig. 2. Data preprocessing

Next we define the word length which will represent the number of previous words that will determine our next word. Prev words array is defined to keep five previous words and their corresponding next words in the list of next words. We are converting the texts to sequences. This is a way of interpreting the text data into numbers so that we can perform better analyses on them.

We will then create the training dataset. The 'X' will contain the training data with the input of text data. The 'y' will contain the outputs for the training data. So as shown in the Figure. 3, the 'y' contains all the next word predictions for each input 'X'.

```
X = np.zeros((len(sentences), SEQUENCE_LENGTH, len(chars)), dtype=bool)
y = np.zeros((len(sentences), len(chars)), dtype=bool)
for i, sentence in enumerate(sentences):
    for t, char in enumerate(sentence):
        X[i, t, char_indices[char]] = 1
    y[i, char_indices[next_chars[i]]] = 1
```

Fig. 3. X-Training data, y-outputs for the training data

Next we used the Recurrent Neural networks for the next word prediction model. Here we considered the LSTM model, which is a very powerful RNN. We will be building a sequential model. We will then create an embedding layer and specify the input dimensions and output dimensions. It is important to specify the input length as 40 since the prediction will be made on exactly a sequence of 40 characters and we will receive a corresponding response for that. We will then add an LSTM layer to our architecture. We will pass this through a hidden layer using the dense layer function with relu set as the activation. Finally, we pass it through an output layer with the softmax activation. The model summary and the architecture are shown in the Figure. 4 and 6.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 128)               103424

 dense (Dense)               (None, 73)                9417

 activation (Activation)     (None, 73)                0

=================================================================
Total params: 112,841
Trainable params: 112,841
Non-trainable params: 0
_________________________________________________________________
```

Fig. 4. Model summary

We trained the model with 30 epochs and a batch size of 128. The learning rate is set to 0.01. Since the dataset is small, a small number of epochs may be sufficient because the model can quickly learn all the relevant patterns in the data. If the model is trained for too long on a small dataset, problem of overfitting occurs.

```
optimizer = RMSprop(learning_rate= 0.01)
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])

history = model.fit(X, y, validation_split=0.05, batch_size=128, epochs=30, shuffle=True).history
```

Fig. 5. Model training

Next for testing the model we built a python program to predict the next word using our trained model and some essential functions are defined as a part of the program that will be used in the process of prediction. Next word prediction function is created to predict the next word until space is generated. It will do this by iterating the input, which will ask our RNN model and extract instances from it. Sequence of 40 characters had been used as a base for our predictions. Finally the model can be used for prediction. In the Figure. 4, the overall flow of the algorithm starting from preprocessing to final prediction has been represented.

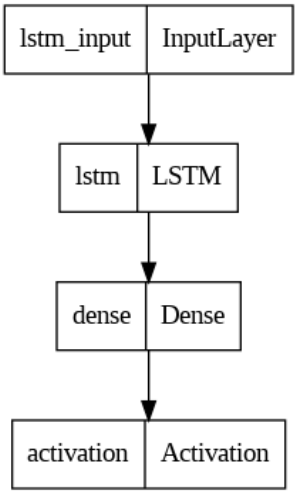

Fig. 6. Model architecture

V. SOFTWARE ARCHITECTURE

The software architecture of our next word prediction model as shown in the Figure. 7 typically consists of the following components:

Data preprocessing: This component involves cleaning and processing the raw text data, such as removing punctuation, tokenizing the text into individual words, and converting the text into a numerical format that can be processed by the model.

Language model: This component is responsible for learning the statistical relationships between words in the training data and predicting the next word based on the context of the previous words. A common approach for this component is to use a neural network, such as a recurrent neural network (RNN) or a long short-term memory (LSTM) network. We have built the model based on LSTM.

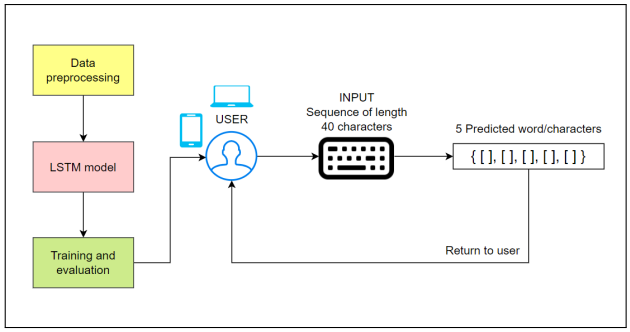

Fig. 7. Software Architecture Diagram

User interface: This component provides a user-friendly interface for users to input text and view the predicted next word. This component may be integrated into various applications, such as mobile keyboards, chatbots, and virtual assistants. In our project we have not built any particular user interface, instead using the simple python program we get input from the user (Figure. 8), until the user enters 0 after which the execution i.e. prediction will be completed.

```
while(True):
    text = input("Enter your line: ")

    if text == "0":
        print("Execution completed...")
        break

    else:
        seq = text[:40].lower()
        print(seq)
        print(predict_completions(seq, 5))
        print()

... Enter your line:
```

Fig. 8. User input

Training and evaluation: This component involves training the language model on a dataset of text data and evaluating its performance on a separate test set. This process may

involve fine-tuning the model parameters, such as the number of layers and neurons in the neural network, to optimize its performance.

Overall, the software architecture of our next word prediction model is designed to process text data present in the ebook dataset, learn the statistical relationships between words, and provide contextually relevant predictions.

VI. IMPLEMENTATION

The dataset contains 1,07,794 words from Project Gutenberg's ebook The Adventures of Sherlock Holmes, by Arthur Conan Doyle (Figure. 9). Project Gutenberg is a library of over 70,000 free eBooks in the public domain. Project Gutenberg's collection includes classic literature, historical documents, and scientific publications, as well as works in multiple languages. The books are available in various digital formats, including epub, kindle and pdf, and can be downloaded or read online for free. In addition to its e-book collection, Project Gutenberg also provides tools and resources for creating and publishing e-books, including guidelines for formatting and copyright information. From here we can get many stories, documentations, and text data which are necessary for our problem statement.

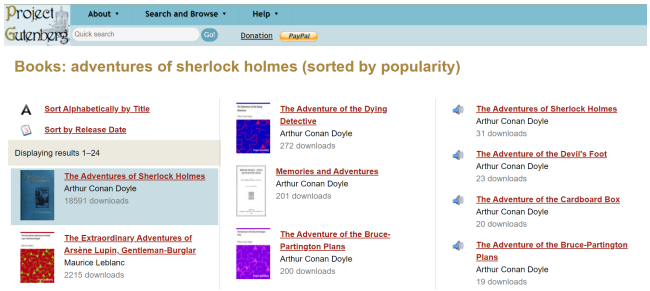

Fig. 9. Our dataset from Project Gutenberg's website

We particularly chose the ebook "The Adventures of Sherlock Holmes" as our dataset. Python libraries such as tensorflow, keras, numpy, and matplotlib are used to implement the model. Google Colaboratory allows developers to write and execute Python code through their browser. Google Colab is an excellent tool for deep learning tasks. It is a hosted Jupyter notebook that requires no setup and has an excellent free version, which gives free access to Google computing resources such as GPUs and TPUs. Overall Google Colab is just a specialized version of the Jupyter Notebook, which runs on the cloud and offers free computing resources. So, Google Colab has been used as the environment for coding and building our model.

VII. RESULTS AND INFERENCES

In this section, we will evaluate the performance of the proposed model. We created a model using an ebook text file as the dataset which will predict the user's sentence after the user has typed 40 letters. The model will understand 40 characters and predict upcoming words/characters using the LSTM neural network that has been implemented using Keras and Tensorflow. The input test cases fed to the model and the corresponding predictions are shown in the Figure.10.

For each of the input entered by the user, the prediction model takes only the first 40 characters including the spaces and predicts the next word. For example, the sentence "I could not help laughing at the ease with which he" has been given as the input and the model considers only the first 40 characters and predicted five strings [th, ndows, ll, sh, fe] that could come after 'wi'. The input test cases shown in the Fig. 10 are taken from our ebook text file dataset and out of five predictions we got four correct.

```
Enter your line: As I passed the well-remembered door, which must always be associated in
as i passed the well-remembered door, wh
['o ', 'ich ', 'en ', 'at ', 'y ']

Enter your line: and clearing up those mysterie which had
and clearing up those mysterie which had
[' been ', '\na ', 'e ', ', ', 's ']

Enter your line: I rang the bell and was shown up to the chamber which had formerly
i rang the bell and was shown up to the
['station ', 'confesse ', 'bedroom. ', 'lady ', 'facts ']

Enter your line: Obviously they have been caused by someone who has very carelessly scraped round the
obviously they have been caused by someo
['ne ', 'ld ', 'tent ', '-deen ', 'dvtably. ']

Enter your line: I could not help laughing at the ease with which he
i could not help laughing at the ease wi
['th ', 'ndows ', 'll ', 'sh ', 'fe ']

Enter your line: 0
Execution completed...
```

Fig. 10. Input and corresponding output

The accuracy graph for the model has been shown in the Figures 11. We can see that the model has achieved an accuracy of around 61% with 30 epochs. Generally, a larger number of epochs allows the model to learn more complex patterns and improve its accuracy on the training data. Since the dataset we have chosen is actually small, a less number of epochs may be sufficient because the model can quickly learn all the relevant patterns in the data.

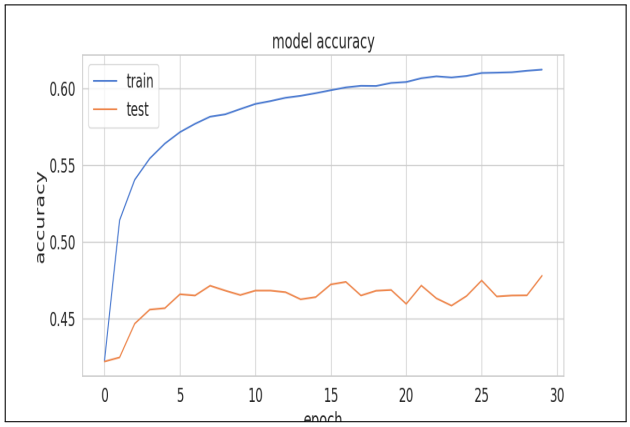

Fig. 11. Accuracy on training and testing data

This paper presents how the system is predicting and correcting the next/target words using LSTM. The next word prediction model which we have developed is fairly accurate on the provided dataset. The overall quality of the prediction is good. However, certain pre-processing steps and certain changes in the model can be made to improve the prediction of the model.

VIII. FUTURE SCOPE

This next word prediction model has more scope on social media for syntax analysis and semantic analysis and has many practical applications in natural language processing, including:

- Next word prediction is commonly used in mobile devices and computer keyboards to suggest the most likely next word as users type, allowing for faster and more accurate input.
- Chatbots use next word prediction to generate contextually relevant responses to user input in real-time, creating a more natural and engaging conversation experience.
- Next word prediction can improve the accuracy and speed of machine translation by predicting the most likely next word in the target language based on the context of the source language.
- Voice assistants such as Amazon Alexa, Google Home, and Apple Siri use next word prediction to improve speech recognition accuracy and generate more natural-sounding responses.
- Next word prediction can be used to aid in language learning by providing contextually relevant suggestions for the next word, helping learners build vocabulary and improve their writing and speaking skills.
- Next word prediction can assist in content creation by suggesting the most likely next word, making the writing process faster and more efficient.

Overall, next word prediction is a powerful tool in natural language processing that can enhance user experience, improve efficiency, and enable new applications.

REFERENCES

[1] A. Rianti, S. Widodo, A. D. Ayuningtyas, and F. B. Hermawan, "NEXT WORD PREDICTION USING LSTM," JOURNAL OF INFORMATION TECHNOLOGY AND ITS UTILIZATION, VOLUME 5, ISSUE 1, JUNE-2022.

[2] A. Tiwari, N. Sengar and V. Yadav, "Next Word Prediction Using Deep Learning," 2022 IEEE Global Conference on Computing, Power and Communication Technologies (GlobConPT), New Delhi, India, 2022, pp. 1-6, doi: 10.1109/GlobConPT57482.2022.9938153.

[3] M. Soam and S. Thakur, "Next Word Prediction Using Deep Learning: A Comparative Study," 2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2022, pp. 653-658, doi: 10.1109/Confluence52989.2022.9734151.

[4] Ambulgekar S., Sanket Malewadikar, Raju Garande, and Bharti Joshi. "Next Words Prediction Using Recurrent NeuralNetworks." In ITM Web of Conferences, vol. 40, p. 03034. EDP Sciences, 2021.

[5] Stremmel, Joel, and Arjun Singh. "Pretraining federated text models for next word prediction." In Future of Information and Communication Conference, pp. 477-488. Springer, Cham, 2021.

[6] J. Yang, H. Wang and K. Guo, "Natural Language Word Prediction Model Based on Multi-Window Convolution and Residual Network," in IEEE Access, vol. 8, pp. 188036-188043, 2020, doi: 10.1109/ACCESS.2020.3031200.

[7] A. F. Ganai and F. Khursheed, "Predicting next Word using RNN and LSTM cells: Stastical Language Modeling," 2019 Fifth International Conference on Image Information Processing (ICIIP), Shimla, India, 2019, pp. 469-474, doi: 10.1109/ICIIP47207.2019.8985885.

[8] S. Sarker, M. E. Islam, J. R. Saurav and M. M. H. Nahid, "Word Completion and Sequence Prediction in Bangla Language Using Trie and a Hybrid Approach of Sequential LSTM and N-gram," 2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT), Dhaka, Bangladesh, 2020, pp. 162-167, doi: 10.1109/ICAICT51780.2020.9333518.

[9] O. F. Rakib, S. Akter, M. A. Khan, A. K. Das and K. M. Habibullah, "Bangla Word Prediction and Sentence Completion Using GRU: An Extended Version of RNN on N-gram Language Model," 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), Dhaka, Bangladesh, 2019, pp. 1-6, doi: 10.1109/STI47673.2019.9068063.

[10] J. Yang, H. Wang and K. Guo, "Natural Language Word Prediction Model Based on Multi-Window Convolution and Residual Network," in IEEE Access, vol. 8, pp. 188036-188043, 2020, doi: 10.1109/ACCESS.2020.3031200.

[11] N. T. K. Naulla and T. G. I. Fernando, "Predicting the Next Word of a Sinhala Word Series Using Recurrent Neural Networks," 2022 2nd International Conference on Advanced Research in Computing (ICARC), Belihuloya, Sri Lanka, 2022, pp. 13-18, doi: 10.1109/ICARC54489.2022.9754174.

[12] S. Singh, "On-Device User-Adaptive Next Word Prediction System," 2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA), Gunupur, India, 2022, pp. 1-4, doi: 10.1109/ICCSEA54677.2022.9936158.

[13] M. Habib, M. Faris, R. Qaddoura, A. Alomari and H. Faris, "A Predictive Text System for Medical Recommendations in Telemedicine: A Deep Learning Approach in the Arabic Context," in IEEE Access, vol. 9, pp. 85690-85708, 2021, doi: 10.1109/ACCESS.2021.3087593.

[14] S. Sarker, M. E. Islam, J. R. Saurav and M. M. H. Nahid, "Word Completion and Sequence Prediction in Bangla Language Using Trie and a Hybrid Approach of Sequential LSTM and N-gram," 2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT), Dhaka, Bangladesh, 2020, pp. 162-167, doi: 10.1109/ICAICT51780.2020.9333518.

[15] K. Terada and Y. Watanobe, "Code Completion for Programming Education based on Recurrent Neural Network," 2019 IEEE 11th International Workshop on Computational Intelligence and Applications (IWCIA), Hiroshima, Japan, 2019, pp. 109-114, doi: 10.1109/IWCIA47330.2019.8955090.

[16] B. Tarján, G. Szaszák, T. Fegyó and P. Mihajlik, "N-gram Approximation of LSTM Recurrent Language Models for Single-pass Recognition of Hungarian Call Center Conversations," 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), Naples, Italy, 2019, pp. 131-136, doi: 10.1109/CogInfoCom47531.2019.9089959.

[17] Stremmel, J., & Singh, A. (2021, April). Pretraining federated text models for next word prediction. In Future of Information and Communication Conference (pp. 477-488). Springer, Cham.