Exploring LLM and Neural Networks Towards Malicious Prompt Detection

Himani Deshpande
Department of Artificial Intelligence & Data Science
Thadomal Shahani Engineering College
Mumbai, India
himani.deshpande@thadomal.org

Dhawal Chaudhari
Department of Artificial Intelligence & Data Science
Thadomal Shahani Engineering College
Mumbai, India
dhawalc1901@gmail.com

Tanmay Sarode
Department of Computer Science
Thadomal Shahani Engineering College
Mumbai, India
tanmaysarode2005@gmail.com

Ayush Kamath
Department of Computer Science Thadomal Shahani
Engineering College
Mumbai, India
ayushkamath123@gmail.com

Abstract*—At the forefront of the Artificial Intelligence Revolution is the Generative AI domain which is making splashes in generation of new content from existing Large Language Models. Large Language Models (LLMs) are flexible, effective tools with many uses. On the other hand, they can be tricked by devious provocation. This work explores the detection of fraudulent prompts intended to produce false or damaging outputs using Long Short-Term Memory (LSTM) networks. During this study we analysed the LTSM model against a neural network model. We found a 91.69% accuracy for the LTSM model and a 92.52% accuracy for the simple neural network. However, the simple neural network had a higher F1 score of 0.73 compared to the LTSM score of 0.69. We found that the standard neural model is more efficient at identifying harmful user prompts. Our research highlights the need for taking into account various architectures of models to reduce the risk of malicious prompting in language learning models.

***Keywords*—Malicious prompting, Neural Networks, Long Short Term Memory Models, Generative AI.**

I. INTRODUCTION

With the introduction of LLMs it has become crucial to find measures to prevent the misuse of various models by using misdirecting and malicious prompting. These models have shown a remarkable ability to generate contextually relevant text by using a simple prompt. They have found many uses in advanced research assistance, automatic customer support and so on. Despite their exceptional performance, there arises a critical need to address exploitation of these models through malicious prompting. Malicious prompting can be defined as creating specific and misleading inputs which in turn exploit the potential of Large Language Models to generate harmful, misleading or unethical outputs. This type of manipulation turns out to be quite hazardous as it can manifest in various forms like producing misinformation, giving heavily biased narratives or even generating offensive content.
Deep Learning is being used in this study to implement LSTMs to detect malicious prompts. It is a subset of machine learning that employs neural networks to extract detailed information from raw input. The term "deep" comes from multiple-layered neural networks, which are commonly employed in deep learning. Deep learning contains various types of Neural Networks such as Convolutional Neural Networks used for image processing, Recurrent Neural Networks used to handle sequential data, Transformers which handle sequential data but with parallelizable potential making them ideal for large datasets and Generative Adversarial Networks which are used for image generation.
Neural Networks are a class of models which have been inspired by the way the human brain and neurons work. They consist of multiple interconnected layers of neurons, where each neuron performs some particular operation on the input data and passes it to the next layer. As a result, they were found to be highly effective in recognizing patterns within data and handling advanced tasks like computer vision and speech recognition.A neural network consists of an input layer, hidden layers and an output layer. The neurons have activation functions like sigmoid or Relu or Sigmoid functions which introduce non-linearity to the model. The main goal of a neural network is to minimize the loss function and find a proper trade-off between variance and bias.
RNNs or Recurrent Neural Networks are specialized versions of the Neural networks used in sequential Data. In RNN the previous output is also used to predict the current out making it different from the vanilla Neural Networks reducing the complexity in the parameters as compared to the neural networks.
A promising approach to detect these malicious prompts is by using Long Short-Term Memory(LSTM) networks.
LSTMs are an advanced variant of Recurrent Neural Networks and ameliorate them by countering the vanishing and exploding gradient problem. They do so by using long-term memory which can capture long-term dependencies and as a result, be used to analyze input prompts and detect malicious intent. This capability is crucial for developing robust and safe systems that can flag any malicious content before they are able to influence the output. This paper aims to construct a model using LSTMs to detect malicious prompts and compare it with a simple neural network model

II. LITERATURE REVIEW

A study conducted on 78 real world prompts classified into 10 categories proved that jailbreak prompts done on ChatGPT can bypass the restrictions imposed by OpenAI. Furthermore, analysis proved that prompts have become more sophisticated and effective. The study also provided solutions for preventing jailbreaks [1]. Another study introduced HOUY1 a black box methodology to expedite the prompt injection process on different LLMs to see whether they are susceptible to prompt injection attacks or not. Their study showed that 31 out of the 36 LLMs are vulnerable to prompt injection attacks [2].

Another study presents the challenges, opportunities and limitations Generative AI offers in the cybersecurity paradigm. This study showed how ChatGPT is affected by common jailbreaking techniques to bypass its ethical and privacy safety guards. It also shows how ChatGPT can be used to unleash various attacks demonstrating use of GenAI in cyber offense. It also illustrates several open challenges and problems pertinent to cybersecurity[3]. Another study conducted in Hong Kong proposed an evaluation framework for judging the resilience of LLM-integrated applications to prompt injection attacks. The study revealed that newer applications are more resilient to such attacks [4].

A study conducted in the Hong Kong Polytechnic University proposed a defence strategy against prompt injection attacks called the 'Signed-Prompt' method which is successfully able to differentiate between genuine user commands and commands from signed attacker-derived prompts, thereby preventing prompt injection attacks [5]. Another study conducted introduced a momentum-enhanced optimization algorithm for solving challenges posed by unclear prompt injection attack objectives [6].

A study conducted introduced the FuzzLLM, a new and universal framework to unearth jailbreak vulnerabilities in Large Language models or LLMs by using fuzzing techniques. It achieves this by leveraging templates that merge jailbreak constraints with prohibited questions [7]. Another study performed an excellent analysis on the effectiveness of different types of jailbreaks on various LLMs while also discussing the limitations and the challenges of current methods for mitigating jailbreaks [8].

A study demonstrated how ChatGPT can be jailbroken to perform cyberattacks and also showed how prompt engineering can aid in AI-powered attacks by using the DAN prompt to remove ChatGPT's ethical boundaries[9].
A study conducted in China proposed a novel approach for the detection of malicious web pages using LLMs or large language models by analyzing the web page content to improve detection performance without requiring large amounts of training data[10].

III. METHODOLOGY

The dataset used for the model is an amalgamation of a custom created dataset, along with the consisting of the prompt and whether it is a malicious prompt or not. In order to make the data compatible for the model, columns irrelevant to the model were removed. For additional preprocessing, the pandas library was used to create the respective data frames used for training. An 80:20 train test split was done on the dataset. Then for the embedding of the prompts, the max_words was set to 10000 and the max_length to 100.

1. LSTM Model:

The proposed model implements the use of the very well-known LSTM model also known as Long Short term memory. LSTMs are a subset of the Recurrent Neural Networks or the RNNs. The Recurrent Neural Networks are machine learning algorithms which are used when we are dealing with sequential data. Instead of using a single input and letting the algorithm predict the output, in RNNs we use multiple inputs and feed them through the RNN network using the same weights and biases in turn using the previous input for predicting the current output.
One major drawback to the base RNNs is the vanishing/exploding gradient problem which makes the algorithm inefficient in predicting accurate values. The exploding gradient problem occurs when the weights are set to be greater than 1. Due to this the original input gets exponentially larger losing its original value. The vanishing gradient problem is the exact opposite of the exploding gradient, when we set the weights less than 1 when the gradient is passed onto the next RNN unit it is diminished due to getting multiplied with the small gradient leading to the vanishing gradient problem. In order to overcome the vanishing /exploding gradient problem LSTMs were introduced.

$$a^{<t>} = g1(W_{aa} a^{<t-1>} + W_{ax} x^{<t>} + b_a) \quad (1)$$

$$y^{<t>} = g2(W_{ya} a^{<t>} + b_y) \quad (2)$$

Equations 1 and 2 represent the equations for a basic RNN for each time step t, a<t> represents the activation, and y<t> the output where Wax, Waa, Wya, ba, by are the weights and biases respectively and g1 and g2 represents the activation functions.

LSTMs consist of two parts,the Long Term Memory which is also referred to as cell state and a Short term memory which is referred to as hidden state.The cell state goes through the whole LSTM with very minor linear interactions. This ensures that the information remains fairly consistent over many time steps and allows a network to work with long term dependencies. There are three gates the forget gate,input gate and the output gate which control the flow of information by ensuring only relevant information goes through. The hidden state gets updated at every time step and contains information about the previous states. It also serves to predict the next output and forms a link between the current input and the cell state.

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \quad (3)$$

Equation 3 represents the equation for the input gate where wi and bi represent the weights and biases for the LSTM and h<t-1> represents the output of the previous LSTM block and xt is the current input to be given to the LSTM block. σ represents the sigmoid function. The input gate controls the percentage of new information that is to be added to the cell state in every time step. The previous hidden state is used with the current input to produce the input gate vector i and candidate cell state.

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \quad (4)$$

Equation 4 represents the equation for the forget gate. The forget gate determines the amount of information to be discarded from the cell state. It uses the previous hidden state and current input and then applies a sigmoid activation function to them to obtain an output. The result is between 1 and 0 and this decides how much of the information is to be remembered.

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \quad (5)$$

Equation 5 represents the equation for the output gate. The output gate decides the output of the LSTM cell for that time step. It uses previous hidden state value and the current input value to give output for the cell. This gate is responsible for updating the short term memory.

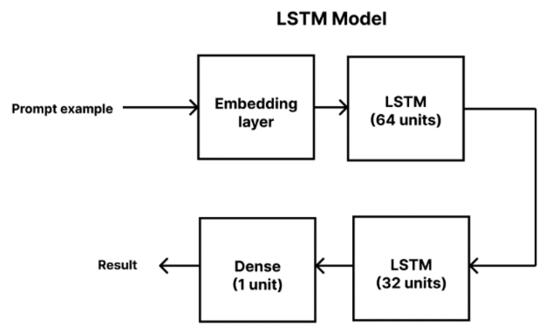

Fig.1. Architecture for the LSTM model

Fig. 1 illustrates the block diagram for the LSTM model used for Malicious prompt detection. It contains two LSTM layers followed by a dense layer for the classification of the prompt to be either malicious or not. First a prompt is taken and it is embedded into numbers so as to pass it to the LSTM. The keras embedding layer is used to map each word to integer and then produce a continuous dense vector to use in the neural network. This output is fed to to the first LSTM layer which contains 64 units. This layer tries to capture the temporal dependencies and also tries to find patterns in the data. Then this data is passed to next LSTM layer which contains 32 units and is able to capture some complex patterns and hierarchies. Finally, a 1 unit dense layer is used with a sigmoid activation function to provide a binary output that is classify the prompt as malicious or safe. The final result is a single value representing if the prompt is safe or malicious.

2. Simple Neural Network Model

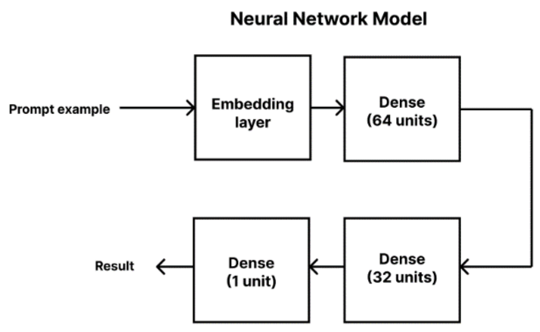

Fig 2. Neural network architecture

Fig. 2 illustrates the block diagram for a simple Neural Network model used for Malicious prompt detection. The input to the model is a prompt that needs to be classified into malicious or benign. Keras embedding layer is used to convert each word to an integer and produce continuous vector representations and then passed to the dense layer.

The dense layer contains 64 units and performs a linear transformation followed by the ReLU function to the input data. This allows the network to capture patterns and dependencies of the data. Then this data is fed to the second dense layer which contains a single unit with a sigmoid function which gives the final output on whether the prompt is malicious or not.

Fig 3 shows the flowchart for the basic algorithm followed while the development of both the models. It consists of successive steps followed by the models starting from “prompt” at the top and lead up to “Output (Malicious and Benign)”. This algorithm has adapted the deep learning approach with the use of two LSTM layers.

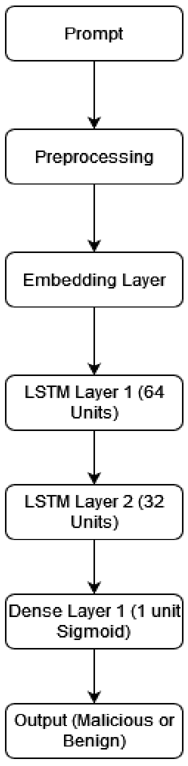

Fig.3. Flowchart of the Algorithm

Algorithm
Input: User prompt
Output: Classification whether malicious or benign.

1.Preprocessing :

- Remove irrelevant information from prompt
- Tokenize the prompt
- Apply text clearing like lemmatization.

2. Embedding:

- Convert preprocessed prompts into numerical representations using word embedding technique. Model Processing:

3. Model Processing
3.1. For LSTM model:

- Feed the embedded prompt sequence into the first LSTM layer.
- The LSTM layer processes the sequence, capturing long-term dependencies within the prompt.
- Pass the output of the first LSTM layer to the second LSTM layer for further pattern extraction.

3.2 For Simple Neural Network

- Feed the embedded prompt sequence into the dense layer.
- The dense layer performs a linear transformation and introduces non-linearity to identify relationships within the prompt data.

4. Classification

- Pass the output from the final layer (LSTM layer in LSTM model or dense layer in simple model) to a dense layer with one unit.
- Apply a sigmoid activation function to the output of the final dense layer.
- The sigmoid function outputs a value between 0 and 1, representing the probability of the prompt being malicious.

5. Decision

- Define a threshold value (e.g., 0.5).If the output probability from the sigmoid function is greater than or equal to the threshold, classify the prompt as "Malicious."Otherwise, classify the prompt as "Benign.

The above algorithm processes the user prompts to classify them as benign or malicious. The first step is preprocessing the prompt by using tokenization and further using text clearing technique lemmatization which is the process of breaking down words into the root meaning to identify similarities to other words. Following preprocessing, the words are converted into numerical values for the model using word embeddings. Once the numerical values are obtained they are passed through the LSTM units which are able to capture long term dependencies or in case of the simple neural networks, passed through the dense layer. Finally, the model's output is passed through the sigmoid activation function which classifies the prompt as benign or malicious.

IV. RESULTS AND DISCUSSIONS

Precision of a model is the ratio of the accurately predicted positives to all the predicted positive observations. Precision is calculated as shown in (6),

$$Precision = \frac{TP}{TP+FP} \quad (6)$$

Where the number of true positives is given by TP and the number of false positives by FP .

Recall of a model which is also known as sensitivity is a ratio of the number of accurately predicted positives to all the predictions in the class. Recall is given by (7),

$$Recall = \frac{TP}{TP+FN} \quad (7)$$

Where number of true positives is given by TP and number of false negatives is given by FN.

F1 score of a model can be defined as the harmonic mean of precision and recall as shown in 8.

$$F1score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (8)$$

Models	Precision	Recall	F1-Score
LSTM Model	0.75	0.63	0.69
Simple Neural Network Model	0.75	0.72	0.73

TABLE 1. TRAIN AND TEST ACCURACIES OF MODELS

Table 1 is a comparison of the train and test set accuracy of both the models. On observing table 1 we can see that the neural network model has more test and train accuracy indicating that the neural network model is more efficient in classifying whether the prompt is malicious or not.

Models	Train Accuracy	Test Accuracy
LSTM Model	99.11	91.69
Simple Neural Network Model	**99.51**	**92.52**

TABLE 2. PRECISION, RECALL AND F1-SCORE COMPARISON

It can be observed from table 2 that the F1 score of Neural Network Model us greater than that of the LTSM model indicating that Neural Network Model is able to distinguish and recognise malicious prompts with a higher efficiency between the two also indicating that it has a lower amount of false positives.

V. CONCLUSION

This research study has successfully investigated how well Large Language Models (LLMs) can detect harmful prompts using Long Short-Term Memory (LSTM) networks. The performance metrics showed us that a simple neural network is better than the LTSM model in the identification of malicious prompts even though LTSM has a higher accuracy. This concludes that the straightforward model is more effective accurately and computationally for malicious prompt detection.

The future scope could be to find the factors that lead to the simpler model outperforming the LTSM model. Also integration of domain-specific expertise into the model training process. Other methods could be adding multiple layers of security into the model like post prompting, paraphrasing and sequence enclosing on detection of malicious content.

VI. REFERENCES

[1] Liu, Yi et al. "Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study." *ArXiv* abs/2305.13860 (2023): n. Pag.

[2]Liu, Yi et al. "Prompt Injection attack against LLM-integrated Applications." *ArXiv* abs/2306.05499 (2023): n. Pag.

[3]M. Gupta, C. Akiri, K. Aryal, E. Parker and L. Praharaj, "From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy," in *IEEE Access*, vol. 11, pp. 80218-80245, 2023, doi: 10.1109/ACCESS.2023.3300381.

[4]Yip, D. W., Esmradi, A., & Chan, C. F. (2023). *A Novel Evaluation Framework for Assessing Resilience Against Prompt Injection Attacks in Large Language Models*. https://doi.org/10.1109/csde59766.2023.10487667

[5]Suo, Xuchen. "Signed-Prompt: A New Approach to Prevent Prompt Injection Attacks Against LLM-Integrated Applications." *ArXiv* abs/2401.07612 (2024): n. Pag.

[6] X. Liu, Z. Yu, Y. Zhang, N. Zhang, and C. Xiao, "Automatic and Universal Prompt Injection Attacks against Large Language Models," *arXiv.org*, Mar. 07, 2024. https://arxiv.org/abs/2403.04957.

[7] D. Yao, J. Zhang, I. G. Harris, and M. Carlsson, "FuzzLLM: A Novel and Universal Fuzzing Framework for Proactively Discovering Jailbreak Vulnerabilities in Large Language Models," *arXiv.org*, Sep. 11, 2023. https://arxiv.org/abs/2309.05274.

[8] A. Rao, S. Vashistha, A. Naik, S. Aditya, and M. Choudhury, "Tricking LLMs into Disobedience: Formalizing, Analyzing, and Detecting Jailbreaks," *arXiv.org*, Feb. 26, 2024. https://arxiv.org/abs/2305.14965

[9]Alotaibi, L., Seher, S., & Mohammad, N. (2024). *Cyberattacks Using ChatGPT: Exploring Malicious Content Generation Through Prompt Engineering*. https://doi.org/10.1109/icetsis61505.2024.10459698

[10]Li, L., & Gong, B. (2023). *Prompting Large Language Models for Malicious Webpage Detection*. https://doi.org/10.1109/prml59573.2023.10348229