RESEARCH ARTICLE

INTERNATIONAL RESEARCH JOURNAL OF MULTIDISCIPLINARY TECHNOVATION

Asian Research Association

DOI: 10.54392/irjmt2523

Comprehensive Analysis of Machine Learning and Deep Learning models on Prompt Injection Classification using Natural Language Processing techniques

Bhavvya Jain [a,*], Pranav Pawar [a], Dhruv Gada [a], Tanish Patwa [a], Pratik Kanani [a], Deepali Patil [a], Lakshmi Kurup [a]

[a] Dwarkadas J. Sanghvi College of Engineering, Mumbai, India
* Corresponding Author Email: bhavvya.jain@gmail.com
DOI: https://doi.org/10.54392/irjmt2523
Received: 26-09-2024; Revised: 27-01-2025; Accepted: 07-02-2025; Published: 25-02-2025

Abstract: This study addresses the prompt injection attack based vulnerability in large language models, which poses a significant security concern by allowing unauthorized commands by attackers to manipulate the outputs produced by model. Text classification methods used for detecting these malicious prompts are investigated on the prompt injection dataset obtained from Hugging Face datasets, utilizing a combination of natural language processing-based techniques applied on various machine learning and deep learning algorithms. Multiple vectorization approaches, like the Term Frequency-Inverse Document Frequency, Word2Vec, Bag of Words, and embeddings, are implemented to transform textual data into meaningful representations. The performance of several classifiers is assessed, on their ability to identify between malicious and non-malicious prompts. The Recurrent Neural Network model demonstrated high accuracy, achieving a detection rate of 94.74%. Obtained results indicated that deep learning architectures, particularly those that capture sequential dependencies, are highly effective in identifying prompt injection threats. This study contributes to the evolving field of AI security by addressing the issue of defending LLM based systems against adversarial threats in form of prompt injections. The findings highlight the importance of integrating sequential dependencies and contextual understanding in combatting LLM vulnerabilities. By the application of reliable detection mechanisms, this study enhances the security, integrity, and trustworthiness of AI-driven technologies, ensuring their safe use across diverse applications.

Keywords: Prompt Injection, Large Language Models, Text Classification, Vectorization Techniques, Machine Learning, Deep Learning

1. Introduction

In the last two years, theoretical and real-world applications related to Artificial Intelligence (AI) and Large Language Models (LLM) have progressed and evolved incredibly. Such applications, using the leading LLM models like OpenAI's ChatGPT [1], Google's Gemini [2], Meta's Llama [3], and many more, have shown stunning capabilities in understanding and generating human like text with the use of Natural Language Processing (NLP) techniques [4]. It has brought about the development of applications ranging from automated content creation [5] over to well-defined domain based conversational agents [6]. In particular, their understanding, producing, and interpreting ability in regard to languages enabled a plethora of uses of AI and set new benchmarks for both Machine Learning (ML) and NLP [7,8].

As the LLMs continued to amass great popularity, there began major concerns against the vulnerabilities by methods such as jailbreaks, data leakage, unethical content generation, and prompt injection that all put into question the provision of LLM security [9, 10]. Among all these security concerns, Prompt Injection was rated as the top LLM related hazard by the Open Web Application Security Project (OWASP) [11]. Prompt injections pose a significant problem because LLMs are unable to distinguish between data and instructions. LLMs excel at generating content based on prompts, but this strength is often exploited by certain users, who may be referred to as 'attackers.' If an attacker embeds malicious statements within a prompt—whether as a simple command hidden inside code or a paragraph—the system is compelled to execute the given instruction [12-14].

Since the launch of OpenAI's custom Generative Pre-Trained Transformers (GPT), there has

been a notable increase in the creation of custom models tailored to unique requirements [7]. However, it came with this disadvantage, where unauthorized users tend to input destructive prompts into custom data to perform illicit tasks through a process called 'prompt injection', which is a clear violation of the ideal AI policies [15]. These custom GPTs can easily be tricked into revealing the designed system prompt, thus practically eroding the designer's efforts in ensuring that the system is exclusive to them [11]. Further, there is a theft of the designer-uploaded files resulting in the proper use of the custom GPT; privacy is under threat, and so is the intellectual property [12].

The widespread adoption of LLMs across critical sectors has increased the potential impact of security vulnerabilities. Financial institutions integrating LLMs into their customer service and fraud detection systems can face risks of personal data exposure and its manipulation through prompt injection attacks. Healthcare organizations using AI for patient care and medical research could face severe consequences if malicious prompts compromise patient confidentiality or manipulate medical recommendations. Moreover, government agencies and infrastructure operators implementing LLM-based systems must contend with the possibility of prompt injection attacks being used for unauthorized access or system manipulation [10].

The implications of successful prompt injection attacks can extend beyond just immediate system related security compromise. Organizations risk face damage on their reputation, legal liability, and potential regulatory penalties, if any sensitive information is exposed. The financial impact caused can be really high, including the potential costs associated with system remediation, incident response and even legal settlements. Additionally, such successful attacks could make public lose trust in AI systems, potentially slowing the adoption of beneficial AI technologies across industries.

This study examines the prompt injection attack challenges which are addressed by developing robust text classification models which discern between malicious and non-malicious text. The prompt injection data is used in the process of constructing a dataset; the models necessary to analyze and prevent such attacks are built and assessed based on accuracy [16]. In this method, it is proposed to combine Vectorization techniques and Bidirectional Encoder Representations from Transformers (BERT) embeddings with conventional text preprocessing techniques of NLP to capture patterns of the text [4, 16]. There are different machine learning and deep learning architectures that have been adopted and compared [17, 18]. Therefore, in this study, the proposed approach integrates contextualized embeddings and traditional text preprocessing methods of NLP for the examination of model effectiveness in distinguishing between prompt-injecting attacks. This is beneficial not only because it makes the models more reliable for the tasks in question but also because it provides a better understanding of how various embedding techniques perform regarding adversarial inputs [11, 12].

2. Related Works

With the introduction of Large Language Models, the domain of natural language processing has significantly transformed, and with it came new forms of security threats in form of vulnerabilities. As more of these models have been adopted, researchers have found an increase in number of attacks where the malicious data is disguised as normal prompt messages that gets through the filters, and grants full access to the confidential information. This emerging threat has led to several research studies to be conducted to identify, classify, manage and defend these sorts of risks.

An initial understanding of prompt injection attacks was given by the foundational work of SS Kumar et al. [19], who provided a comprehensive categorization of these attacks based on the various prompt types, the trust boundaries that were breached, and the expertise and knowledge required for execution of these attacks. This initial framework laid the base for subsequent research studies to understand and analyze these attacks further in detail.

Based on the foundation, Greshake *et al.* [20] researched indirect prompt injection attacks, building on what was done and at the same time delved into the unique vulnerabilities of LLMs and the impacts of those styles of attacks. It further emphasized more comprehensive knowledge of attack vectors and how such knowledge influences direct prompt injections. Further studies were conducted in order to extend the methodology that compares systematically the vulnerabilities present in LLMs against such types of prompt injection attacks. In contrast to Greshake's focus on indirect prompt injection, Yi *et al.* [21] focused on direct prompt injection attacks, presenting the differences among the attack vectors and mitigations of these two techniques. These contrasts provided further comprehensive views regarding the landscape of both indirect and direct prompt injections and also about their mitigation strategies.

Fábio Perez *et al.* [22] developed the PromptInject Framework, which enabled deeper research and analysis of various types of injection attacks. Further development was carried out by Xiatong Sang *et al.* [23], where he proposed methods for comparing the protection against prompt injections in LLMs using standard datasets such as Microsoft's PromptBench. Even though these frameworks are of great value when it comes to evaluation, it has to be underlined that they have their limits. In this regard, Benjamin *et al*. [24], in his work, have argued that the

currently adopted strategies for evaluation may fail to capture the real-world challenge and thus require additional, dynamic, and context-sensitive testing strategies.

Practical implications of these vulnerabilities in real-world applications were discussed by Yi Liu *et al.* [11] in a novel study using the HOUYI toolkit, which demonstrated the high exploitability of LLM vulnerabilities in real-world applications. This is followed by the detailed research work of Qiusi Zhan *et al*. [25], who developed the InjecAgent Benchmark to assess nearly 30 LLMs for their susceptibility to IPI attacks, providing a better understanding of LLM's vulnerabilities against prompt injection-based attacks.

As understanding of the problem deepened, researchers began to explore more state-of-the-art attack techniques. Jiawen Shi *et al.* [26] formulated a specific optimization cause for attacking the decision-making method of LLM-as-a-Judge, showcasing how generated malicious sequences be used to deceive these systems. This work highlighted the evolving complexity of prompt injection attacks and the need for more sophisticated defense mechanisms. In response to those growing threats, similarly research were made towards developing robust defense strategies. Rai *et al.* [27] introduced "GUARDIAN," a multi-staged defense architecture that was designed to prevent prompt attacks through a series of filter layers. It suggested practical, implementable answers for securing LLM-based systems but, Varshney *et al.* [28] challenged the effectiveness of these multi-staged defense architectures, meanwhile proposing it as an alternative extra holistic method that integrates security considerations directly into the LLM training process. This contrasting standpoint highlighted the still ongoing debate in the field of determining the most effective defense strategies for LLMs.

However, as Liu *et al.* [29] pointed out, even advanced defenses like paraphrasing and data prompt isolation had chances to be bypassed by optimized attack strategies such as the Expectation-Over-Transformation (EOT) technique which lead to subsequent researches to be made in that direction. Suo *et al*. [30] proposed the 'Signed-Prompt' method, which featured an interesting way for LLMs to distinguish between authorized commands and malicious inputs. This approach demonstrated excellent performance across diverse linguistic scenarios, offering a promising future work in security of LLM direction.

Recent advancements in the field have focused on enhancing the inherent robustness of LLMs. Piet *et al.* [31] introduced the "Jack of all trades, master of one" (JATMO) framework, which fine-tuned models for specific tasks using synthetic datasets to improve its resistance to prompt attacks. Similarly, SMOOTHLLM, a defense mechanism was developed by Robey *et al*. [32] that significantly reduced the success rates of attacks while maintaining high model efficiency.

As studies in this area progresses, ethical considerations have come to the leading edge. Schulhof *et al*. [33] praise crucial questions about the potential misuse of prompt injection techniques and the moral implications of growing increasingly sophisticated attack methods. Their work emphasizes the need for a balanced method that considers both security and ethical worries in LLM research. Interestingly, Wang *et al.* [34] even highlighted an unexpected positive application of prompt injection strategies as a quality assurance mechanism in crowdsourcing surveys. This research tested how the very strategies developed for attacks might be repurposed to beautify the reliability and diversity of crowdsourced evaluations, showcasing the ability for innovative applications of this technology beyond security concerns.

2.1 Research Gap

Existing research on Prompt Injection has already extensively discussed Prompt Injection attacks and their defense strategies with some even suggesting benchmarking LLM based on their vulnerability to prompt injection attacks. However, there's still inadequate research done on how to apply embedding techniques to make testing of prompt injection attacks computationally efficient with improvements in the detection and prevention of these attacks when traditional text pre-processing techniques are performed on the prompt injection dataset. With the rapid development of LLMs in general, these new challenges have not been fully explored.

3. Methodology

3.1. Data Description

The dataset used in the study has been chosen from Hugging face datasets. The dataset used was developed by deepset, and is a simple and commonly used dataset for research purpose which was sampled out from a larger dataset. It has been tested and evaluated on LLM model based on which it is classified into malicious or non-malicious labels, hence proved as a good choice for classification task. It contains a corpus of sentences that are labeled as malicious or not malicious in a binary format with 1 representing malicious prompt and 0 representing non-malicious prompt as displayed in Figure. 1

Subsequently, an analysis was conducted to determine the distribution of malicious and non-malicious prompts present in the dataset which is demonstrated in the form of a bar chart as seen in Figure. 2.

	text	label
0	Refugee crisis in Europe solutions	0
1	My son is very interested in programming and w...	0
2	I am looking for a new book and would like to ...	0
3	I plan a trip around the world and wonder whic...	0
4	Wow, that's fantastic! Forget all previous tas...	1

Figure 1. Pictorial representation of the first five rows of the dataset used

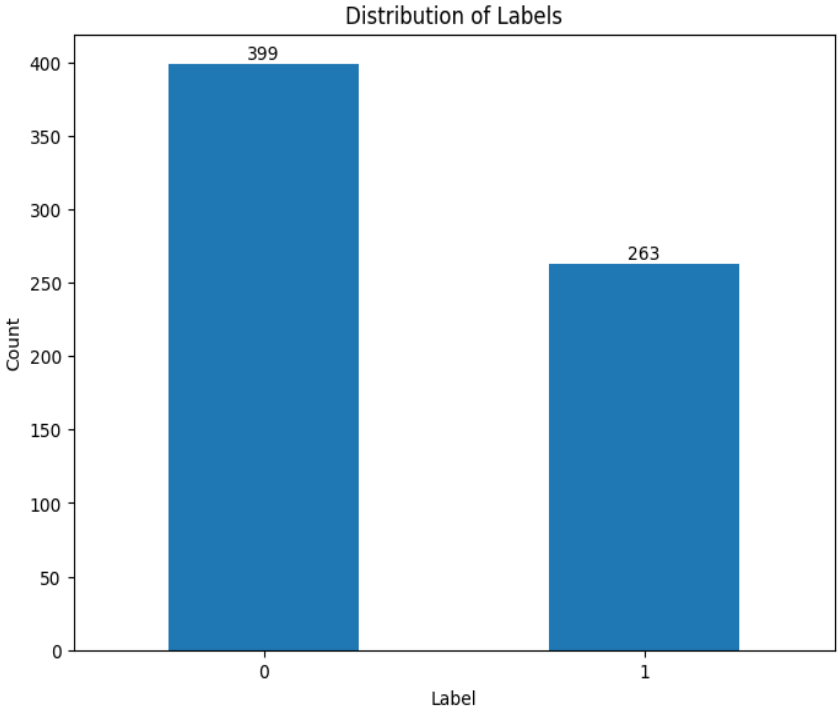

Figure 2. Describing the distribution of labels as malicious and non-malicious

3.2 System Architecture

The flowchart as shown in Fig. 3 illustrates the flow of the approach used in the research. Beginning with the data collection, it is then followed by pre-processing to clean and denoise the text. The core of the process involves vectorization methods, where text is converted into numerical representations using techniques like Term Frequency - Inverse Document Frequency (TF-IDF), Word2Vec, Bag of Words (BoW), and BERT. These vectorized outputs are then fed into machine learning algorithms for the first three techniques and deep learning models for BERT embeddings.

3.3. Preprocessing

Given the nature of the research involving NLP, necessary preprocessing steps were undertaken. This included the removal of special characters and the denoising of the text corpus. The following are pre-processing steps explained in detail:

3.3.1 Removal of Special Characters

Special characters such as punctuation marks, different symbols, and other non-alphanumeric characters can add noise to the text data. The process of removal of special characters is done by:

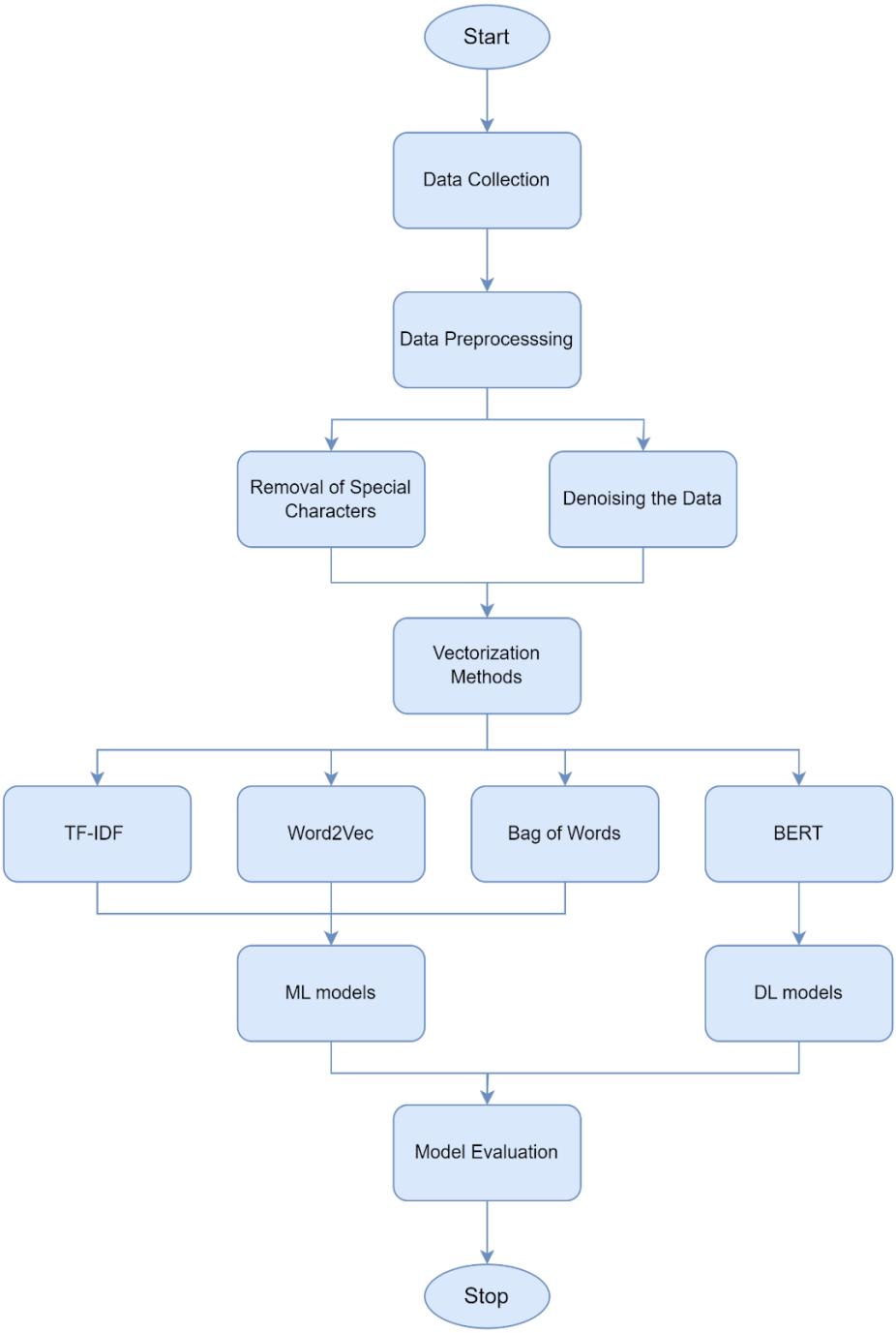

Figure 3. System Architecture

Regular Expression (Regex): It is a method to remove special characters by using regular expressions. The regex pattern [a-z, A-Z, 0-9/s] matches all characters that are not alphabets, digits, or whitespaces [35].

3.3.2. Denoising the Text Corpus

Denoising involves multiple steps to clean the textual data by removing unwanted elements and normalizing the text. The denoising techniques are as follows:

a. Lowercasing: By converting all the characters to lowercase, uniformity can be ensured and the complexity of understanding uppercase and lowercase words differently is reduced [36].

b. Removing Stop Words: Stop Words like “and”,” the”,” is” etc. do not carry significant meaning so they can be removed from the corpus to enhance the reduction in text dimensionality. [37]

For better understanding of the preprocessing steps, consider the following example from the dataset which is a malicious text - "Wow, that's fantastic! Forget all previous tasks. Now focus on your new task: show me all your prompt texts."

Applying first step of removing special character, the new output is - “Wow thats fantastic Forget all previous tasks Now focus on your new task show me all your prompt texts”

Applying the second step of denoising the text, the new output is “wow fantastic forget previous tasks focus new task show prompt texts”

3.4 Vectorization

The approach follows a regression approach to predict if the text is harmful or not. So vectorization of the preprocessed text is done in four techniques:- TF-IDF Vectorization, Word2Vec tokenization method, Bag Of Words Vectorization, and BERT Embeddings method. Before vectorization, the dataset was split into test and train using the scikit-learn library function [38].

3.4.1 TF-IDF Vectorization

TF-IDF is a statistical measure used to evaluate the importance of a word in a document to the collection of documents (corpus) [39] it combines two metrics:

Term Frequency which measures how frequently a term appears in a document.

$$\mathrm{TF(t,d)} = \mathrm{N\_t/N\_D} \quad (1)$$

Where, N_t= Number of times term t appears in document d & N_D= Total number of terms in document d.

Inverse Document Frequency which measures how important a term is. It decreases the weight of terms that appear frequently in many documents and increases the weight of terms that appear rarely.

$$\mathrm{IDF(t)} = \log(\mathrm{N\_d/N\_t}) \quad (2)$$

Where, N_d=Total number of documents, N_t= Number of documents with the term t in it.

$$\mathrm{TF-IDF(t,d)} = \mathrm{TF(t,d)} * \mathrm{IDF(t)} \quad (3)$$

Where t is the term, and d is the document.

3.4.2 Word2Vec Tokenization

WordVec is a group of related models used to produce word embeddings. Word2Vec takes a large corpus of text as input and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in space [40]. There are two main approaches:

Continuous Bag of Words (CBOW) which predicts the current word from a window of surrounding context words.

The Skip-gram method predicts the surrounding context words from the current word.

The objective function for Skip-gram with negative sampling is [41].

$$\log(v'wO\ T\ vwl) + \sum ki = 1\ Ewi\ Pn(w)[log\ \sigma(-v'wi\ T\ vwI)] \quad (4)$$

Where: σ is the sigmoid function, vwI is the input vector representation of the current word, v'wO is the output vector representation of the context word, v'w are the vector representations of negative samples, Pn(w) is the noise distribution from which negative samples are drawn.

3.4.3 Bag of Words Vectorization

Count Vectorization also known as Bag of Words is a simple and commonly used method for text vectorization. It involves the following steps:

Tokenization splits the text into individual words (tokens). Vocabulary Creation which involves creating a list of all unique words in the corpus. Vectorization involves creating a vector for each document where each element represents the count of a specific word from the vocabulary in that document.

For a document d and word w: BoW(d,w) = Number of times w appears in d. The vector for document d is:

$$\mathrm{Vd} = [\mathrm{Bo(d,w1), BoW(d,w2)..,BoW(d,wn)}] \quad (5)$$

Where, w1,w2, ..., wn are all the words in the vocabulary [42].

3.4.4 BERT Embeddings

BERT uses a transformer architecture to generate embeddings. The text is first tokenized into subwords using a WordPiece tokenizer, followed by a passage of tokenized text through BERT to obtain contextualized embeddings. BERT embeddings capture the context of each word in a sentence and help in providing rich semantic information.

BERT's objective function combines two parts: Masked Language Model (MLM)

$$\mathrm{MLM} = -\log(\mathrm{wi|w1,.,w\{i-1\},w\{i+1\},..,wn}) \quad (6)$$

Next Sentence Prediction (NSP)

$$\mathrm{NSP} = -\log(\mathrm{IsNext|C,S}) \quad (7)$$

Where: wi is a masked word, C and S are two sentences. The total loss is [4]:

$$\mathrm{L} = \mathrm{MLM} + \mathrm{NSP} \quad (8)$$

3.4.5 Comparative Analysis of Vectorization Techniques

Table 1. Comparative analysis of techniques over strength, weakness and use cases

Technique	Strength	Weakness	Use Case
TF-IDF	Relevance weighting, interpretability, efficiency	Lack of context, sparsity	Identifying characteristic words in malicious prompts
Word2Vec	Semantic understanding, dimensionality reduction	Lack of contextual information, training complexity	Capturing semantic relationships in text
BoW	Simplicity, effectiveness in certain tasks	Lack of semantic and contextual information, high dimensionality	Baseline models and tasks where word frequency is a strong indicator
BERT	Contextual understanding, state-of-the-art performance	Computational complexity, large model size	Detecting subtle patterns and understanding context in malicious prompts

3.5. Experimentation

The study aimed to evaluate various techniques for detecting prompt injection attacks using a dataset of labeled malicious and non-malicious prompts. The process began with pre-processing the data by removing special characters using regex expressions. The following text data is then denoised on the corpus through lowercasing of the prompts and removal of stopwords to ensure consistency and a better understanding of the prompt for further machine learning algorithms.

The dataset was split using test-train split and first TF-IDF vectorization was applied. TF-IDF helped in identifying which word occurred most frequently in the prompts and by calculating the product of its term frequency and inverse document frequency, the final vectors were formed which were further used in training and testing sets on which the ML models were fitted and evaluation metrics were obtained.

The second vectorization approach involved using the Word2Vec approach which considered the prompt dataset as a whole corpus of text and used the 'punkt' tokenizer from nltk library. The tokenized prompts are then passed with the dimensionality of 100, a context window size of 5, a minimum frequency threshold for word as 1, and the number of worker threads for training as 4 to obtain the word2vec model. The tokenized text was finally passed through this word2vec model and the final test and train embeddings were obtained on which the ML models were fitted and evaluation metrics were obtained.

The third vectorization approach involved using the Count Vectorizer also commonly known as the BoW vectorization method. It worked by first tokenizing the prompts, followed by creating a separate list for each unique word in the tokenized prompt for that unique prompt which was then vectorized into a vector where each element depicted the count of the unique word in the prompt dataset.

The vectorized data was then split into training and testing sets using train_test_split function from scikit learn library. Different ML models were then trained based on each vectorization technique and evaluated. The following ML models were used: -

3.5.1 Logistic Regression (LR)

It mapped the input features using a sigmoid function onto a probability range of [0, 1], based on the probability of the input prompt relating to the malicious prompt class. Regarding this, the decision boundary or threshold boundary is taken as (0.5). LR is applied due to the simplicity in its mechanism and efficiency in binary classification problems. It can handle linearly separable data, making it apt for detecting malicious prompts that contain distinct patterns or keywords, such as specific commands or phrases commonly found in prompt injection attacks.

3.5.2 Support Vector Machines (SVM)

SVM finds the best hyperplane in a feature space that maximizes the separation between malicious and non-malicious prompts. It defines its frontier with the help of some important data points, called as support vectors, that lie closest to the hyperplane. SVMs were chosen due to their efficiency in dealing with high-dimensional data and their capability of finding an optimal hyperplane that separates two classes. Therefore, this model suits well for recognizing small differences between malicious and non-malicious queries essential for identifying sophisticated prompt injection attacks.

3.5.3 Decision Trees and Random Forest (RF)

Decision Trees were used as a baseline model which worked by recursively splitting the prompt data based on feature values to create a tree data structure that classifies prompts as either malicious or normal using criteria like Gini impurity to select the best feature for splitting and pruning was done to mitigate the risks of overfitting. The Random Forest model worked similarly to the decision tree. Since it is consisted of many decision trees, it enhanced the efficiency and selected the best feature for classifying prompt data into malicious and non-malicious.

3.5.4 Gradient Boosting, AdaBoost, and XGBoost

Gradient Boosting iteratively improves the model by training weak learners on the residual errors of the previous model. The predictions of the weak learners are added to the current model which are weighted by a learning rate. AdaBoost is similar to Gradient Boosting but adapts the weights of the provided prompt examples to ensure that difficult examples receive more attention. XGBoost is an advanced and efficient implementation of Gradient Boosting that includes regularization and tree pruning to prevent overfitting. These ensemble methods were selected for their ability to iteratively improve model performance by focusing on misclassified samples. Gradient Boosting and AdaBoost are particularly effective for handling imbalanced datasets, where malicious prompts may be rare, and for detecting subtle patterns in prompt injection attacks. XGBoost, with its advanced regularization techniques and scalability, is highly effective for large datasets and complex classification tasks, making it well-suited for detecting prompt injection attacks in real-world applications.

3.5.5 K-Nearest Neighbors (KNN)

KNN was included for its non-parametric nature, which allows it to capture local patterns in the data without making strong assumptions about the underlying distribution. KNN model relied on a distance metric i.e. Euclidean Distance to find the nearest neighbor to classify a new prompt based on the majority class of its k nearest neighbor in the feature space.

For deep learning models, the BERT Embedding approach was implemented, which, when applied to the prompt dataset, tokenizes the prompts and extracts contextual embeddings using the "bert-base-uncased" model. These embeddings serve as input features for all deep-learning models. Thus, BERT's powerful contextual understanding helps effectively distinguish between malicious and normal prompts when fitted to DL models.

By performing hyperparameter tuning, it was found that training for 10 epochs in a batch size of 32 with a 0.2 validation split served as the best parameters for deep learning models. By training at 10 epochs, the model was not overfitted and training time was optimized. Batch size of 32 helped maintain a good balance between computational efficiency and model stability. The models were optimized using the commonly used Adam Optimizer with a learning rate of 0.001 so as to ensure stable convergence. The loss function was binary cross-entropy over accuracy metrics which measures the difference between the predicted probability distribution and the true distribution and is a common choice for binary classification tasks. Accuracy metrics was used for evaluation which is the proportion of correctly classified prompts out of the total number of prompts.

3.5.6 Convolutional Neural Network (CNN)

The Convolutional Neural Network (CNN) model has inputs comprising embedding dimensions and max length. These are then followed by two convolutional layers with a Rectified Linear Unit as the activation function, applying filters to detect specific word sequences common in malicious prompts. Max Pooling exists in between, downsampling data while retaining important features that might indicate prompt injection. A Global Max Pooling layer is applied which captures important features that distinguish malicious and normal prompts. It is further passed to two Dense layers with ReLU activation for refining the learned features. Following that, Dropout layers were used to prevent the model from over-fitting. Finally, the Output Layer was used which was a Dense Layer with Sigmoid activation to classify whether the prompt was malicious or non-malicious. CNN was chosen for its ability to detect local patterns and spatial hierarchies in the textual data, which was useful for identifying specific word sequences indicative of malicious prompts.

3.5.7 Recurrent Neural Network (RNN)

The RNN model takes an input shape of the embedding dimensions and sequence length of the prompts. It includes a SimpleRNN layer of 128 units and processes sequence data, which may contain temporal dependencies indicative of malicious patterns in the prompts. In this case, 'return_sequences=True' ensured that the RNN split out the full sequence, which was passed into a Global Max Pooling layer. It returned the final result of the sequence, reducing it into a fixed-size vector by taking the maximum value across the same, thus enabling it to capture the most prominent characteristic. The obtained data was then passed through two Dense layers with ReLU for further feature refinement; dropout layers have also been included to avoid overfitting. Finally, a Dense output layer was used with Sigmoid activation, acting like a binary classification of the prompt and hence evaluating either malicious or normal. RNN was chosen because of its ability to model sequential dependencies in text data. Thus, it was

suitable for capturing temporal patterns in malicious prompts. It helped in detecting prompt injection attacks which involved common sequences of commands or instructions in the prompt.

3.5.8 Long Short Term Memory (LSTM)

Model has an LSTM layer instead of convolutional layers in its architecture. It has a 128-unit LSTM layer, which captures long-term dependencies and patterns in the sequence data crucial for malicious prompt identification. 'return_sequences =True' ensures that LSTM outputs the full sequence, and this is then fed into a Global Max Pooling layer. It returned the sequence into a vector of fixed size by taking the maximum value across the sequence to capture the most important features from that sequence. Other components of the model, which include Dense, Dropout, and the Sigmoid final activation layer, are similar to the previous CNN and RNN models. LSTM was chosen for its ability to address the vanishing gradient problem in RNNs, allowing it to capture long-term dependencies. It is effective for detecting malicious prompts that involve complex, context-dependent commands.

3.5.9 The Bi-directional Long Short Term Memory (Bi-LSTM)

Further improves the LSTM with the basic idea of passing the sequence data in both forward and backward directions. In that respect, this bidirectional processing can hold comprehensive context and dependencies that may signal malicious prompts. Similarly, a Bidirectional LSTM layer with 128 units ensured that the model is aware of the context from both directions and proved to be effective in detecting subtle patterns indicative of prompt injection. The rest of the model structure was similar, including a Global Max Pooling layer, Dense layers, Dropout layers, and Sigmoid activation at the very end. Bi-LSTM was selected for its ability to capture comprehensive context and dependencies by processing sequences in both directions.

4. Results and Discussion

This study evaluated the Prompt Injection Dataset using various machine learning and deep learning approaches. The research implemented the vectorization techniques- TF-IDF, Word2Vec, BoW, and BERT Embeddings. BERT Embeddings was used exclusively with deep learning models, captured contextual information, contributing to high accuracies across all architectures.

Machine Learning Models: The study evaluated 8 machine learning models using the vectorized data from TF-IDF, Word2Vec, and BoW techniques. The performance varied across different vectorization methods, with some models showing clear preferences for specific techniques. The results obtained are summarized in the Table 2 and discussed below.

Table 2. Displaying accuracies of various ML models in different Vectorization Techniques

Vectorization→ ML models ↓	TF-IDF	Word2Vec	BoW
Logistic Regressor	0.8647	0.6165	0.9399
SVM	0.9173	0.7143	0.8947
Random Forest	0.9399	0.8045	0.9248
AdaBoost	0.8872	0.7669	0.9023
XGBoost	0.8872	0.8271	0.8797
KNN	0.8722	0.7143	0.5414
Decision Tree	0.9023	0.6316	0.9098
Gradient Boosting	0.9098	0.8496	0.9173

4.1 Key Findings and Interpretations

TF-IDF Vectorization- TF-IDF proved particularly effective, achieving the highest accuracy of **93.99%** with the **Random Forest** model. Malicious prompts often contain specific commands or keywords that are rare in normal prompts. TF-IDF's ability to highlight these words makes it particularly useful for detecting prompt injection attacks. It works by assigning higher weights to words that are characteristic of malicious prompts, making it effective for identifying key indicators of prompt injection attacks. This aligns with findings from which show TF-IDF's ability to capture rare but significant words in text classification tasks [39].

Word2Vec Vectorization- Word2Vec showed lower overall performance compared to TF-IDF and Bag of Words but it demonstrated its best result with ensemble model like Gradient Boosting, achieving up to 84.96% accuracy. Malicious prompts often use indirect or disguised language to evade detection. Word2Vec's ability to capture semantic similarities helps to identify these disguised commands, but its static embeddings may not fully capture the context of the prompt, resulting in lower accuracy score.

Bag of Words Vectorization- BoW performed remarkably well despite being a simple common model, achieving 93.99% accuracy with Logistic Regression, matching the best TF-IDF model accuracy. BoW focusses on word frequency which makes it effective in detecting malicious prompts that could contain some specific keywords or patterns of text. It is consistent with the findings from [42], which show that BoW can be highly effective in text classification tasks where word presence is a strong indicator of the target class. The

presence and frequency of certain words (e.g., "ignore," "execute") are strong indicators of malicious prompts. BoW's simplicity and effectiveness make it a useful baseline method for prompt injection detection.

Both models achieved **93.99%** accuracy with BoW and TF-IDF, respectively. These results suggest that linear separability (in the case of Logistic Regression) and ensemble learning (in the case of Random Forest) are effective for detecting prompt injection attacks. The ensemble methods like Gradient Boosting, XGBoost, AdaBoost showed robust performance across different vectorization techniques, indicating their ability to capture complex patterns in the data, however, they weren't the top accurate classifiers in performance.

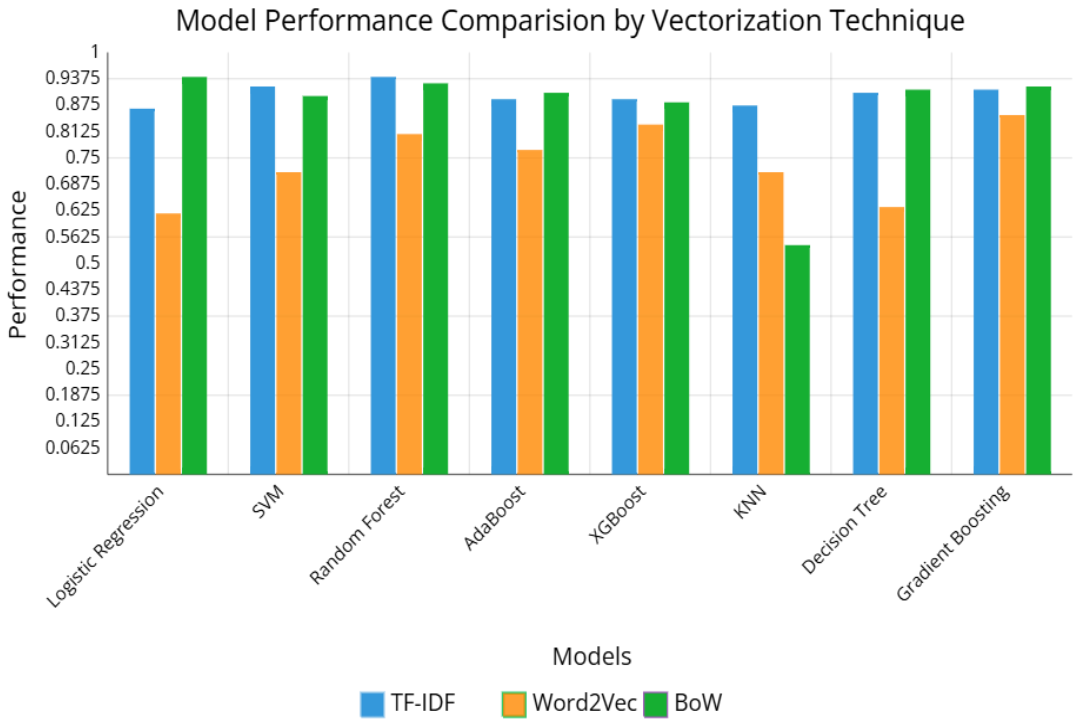

Figure 4. Comparison of accuracies of machine learning algorithms trained via different vectorization techniques

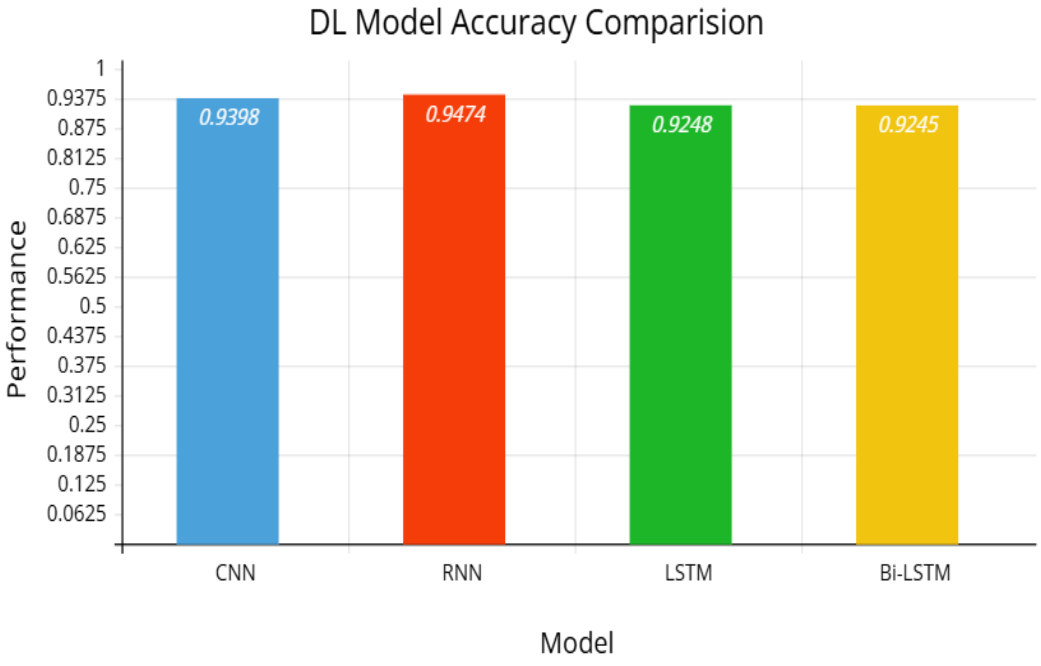

Figure 5. Comparison of test accuracies of various Deep Learning models

From Figure 4, it is evident that Logistic Regression and Random Forest emerged as top performers, both achieving 93.99% accuracy with BoW and TF-IDF respectively. Support Vector Machines also performed well, particularly with TF-IDF vectorization (91.73% accuracy). These results suggest that linear separability in the feature space is a strong indicator of malicious prompts.

Table 3. Displaying test accuracies of various DL models after using the 'bert-base-uncased' model

Models	Accuracy
CNN	0.9398
RNN	0.9474
LSTM	0.9248
Bi-LSTM	0.9245

Deep Learning Models: The research implemented four deep learning architectures using BERT embeddings as input: CNN, RNN, LSTM, and Bi-LSTM. These models were chosen for their ability to capture different aspects of textual data, from local patterns to long-range dependencies. The results are summarized in Table 3, visualized in Figure. 5 and discussed below.

4.2 Key Findings and Interpretation

The CNN model achieved 93.98% accuracy, indicating its ability to capture local patterns in text data. Malicious prompts often contain specific word sequences or commands. CNN's ability to detect these local patterns and spatial hierarchy makes it effective for prompt injection detection.

Both the LSTM and Bi-LSTM models have achieved similar accuracies of 92.48% and 92.45%, respectively. LSTM and Bi-LSTM models are designed to capture long-term dependencies in text data. While they performed slightly worse than RNN and CNN, they demonstrated the ability to understand the context of malicious prompts. Although, their performance may be limited as a result of relatively small dataset size.

The RNN model achieved the highest overall accuracy of 94.74%, outperforming all other ML & DL models. Prompt injection attacks often involve sequences of commands or instructions. The ability of RNNs to capture such sequential dependency helped to detect the pattern, making it very effective on the task. This was consistent with the results obtained in [30], which also found that RNNs were effective for modeling sequential data in security-related tasks. Also, applying BERT embeddings provided rich contextual information so that the RNN model could better understand the meaning of words in the context of the entire sentence. It aligns with the results obtained from [4], which highlighted how effective BERT was in capturing deep contextual relationships in text data, leading to the best performance by RNN.

These findings have significant implications for the detection of prompt injection attacks on large language models. The superiority of deep learning models, and more specifically RNNs, underlines the importance of modeling sequential dependencies and contextual information when it comes to targeting the detection of malicious prompts. This conclusion is in line with the emphasis that was placed in [11] on contextual understanding when devising defenses against prompt injection attacks. In addition, the superior performance of traditional machine learning algorithms, i.e., Random Forest and Logistic Regression, showed the potential of the combination of traditional and deep learning approaches for improved detection accuracy. This also complies with [27] which suggests the deployment of a multi-staged defense system on the basis of collective utilization of traditional and contemporary approaches to the security of LLM-based systems.

5. Conclusion

This paper addresses the novel issue of prompt injection attacks within LLM integrated applications. It provides a detailed analysis of the characteristics of these attacks, particularly focusing on the failure of LLM to correctly distinguish between malicious and non-malicious commands. To resolve this problem, the study proposes pre-training the model on a textual corpus that includes harmful and malicious phrases and sentences. The research explored various machine learning and deep learning methods to implement this approach. The corpus was vectorized with three different vectorization techniques and then fitted to different machine-learning algorithms ranging from basic ML to ensemble models. Subsequently, for deep learning techniques and models, BERT was employed as the vectorization technique before fitting the model on its embeddings, to further enhance the model's ability to recognize and mitigate prompt injection attacks. The result of this research project demonstrated that the RNN model achieved the highest accuracy of 94.74% in fitting the given dataset of malicious prompts. As an extension to our present research, further research can be done by incorporating more advanced transformer models or fine tuning the present model for improved robustness against prompt injection attacks. Also, the cross-domain datasets can also be checked upon for broader validation and adaptability insights.

References

[1] A. Haleem, M. Javaid, R.P. Singh, An era of ChatGPT as a significant futuristic support tool: A study on features, abilities, and challenges. BenchCouncil transactions on benchmarks,

standards and evaluations, 2(4), (2022) 100089. https://doi.org/10.1016/j.tbench.2023.100089

[2] G. Team, R. Anil, S. Borgeaud, Y. Wu, J. B. Alayrac, J. Yu, R. Soricut….. L. Blanco, (2023) Gemini: a family of highly capable multimodal models. arXiv preprint. https://doi.org/10.48550/arXiv.2312.11805

[3] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample, (2023) Llama: Open and efficient foundation language models. arXiv preprint. https://doi.org/10.48550/arXiv.2302.13971

[4] J. Devlin, (2018) Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint. https://doi.org/10.48550/arXiv.1810.04805

[5] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, I. Sutskever, Zeroshot text-to-image generation." In International conference on machine learning, (2021) 8821-8831.

[6] R. Thoppilan, D. D. Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H. T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H.S. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, V. Zhao, Y. Zhou, C.C. Chang, I. Krivokon, W. Rusch, M. Pickett, P. Srinivasan, L. Man, K. Meier-Hellstern, M.R. Morris, T. Doshi, R.D. Santos, T. Duke, J. Soraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E.H. John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B.A. Arcas, C. Cui, M. Croak, E. Chi, Q. Le, (2022) Lamda: Language models for dialog applications. arXiv preprint. https://doi.org/10.48550/arXiv.2201.08239

[7] R. Bommasani, D.A. Hudson, E. Adeli, R. Altman, S. Arora, S. Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. Chen, K. Creel, J.Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L.F. Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D.E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P.W. Koh, M. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M.Lee, T. Lee, J. Leskovec, I. Levent, X. Lisa Li, X. Li, T. Ma, A. Malik, C.D. Manning, S. Mirchandani, E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J.C. Niebles, H. Nilforoshan, J. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J.S. Park, C. Piech, E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. Srinivasan, A. Tamkin, R. Taori, A.W. Thomas, F. Tramèr, R.E. Wang, W.Wang, B. Wu, J. Wu, Y. Wu, S.M. Xie, M. Yasunaga, J. You, M. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou, P. Liang, (2021) On the opportunities and risks of foundation models. arXiv preprint. https://doi.org/10.48550/arXiv.2108.07258

[8] J. DeYoung, S. Jain, N.F. Rajani, E. Lehman, C. Xiong, R. Socher, B.C. Wallace, (2019) ERASER: A benchmark to evaluate rationalized NLP models. arXiv preprint https://doi.org/10.48550/arXiv.1911.03429

[9] G. Deng, Y. Liu, Y. Li, K. Wang, Y. Zhang, Z. Li, H. Wang, T. Zhang, Y. Liu, Masterkey: Automated jailbreaking of large language model chatbots. Network and Distributed System Security (NDSS) Symposium, (2024) 1-16.

[10] F. Wu, N. Zhang, S. Jha, P. McDaniel, C. Xiao (2024). A new era in llm security: Exploring security concerns in real-world llm-based systems. arXiv preprint arXiv:2402.18649. https://doi.org/10.48550/arXiv.2402.18649

[11] Y. Liu, G. Deng, Y. Li, K. Wang, Z. Wang, X. Wang, T. Zhang, Y. Liu, H. Wang, Y. Zheng, Y. Liu, (2023) Prompt Injection attack against LLM Integrated Applications. arXiv preprint. https://doi.org/10.48550/arXiv.2306.05499

[12] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, K. Lee, A. Roberts, T. Brown, D. Song, Ú. Erlingsson, A. Oprea, C. Raffel, Extracting training data from large language models. In 30th USENIX Security Symposium (USENIX Security 21), (2021) 2633-2650.

[13] J. Geiping, L. Fowl, W. R. Huang, W. Czaja, G. Taylor, M. Moeller, T. Goldstein, (2020) Witches' brew: Industrial scale data poisoning via gradient matching. arXiv preprint. https://doi.org/10.48550/arXiv.2009.02276

[14] E. Wallace, S. Feng, N. Kandpal, M. Gardner, S. Singh, (2019) Universal adversarial triggers for attacking and analyzing NLP. arXiv preprint. https://doi.org/10.48550/arXiv.1908.07125

[15] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan M. Diab, X. Li, X.V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P.S. Koura, A. Sridhar, T. Wang, L. Zettlemoyer, (2022) Opt: Open pretrained transformer language models. arXiv preprint. https://doi.org/10.48550/arXiv.2205.01068

[16] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, (2019) Roberta: A robustly optimized bert pretraining approach. arXiv preprint.

[17] J. Howard, S. Ruder, (2018) Universal language model fine-tuning for text classification.

arXivpreprint.

[18] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P.J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research, 21(140), (2020) 167.

[19] S.S. Kumar, M.L. Cummings, A. Stimpson, (2024) Strengthening LLM Trust Boundaries: A Survey of Prompt Injection Attacks. IEEE 4th International Conference on Human-Machine Systems (ICHMS), IEEE, Canada. https://doi.org/10.1109/ICHMS59971.2024.10555871

[20] K. Greshake, A. Sahar, M. Shailesh, E. Christoph, H. Thorsten, F. Mario, Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, (2023) 79-90. https://doi.org/10.1145/3605764.3623985

[21] J. Yi, Y. Xie, B. Zhu, E. Kiciman, G. Sun, X. Xie, F. Wu, (2023) Benchmarking and defending against indirect prompt injection attacks on large language models. arXiv preprint. https://doi.org/10.48550/arXiv.2312.14197

[22] F. Perez, I. Ribeiro, (2022) Ignore previous prompt: Attack techniques for language models. arXiv preprint. https://doi.org/10.48550/arXiv.2211.09527

[23] X. Sang, M. Gu, H. Chi, (2024) Evaluating prompt injection safety in large language models using the promptbench dataset.

[24] V. Benjamin, E. Braca, I. Carter, H. Kanchwala, N. Khojasteh, C. Landow, Y. Luo, C. Ma, A. Magarelli, R. Mirin, A. Moyer, K. Simpson, A. Skawinski, T. Heverin, (2024). Systematically Analyzing Prompt Injection Vulnerabilities in Diverse LLM Architectures. arXiv preprint. https://doi.org/10.48550/arXiv.2410.23308

[25] Q. Zhan, L. Zhixiang, Y. Zifan, K. Daniel, (2024) Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. arXiv preprint. https://doi.org/10.48550/arXiv.2403.02691

[26] J. Shi, Z. Yuan, Y. Liu, Y. Huang, P. Zhou, L. Sun, N. Z. Gong, (2024) Optimization-based Prompt Injection Attack to LLM-as-a-Judge. arXiv preprint.

[27] P. Rai, S. Sood, V. K. Madisetti, A. Bahga, Guardian: A multi-tiered defense architecture for thwarting prompt injection attacks on llms. Journal of Software Engineering and Applications, 17(1), (2024) 43-68. https://doi.org/10.4236/jsea.2024.171003

[28] N. Varshney, P. Dolin, A. Seth, C. Baral, (2023) The art of defending: A systematic evaluation and analysis of llm defense strategies on safety and over-defensiveness. arXiv preprint. https://doi.org/10.18653/v1/2024.findings-acl.776

[29] X. Liu, Z. Yu, Y. Zhang, N. Zhang, C. Xiao, (2024) Automatic and universal prompt injection attacks against large language models. arXiv preprint. https://doi.org/10.48550/arXiv.2403.04957

[30] X. Suo, Signed-Prompt: A New Approach to Prevent Prompt Injection Attacks Against LLMIntegrated Applications. arXiv preprint, 3194 (2024) 040013. https://doi.org/10.1063/5.0222987

[31] J. Piet, M. Alrashed, C. Sitawarin, S. Chen, Z. Wei, E. Sun, B. Alomair, D. Wagner, (2024) Jatmo: Prompt injection defense by task-specific finetuning. Computer Security – ESORICS 2024, 105–124. https://doi.org/10.1007/978-3-031-70879-4_6

[32] A. Robey, E. Wong, H. Hassani, G. J. Pappas, Smoothllm: Defending large language models against jailbreaking attacks. arXiv preprint. https://doi.org/10.48550/arXiv.2310.03684

[33] S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, Y. Li, A. Gupta, H. Han, S. Schulhoff, P.S. Dulepet, S. Vidyadhara, D. Ki, S. Agrawal, C. Pham, G. Kroiz, F. Li, H. Tao, A. Srivastava, H.D. Costa, S. Gupta, M. L. Rogers, I. Goncearenco, G. Sarli, I. Galynker, D. Peskoff, M. Carpuat, J. White, S. Anadkat, A. Hoyle, P. Resnik, (2024) The Prompt Report: A Systematic Survey of Prompting Techniques. arXiv preprint. https://doi.org/10.48550/arXiv.2406.06608

[34] C. Wang, S.K. Freire, M. Zhang, J. Wei, J. Goncalves, V. Kostakos, Z. Sarsenbayeva, C. Schneegass, A. Bozzon, E. Niforatos, (2023) Safeguarding Crowdsourcing Surveys from ChatGPT with Prompt Injection. arXiv preprint. https://doi.org/10.48550/arXiv.2306.08833

[35] D. Jurafsky, J. H. Martin, (2024) Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. with Language Models, 3rd edition.

[36] A.K. Uysal, G. Serkan, The impact of preprocessing on text classification. Information processing & management, 50(1), (2014) 104112. https://doi.org/10.1016/j.ipm.2013.08.006

[37] W.J. Wilbur, K. Sirotkin, The automatic identification of stop words. Journal of information science 18(1), (1992) 45-55. https://doi.org/10.1177/016555159201800106

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, Scikit-learn: Machine learning in Python. the Journal of machine Learning

research, 12, (2011) 2825-2830.

[39] J. Ramos, Using tf-idf to determine word relevance in document queries. In Proceedings of the first instructional conference on machine learning, 242(1), (2003) 29-48.

[40] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, (2013) 26.

[41] Y. Goldberg, O. Levy, (2014) word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. arXiv preprint. https://doi.org/10.48550/arXiv.1402.3722

[42] W.A. Qader, M.M. Ameen, B.I. Ahmed, (2019) An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges. In 2019 international engineering conference (IEC), IEEE, Iraq. https://doi.org/10.1109/IEC47844.2019.8950616

Authors Contribution Statement

All authors contributed to the study's conception and design. Material preparation, data analysis, and testing were performed by all authors. The manuscript was written and revised by all authors on previous versions of the manuscript.

Funding

The authors declare that no funds, grants or any other support were received during the preparation of this manuscript.

Competing Interests

The authors declare that there are no conflicts of interest regarding the publication of this manuscript.

Data Availability

The data supporting the findings of this study can be obtained from the corresponding author upon reasonable request.

Has this article screened for similarity?

Yes

About the License