# Detecting and Mitigating Prompt Injection Attacks using quality dataset

Safwan Shaheer
Department of Computer Science (CS)
School of Data and Sciences (SDS)
Dhaka, Bangladesh
safwan.shaheer@g.bracu.ac.bd

G. M. Refatul Islam
Department of Computer Science and Engineering (CSE)
School of Data and Sciences (SDS)
Dhaka, Bangladesh
gm.refatul.islam@g.bracu.ac.bd

Mohammad Rafid Hamid
Department of Computer Science and Engineering (CSE)
School of Data and Sciences (SDS)
Dhaka, Bangladesh
mohammad.rafid.hamid@g.bracu.ac.bd

Md. Abrar Faiaz Khan
Department of Computer Science and Engineering (CSE)
School of Data and Sciences (SDS)
Dhaka, Bangladesh
md.abrar.faiaz.khan@g.bracu.ac.bd

Md. Omar Faruk
Department of Computer Science and Engineering (CSE)
School of Data and Sciences (SDS)
Dhaka, Bangladesh
md.omar.faruk@g.bracu.ac.bd

Abstract—Prompt injection attacks are a growing concern in various domains, as they can compromise the security and stability of critical systems, from power grids to large scale web applications. In this research, we propose a novel and comprehensive approach for detecting and mitigating prompt injection attacks by creating a highly curated and high quality dataset, leveraging techniques from the "Orca: Progressive Learning from Complex Explanation Traces of GPT-4" paper. We build upon pre-existing datasets from HuggingFace, namely HackAPrompt-Playground-Submissions, and HackAPrompt-AICrowd-Submissions, and augment them using the Orca techniques. Our proposed solution aims to improve the detection and mitigation of prompt injection attacks, ensuring the security and stability of the targeted systems.

Index Terms—prompt injection attacks, dataset augmentation, huggingFace datasets, progressive learning, detection & mitigation, llm alignment

## I. Introduction

Prompt injection attacks have emerged as a pressing and formidable threat, eliciting mounting concern due to their potential to jeopardize the security and stability of critical systems across diverse domains, encompassing essential infrastructure such as power grids and mission-critical web applications. These attacks exploit the vulnerabilities of Large Language Models (LLMs) like GPT-3.5, BERT etc. which are advanced AI models designed to generate human-like text by processing and understanding vast amounts of training data.

Prompt injection involves maliciously manipulating or injecting harmful content into the initial prompt provided to LLMs, with the intention of influencing their generated output. This manipulation can lead to biased, misleading, or even harmful responses, compromising the integrity and reliability of decision-making processes. The impact of prompt injection attacks is far-reaching, undermining trust in AI-generated content and potentially leading to misinformation, compromised security, and harmful outcomes.

Prompt engineering works due to the inherent nature of LLMs. These models leverage deep learning techniques to analyze patterns, context, and semantics in text, enabling them to generate coherent and contextually relevant responses. Attackers exploit this capability by carefully crafting prompts to introduce biased or malicious content, influencing the model's output in undesirable ways.

In this research, we present an approach to effectively detect and mitigate prompt injection attacks on LLMs. To tackle the challenges posed by these attacks, we focus on constructing a meticulously curated and exceptionally high-quality dataset. To achieve this, we draw inspiration from the "Orca: Progressive Learning from Complex Explanation Traces of GPT-4" paper and leverage its techniques. In order to build upon existing resources, we incorporate pre-existing datasets from HuggingFace, namely HackAPrompt-Playground-Submissions and HackAPrompt-AICrowd-Submissions. By applying the Orca techniques, we enhance and augment these datasets, ensuring they capture the diverse range of prompt injection attack scenarios accurately. This approach

empowers our research to offer improved detection and mitigation strategies, fortifying the security and stability of critical systems against prompt injection attacks.

Neglecting to tackle prompt injection attacks can have severe consequences. It not only compromises the reliability of AI-generated content but also perpetuates biases, hindering progress towards fairness and equity in LLMs. Our research aims to contribute to the field by providing effective detection and mitigation strategies, ultimately ensuring the responsible use and deployment of LLMs in critical systems.

## II. Related Work

The literature on security risks in Large Language Models (LLMs) has identified various attack vectors, such as jailbreaks, privacy attacks, and prompt injection attacks. In this section, we review the key papers that address these aspects and their implications on LLM-integrated applications.

### A. Jailbreaks in LLMs

The paper "Tricking LLMs into Disobedience: Understanding, Analyzing, and Preventing Jailbreaks" [1] presents a formalism and taxonomy for jailbreaks in LLMs, surveying existing jailbreak methods and their effectiveness on open-source and commercial LLMs, such as GPT 3.5, OPT, BLOOM, and FLAN-T5-xxl. The study aims to bridge the gap in understanding and analyzing these attacks and their mitigations, proposing a limited set of prompt guards and discussing their effectiveness against known attack types.

### B. Privacy Attacks on ChatGPT

"Multi-step Jailbreaking Privacy Attacks on ChatGPT" [2] investigates the privacy threats from OpenAI's ChatGPT and the New Bing enhanced by ChatGPT. The authors follow previous training data extraction attacks on ChatGPT and propose a multi-step jailbreaking prompt to extract personal information. The study shows that application-integrated LLMs can act as effective misinformation generators, leading to a significant degradation in the performance of Open-Domain Question Answering (ODQA) systems.

### C. Indirect Prompt Injection

In "Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection" [3], the authors introduce the concept of Indirect Prompt Injection (IPI) to compromise LLM-integrated applications. The study deconstructs the complexities and implications of prompt injection attacks on actual LLM-integrated applications and forms HouYi, a novel black-box prompt injection attack technique, which draws inspiration from existing methods.

### D. Prompt Injection Attacks

"Prompt Injection attack against LLM-integrated Applications" [4] provides an in-depth analysis of the security risks associated with prompt injection attacks on actual LLM-integrated applications and their potential impact on LLM-integrated systems. Prompt injection attacks aim to elicit unintended responses from LLM-based tools, potentially achieving unauthorized access, manipulating responses, or deceiving users.

### E. Goal Hijacking and Prompt Leaking

"Ignore Previous Prompt: Attack Techniques For Language Models" [5] investigates two types of attacks—goal hijacking and prompt leaking—against LLMs like GPT-3. The authors propose a framework called PromptInject for mask-based iterative adversarial prompt composition. The study demonstrates that even low-aptitude but ill-intentioned agents can easily exploit GPT-3's stochastic nature, creating long-tail risks. The code for PromptInject is available for further research and analysis.

These papers highlight the growing concerns and challenges related to security risks in LLMs. As LLMs continue to advance and become more integrated into various applications, it is crucial for researchers and developers to address these security risks and develop effective mitigation strategies to ensure the safe and responsible use of these powerful models.

## III. Methodology

### A. Integrating ORCA techniques for Dataset Augmentation

The paper "Orca: Progressive Learning from Complex Explanation Traces of GPT-4" presents a novel approach to enhance the capability of smaller models through imitation learning, drawing on the outputs generated by large foundation models (LFMs) like GPT-4 [6]. The model, named Orca, learns from detailed output generated by GPT-4, including explanation traces, step-by-step thought processes, and other complex instructions, guided by teacher assistance from ChatGPT. We will follow the steps below in order to integrate ORCA's technique in our dataset augmentation pipeline:-

- Generate detailed explanations and step-by-step processes related to prompt injection attacks using GPT-4.
- Select and sample data points carefully to ensure a well-balanced and representative dataset.
- Use ChatGPT's guidance to create diverse and informative data points.
- Add the new data points to the existing dataset to improve its quality and coverage of prompt injection attack situations.

B. Preprocessing and Augmentation of HackAPrompt-Playground-Submissions Dataset

The HackAPrompt-Playground-Submissions dataset is a part of the HackAPrompt 2023 competition, which focuses on AI safety and prompt injection attacks [7]. The dataset contains user-generated prompts, user inputs, and completions, along with expected completions, token counts, and labels indicating whether the prompt injection was successful or not. We will follow the steps below in order to integrate LIMA's technique in our synthetic data generation pipeline:-

- Data Deduplication: Remove duplicate entries in the dataset to ensure diversity and prevent overfitting.
- Data Cleaning: Filter out any irrelevant or noisy data, such as entries with incorrect labels or missing information.
- Data Visualization: Analyze the dataset using visualizations to identify patterns, trends, and potential issues.
- Dataset Augmentation: Apply the ORCA methodologies to generate additional features. For instance, using ORCA to generate complex explanation traces and step-by-step thought processes.
- Classify Types of Prompt Injection Attacks: Analyze and classify the types of prompt injection attacks present in the dataset to better understand the attack patterns and improve detection accuracy.
- Feature Engineering: Introduce new features to the dataset that can help improve the model's accuracy in detecting prompt injection attacks. Some potential features include: Time-based features, Text-based features and statistical features
- Quality Assessment: Continuously check the quality of the augmented dataset, ensuring that it maintains a high standard.

Data Cleaning: A comprehensive data cleaning procedure was performed for maintaining the standard and integrity of the dataset. To ensure subsequent inspection and clarification to be precise and authentic this action is committed to remove irrelevant and noisy data.

- In the beginning, some columns have been removed from the dataset they seemed to be needless into the research objectives.

- Base on the 'level' column the dataset have been classified to make it easier for analysis and this sorting in ascending order of difficulty level grant us for a organized investigation of the data.

- By removing any rows with 'null' values, The dataset was cleaned up. This process made sure that only full relevant data points would be included for further analysis.

- To standardize the data, many preprocessing steps were used in some columns. New columns were created that were the copy of 'completion' and 'expected_completion' columns and those columns were changed to lowercase. Additionally, non-alphabetic characters as well as preceding and following whitespaces were eliminated from the 'completion_lower' column.

- The comparison of the 'completion_lower' and 'expected_completion_lower' columns was an essential step in the data cleansing process. Entries were noted as 'True' when the two values matched. it indicates that the prompts had been correctly completed.

- To improve the dataset's logical flow and connect it with the study goals, column reordering was done.

Through these data cleaning steps, the dataset was meticulously refined to eliminate noise and inconsistencies, setting the stage for subsequent analysis and meaningful interpretation.

## IV. Data Visualization

### A. Analysis of Model Performance on Answer Correctness

In our evaluation of various language models' performance, we analyzed the distribution of correct versus incorrect answers. Our primary focus was on three models: FlanT5-XXL, gpt-3.5-turbo, and text-davinci-003. The FlanT5-XXL and gpt-3.5-turbo models showcased relatively comparable performance in terms of their accuracy ratios, with FlanT5-XXL marginally outperforming gpt-3.5-turbo. However, both models exhibited a high absolute count of incorrect answers. In contrast, the text-davinci-003 model, while producing fewer overall answers, had the lowest accuracy percentage among the three.

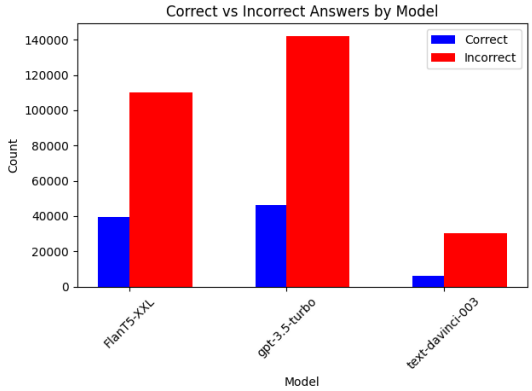


Fig. 1: Correct vs Incorrect Answers by models

B. Examination of Correlation Coefficients: Correctness vs. Token Count

Our findings suggest a marginal negative relationship between the correctness of an answer and its token count ($r = -0.054$). This intimates that as answers slightly increase in length, there's a negligible propensity for them to be incorrect. Conversely, shorter answers exhibit a minuscule tendency towards correctness. Nevertheless, the magnitude of this correlation is proximate to 0, indicating that the association is considerably weak.

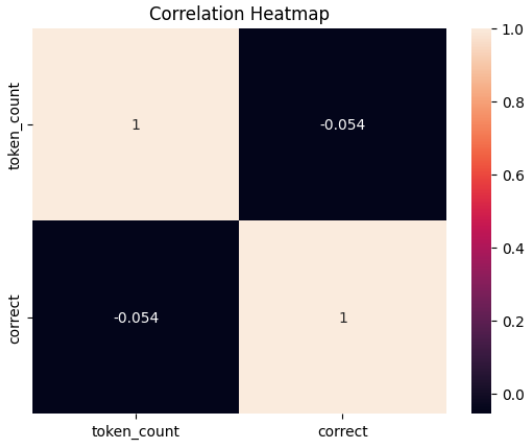


Fig. 2: Correlations of correct and token count

C. Analysis of Answer Length by Correctness: Insights from a Boxplot

The disparities in token count distributions between correct and incorrect categorizations emphasize the model's variable response lengths. The outliers in incorrect answers suggest potential challenges the model faces with extended input. This revelation merits further investigation into model behaviors and the factors influencing these variances.

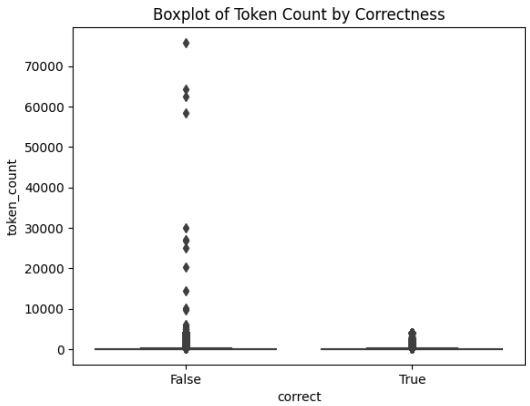


Fig. 3: Box plot of token count by correctness

D. Analysis of model result grouped by model name and level

We calculate the mean of integer version of the correct column (0 or 1) for all the different groups of model and levels. We then create a bar plot to visually represent the mean results of different models at various levels. The bar-graph indicates that the only levels 0 to 9 have decently high means. Having a high mean is good in the sense that it means prompt injections were successful and this will allow our classification models learn more about the types of prompts the model is more susceptible to.

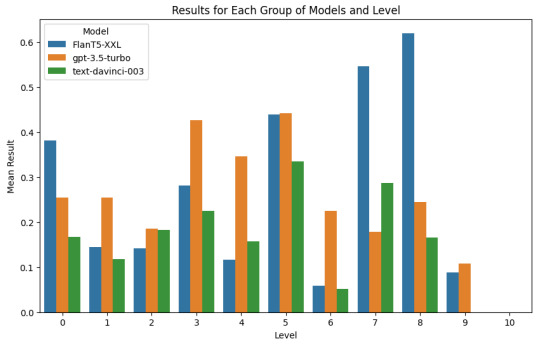


Fig. 4: Box plot of token count by correctness

## V. Training Process

In our research, we aimed to detect and mitigate prompt injection attacks using a combination of pre-trained sentence embeddings and custom neural network architectures. The training process involved the following steps:

1) Data Preprocessing: We preprocessed the text data by removing newline characters and splitting it into training and testing sets using an 80/20 split. This ensured that our model had a clean and structured input for training and evaluation.
2) Generating Sentence Embeddings: We used the SentenceTransformer library with the 'all-MiniLM-L6-v2' pre-trained model to generate sentence embeddings for our dataset. This model has been shown to perform well in various NLP tasks, including sentiment classification. By using these embeddings as input to our custom neural network, we were able to effectively capture the semantic meaning of the text.
3) Defining the Neural Network Architecture: We designed a custom LSTMClassifier model, which is a type of recurrent neural network (RNN) architecture. This model is particularly suitable for sequence-based tasks, such as text classification. The LSTMClassifier consists of an LSTM layer followed by a fully connected layer for classification. The LSTM layer is responsible for capturing the sequential information in the input embeddings, while the

fully connected layer maps the LSTM output to the final class probabilities.

4) Hyperparameters: We used a hidden size of 128 for the LSTM layer, which determines the number of hidden units in the layer. A batch size of 32 was used for training, which is the number of samples processed before updating the model's weights. The learning rate was set to 1e-4, which controls the step size during the optimization process. We trained the model for 500 epochs, which is the number of times the entire dataset is passed through the model during training.
5) Training the Model: We trained the LSTMClassifier using the generated sentence embeddings as input and the corresponding labels as targets. During training, the model learned to capture the relationships between the input embeddings and the target labels, allowing it to effectively classify the text based on the presence of prompt injection attacks.
6) Evaluation: We evaluated the performance of our model on the test dataset, which was not used during training. This allowed us to assess the model's ability to generalize to new, unseen data. We tracked various performance metrics, such as accuracy, precision, recall, and F1 score, to gain a comprehensive understanding of the model's performance.
7) Experimenting with classical ML Models: We also experimented with classical machine learning models like Random Forest Classifier (RFC) and Naive Bayes Classifier for the classification task. In this case, we did not use Sentence Transformers embeddings but relied on the CountVectorizer to convert the text data into a matrix of token counts. This approach allowed us to compare the performance of different models and feature extraction techniques for the prompt injection detection task.

## VI. Evaluation and Conclusion

The evaluation of the LSTM and FNN models on the task of detecting prompt injection attacks yielded interesting findings. Both models exhibited relatively comparable performance metrics, indicating their ability to discern instances of attacks from benign prompts. The LSTM model achieved an accuracy of 75%, precision of 73%, recall of 75%, and an F1-score of 71%. On the other hand, the FNN model demonstrated an accuracy of 74%, precision of 72%, recall of 74%, and an F1-score of 71%.

These results highlight the potential of both models to effectively identify prompt injection attacks, albeit with slight variations in their performance. It is evident that LSTM slightly outperformed the FNN in terms of accuracy, precision, and recall. The LSTM's ability to capture sequential dependencies within the data might have contributed to this. However, the differences in performance between the two models were marginal. Also both models actually havent performed very very well as their scores are all $< 76\%$. The performances needs to be improved to be able to use them in production grade applications. To that end we suggest the following future work:

- Model Hybridization: Investigate hybrid models that combine the strengths of both LSTM and FNN architectures, potentially leverage the sequential analysis capabilities of LSTM while benefiting from the simpler architecture of FNN.

- Feature Engineering: Explore the incorporation of domain-specific features or embeddings to enhance the models' understanding of prompt semantics. This could involve pre-trained language models or customized embeddings tailored to the context of prompt injection attacks.

- Ensemble Techniques: Explore ensemble techniques, such as model averaging, to use collective power of multiple models for improved detection accuracy.

- Adversarial Attacks: Investigate the robustness of the developed models against adversarial attacks specifically designed to evade detection. Adversarial training and evaluation can enhance the models' resistance to real-world evasion attempts.

- Real-time Detection: Develop real-time detection systems that can process prompts and identify potential attacks in dynamic, rapidly changing environments.

- Scalability and Generalization: Extend the evaluation to larger and more diverse datasets to assess the models' scalability and generalization capabilities across different domains and languages.

## References


[1] A. Rao, S. Vashistha, A. Naik, and S. Ray, "Tricking llms into disobedience: Understanding, analyzing, and preventing jailbreaks," 2023.

[2] H. Li, D. Guo, W. Fan, M. Xu, J. Huang, and Y. Song, "Multi-step jailbreaking privacy attacks on chatgpt," 2023.

[3] K. Greshake, M. Backes, and Y. Zhang, "Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection," 2023.

[4] ——, "Prompt injection attack against llm-integrated applications," 2023.

[5] F. Perez and I. Ribeiro, "Ignore previous prompt: Attack techniques for language models," 2022.

[6] S. Mukherjee, A. Mitra, G. Jawahar, S. Agarwal, H. Palangi, and A. Awadallah, "Orca: Progressive learning from complex explanation traces of gpt-4," 2023.

[7] "HackAPrompt competition: A step towards AI safety," https://pub.towardsai.net/hackaprompt-competition-a-step-towards-ai-safety-45ae56b40791, May 2023, accessed: 2023-7-11.