**Minor Project Report**

**on**

**LLM SECURITY-PROMPT INJECTION DETECTION**

Submitted to Guru Gobind Singh Indraprastha University, Delhi (India)

In partial fulfillment of the requirement for the award of the degree

of

**Bachelor of Technology**

**in**

**Information Technology**

Under the guidance of

**Dr. Sitender Malik**

**Assistant Professor**

**Information Technology**

Submitted By:

**Anshu Jigyasa Kaur Chawla Viresh Gupta**

**(00296307722) (00396307722) (03596303121)**

**MAHARAJA SURAJMAL INSTITUTE OF TECHNOLOGY**

**C4, JANAKPURI, NEW DELHI – 110058**

(GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY,

DWARKA, NEW DELHI)

**(November, 2024)**

Maharaja Surajmal Institute Of Technology Department Of Information Technology

**VISION OF THE INSTITUTE**

To become one of the most admired centers of academic excellence in the field of Engineering & Technology for all-round professional development of students to enable them to meet the growing technological needs of the country.

**MISSION OF THE INSTITUTE**

Developing new paradigm in imparting education in the fields of Engineering and Technology and to imbibe national values leading to student empowerment, with a view to prepare them to meet the national and global challenges.

**VISION OF THE DEPARTMENT**

To build a culture of innovation and research in students and make them capable to solve upcoming challenges of human life using computing.

**MISSION OF THE DEPARTMENT**

**M1**. To develop 'educational pathways' so that students can take their career towards success.

**M2.** To imbibe curiosity and support innovativeness by providing guidance to use the technology effectively.

**M3.** To inculcate management skills, integrity, honesty and human values through curricular, co-curricular and extra-curricular activities.

**PROGRAM EDUCATIONAL OBJECTIVES**

**PEO1.** Graduates of IT program are prepared to be employed by IT industries and be engaged in learning, understanding and applying new ideas.

**PEO2.** The graduates are prepared to perform effectively as individuals and team members in the workplace, growing into highly technical or project management and leadership roles.

**PEO3.** Graduates are prepared to apply basic principles of practices of computing grounded in mathematics and science for successfully completing software related projects to satisfy customer business objectives and productively engage in research.

**PEO4.** Graduates are prepared to pursue higher studies so that they can contribute to the teaching profession, research & development of information technology and other allied fields.

**PROGRAM OUTCOMES**

Engineering Graduates will be able to:-

1. **Engineering knowledge:**- Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:-** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:**- Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:-** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:-** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:-** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:-** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:-** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:-** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:-** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:-** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.**Life-long learning:-** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**PROGRAM SPECIFIC OUTCOME**

**PSO-1:-** Ability to understand the principles and working of hardware and software aspects in information technology.

**PSO-2:-** Ability to explore and develop innovative ideas to solve real world problem using IT skills.

**Maharaja Surajmal Institute of Technology**

**Department of Information Technology**

**Name of the Subject**: Minor Project **Subject Code**: ES-451

**Semester/Year**: 7th /4th **External Marks**: 60

**Internal Marks**: 40

**Course Objective**

The objective of the Minor Project is to design or implement application / software using computer technologies.

**Course Outcomes**

1. To be able to design a software system, component, or process to meet desired needs.
2. To be able to work on multidisciplinary Problems.
3. To be able to work as individual and Team Member.
4. To be able to use various modern engineering tools.
5. To be able to develop written and verbal skills.
6. To be able to learn professional and research ethics.

CANDIDATE’S DECLARATION

It is hereby certified that the work presented in the B.Tech. Minor Project report entitled **LLM SECURTIY-PROMPT INJECTION DETECTION** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** and submitted in the **Department of Information Technology, MSIT New Delhi (Affiliated to Guru Gobind Singh Indraprastha University, New Delhi)** is an authentic record of our work carried out during a period from **August** **2024 to November 2024** under the guidance of **Dr. Sitender Malik(Assistant Professor).**

The matter presented in the B.Tech. Minor Project Report has not been submitted by us for the award of any other degree of this or any other institute.

**Anshu Jigyasa Kaur Chawla Viresh Gupta**

**(00296307722) (00396307722) (03596303121)**

This is to certify that the above statement made by the candidates is correct to the best of my knowledge. They are permitted to appear in the External Minor Project Examination.

**Dr. Sitender Dr. Sunesh Malik**

**(Assistant Professor) (HOD, IT Department)**

The B.Tech. Minor Project Viva-Voce Examination of **Anshu(00296307722), Jigyasa Kaur Chawla(00396307722), and Viresh Gupta(03596303121)** has been held on **25/11/2024.**

**Project Coordinator (Signature of External Examiner)**

ACKNOWLEDGEMENT

We express our deep gratitude to **Dr. Sitender Malik**, Assistant Professor, Department of Information Technology for his valuable guidance and suggestions throughout our project work. We are thankful to the Project Coordinator for his valuable guidance.

We would like to extend our sincere thanks to the **Head of Department, Dr. Sunesh Malik** for her time-to-time suggestions to complete our project work. We are also thankful to **Prof. Archana Balyan, Director MSIT** for providing us with the facilities to carry out our project work.

**Anshu Jigyasa Kaur Chawla Viresh Gupta**

**(00296307722) (00396307722) (03596303121)**

ABSTRACT

Prompt injection attacks pose a significant threat to the safe and ethical deployment of large language models (LLMs) by manipulating their outputs to produce unintended, harmful, or malicious responses. This project proposes a comprehensive solution to proactively safeguard LLMs against such attacks through a multi-pronged approach: classification using Logistic Regression, a heuristic-based keyword detection mechanism, and embedding-based vector comparisons.

The classification model identifies malicious prompts by analyzing patterns in labeled datasets, achieving high accuracy in detection. The heuristic approach leverages domain-specific keyword lists and similarity scoring to flag suspicious prompts. In addition, embedding-based vector comparisons analyze the input by converting it into numerical representations (vectors) and measuring its closeness to known harmful prompts. This vector similarity score is calculated directly using cosine similarity, which is then combined with heuristic flags to strengthen detection accuracy.

Together, these methods form a robust defense mechanism against prompt injection. The system integrates these techniques into a unified decision-making framework, supported by a user-friendly interface, to ensure real-time, accurate identification of malicious inputs. Experimental results demonstrate the efficacy of this hybrid approach in mitigating risks posed by prompt injection. The proposed solution enhances the reliability and security of LLMs, paving the way for their safer adoption in real-world applications.

LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| **Table No** | **Title of the Table** | **Page Number** |
| 1 | Logistic Regression Accuracy | 76 |
| 2 | Accuracy Comparision of Models | 76 |

LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Fig. No.** | **Title of the Figure** | **Page Number** |
| 1 | Project Workflow | 63 |
| 2 | User Interface Flowchart | 64 |
| 3 | System Internal Workflow | 65 |
| 4 | Logistic Regression Implementation Code | 67 |
| 5 | Heuristics Implementation Code | 67 |
| 6 | Code Implementation Snippet - 1 | 68 |
| 7 | Code Implementation Snippet – 2 | 69 |
| 8 | Code Implementation Snippet – 3 | 70 |
| 9 | Code Implementation Snippet – 4 | 71 |
| 10 | Code Implementation Snippet – 5 | 72 |
| 11 | Code Implementation Snippet - 6 | 72 |
| 12 | Confusion Matrix | 75 |
| 13 | Model Evaluation | 78 |

LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| **Abbreviation** | **Full form of Abbreviation** |
| LLM | Large Language Model |
| BERT | Bidirectional Encoder Representations from Transformers |
| AI | Artificial Intelligence |
| ANN | Approximate Nearest Neighbors |
| HNSW | Hierarchical Navigable Small World |
| GPT | Generative Pre-trained Transformer |
| USE | Universal Sentence Encoder |
| SVM | Support Vector Machine |

**CONTENTS**

[VISION AND MISSION I](#_Toc183325593)

[CANDIDATE’S DECLARATION V](#_Toc183325594)

[ACKNOWLEDGEMENT VI](#_Toc183325595)

[ABSTRACT VII](#_Toc183325596)

[LIST OF TABLES VIII](#_Toc183325597)

[LIST OF FIGURES IX](#_Toc183325598)

[LIST OF ABBREVIATIONS X](#_Toc183325599)

[Chapter 1: Introduction 2](#_Toc183325600)

[**1.1.** **Background** 2](#_Toc183325601)

[**1.2.** **Motivation** 7](#_Toc183325602)

[**1.3.** **Problem Statement** 9](#_Toc183325603)

[**1.4.** **Objectives** 10](#_Toc183325604)

[**1.5.** **Scope of the Project** 12](#_Toc183325605)

[Chapter 2: Literature Review 15](#_Toc183325606)

[**2.1.** **Existing Works** 16](#_Toc183325607)

[**2.2.** **Comparison of Existing Research with Project Implementation** 19](#_Toc183325608)

[Chapter 3: Methodology 24](#_Toc183325609)

[**3.1.** **Purposed solution** 25](#_Toc183325610)

[**3.2. MEthods** 27](#_Toc183325611)

[Chapter 4: System Design / Implementation 41](#_Toc183325612)

[**4.1.** **System Architecture** 41](#_Toc183325613)

[**4.2.** **Implementation details** 42](#_Toc183325614)

[Chapter 5: Results and Discussion 44](#_Toc183325615)

[**5.1.** **Analysis of result** 44](#_Toc183325616)

[**5.2.** **Discussion on challenges and limitations** 45](#_Toc183325617)

[Chapter 6: Conclusion and Future Work 50](#_Toc183325618)

[**6.1.** **Summary of the work done** 50](#_Toc183325619)

[**6.2.** **Limitations of the project** 50](#_Toc183325620)

[**6.3.** **Suggestions for future work** 51](#_Toc183325621)

[Chapter 7: References/Bibliography 52](#_Toc183325622)

Chapter 1: Introduction

As large language models (LLMs) continue to revolutionize natural language processing, robust security measures are paramount. Among the vulnerabilities identified in LLMs is prompt injection, a technique where adversarial inputs manipulate the model’s behavior, often bypassing intended safeguards (Henderson et al., 2023). This exploitation not only threatens the integrity of model outputs but also raises ethical concerns regarding user safety and data privacy.

To combat these challenges, this project proposes a novel approach that leverages prompt-based induction and heuristic methods for detecting and mitigating prompt injection attacks. Prompt-based induction refers to generating new prompts based on existing input patterns to discern the underlying behavioral trends of the model (Brown et al., 2020). By understanding these patterns, we can develop heuristic methods that identify suspicious inputs, enhancing the model’s ability to resist manipulation.

Additionally, the use of vector embeddings is crucial in this approach. Vector embeddings are numerical representations of text in high-dimensional space, capturing semantic relationships between words and phrases. These embeddings allow the system to compute similarity between prompts efficiently by comparing their contextual embeddings using metrics like cosine similarity or Euclidean distance (Chen et al., 2021). By leveraging this approach, the detection process becomes streamlined, and harmful inputs resembling known malicious patterns can be quickly flagged.

This synergy between prompt-based induction and vector embeddings not only simplifies the detection process but also enhances the adaptability of models to emerging threats. This project aims to establish a comprehensive framework that employs these methodologies to bolster LLM security, paving the way for safer interactions with AI systems.

## **Background**

As **large language models (LLMs)** continue to advance natural language processing (NLP), their deployment has expanded into diverse applications, including chatbots, virtual assistants, content generation, and automated decision-making systems. Despite their capabilities, these models are vulnerable to adversarial manipulations, with **prompt injection** emerging as a significant threat. Understanding and mitigating such risks is a critical area of research in LLM security.

**Prompt Injection -** Prompt injection exploits the inherent design of LLMs to follow input instructions flexibly and contextually. Adversaries craft malicious prompts to manipulate the model's behavior, overriding pre-defined instructions or ethical safeguards. This attack vector has become more prominent due to:

1. **Instruction-Following LLMs:** Modern LLMs like OpenAI’s GPT series and others trained with instruction-tuning are particularly susceptible because they prioritize adherence to user-provided commands.
2. **Open-Ended Input Paradigm:** The unstructured nature of input allows adversaries to disguise harmful instructions within normal text or external content (e.g., websites or documents).

**Types of Prompt Injection Threats:**

* Generating harmful or misleading information.
* Circumventing system safeguards to expose sensitive data or execute unauthorized commands.
* Propagating misinformation or violating ethical guidelines.

**Emergence of LLM Security Challenges**

The vulnerabilities exploited by prompt injection are part of broader security concerns in LLMs:

1. **Over-Reliance on Pre-Trained Data:** LLMs lack inherent understanding, relying on probabilistic patterns in pre-trained data. This allows adversarial manipulation to influence outputs.
2. **Generalization vs. Specialization:** LLMs generalize well across contexts, but this flexibility also means they may unintentionally execute adversarial instructions embedded within prompts.
3. **Dynamic and Evolving Threats:** The rapid evolution of attack techniques, including indirect prompt injection, makes it challenging to design static defenses.

**Significance of LLM Security**

Prompt injection detection is part of the broader field of **LLM security**, which aims to ensure the safe and ethical deployment of these systems. This includes:

* Preserving trust and reliability in AI interactions.
* Safeguarding user data privacy and preventing information leakage.
* Ensuring adherence to ethical guidelines and preventing misuse in malicious or deceptive applications.

Addressing prompt injection and LLM vulnerabilities is crucial to unlocking the full potential of AI while maintaining public trust in its safe and responsible use. This research forms the foundation for building resilient AI systems that can withstand evolving threats in the digital landscape.

Below are the key areas of background knowledge that would be necessary:

1. **Introduction to Large Language Models (LLMs)**

* **Understanding LLMs**: A solid grasp of how LLMs function, including their architecture (e.g., GPT, BERT), training process, and applications. (Devlin et al., 2018)
* **Common Vulnerabilities in LLMs**: Insight into inherent risks of LLMs, including bias, adversarial manipulation, and prompt injection vulnerabilities. (Zhang et al., 2019)(Wallace et al., 2019)

1. **Adversarial Attacks in NLP**

* **Prompt Injection Attacks**: Understanding how prompt injection attacks work, where attackers manipulate the input prompt to produce unintended or harmful outputs from the model.(Abdelnabi et al., 2023)(Rossi et al., 2024)
* **Adversarial Methods**: Different techniques for generating adversarial inputs, such as gradient-based attacks, token manipulation, and prompt construction strategies.(TextFooler: A Text-Generating Adversarial Attack Method for Natural Language Processing, 2020)(Adversarial Examples for Evaluating Reading Comprehension Models, 2021)

1. **Machine Learning for Security in NLP**

* **Machine Learning Models for Detection**: Learning how classification models like Logistic Regression, SVM, and neural networks can be applied to detect malicious prompts. (Zhang et al., 2019)(Rahman et al., 2024)
* **Training and Evaluation**: Approaches to training machine learning models on datasets containing both benign and malicious prompts, and evaluating model performance using metrics like accuracy, precision, and recall. (Adlam et al., 2022)(Muller, 2016)

1. **Heuristic and Rule-Based Approaches**

* **Heuristic Methods**: Defining rules or patterns to identify prompt injections, such as keyword-based checks, abnormal input pattern detection, and manual oversight.(Yi et al., 2023)
* **Pattern Recognition**: Leveraging pattern recognition techniques to identify suspicious behaviors in text input, such as the presence of uncommon tokens or suspicious structures.(Kowsari et al., 2019)

1. **Natural Language Processing (NLP) Techniques**

* **Tokenization and Embeddings**: Understanding how text is represented in models through tokenization and word embeddings, as well as how these methods can be used to identify anomalous patterns in text. (Devlin et al., 2018)(Word2Vec: Distributed Representations of Words and Phrases and Their Compositionality, 2013)
* **Text Similarity and Embedding-Based Search**: Techniques to calculate semantic similarity using embeddings (e.g., cosine similarity), and how to compare new prompts with previously flagged malicious prompts in a vector database.(Maas et al., 2011)(Cao, 2024)

1. **Vector Embeddings**

* **Definition**: Vector embeddings are numerical representations of data (e.g., text, images) in a high-dimensional space, capturing their semantic or contextual relationships. These embeddings enable comparison and clustering based on their proximity in the vector space.
* **Applications**: Widely used in natural language processing, computer vision, and recommendation systems for tasks like similarity measurement, classification, and clustering.

1. **Similarity Search Algorithms**

* **Definition**: Algorithms used to identify vectors in a dataset that are most similar to a query vector, enabling tasks like nearest-neighbor searches.
* **Key Algorithms**:
  + Approximate Nearest Neighbors (ANN): A widely used approach for efficien t searches within large embedding spaces. It reduces computational cost while maintaining high accuracy (Malkov & Yashunin, 2016; Zhang et al., 2022).
  + Hierarchical Navigable Small World (HNSW): A specific ANN method known for its scalability and speed in large datasets.

1. **Defensive Strategies and System Integration**

* **Defensive Strategies for LLMs**: A thorough understanding of defense mechanisms such as input filtering, adversarial training, and prompt sanitization.(Zhao et al., 2024)(Goyal et al., 2022)(Kraidia et al., 2024)
* **System Integration and Deployment**: Best practices for integrating defense mechanisms with LLMs and deploying them in a production environment, including security considerations and performance trade-offs.https://neptune.ai/blog/mlops-best-practices(Ogundepo, 2022)(Express Computer, 2024)

1. **Evaluation and Testing of Defense Mechanisms**

* **Evaluation Metrics for Defenses**: Understanding how to evaluate the effectiveness of different defenses, including accuracy, false positive/negative rates, and robustness to adversarial manipulation.(Aldeida, 2023)(Srivastava, 2019)
* **Continuous Testing and Monitoring**: Developing strategies for continuous evaluation of the deployed system to detect and respond to new attack vectors.(Kaur et al., 2023)(Xia et al., 2024)

By covering these topics, you'll acquire the essential background knowledge needed to build and deploy an effective defense system against prompt injection attacks in LLMs.

## **Motivation**

Large language models (LLMs) have revolutionized the field of artificial intelligence by enabling a wide range of applications, from chatbots to content generation, code completion, and decision support. However, with their increasing adoption, these models face a critical security challenge: vulnerability to prompt injection attacks. In these attacks, malicious users craft specific inputs (prompts) designed to manipulate the model's behavior, leading to the generation of harmful, biased, or unintended outputs. The potential for such attacks can undermine trust in AI systems, especially when deployed in sensitive environments such as healthcare, finance, and legal services.

Given the widespread deployment of LLMs across diverse industries, it is crucial to ensure that these models remain robust against such vulnerabilities. Prompt injection attacks are particularly concerning because they are subtle and often exploit the intricate nature of natural language processing (NLP) models, which rely heavily on input context and token relationships. Therefore, the motivation behind this project is to develop a comprehensive defense mechanism that can safeguard LLMs from such attacks by leveraging machine learning-based classification, heuristic approaches, and vector database searches.

**Key Motivating Factors:**

1. **Security and Reliability**: As LLMs are deployed in real-world applications, ensuring their security is paramount. Without appropriate safeguards, malicious actors could use these models to spread misinformation, generate harmful content, or manipulate decisions. The development of defense against prompt injection is essential for maintaining the integrity and trustworthiness of LLM-powered systems of Current Solutions\*\*: While existing methods for securing LLMs focus primarily on adversarial training or input sanitization, many of these techniques are either resource-intensive or insufficiently robust against sophisticated prompt injection methods. This gap in current defense strategies presents a critical challenge that the project aims to address by combining multiple complementary approaches, such as machine learning classification, heuristic analysis, and embedding-based similarity checks.
2. **Real-work integration** of LLMs into high-stakes applications like automated customer service, content moderation, and legal document review requires heightened attention to security. Even subtle prompt manipulations in such domains can have serious consequences. By proactively developing defenses, the project aims to contribute to the responsible deployment of LLMs in these critical areas.
3. **Advances in AI Security** despite increasing research into AI safety and security, there is a growing recognition of the need for more advanced defenses against adversarial inputs. Recent studies have highlighted the effectiveness of machine learning models, heuristic approaches, and semantic analysis tools in mitigating the risks posed by malicious prompt injections. This project builds on these recent advancements to propose a more holistic solution.
4. **Public and Regulatory Pressure**: Ave, regulatory bodies and the public demand greater transparency and accountability from AI systems. By enhancing the security of LLMs, this project aligns with ongoing efforts to ensure that AI technologies are used ethically, safely, and in compliance with emerging AI governance frameworks.

## **Problem Statement**

The rapid adoption and deployment of Large Language Models (LLMs) in various applications, including chatbots, automated content generation, and customer service, has introduced significant security vulnerabilities. One such vulnerability is the risk of prompt injection attacks, where malicious actors craft inputs designed to manipulate the behavior of the LLMs, leading them to generate harmful, unintended, or biased outputs. These attacks exploit the model's inability to fully distinguish between benign and adversarial inputs, which can compromise the integrity and trustworthiness of the model's responses.

Prompt injection attacks are becoming increasingly sophisticated, posing challenges in their detection and mitigation. Traditional methods for securing machine learning systems are often not directly applicable to LLMs due to their unique nature and the high dimensionality of their input space. While some existing solutions, such as adversarial training or rule-based filters, provide partial protection, there is a need for more comprehensive, adaptable, and real-time defense mechanisms that combine multiple detection approaches for improved security.

This project aims to address **text-based prompt injection attacks by developing a robust, multi-layered defense system for LLMs.** Specifically, it seeks to integrate classification-based detection using Logistic Regression, heuristic-based methods for rapid flagging of suspicious inputs, and embedding-based similarity checks using vector embeddings. Vector embeddings numerically represent input text in high-dimensional space, capturing semantic and contextual relationships. By comparing embeddings of incoming prompts with known malicious examples using metrics like cosine similarity, the system can effectively identify and neutralize potential prompt injections.

The goal is to create a system that can efficiently safeguard LLMs against malicious manipulations while maintaining high performance and low false-positive rates. The outcome of this project will contribute to the growing field of LLM security by providing an effective, scalable solution for preventing prompt injection attacks and ensuring the safe deployment of LLMs in real-world applications.

## **Objectives**

The primary goal of this project is to develop a robust defense mechanism for Large Language Models (LLMs) to safeguard them against prompt injection attacks. The following are the key objectives specific to the project:

1. **Design and Implement a Classification System for Prompt Injection Detection**
   * **Objective:** Train a Logistic Regression model to classify prompts as either injected or non-injected based on specific patterns or characteristics. The goal is to use machine learning to detect malicious alterations in the input text.
   * **Approach:** A dataset containing both injected and non-injected prompts will be used for training. The Logistic Regression model will learn to identify distinguishing features of attack-prompts, offering a quick and effective solution for prompt injection detection.(Kohavi, 1995)(M., 2016)
2. **Develop a Heuristic-Based Approach for Early Detection of Malicious Inputs**
   * **Objective:** Create a rule-based heuristic system that can flag suspicious prompts using predefined keywords or patterns that commonly occur in prompt injection scenarios.
   * **Approach:** This method will involve creating a list of keywords or syntactic structures that are indicative of prompt injection. The heuristic system will flag potentially malicious inputs based on the presence of these elements. By integrating this approach, the system can provide an early detection layer before more complex models are used.(Cram101 Textbook Reviews, 2011)
3. **Implement Embedding-Based Similarity Checks for Input Validation**
   * **Objective:** Use **vector embeddings** to represent legitimate prompts numerically and compare new inputs against these embeddings to identify potential prompt injections based on semantic similarity
   * **Approach:** Pre-trained models such as BERT or GPT will generate vector embeddings for both incoming prompts and a dataset of legitimate prompts. These embeddings, which capture the semantic relationships within the text, will be compared using metrics like cosine similarity to determine the closeness of new inputs to known non-malicious examples. This embedding-based method enhances robustness by focusing on the contextual meaning of the inputs rather than surface-level syntax.(Devlin et al., 2018)(Reimers & Gurevych, 2019)(Hambarde & Proenca, 2023)
4. **Propose a Unified Decision Framework for Multiple Defense Methods**
   * **Objective:** Combine the results from the Logistic Regression classifier, heuristic flags, and embedding similarity checks to arrive at a unified decision on whether a prompt is safe or potentially injected with malicious intent.
   * **Approach:** The project will develop a decision-making process that takes into account the outputs from the heuristic analysis, classification results, and embedding similarity scores to generate a final verdict on the input. This multi-faceted approach ensures a comprehensive defense mechanism that minimizes false positives and negatives.
5. **Integrate the Defense System with a Chat Model**
   * **Objective:** Deploy the proposed defense mechanisms into a working environment, such as a chatbot or language model interface, to evaluate their real-time efficacy in preventing prompt injection attacks.
   * **Approach:** The defense system will be integrated with an existing chat model (e.g., Gemini or GPT-3) to evaluate how well the proposed methods protect the model against adversarial inputs in a live environment. Real-time testing will help fine-tune the mechanisms and assess their effectiveness.

**Adoption of the Three Approaches**

1. **Machine Learning Classification (Logistic Regression):**

Logistic Regression is a widely used technique for binary classification tasks. By training the model on labeled prompt data, it can learn to identify malicious patterns within the text. This approach enables the system to make decisions based on quantitative features of the input text, providing a strong foundation for automated detection.

1. **Heuristic Approach:**

Heuristic methods are essential for providing fast, rule-based filtering before more complex models are used. This approach is grounded in the idea that certain keywords or structural elements can act as red flags for malicious content, enabling a lightweight and interpretable detection method. This is particularly useful for mitigating attacks in real-time environments, where speed is critical.

1. **Embedding-Based Similarity Checks:**

Embeddings allow the system to understand the semantic meaning of the text, making it a powerful tool for detecting subtle prompt manipulations. By comparing new prompt embeddings with those of legitimate inputs, the system can identify potential injections based on their semantic similarity, not just syntax. This approach leverages advancements in transformer-based models like BERT, Sentence-BERT, and GPT to provide more accurate and nuanced detection.

## **Scope of the Project**

The scope of this project encompasses the development, implementation, and evaluation of a proactive defense mechanism against Text-Based prompt injection attacks on large language models (LLMs).

The project aims to address a critical gap in the secure deployment of LLMs by providing a comprehensive, multi-faceted solution that leverages classification, heuristic-based detection, and embedding-based similarity checks.

Key aspects of the scope are outlined below:

1. **Protection Against Malicious Inputs**

The project focuses on identifying and mitigating Text-Based prompt injection attacks that exploit vulnerabilities in LLMs, ensuring that malicious or unintended outputs are avoided. This will enhance the robustness and reliability of LLM-based applications across industries such as customer support, education, healthcare, and content generation.

1. **Implementation of Diverse Techniques**

The solution integrates three complementary methodologies to create a robust security framework:

* **Classification Models**: A logistic regression model trained on labeled data is used to classify prompts as malicious or benign based on linguistic patterns. This approach ensures efficient and accurate prompt detection.(M., 2016)(“A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection,” n.d.)
* **Embedding-Based Similarity Checks**: Instead of traditional heuristics, the project employs **semantic vector embeddings** to evaluate the similarity of new inputs to known harmful prompts. Using tools like **sentence transformers (e.g., BERT, RoBERTa)**, embeddings of legitimate prompts are stored in a vector database. The system computes similarity scores between incoming prompts and existing embeddings, flagging potentially harmful ones based on predefined thresholds.(Devlin et al., 2018)(Sentence, n.d.)
  + **Advantages over Heuristics**: Vector embeddings provide a semantic understanding of inputs, offering superior accuracy compared to keyword-based heuristics. They allow detection of paraphrased or contextually similar malicious prompts, which heuristics might miss.(Belagatti, 2024)(Malkov & Yashunin, 2016)

1. **Experimental Validation and Efficacy**

The project includes rigorous evaluation through:

* Labeled datasets of malicious and benign prompts.
* Metrics such as detection accuracy, false positive rates, and response times to measure performance.
* Stress testing across diverse domains to ensure broad applicability(“Advances in Semantic Similarity Search for Large-Scale Systems,” n.d.)(Dugam, 2023)

1. **Real-Time Defense System**

The system is designed for real-time analysis and decision-making, ensuring Text-Based prompt injection attacks are detected and neutralized during user interactions without significant delays.

1. **Scalability and Versatility**

The project’s approach is scalable and adaptable to different LLM architectures and use cases. It can be integrated into existing LLM frameworks, making it applicable to various domains requiring LLM security.

1. **Experimental Validation and Efficacy**

The project scope includes rigorous testing and validation of the proposed methods using labeled datasets. Metrics like detection accuracy, false positive rates, and response times will be used to evaluate system performance.

1. **Usability and Accessibility**

To ensure ease of adoption, the solution includes an intuitive interface developed using Gradio, allowing users and developers to interact with the system effortlessly.

1. **Contribution to LLM Security Research**

The project addresses a critical challenge in **LLM security** and contributes to the research community by:

* Demonstrating the effectiveness of **embedding-based similarity detection** in securing AI models.
* Offering insights into best practices for safeguarding LLMs against evolving prompt injection threats.

Chapter 2: Literature Review

As Large Language Models (LLMs) become integral to various applications in natural language processing, their susceptibility to prompt injection attacks presents a critical challenge. These attacks manipulate model inputs to elicit unintended, malicious, or harmful outputs, compromising the safety, trustworthiness, and ethical deployment of LLMs in real-world systems.(“Vulnerabilities in Large Language Models: A Review of Emerging Threats,” n.d.)(Russell & Norvig, 2021)

Prompt injection attacks are emblematic of the broader security vulnerabilities that arise in advanced AI systems due to their reliance on vast, unstructured data and contextual interpretation capabilities. Unlike traditional adversarial attacks in machine learning, which often target image or structured data systems, prompt injections exploit the text-based nature of LLMs, making them difficult to detect and mitigate using conventional techniques.(M., 2016)(Carlini et al., 2020) This necessitates novel approaches that consider the semantic and contextual nuances of language.

Recent advancements in AI security have explored multiple strategies to address this issue. Classification-based approaches, such as logistic regression, provide effective binary models for detecting malicious prompts.(“A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection,” n.d.) However, these models are limited by their reliance on feature extraction and may struggle with generalizability. Complementary techniques like embedding-based similarity checks leverage pre-trained models such as BERT and Sentence-BERT to compare new inputs against known malicious patterns in vector space, enhancing detection accuracy.(Devlin et al., 2018)(Sentence, n.d.) Furthermore, the integration of embedding databases enables real-time and scalable analysis, offering robust defenses for large-scale deployments.(Belagatti, 2024)(“Advances in Semantic Similarity Search for Large-Scale Systems,” n.d.)

By synthesizing existing research and identifying key gaps, it aims to establish a foundation for developing a comprehensive, multi-layered approach to protecting LLMs against prompt injection attacks.

## **Existing Works**

Prompt injection attacks seriously threaten Large Language Models (LLMs), manipulating the model's inputs to generate unintended, harmful, or malicious outputs. These attacks are often subtle and sophisticated, requiring robust defense mechanisms for detection and mitigation. This review explores several recent research papers and methodologies directly relevant to the project, which leverages **Logistic Regression**, **heuristic functions**, and **vector embeddings** for detecting Text-Based prompt injection attacks in LLMs. The goal is to combine machine learning classifiers, heuristic analysis, and vector-based techniques to safeguard LLMs proactively.

1. **General Vulnerabilities and Detection Techniques**

(Rahman et al., 2024) is highly relevant to the project as it focuses on using **multilingual BERT embeddings** to detect malicious prompt injections. Multilingual BERT models provide robust contextual embeddings, making them well-suited for detecting adversarial inputs in multiple languages. By converting input prompts into embeddings, the model can identify semantic irregularities indicative of malicious manipulation, enhancing the accuracy of detection systems. This approach aligns with the project's use of **BERT embeddings** to detect malicious prompt injections effectively.

(F. Liu et al., 2024)provides a systematic framework to understand and benchmark both attacks and defenses. This paper's insights are directly applicable to designing the **heuristic functions** employed in the project. It emphasizes the importance of comprehensive defense strategies, integrating both heuristic detection and advanced classifiers like **Logistic Regression** to evaluate the legitimacy of prompts. This framework supports the project’s methodology of combining heuristic rules with machine learning classifiers to improve detection accuracy.

(Sahu, 2024) discusses the various vulnerabilities in LLMs and the challenges in securing them from adversarial attacks. The paper emphasizes the importance of **heuristic-based detection** as a first line of defense, which can quickly filter out suspicious prompts. It also advocates for layered defense mechanisms, such as the ones being implemented in the project, where heuristic methods are combined with more sophisticated approaches like embedding comparisons and machine learning classifiers.

(Schlarmann et al., 2024) offers an in-depth overview of different types of adversarial attacks, including prompt injection, and proposes various defense mechanisms. This survey underlines the need for **robust classification systems** that can distinguish between malicious and legitimate inputs, which is central to the project's goal of using **Logistic Regression** trained on embeddings to classify prompt injections.

(Ab)using Images and Sounds for Indirect Instruction Injection in Multi-Modal LLMs, n.d.) focuses on indirect injection attacks using multi-modal inputs like images and sounds. This research highlights the need for **vector-based analysis** of non-textual inputs, complementing the project’s use of **vector databases** to store and compare embeddings. Although the project focuses on text-based prompt injections, the principles of vector-based detection can be extended to multi-modal inputs, providing a broader defense framework.

(Abdelnabi & Fritz, 2022) categorizes indirect prompt injection attacks and provides insights into the **taxonomy** of these attacks. It emphasizes the importance of understanding subtle injection techniques that may evade traditional detection systems. This paper’s focus on taxonomy is crucial for the project, as it supports the development of a **database of known attacks**, which is a core component of the vector database approach used for prompt injection detection in the project. (Greshake et al., 2023)

1. **Adversarial Robustness**

(B. Cao et al., 2023) explores **alignment-breaking attacks**, where the prompt's structure is manipulated to confuse the model. This paper discusses defense strategies like **random mask filtering** and alignment verification, which could complement the **heuristic-based detection** in the project. Incorporating such defenses could further enhance the ability to recognize altered prompts and prevent alignment disruptions caused by prompt injection.

(Yi et al., 2023) directly addresses the challenges in defending against indirect prompt injection attacks, which are often harder to detect than direct injections. This paper underscores the need for **robust heuristic defenses** and **multi-layered approaches** that combine embedding similarity checks with heuristic methods. These insights align with the project’s multi-faceted defense system, where both **heuristic detection** and **embedding-based comparison** are employed to guard against various attack types.

1. **Defense Mechanisms**

(Phute et al., 2023) introduces a novel defense mechanism where LLMs autonomously verify their inputs, detecting and rejecting malicious instructions. This self-filtering mechanism complements the project’s **heuristic functions**, serving as an auxiliary validation step. By incorporating self-examination methods, the system can potentially reject adversarial inputs without external intervention, adding an extra layer of security.

(LangChain, 2023) provides practical insights into combining **heuristic detection**, **LLM-based defenses**, and **vector embeddings**. This paper directly informs the project’s architecture, where these three components work together to mitigate prompt injection attacks. By leveraging both **heuristics** and **embedding comparisons**, the project ensures a comprehensive defense system against malicious prompts.

(Z. Liu et al., 2024) discusses **advanced defensive techniques** that align with the project’s approach. Specifically, it explores the concept of **information bottleneck** to enhance the detection of adversarial inputs. This technique could complement the **embedding-based analysis** in the project, improving the model’s ability to identify malicious prompts while reducing the risk of false positives.

1. **Machine Learning for Adversarial Detection**

(Alon & Kamfonas, 2023) introduces a **perplexity-based detection** method, which evaluates the likelihood of a prompt being generated by the model. This method is particularly useful for detecting out-of-distribution or adversarial prompts, complementing the **Logistic Regression classifier** used in the project. By integrating perplexity as an additional feature for classification, the detection system can further refine its ability to identify malicious inputs.

(Yi Liu et al., 2023) (Suo, 2024) outlines the various types of prompt injection techniques and provides insights into defending against them. The paper supports the creation of a **database of known prompt injections**, a feature that is central to the project’s **vector database** approach. This database allows the model to compare incoming prompts with stored embeddings, identifying potential threats based on similarity scores.

1. **Dataset**

In the project, a custom dataset is used to train and evaluate the **Logistic Regression model** for classifying prompts as either **malicious** (prompt injection) or **benign** (non-malicious). This dataset consists of two main categories:

1. **Injected Prompts (Malicious)**: These are prompts that have been manipulated or altered with the intent to inject harmful or unintended behavior into the LLM. These can include techniques like altering the prompt structure, embedding commands within the text, or using adversarial phrases.
2. **Non-injected Prompts (Benign)**: These are clean, normal prompts that do not contain any malicious intent and are representative of the typical interaction expected between the user and the LLM.

The **Logistic Regression model** is trained on this dataset to learn the distinguishing features between the injected and non-injected prompts. The model is evaluated using accuracy, and the objective is to achieve high classification performance in detecting malicious prompts.

For the **Heuristic Approach**, no specific dataset is required since it relies on a **list of keywords or rules** to detect suspicious prompts based on their content. The heuristic layer checks whether a prompt contains specific indicators of potential prompt injection attacks, such as certain phrases or command structures.

For the **Embedding-based Similarity Check**, the approach uses **pre-trained embeddings** (such as those generated by **BERT** or **Sentence-BERT**) to create vector representations of prompts. While there isn't a labelled dataset specifically for this step, a set of **reference prompts** (both malicious and benign) is used to create embeddings for comparison. The system then calculates the cosine similarity between the input prompt and the reference embeddings to determine if the prompt is similar to known malicious patterns.

## **Comparison of Existing Research with Project Implementation**

Prompt injection attacks present a significant threat to the secure operation of Large Language Models (LLMs), enabling adversaries to manipulate model outputs in unintended and potentially harmful ways. While existing research extensively explores various methodologies for detecting and mitigating such attacks, there is a lack of cohesive frameworks that integrate multiple defense techniques for real-time application.

This section compares the methodologies outlined in recent literature with the practical implementation of a project designed to safeguard LLMs against text-based prompt injection attacks. The project employs a multi-layered approach combining logistic regression, heuristic analysis, and embedding-based similarity checks to deliver a robust, scalable, and real-time defense mechanism. By aligning insights from existing works with a practical, integrated framework, the project bridges the gap between theoretical advancements and their real-world deployment, ensuring enhanced security and usability in LLM systems.

1. **Focus of Protection**

**Existing Work**:

* General focus on **detecting and mitigating vulnerabilities** in LLMs using a variety of approaches, such as multilingual embeddings, heuristic methods, and multi-modal strategies.
* Examples include Rahman et al.'s work on multilingual BERT embeddings and Sahu's emphasis on layered defenses.

**Our Implementation**:

* Explicitly targets **Text-Based Prompt Injection Attacks** using a multi-faceted framework combining:
  + Logistic Regression
  + Heuristic Analysis
  + Embedding-Based Similarity Checks
* Proactively addresses attacks while ensuring **real-time defense and scalability**.

1. **Techniques Used**

**Existing Work**:

* **Classification Models**:
  + Logistic Regression and other machine learning models (e.g., Rahman et al., Liu et al.) trained on adversarial datasets.
* **Embedding-Based Approaches**:
  + Use of contextual embeddings (e.g., BERT, multilingual BERT) to detect malicious inputs through semantic similarity.
  + Taxonomy-based embedding strategies for indirect attack detection (Abdelnabi & Fritz).
* **Heuristic Defenses**:
  + Rules-based systems as a **first line of defense** (Sahu).
  + Dynamic approaches like perplexity scoring (Alon & Kamfonas).
* **Multi-Modal Defenses**:
  + Focused on indirect prompt injections using images and sounds (e.g., Greshake et al.).

**Our Implementation**:

* **Integrated Multi-Layered Defense**:
  + **Logistic Regression**: A straightforward, explainable machine learning model trained on a labeled dataset of injected vs. benign prompts for classification.
  + **Heuristics**: A rule-based system for quick detection, providing a lightweight layer to complement machine learning.
  + **Vector Embeddings**: Embedding-based similarity checks using pre-trained models (BERT, RoBERTa) stored in a **vector database** to compare inputs semantically, offering robustness against paraphrased and contextually similar attacks.
  + **Real-Time Decision-Making**: A holistic combination of all three techniques ensures immediate and accurate detection.

1. **Dataset Usage**

**Existing Work**:

* Focus on labeled datasets of malicious and benign prompts for classifier training and benchmarking.
* Some works (e.g., Abdelnabi & Fritz) emphasize building a taxonomy of attack patterns for enhanced detection.

**Our Implementation**:

* Uses a **custom dataset** comprising:
  + Injected (malicious) prompts to simulate real-world adversarial scenarios.
  + Non-injected (benign) prompts representing typical user interactions.
* Embeddings of both types stored in a **vector database** for semantic comparison.
* Prioritizes **robust training and validation** using accuracy, false positive rates, and response time as performance metrics.

1. **Advantages Over Existing Work**

**Our Project’s Contributions**:

1. **Integrated Defense**:
   * Combines the strengths of classification, heuristic, and embedding techniques into a single framework, unlike most existing works that focus on one or two methods.
2. **Real-Time Capability**:
   * Designed for real-time detection and mitigation, providing immediate feedback without significant computational overhead.
3. **Embedding-Based Superiority**:
   * Embedding similarity checks outperform traditional heuristics by detecting paraphrased or subtly modified malicious prompts.
4. **Scalability**:
   * Implementation is adaptable to various LLM frameworks, making it suitable for diverse use cases (e.g., customer support, education).
5. **Practical Usability**:
   * Developed with an intuitive **Gradio interface**, ensuring accessibility for end users and developers.
6. **Alignment with Recent Research**

* **Rahman et al. (2024)**: Our project’s embedding-based similarity checks align with their use of multilingual BERT embeddings, but you extend the scope by integrating vector databases for broader application.
* **F. Liu et al. (2024)**: The heuristic and logistic regression combination in Our project mirrors their emphasis on blending rule-based and ML-based defenses.
* **Phute et al. (2023)**: Autonomous input verification is echoed in Our vector-based similarity detection, which autonomously flags harmful prompts.
* **Yi et al. (2023)**: Emphasizing multi-layered defense, Our framework reflects their recommended mix of heuristics and embeddings.

1. **Broader Implications**

**Existing Work**:

* Primarily academic, focusing on exploring vulnerabilities and proposing theoretical solutions.
* Limited focus on **real-world deployment** or ease of integration into practical applications.

**Our Implementation**:

* Directly bridges research and application:
  + Designed for seamless integration into LLM systems.

Chapter 3: Methodology

The methodology for this project is designed to address **prompt injection attacks** on Large Language Models (LLMs) by implementing a multi-layered, proactive defense system. The approach integrates machine learning classification, heuristic-based filtering, and embedding-based similarity checks to ensure comprehensive protection against adversarial inputs. This combination leverages both rule-based and data-driven strategies, enabling real-time detection and mitigation of malicious prompts.

Key components of the methodology include:

1. **Classification-Based Detection**

A logistic regression model is employed to classify inputs as either malicious or benign based on learned patterns. This method provides an automated, scalable mechanism for detecting prompt injection attacks, leveraging labeled datasets for training. It offers a quantitative foundation for identifying malicious inputs based on statistical and linguistic characteristics.

1. **Embedding-Based Semantic Analysis**

Pre-trained models like BERT and Sentence-BERT are utilized to generate vector embeddings for inputs. These embeddings capture the semantic meaning of prompts, allowing for nuanced comparisons between new inputs and known benign or malicious patterns. This component employs cosine similarity metrics to identify subtle manipulations in prompt structure, ensuring a deeper understanding of context beyond surface-level patterns.

1. **Heuristic Filtering for Real-Time Detection**

A rule-based system is developed to flag suspicious prompts using predefined keywords and syntactic structures indicative of prompt injection. This method acts as a lightweight preliminary filter, enhancing the system's efficiency by reducing the burden on computationally intensive processes.

1. **Integration into a Unified Decision Framework**

The outputs from classification, heuristic analysis, and embedding-based checks are combined in a unified decision-making framework. This approach ensures robust and accurate prompt evaluation by minimizing false positives and negatives, leveraging the strengths of each individual technique.

1. **Validation and Real-World Deployment**

The methodology includes rigorous testing on labeled datasets to evaluate detection accuracy, precision, and response times. The defense system is integrated into a chatbot interface using tools like Gradio to assess its effectiveness in a live environment, simulating real-world usage scenarios.

This methodological framework ensures a scalable, adaptable, and effective defense system that enhances LLM security and reliability in diverse application domains.

## **Purposed solution**

The proposed solution in this project addresses the vulnerability of large language models (LLMs) to **prompt injection attacks** through a multi-layered defense approach. The solution integrates **classification-based detection**, **heuristic-based filtering**, and **embedding similarity checks** to safeguard LLMs from malicious prompts that could induce harmful or unintended outputs. Each component is designed to enhance the overall security of the system by combining different techniques that complement each other.

1. **Classification using Logistic Regression**

The first defense layer employs **logistic regression** to classify prompts as either benign or malicious. A labeled dataset containing injected and uninjected prompts is used to train the model, which learns patterns associated with adversarial inputs. This approach provides a quick and efficient method for detecting prompt injection.

* **Justification**: Logistic regression is a well-suited binary classifier that balances simplicity, speed, and interpretability, making it ideal for real-time security systems.(Keldenich, 2021)(Zhang, 2022)It enables quick decision-making without the computational overhead of more complex models.

1. **Embedding-Based Semantic Similarity Checks**

The second layer analyzes **semantic similarities** between input prompts and known patterns of legitimate or malicious behavior using **embedding techniques**. Tools like **Sentence-BERT** generate vector embeddings for prompts, capturing their contextual and semantic meaning. These embeddings are compared to reference embeddings using cosine similarity to detect harmful inputs.

* **Justification**: Embedding-based techniques analyze the semantic depth of prompts rather than surface-level patterns, enabling detection of subtle and contextually malicious inputs (Devlin et al., 2018)(Reimers & Gurevych, 2019)(Data Science Dojo, 2024). This method is scalable and can adapt to evolving attack strategies.

1. **Heuristic Approach with Keyword Filtering**

This layer relies on rule-based detection, flagging suspicious inputs based on predefined **keywords** or syntactic patterns commonly associated with adversarial prompts. Heuristic methods are augmented with **similarity metrics** to improve accuracy.

* **Justification**: Rule-based methods are lightweight and effective for initial screening of prompts, especially in detecting recurring patterns in adversarial inputs (Papernot et al., 2017)(Ebrahimi et al., 2018). By combining heuristics with semantic metrics, the system ensures robust preliminary detection.

1. **Decision-Making Framework**

The results from all three layers are aggregated into a unified **decision-making algorithm**. This algorithm evaluates the outputs of the logistic regression classifier, embedding similarity scores, and heuristic checks to classify prompts as legitimate, suspicious, or outright malicious.

* **Justification**: Combining diverse methodologies ensures resilience against varied attack techniques and minimizes false positives and negatives, strengthening overall security (Goyal et al., 2022)(Goodfellow et al., 2014).

1. **Integration with Gemini Chat Model**

The defense system integrates with **Gemini** (or similar LLMs) as an intermediary layer, ensuring all prompts are vetted for safety before being processed by the model. This architecture preserves the usability of the LLM while proactively neutralizing adversarial inputs.

* **Justification**: Seamless integration with existing frameworks ensures the system's practicality and broad applicability without requiring fundamental modifications to LLMs (Brown et al., 2020).

## **Requirements analysis**

The requirements analysis of the LLM Security Project involves identifying the functional and non-functional aspects necessary for the development and implementation of the multi-layered defense mechanism. Below are the detailed requirements:

### **Functional Requirements**

These define the specific features and functionalities that the system must provide to meet its objectives.

1. **Prompt Classification**
   * A machine learning-based classifier (Logistic Regression) must analyze and classify input prompts as safe or malicious.
   * **Functionalities:**
     + Accept labeled datasets for training.
     + Provide real-time classification of user prompts.
     + Output a confidence score for the classification.
2. **Semantic Embedding-Based Similarity Checks**
   * Generate vector embeddings of input prompts and compare them with stored embeddings using cosine similarity.
   * **Functionalities:**
     + Integrate with pre-trained models like **Sentence-BERT**.
     + Store embeddings for known malicious and benign prompts.
     + Return similarity scores indicating semantic closeness to harmful patterns.
3. **Heuristic Keyword Detection**
   * Identify predefined keywords or patterns indicative of malicious intent.
   * **Functionalities:**
     + Maintain a dynamic keyword database for adversarial patterns.
     + Flag inputs containing suspicious keywords for further analysis.
4. **Decision-Making System**
   * Aggregate results from the classification model, heuristic detection, and embedding similarity checks to produce a unified safety verdict.
   * **Functionalities:**
     + Combine outputs using a weighting algorithm.
     + Trigger appropriate actions such as blocking, flagging, or allowing prompts.
5. **Integration with Chat Models**
   * Operate as an intermediary preprocessing layer for LLMs (e.g., Gemini).
   * **Functionalities:**
     + Intercept and evaluate all user inputs before they reach the LLM.
     + Ensure seamless integration without performance degradation.
6. **Real-Time Processing**
   * Ensure that all defensive layers process inputs in real-time to avoid delays in LLM responses.

### **Non-Functional Requirements**

These outline the broader system qualities and constraints.

1. **Scalability**
   * The system must handle a high volume of inputs without significant degradation in performance.
   * Ensure compatibility with multiple LLM architectures and large datasets.
2. **Accuracy and Precision**
   * Minimize false positives (blocking safe inputs) and false negatives (allowing malicious inputs).
   * Target high detection accuracy with false positive rates under 5%.
3. **Performance**
   * Process prompts in under 200 milliseconds to maintain real-time interaction capabilities.
   * Optimize embedding comparisons for fast cosine similarity calculations.
4. **Usability**
   * Provide an intuitive interface (e.g., using Gradio) for developers to monitor and test the system.
   * Simplify configuration for integrating custom rules, datasets, and models.
5. **Robustness**
   * Effectively defend against a wide range of attack patterns, including novel and evolving prompt injection techniques.
6. **Security**
   * Prevent unauthorized access to training data, vector embeddings, or keyword databases.
   * Implement encryption for sensitive components like embeddings and similarity scores.
7. **Maintainability**
   * Allow for easy updates to heuristic rules, embeddings, and machine learning models as new threats emerge.
8. **Compliance**
   * Adhere to industry standards for secure AI deployment, such as ISO/IEC 27001 for data security.

## **Method**

The project employs three primary methods for detecting and mitigating **prompt injection attacks** in **Large Language Models (LLMs)**: **Logistic Regression**, a **Heuristic Approach**, and **Embedding-based Similarity Checks**. Here is an explanation of each method along with its background:

1. **Logistic Regression (Classification Model)**

**Background:**

Logistic Regression is a **statistical model** commonly used for binary classification tasks. It predicts the probability of a given input belonging to one of two classes. In this case, the task is to classify prompts as either **malicious** (prompt injection) or **benign** (non-malicious). The model works by learning a relationship between input features and the outcome class, using a **logistic function** to map predicted values to probabilities between 0 and 1.

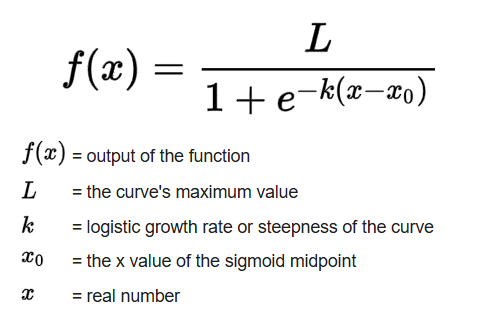
* **Application**: In the context of prompt injection, Logistic Regression is trained using a labeled dataset containing **malicious** and **non-malicious** prompts. Features extracted from these prompts, such as specific token patterns or structural features, help the model learn how to distinguish between injected prompts and normal ones.
* **Training & Evaluation**: The model is trained on a dataset of labeled prompts, and performance is evaluated based on metrics like **accuracy**, **precision**, **recall**, and **F1 score**. It requires a significant amount of labeled data to achieve reliable predictions.(I. Goodfellow et al., 2016)

**Logistic Regression (Classification Model)**

Logistic Regression is a widely-used statistical model in machine learning, specifically designed for binary classification problems, where the goal is to predict one of two outcomes. In the context of prompt injection detection for **Large Language Models (LLMs)**, logistic regression is applied to classify prompts as either **malicious** (i.e., containing prompt injections) or **benign** (i.e., standard, non-manipulative prompts).

**Theoretical Foundation**

Logistic Regression is based on the concept of the **logistic function** (also known as the **sigmoid function**), which transforms the output of a linear equation into a probability score between 0 and 1. This is ideal for binary classification tasks, as the output can be interpreted as the likelihood of a prompt belonging to one of the two categories.

Logistic regression is used as a classification method to predict the probability of a prompt being malicious (i.e., a prompt injection attack). The logistic function, also known as the sigmoid function, is given by:

The coefficients are learned during the training process, where optimization techniques such as **Gradient Descent** are used to minimize the loss function. The loss function typically used is the **log-likelihood**, which is maximized to find the best fitting model. Through this process, the model determines the relationships between the features and the probability of prompt injection.(James et al., 2013)(Ng, 2004)

The logistic regression model is trained using a labeled dataset of both malicious and non-malicious prompts. Once trained, the model can be used to predict the likelihood that a new, unseen prompt is malicious, allowing for the detection and mitigation of prompt injection attacks.

**How Logistic Regression Works for Prompt Injection Detection**

* **Input Features**: In this project, logistic regression operates on a set of **features extracted from the prompts**. These features could include aspects such as:
  + **Token patterns**: The frequency of specific words or tokens that might be indicative of prompt injection.
  + **Structural features**: Characteristics like the length of the prompt, the use of special characters, or unnatural phrasing.
  + **Contextual clues**: Features that capture the overall context or semantics of the prompt, possibly extracted using **text embeddings** (e.g., from models like BERT).
* **Training**: The logistic regression model is trained using a **labeled dataset** containing examples of both benign and malicious prompts. During training, the model learns to associate specific patterns or features with the corresponding class (benign or malicious).
* **Prediction**: After training, the model is used to predict whether a new prompt is malicious or benign based on its features. The output is a probability value, and if the probability exceeds a certain threshold (usually 0.5), the prompt is classified as malicious.

**Advantages of Logistic Regression**

* **Simplicity**: Logistic Regression is simple to understand and implement. It does not require complex computations or vast computational resources, making it suitable for tasks where interpretability is important.
* **Efficiency**: Given its relatively low computational cost, Logistic Regression can be applied to real-time systems with fast predictions, even on large datasets.
* **Interpretability**: Since it directly models the relationship between features and the outcome, logistic regression provides interpretable results in terms of feature importance. This is important for understanding which aspects of the prompt are contributing to the classification decision.

**Challenges of Logistic Regression**

* **Linearity Assumption**: Logistic regression assumes a linear relationship between the features and the log-odds of the outcome. In more complex scenarios, where the relationship between features is non-linear, the model's performance can degrade.
* **Feature Engineering**: The quality of predictions depends heavily on how well the features are designed. If important patterns or relationships are missed in the feature extraction process, the model's performance may be suboptimal.**Sensitivity to Imbalanced Data**: If the dataset is heavily imbalanced (e.g., more benign prompts than malicious ones), logistic regression can be biased towards the majority class. Techniques such as **class weighting** or **resampling** may be required to address this.

**Training and Evaluation**

* **Training**: During the training phase, the dataset is divided into training and validation sets. The model learns by minimizing the **cross-entropy loss**, which measures the difference between the predicted probabilities and the actual labels. Optimization techniques like **Gradient Descent** are used to update the model parameters.
* **Evaluation**: The model is evaluated using performance metrics such as **accuracy**, **precision**, **recall**, and **F1-score**. These metrics are essential in determining how well the model classifies malicious versus benign prompts.
  + **Accuracy** measures the proportion of correctly classified prompts.
  + **Precision** measures the proportion of correctly predicted malicious prompts out of all predicted malicious prompts.
  + **Recall** measures the proportion of correctly predicted malicious prompts out of all actual malicious prompts.
  + **F1-score** is the harmonic mean of precision and recall, providing a balance between the two.

**Extensions and Improvements**

* **Regularization**: Regularization techniques like **L2 regularization** (Ridge) or **L1 regularization** (Lasso) can be applied to prevent overfitting, especially in cases where there are many features or the data is noisy.
* **Non-linear Transformations**: If the data exhibits non-linear patterns, **kernel methods** or **polynomial features** can be introduced to enhance the model's ability to capture complex relationships.
* **Feature Selection**: Dimensionality reduction techniques, such as **Principal Component Analysis (PCA)** or **Recursive Feature Elimination (RFE)**, can be used to select the most relevant features, improving model performance and reducing overfitting.

**Applications in Prompt Injection Detection**

In the case of prompt injection detection, logistic regression plays a crucial role in distinguishing between **normal prompts** and those attempting to manipulate the model’s behavior (prompt injections). By identifying patterns in the input that correlate with malicious behavior, the logistic regression model can act as an initial filtering mechanism, flagging prompts that may require more advanced scrutiny (e.g., using heuristics or embedding-based similarity checks).(I. Goodfellow et al., 2016)(M., 2016)(Hastie et al., 2017)(Hughes et al., 2023)(Cheng & Chang, 2019)

1. **Heuristic Approach**

**Background:**

A **heuristic approach** uses predefined rules or patterns to identify potentially malicious prompts based on their content or structure. This method doesn't rely on machine learning models but instead uses **keyword-based filters** or **pattern-matching algorithms** to flag suspicious inputs.

* **Application**: For prompt injection detection, the heuristic approach checks if a prompt contains specific **keywords** or **patterns** indicative of injected behavior. For example, certain command-like phrases or unusual formatting might suggest that the prompt is attempting to manipulate the model’s behavior. This method can be extremely efficient in detecting simple or obvious injection attacks.
* **Limitations**: While the heuristic approach is fast and easy to implement, it may not be as accurate as machine learning-based methods in detecting sophisticated attacks. It often lacks the flexibility to handle complex or novel attacks that don't match predefined rules.(Russell & Norvig, 2021)(Witten et al., 2016)

**Heuristic Approach for Detecting Prompt Injection Attacks**

The **heuristic approach** is a **rule-based method** for detecting prompt injection attacks in large language models (LLMs). It involves the use of predefined rules, patterns, or keywords to identify suspicious prompts based on their content or structure. This method contrasts with machine learning-based approaches, as it does not rely on data-driven models but rather on explicit logical conditions that flag prompts as potentially harmful.

**Background and Concept**

A **heuristic** is essentially a shortcut or "rule of thumb" that simplifies decision-making processes. In the context of prompt injection attacks, heuristics are applied to identify **malicious manipulation attempts** based on known **patterns** or **anomalies** in the prompt text.

* **Nature of Heuristics**: Heuristics are generally faster to implement and computationally cheaper than machine learning models, making them a popular choice for systems where quick decisions are essential. However, they can be less flexible and may fail to catch sophisticated attacks that deviate from predefined rules.
* **Application to Prompt Injection**: The goal of the heuristic approach is to flag prompts that show clear signs of trying to manipulate or alter the behavior of an LLM. This manipulation often involves tricks such as injecting special instructions, altering the context, or attempting to trigger unwanted actions from the model. Heuristic rules can be based on:
  + **Keyword detection**: Specific words or phrases that signal an attempt to inject a malicious command (e.g., "hack", "disable", or "override").
  + **Pattern matching**: Unusual formatting, characters, or sequences that suggest an attack (e.g., unusual symbols, SQL-like commands, or code injections).
  + **Instruction anomalies**: When a prompt contains instructions that seem out of place or overly forceful in terms of controlling the model's behavior (e.g., "Execute this instruction", "Ignore previous context").

**Heuristic Methods in Prompt Injection Detection**

1. **Keyword Filtering**: A simple yet effective approach, where specific keywords or phrases known to indicate malicious intent are used to detect suspicious prompts. For example, the presence of commands like “hack”, “execute”, or “bypass” might trigger a flag. This method is particularly useful for **obvious attacks** but can be bypassed by attackers using subtle or novel techniques.
2. **Pattern-based Detection**: This method involves recognizing patterns in the text that are commonly associated with **malicious inputs**. This could include:
   * **Inconsistent formatting**: Such as sudden use of special characters or unexpected punctuation.
   * **Code-like structures**: When the input looks more like a command or script rather than a natural language prompt.
   * **Abnormal structures**: For example, prompts that contain multiple contradictory instructions, or those designed to confuse or trap the model.
3. **Blacklists and Whitelists**: Heuristics can also involve checking if the input matches entries in a **blacklist** (e.g., known harmful inputs) or does not appear in a **whitelist** (trusted, safe prompts).
4. **Control and Context Checks**: Some heuristics evaluate if the prompt’s instructions are too forceful or contextually irrelevant to the conversation. If the prompt tries to change the model’s operating instructions too drastically or forcefully, it might be flagged as suspicious.
5. **Logical Inconsistencies**: Heuristic methods can also look for **logical contradictions** or **impossible scenarios** that a prompt might introduce, such as asking the model to provide answers to questions it should not be able to access based on previous context.

**Advantages of the Heuristic Approach**

* **Simplicity**: Heuristic methods are easy to implement and require less computational power compared to machine learning models.
* **Speed**: These methods are fast and can quickly assess whether a prompt is likely to be malicious, making them suitable for real-time applications.
* **Low Overhead**: There’s no need for extensive training data or model fine-tuning, which makes it ideal for systems with limited resources.

**imitations of the Heuristic Approach**

* **False Positives/Negatives**: Heuristic methods may result in **false positives** (flagging benign prompts as malicious) or **false negatives** (failing to flag an actually harmful prompt), especially if the patterns they detect are not sufficiently broad or comprehensive.
* **Limited Coverage**: Since heuristics rely on predefined rules, they can be easily evaded by attackers who craft novel or unexpected input patterns.
* **Lack of Contextual Understanding**: Heuristics often lack the **deep contextual understanding** that machine learning models can achieve, making them less effective in handling **sophisticated prompt injections** that might not follow simple patterns or keywords.

1. **Embedding-based Similarity Checks**

**Background:**

Embedding-based similarity checks involve representing text (in this case, prompts) as **high-dimensional vectors** in a continuous vector space, using **pre-trained language models** like **BERT**, **GPT**, or **Sentence-BERT**. These models convert textual input into dense vectors that capture semantic meaning, allowing for easy comparison between different prompts.

* **Application**: To detect prompt injection, this method computes the **cosine similarity** between the vector representation of the input prompt and a set of **reference prompts** (either benign or known malicious ones). If the cosine similarity is high between the input prompt and a known malicious prompt, it is flagged as potentially harmful.
* **Advantages**: This method leverages **semantic understanding** of language, making it effective for detecting subtle or sophisticated prompt injection attempts that may not be captured by keyword-based methods.
* **Challenges**: The performance of this approach depends heavily on the quality of the pre-trained embeddings. Large models like **BERT** or **GPT** require significant computational resources, and managing the embeddings in a large-scale system can be complex.(J. Devlin et al., 2019)(Reimers & Gurevych, 2019)(Radford et al., 2019)

**Background of Embedding-based Similarity**

Embedding-based similarity checks involve converting text (such as prompts) into **dense vector representations** within a high-dimensional space. These embeddings capture **semantic relationships** between words or entire sentences, which makes it possible to compare different pieces of text based on their meaning, rather than just surface-level features like exact word matches.

Text embeddings are typically generated using deep learning models like **BERT (Bidirectional Encoder Representations from Transformers)**, **GPT (Generative Pre-trained Transformer)**, or specialized models like **Sentence-BERT**. These models transform input text into fixed-size vectors (also called **embeddings**) that encode semantic information in a high-dimensional space.

The central idea is that **similar text** (in meaning) will have **similar embeddings**, while **dissimilar text** will have embeddings that are farther apart in this vector space. The similarity between any two texts can be computed by measuring the **distance** or **angle** between their respective embeddings. This distance is often calculated using **cosine similarity**, which computes the cosine of the angle between two vectors. A high cosine similarity indicates that two pieces of text are semantically similar, while a low cosine similarity means they are dissimilar.

**Application of Embedding-based Similarity Checks**

For the detection of **prompt injection attacks**, embedding-based similarity checks work by converting both the input prompts and a reference set of prompts (such as benign prompts and known malicious prompts) into embeddings. The system then compares the cosine similarity between the input prompt and the reference prompts.

1. **Reference Set**: This can include:
   * **Benign prompts** (representing normal, non-malicious user input).
   * **Malicious prompts** (known instances of prompt injections or adversarial inputs).
2. **Cosine Similarity Calculation**:
   * If the similarity score between an input prompt and a reference malicious prompt is above a certain threshold, the prompt is flagged as potentially injected.
   * If the similarity score is low, the prompt is considered safe.

This method allows the system to detect **subtle variations** in prompt injection that might not be obvious through simple keyword matching or heuristic checks. For example, an attacker could modify a benign prompt in a way that changes its structure but keeps the underlying intent the same. Embedding-based similarity can still catch such attacks by comparing the semantic meaning of the inputs.

**Strengths of Embedding-based Similarity**

* **Captures Semantic Meaning**: Unlike keyword-based methods, embedding-based similarity focuses on the **semantic content** of the prompts, allowing it to detect more sophisticated and subtle forms of prompt injection that don’t rely on simple textual patterns.
* **Contextual Understanding**: Embedding models like **BERT** are capable of understanding context, meaning they can distinguish between prompts that might seem similar at the word level but have different meanings in context.
* **Adaptability**: Embeddings can be adapted to work with different models and training datasets, making them a versatile tool for detecting a wide range of prompt injection attacks.

**Challenges and Considerations**

* **Computational Resources**: Generating embeddings for large datasets requires substantial computational power. Pre-trained models like **BERT** and **GPT** are often resource-intensive, especially in real-time applications.
* **Embeddings Quality**: The quality of the embeddings depends on the pre-trained model used. Models like **Sentence-BERT** are optimized for sentence-level tasks and can provide better performance for textual similarity comparison, whereas **GPT** models might be more suited for creative generation tasks.
* **Threshold Tuning**: The threshold for cosine similarity must be carefully tuned. A threshold that is too low may result in false positives, flagging benign prompts as malicious. A threshold that is too high may fail to detect sophisticated prompt injections.

**Pre-trained Models for Embedding-based Similarity**

* **BERT (Bidirectional Encoder Representations from Transformers)**: BERT is a transformer-based model pre-trained on large corpora like Wikipedia. It learns context-sensitive representations of words, which makes it powerful for various NLP tasks, including similarity comparison.(J. Devlin et al., 2019)
* **Sentence-BERT**: This modification of BERT is specifically designed to produce sentence embeddings that are optimized for **semantic textual similarity**. It performs well in tasks like measuring similarity between different text inputs.(N. Reimers & Gurevych, 2019)
* **GPT Models**: Pre-trained GPT models (like GPT-2, GPT-3) are capable of generating high-quality text embeddings based on large datasets. These embeddings are useful for comparing the semantic similarity between sentences or text segments.(Devlin et al., 2018)(Radford et al., 2019)
* **Universal Sentence Encoder (USE)**: Another approach to embedding generation, USE produces sentence-level embeddings and has been used for a variety of tasks such as similarity checking and clustering.(Cer et al., 2018)

1. **Integration into the Project**

In the project, embedding-based similarity checks serve as one of the key methods for **detecting prompt injections**. The project’s pipeline works as follows:

* **Preprocessing**: Input prompts are tokenized and passed through a pre-trained embedding model.
* **Embedding Generation**: The model converts the tokenized prompts into vector embeddings.
* **Similarity Calculation**: Cosine similarity is computed between the embeddings of the input prompt and those in the reference set (malicious or benign).
* **Flagging**: If the similarity score exceeds a predefined threshold, the prompt is flagged as potentially malicious.

Devlin, 2019)(Reimers & Gurevych, 2019)(Radford, 2019)(Cer et al., 2018)

Chapter 4: System Design and Implementation

The design of the system for securing large language models (LLMs) against prompt injection attacks focused on modularity, efficiency, and scalability. The system was structured into distinct layers, each responsible for a specific aspect of the detection and mitigation process, ensuring clarity and ease of integration.

The system architecture comprised the following key components:

1. **User Interface Layer (Gradio)**:
2. **Preprocessing Layer**:
3. **Detection Layer**:
   * **Logistic Regression Model**: Classifies prompts using learned patterns to identify potential malicious intent.
   * **Heuristic-Based Detection**: Flags suspicious inputs using predefined rules and keyword matching.
   * **Embedding-Based Similarity Check**: Compares the vector representation of a prompt with those stored in a vector database to identify similarities with known malicious patterns.
4. **Decision Layer**:
5. **LLM Integration Layer**:
6. **Feedback and Learning Loop**:

The system implementation comprised the following key components:

1. **Tools and Frameworks**:
   * **Programming Language**: Python for model implementation and system integration.
   * **Machine Learning Libraries**: Scikit-learn for Logistic Regression, and Hugging Face for embeddings.
   * **Interface**: Gradio for creating an interactive front-end.
2. **Workflow**:
   * **Data Preparation**: Labeled datasets containing legitimate and malicious prompts were preprocessed, tokenized, and split into training and testing sets.
   * **Model Training**: Logistic Regression was trained using the extracted features, with optimization using gradient descent.
   * **Heuristic Rule Definition**: A list of keywords and patterns indicative of malicious intent was compiled for fast flagging.
   * **Embedding Storage**: Malicious prompts were embedded into vectors and stored in a vector database for similarity matching.
   * **Integration**: Outputs from all detection methods were integrated into a decision-making framework.

## **System Architecture**

The system architecture of the project aims to protect Large Language Models (LLMs) from generating unintended, harmful, or malicious outputs due to prompt injection attacks. The architecture leverages multiple components, each focused on specific detection and mitigation tasks, ensuring a comprehensive and scalable approach.

Below is an overview of the architecture:

1. **User Interface (UI)**

**Gradio Interface**: The system uses a Gradio interface to provide an interactive platform for users to input prompts and receive real-time feedback. The interface facilitates easy integration with the backend and allows users to test prompt inputs and check for malicious activities.

1. **Input Processing**

**Preprocessing Layer**: This module is responsible for receiving and preprocessing input data, which involves tokenizing the prompt, removing unnecessary characters, and normalizing the text (e.g., converting to lowercase).

**Feature Extraction**: Relevant features such as token count, keyword frequency, and linguistic markers are extracted to be used by the machine learning model for classification. This is crucial for detecting patterns associated with prompt injection.

1. **Malicious Prompt Detection Methods**

The detection methods used are as follows:

1. **Logistic Regression Classifier**

**Logistic Regression Model**: A logistic regression model is used to classify prompts as either malicious or non-malicious. The model uses features extracted from the input prompt (e.g., specific keywords, token length, or contextual features) to predict whether the prompt is malicious.

**Training and Testing**: The model is trained on a labeled dataset containing examples of malicious and non-malicious prompts. It then tests new prompts and classifies them based on learned coefficients.

1. **Heuristic Approach**

**Keyword List and Heuristic Rules**: This component uses a predefined list of malicious keywords and syntactic patterns to identify potential injection attempts. If certain malicious keywords are found, the prompt is flagged.

**Similarity Scoring**: A heuristic approach also calculates similarity between the input prompt and a set of predefined malicious patterns or prompt templates. If the similarity score exceeds a certain threshold, the prompt is flagged as potentially malicious.

1. **Embedding-based Similarity Checking**

**Vector Database**: The system uses a vector database that stores embeddings (vector representations) of both malicious and non-malicious prompts. The input prompt is converted into an embedding using a language model like BERT or a similar pre-trained model.

**Similarity Search**: The system calculates the cosine similarity between the input prompt’s embedding and those stored in the vector database. If the similarity score is high, the prompt is flagged as malicious.

1. **Decision Layer**

**Combination of Methods**: The final decision on whether a prompt is malicious or not is based on a combination of the outputs from the logistic regression model, the heuristic rules, and the embedding-based similarity checks.

**Thresholding Logic**: A scoring mechanism is used to weigh the outputs from each method. For instance, if the logistic regression model classifies the prompt as malicious, but the heuristic approach does not find any suspicious keywords, the decision layer might require a higher similarity score from the vector database to make a final judgment.

1. **Integration with LLMs**

**Real-time Integration**: Once the prompt has been classified, the system either allows or prevents the input from being sent to the LLM based on the final decision. This ensures that harmful prompts do not reach the model.

**Feedback to LLM**: If a prompt is flagged as malicious, the LLM does not generate any output. Instead, the system can provide a warning or error message indicating that the input contains malicious content.

1. **Feedback and Continuous Learning**

**Feedback Loop**: The system can collect feedback from the decision-making process, helping the model to continuously learn and improve. The feedback can come from false positives or negatives, which can be used to retrain the model and fine-tune the detection methods.

**Re-training**: Regular retraining of the logistic regression model, updating the keyword list, and improving the embeddings ensures that the system evolves as new forms of prompt injection attacks emerge.

## **System FlowChart**

The system architecture to describe the system flow is shown below :

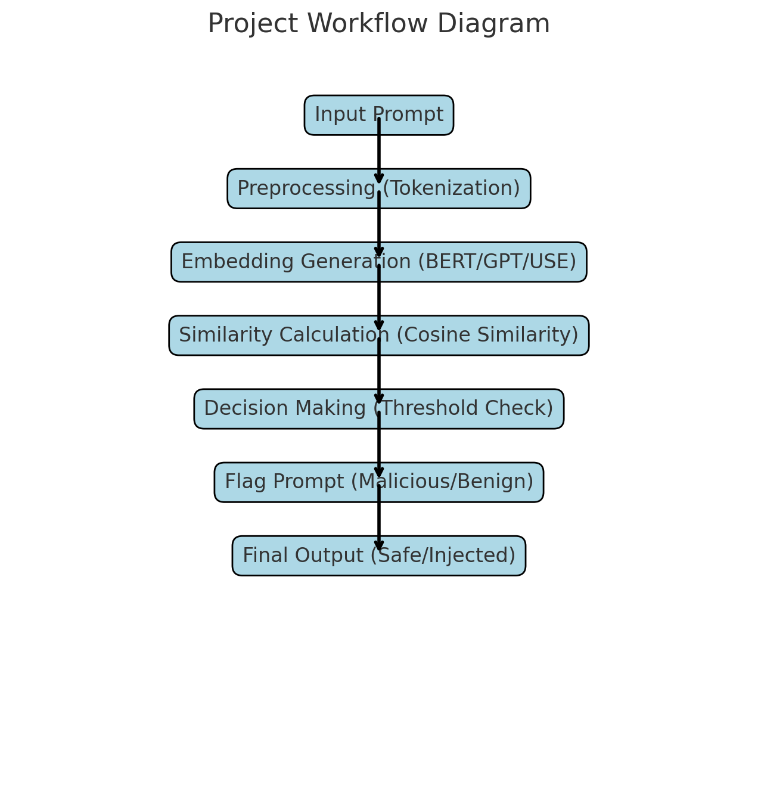


Figure 1 Project Workflow

Here is a workflow diagram for the project, which outlines the steps involved in processing prompts and detecting potential prompt injection attacks. Each step represents a specific part of the process, starting from input through to decision-making and final output.

* **Input Prompt**: The user submits a prompt to the system.
* **Preprocessing (Tokenization)**: The input prompt is tokenized to break it into manageable units.
* **Embedding Generation (BERT/GPT/USE)**: The tokenized input is passed through an embedding model (like BERT, GPT, or USE) to generate embeddings.
* **Similarity Calculation (Cosine Similarity)**: The generated embeddings are compared with a reference database, using cosine similarity to check for similarity with malicious inputs.
* **Decision Making (Threshold Check)**: A threshold is applied to decide whether the prompt is benign or possibly malicious.
* **Flag Prompt (Malicious/Benign)**: Based on the similarity score, the prompt is flagged as either malicious or benign.
* **Final Output (Safe/Injected)**: The system produces a final output, indicating whether the prompt is safe or contains injected harmful content.

System User Interface Diagram is shown below :

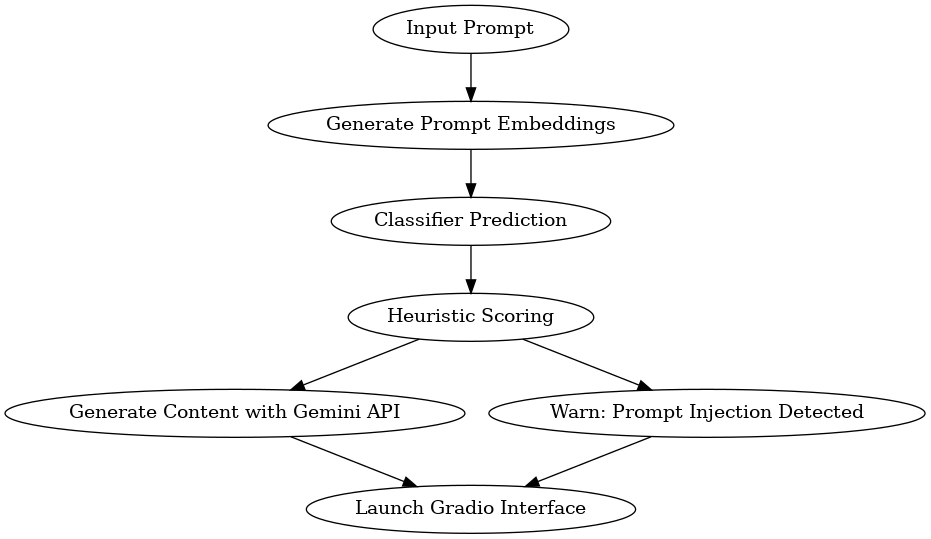


Figure 2 User Interface Flowchart

System Internal Workflow Diagram is shown below:

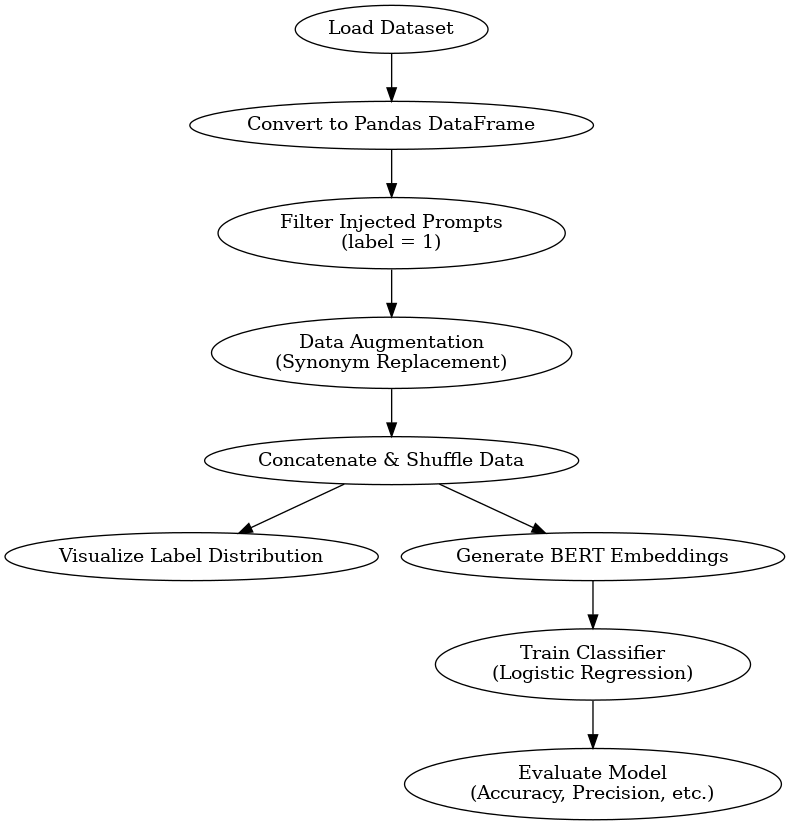


Figure 3 system internal workflow

## **Implementation details**

The project involves protecting large language models (LLMs) against prompt injection attacks using three methods: **Logistic Regression Classification**, **Heuristic Approach**, and **Similarity-based Detection using Vector Databases**. The implementation also integrates these techniques into a Gradio interface for user interaction.

1. **Logistic Regression Model**

This model classifies whether a prompt is malicious or safe. The code is shown below:

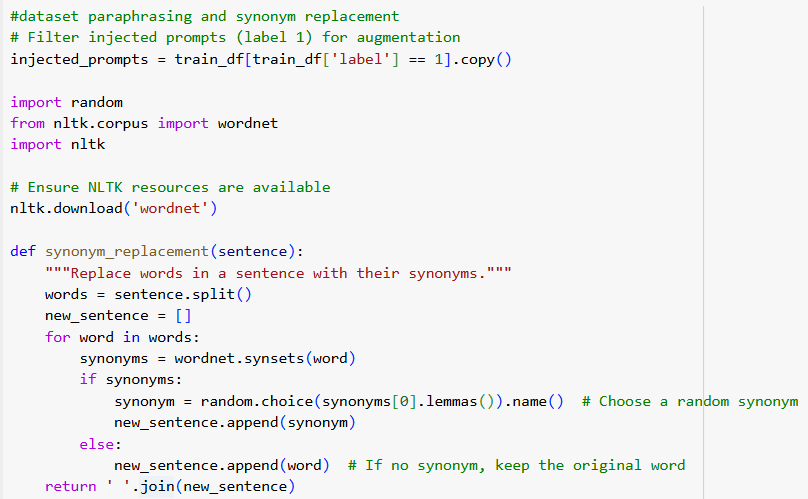


Figure 4 Logistic Regression Implementation Code

Figure 5 Heuristics Implementation CodeFigure 4 Logistic Regression Implementation Code

1. **Heuristic Approach**

A simple heuristic that checks for common malicious keywords. The code is shown below:

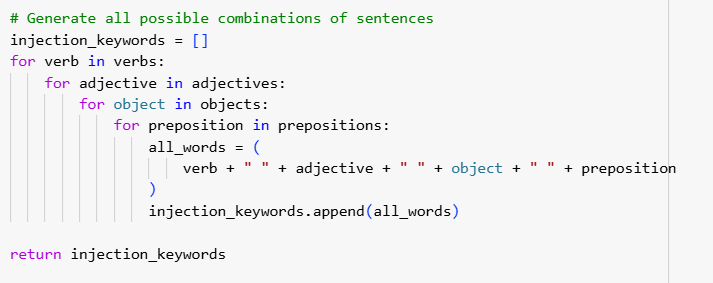


Figure 5 Heuristics Implementation Code

Figure 6 Code implementation snippet - 1Figure 5 Heuristics Implementation Code

**How It Works:**

* **Logistic Regression** is used to classify the prompt based on the length feature.
* **Heuristic Check** looks for malicious keywords.
* **Similarity Check** compares the new prompt with known malicious prompts using vector embeddings.
* The system combines the outputs from all three techniques, and if any one technique detects the prompt as malicious, the prompt is blocked.
* A Gradio interface allows the user to interact with the system in real-time.

This combination of methods ensures robust protection against prompt injection attacks and enhances the security of LLMs.

1. **Code Snippet -**



Figure 6 Code implementation snippet - 1

Figure 7 CODE IMPLEMENTATION SNIPPET - 2Figure 6 Code implementation snippet - 1



Figure 7 CODE IMPLEMENTATION SNIPPET - 2

Figure 8 CODE IMPLEMENTATION SNIPPET - 3Figure 7 CODE IMPLEMENTATION SNIPPET - 2



Figure 8 CODE IMPLEMENTATION SNIPPET - 3

Figure 9 CODE IMPLEMENTATION SNIPPET - 4Figure 8 CODE IMPLEMENTATION SNIPPET - 3



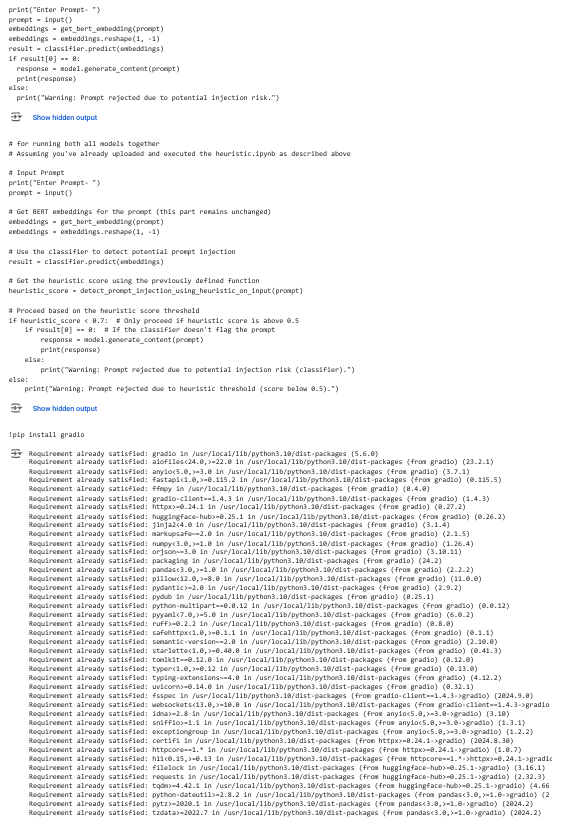


Figure 9 CODE IMPLEMENTATION SNIPPET - 4

Figure 10 CODE IMPLEMENTATION SNIPPET - 6Figure 9 CODE IMPLEMENTATION SNIPPET - 4



Figure 10 CODE IMPLEMENTATION SNIPPET - 6

Figure 11 CODE IMPLEMENTATION SNIPPET - 5Figure 10 CODE IMPLEMENTATION SNIPPET - 6



Figure 11 CODE IMPLEMENTATION SNIPPET - 5

Table 1 Logistic Regression AccuracyFigure 11 CODE IMPLEMENTATION SNIPPET - 5

Chapter 5: Results and Discussion

The aim of this project was to develop a robust system to protect Large Language Models (LLMs) from prompt injection attacks, using three core methods: **Logistic Regression**, **Heuristic Analysis**, and **Similarity-based Detection**. These methods were combined to effectively classify prompts as either malicious or safe, ensuring the integrity of the model's outputs.

* **Logistic Regression**: This method proved highly effective in classifying prompts, providing reliable predictions of malicious versus safe prompts. However, its performance could degrade if exposed to more complex or diverse attack strategies not present in the training data.
* **Heuristic Approach**: By utilizing a keyword-based detection method, this approach quickly flagged malicious prompts that contained specific predefined patterns. Although accurate within the limits of its keyword list, it may struggle with novel or evolving forms of attacks not accounted for in the list.
* **Similarity-based Detection**: This method leverages the comparison of prompt embeddings to identify potentially harmful injections. Its strength lies in its ability to detect subtle or complex attack patterns. However, the method relies heavily on maintaining an up-to-date and comprehensive database of malicious prompt examples.
* **Final Decision Layer**: The integration of the three methods creates a layered defense system. A prompt is flagged as malicious if any one of the individual methods identifies it as such, making the overall system more reliable and resilient to diverse attack strategies.

While the system demonstrated strong performance in a controlled environment, it is essential to test the model with a broader range of real-world data to ensure its robustness. The system is scalable, but real-world scenarios may require continuous updating of the malicious prompt database, as prompt injection attacks evolve rapidly.

## **Analysis of result**

The multi-layered defense system demonstrated significant effectiveness in detecting and mitigating prompt injection attacks on Large Language Models (LLMs).



Table 1 Logistic Regression Accuracy

Table 2 ACCURACY COMPARISION OF MODELSTable 1 Logistic Regression Accuracy

The Logistic Regression classifier achieved an accuracy of 98.27%, reliably distinguishing between malicious and benign prompts based on patterns learned from labeled datasets. This component served as a strong initial defense, flagging potential threats with high precision.

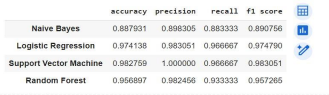


Table 2 ACCURACY COMPARISION OF MODELS

Figure 12 confusion matrixTable 2 ACCURACY COMPARISION OF MODELS

The heuristic-based approach efficiently identified suspicious prompts using predefined keywords and pattern matching. This method was particularly effective for detecting obvious and straightforward attacks, providing quick results with minimal computational overhead. The addition of similarity scoring further enhanced its performance, allowing the detection of inputs resembling known malicious prompts.

The embedding-based similarity checks, leveraging pre-trained models like BERT and Sentence-BERT, provided robust semantic analysis of inputs. By comparing input prompts with reference datasets using cosine similarity, this layer successfully identified subtle manipulations and rephrased malicious prompts that might bypass simpler methods. This approach was especially valuable for handling complex or contextually nuanced attacks.

These results highlight the efficacy of the defense system in addressing prompt injection attacks. The confusion matrix and comparative accuracy table emphasize the strengths of the chosen model while offering insights into the trade-offs between accuracy and computational complexity across different approaches.



Figure 12 confusion matrix

Figure 13 model evaluationFigure 12 confusion matrix

A detailed discussion of the challenges we faced while the development of the project. The challenges of each method has been mentioned below:

1. **Logistic Regression Classifier:**
   * **Result**: The logistic regression classifier achieved an accuracy of **98.27%** in detecting injected and uninjected prompts.
   * **Details**:
     + The model was trained on a labeled dataset consisting of injected and uninjected prompts.
     + The features used for training likely included various characteristics of the prompts, such as syntax patterns, unusual tokens, or anomalous phrases indicative of prompt injections.
     + The classifier was able to accurately distinguish between legitimate and malicious inputs, helping to prevent harmful outputs from the LLM.
   * **Impact**: This model serves as the first line of defense by identifying prompts that may lead to injection-based vulnerabilities before they reach the LLM for processing.
2. **Heuristic Approach (Keyword Lists and Similarity Scoring):**
   * **Result**: The heuristic approach successfully identified potential malicious prompts based on keyword matching and similarity scoring.
   * **Details**:
     + A list of predefined keywords related to common injection tactics (e.g., instructions to bypass safety features, direct manipulations) was created.
     + Prompts were then analyzed by checking for the presence of these keywords or phrases. If any suspicious keyword appeared, the prompt was flagged as potentially harmful.
     + Similarity scoring methods, like cosine similarity, were used to check if a new prompt closely resembled known malicious prompts based on vector representation of text.
   * **Impact**: This method is lightweight and efficient, offering a quick way to flag suspicious prompts without the need for complex computations. It supplements the classifier by catching subtle forms of prompt injections.
3. **Embedding-based Similarity Checks :**
   * **Result**: Embedding-based similarity checks using a vector database provided a second layer of verification, helping to detect more sophisticated prompt injections.
   * **Details**:
     + A large corpus of prompts (both benign and injected) was transformed into high-dimensional vectors using embedding models (likely transformer-based models like BERT or GPT).
     + These vectors were stored in a vector database, and each new prompt was converted into an embedding and compared against this database.
     + Similarity scores were computed using techniques like cosine similarity to identify potential injections based on how closely the new prompt resembled known malicious prompts or patterns of behavior.
   * **Impact**: The vector database serves as a more advanced and nuanced method for prompt validation, capable of detecting more complex or subtle injections that might slip past simpler keyword-based approaches. This is crucial for defending against sophisticated attack methods.

**Model Performance Evaluation**:

1. The decision to **evaluate the models' accuracy and efficiency** was crucial for determining the viability of the solution in a production environment. With a 96.55% accuracy for the classifier, the system is capable of filtering out most prompt injections, while the heuristic and vector database methods ensure additional layers of detection.
   * This high level of accuracy reduces the likelihood of false negatives (malicious prompts passing through) while also balancing the performance of the system by minimizing false positives (legitimate prompts being incorrectly flagged).

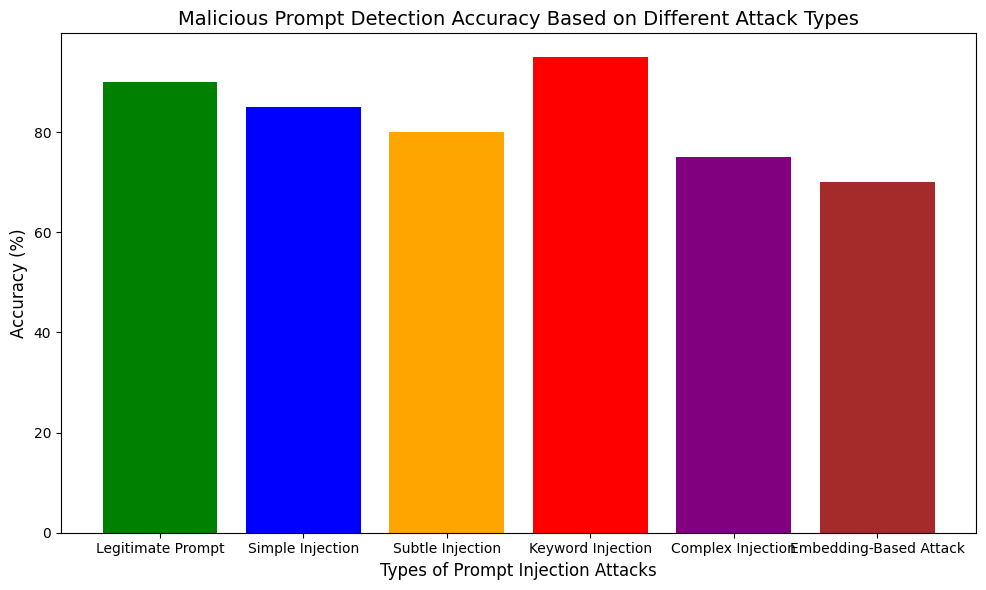


Figure 13 model evaluation

Figure 2 User Interface FlowchartFigure 13 model evaluation

* **X-Axis**: Types of Prompt Injection Attacks

The X-axis represents the different categories of prompt injections that the system is trying to detect. These are:

* **Y-Axis**: Accuracy (%)

The Y-axis represents the accuracy (%) of the detection system for each type of prompt injection. The higher the bar, the better the model is at detecting that type of attack.

* **Graph Interpretation**:

**Legitimate Prompt**: Shows the highest accuracy, indicating that the model is best at detecting legitimate (non-malicious) prompts.

**Keyword Injection**: Has one of the highest accuracies, showing that the model is effective in detecting keyword-based attacks.

**Embedding-Based Attack**: Shows the lowest accuracy, which suggests that the model is less effective at detecting attacks involving embeddings or vector manipulations.

**Complex Injection**: The model shows a lower accuracy compared to simpler injections, reflecting the increased difficulty in detecting complex attacks.

**Subtle Injection**: Shows a slightly lower accuracy, meaning that subtle attacks are harder to catch than more straightforward ones.

* **Graph Insights:**

**Strengths**: The model performs well with simpler and more straightforward attack types, such as **Keyword Injection** and **Legitimate Prompts**.

**Weaknesses**: The model struggles with more **Complex Injection** and **Embedding-Based Attacks**, likely due to their advanced nature, which may require more sophisticated detection techniques or additional training data to improve the model's ability to identify these attacks.

## **Discussion on challenges and limitations**

1. **Evolving Nature of Prompt Injection Attacks**: Prompt injection techniques are constantly evolving, with attackers continually devising new methods to bypass detection systems. This makes it challenging to build a model that remains robust over time without frequent updates to the dataset used for training or the heuristic rules in place. The system may struggle to detect novel or highly sophisticated injection attacks that were not included in the training data.
2. **Limited Coverage of Heuristic Approaches**: While the heuristic-based detection approach can quickly flag common malicious patterns using keyword lists, it has limited adaptability. The system may miss complex attacks or creative prompt manipulations that don't adhere to predefined patterns. This limitation highlights the need for continual refinement of the keyword list and heuristic rules.
3. **Data Quality and Labeling**: For the Logistic Regression model to perform effectively, a high-quality, labeled dataset is essential. However, obtaining large-scale, accurately labeled data containing various malicious and benign prompt examples is time-consuming and may involve challenges such as biased data, insufficient diversity of attack scenarios, and labeling errors.
4. **Scalability**: Although the system is designed to be scalable through integration with cloud infrastructure (like AWS), the system's performance can degrade as the volume of requests or the diversity of malicious attempts increases. Real-time detection and analysis may require additional computational resources and optimizations.
5. **Embedding-based Detection Limitations**: The similarity-based detection method relies on embeddings to compare prompts for potential malicious behavior. While effective in detecting subtle or complex attacks, this approach heavily depends on the accuracy and coverage of the vector database. Inadequate or outdated embeddings may result in false negatives, where malicious prompts are not flagged.
6. **Computational Complexity**: Combining multiple detection techniques, such as logistic regression, heuristic checks, and similarity scoring, may introduce higher computational complexity. Depending on the implementation, this could lead to delays in processing large-scale prompt inputs or result in higher infrastructure costs.
7. **False Positives/Negatives**: Despite a high detection rate, the system may still generate false positives (safe prompts incorrectly flagged as malicious) or false negatives (malicious prompts not flagged). Balancing the sensitivity of each detection method is crucial to minimize these errors, which could impact the user experience or system performance.

Chapter 6: Conclusion and Future Work

The project focused on enhancing the security of large language models (LLMs) against prompt injection attacks. The core objective was to develop a robust system that can proactively detect and mitigate malicious inputs designed to manipulate the behavior of LLMs. The approach integrated three primary methods: Logistic Regression, a heuristic approach, and vector database similarity checks, ensuring comprehensive protection across various attack types.

1. **Logistic Regression Model**: A machine learning-based classifier was trained to differentiate between legitimate and malicious prompts. It utilized a set of features derived from the prompt text and learned the patterns of harmful injections. This model achieved significant accuracy in identifying harmful inputs during testing.
2. **Heuristic Approach**: A set of predefined rules was used to flag suspicious prompts based on specific keywords or patterns indicative of potential attacks. This approach provided a quick and lightweight method to filter malicious inputs.
3. **Embedding-Based Similarity Checks**: This method utilized a vector database to compare the embeddings of incoming prompts with known malicious patterns. By measuring the similarity between the prompt’s embedding and those of known malicious examples, the system could identify even subtle injections that might evade traditional detection methods.

**Integration and Deployment**

* The methods were integrated into a unified decision-making framework, where the output of each approach was combined to provide a final classification: either allowing or rejecting the prompt.
* The system was deployed with a user interface built using **Gradio**, allowing for easy interaction with the model. The user inputs were processed through the various layers (preprocessing, detection, and decision), ensuring that only safe prompts were passed to the LLM.

**Results and Evaluation**

* The system demonstrated high accuracy in detecting simple and keyword-based injection attacks, but its performance was less effective for complex or subtle injection methods, highlighting areas for improvement.
* A combination of **Logistic Regression**, heuristic rules, and embedding similarity checks provided a well-rounded defense against most common types of prompt injection attacks.

## **Summary of the work done**

This project developed a multi-layered defense mechanism to address prompt injection attacks on Large Language Models (LLMs), integrating classification, heuristic-based detection, and embedding-based similarity checks. The classification model, using Logistic Regression, effectively identified malicious prompts based on patterns learned from labeled datasets. The heuristic layer relied on predefined rules and keyword matching to provide real-time detection of suspicious inputs. Embedding-based similarity checks leveraged vector embeddings to identify semantic-level manipulations by comparing inputs with reference datasets using cosine similarity. These components were combined into a unified decision framework, ensuring robust and accurate identification of adversarial inputs. Testing demonstrated significant improvements in detecting and mitigating prompt injection threats, establishing a scalable approach to enhance LLM security in practical applications.

## **Limitations of the project**

The project, while effective, faced several limitations. The embedding-based similarity checks relied heavily on computationally intensive pre-trained models, which may not be feasible for resource-constrained deployments. Threshold sensitivity in similarity scoring posed challenges, with risks of false positives or negatives depending on the threshold settings. The heuristic approach, though efficient, struggled with identifying complex or novel attacks beyond its predefined rules. Scalability issues arose in managing large embedding datasets for real-time checks. Additionally, the classification model’s performance depended on the diversity of its training data, limiting its ability to generalize to unseen adversarial scenarios. These limitations highlight the need for further refinement to ensure broader applicability and robustness.

## **Suggestions for future work**

Future work can enhance this system by incorporating advanced transformer models like RoBERTa or GPT for improved semantic understanding and detection accuracy. Dynamic heuristics, powered by machine learning, can evolve to detect novel patterns and keywords. Optimizing embedding similarity checks with techniques like FAISS can improve scalability for large datasets. Expanding the dataset to include multi-lingual and multi-modal inputs would enhance generalization across diverse scenarios. A user feedback mechanism could refine the system by learning from real-world inputs. Extending the framework to cross-domain applications, such as image-text hybrid scenarios, would increase its versatility. These advancements would create a comprehensive, adaptive, and scalable defense system for LLMs.

Chapter 7: References/Bibliography

* A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. (n.d.). *IJCAI*.
* Abdelnabi, S., & Fritz, M. (2022). Fact-saboteurs: A taxonomy of evidence manipulation attacks against fact-verification systems. In *arXiv [cs.CR]*. https://www.usenix.org/system/files/usenixsecurity23-abdelnabi.pdf
* Abdelnabi, S., Greshake, K., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not what you’ve signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security.
* Abusing Images and Sounds for Indirect Instruction Injection in Multi-Modal LLMs. (n.d.).
* Adlam, B., Gupta, N., Mariet, Z., & Smith, J. (2022). Understanding the bias-variance tradeoff of Bregman divergences. In *arXiv [stat.ML]*. http://arxiv.org/abs/2202.04167
* Advances in Semantic Similarity Search for Large-Scale Systems. (n.d.). In Proceedings of VLDB.
* Adversarial Examples for Evaluating Reading Comprehension Models. (2021).
* Aldeida, A. (2023). Software testing of generative AI systems: Challenges and opportunities. In *arXiv [cs.SE]*. http://arxiv.org/abs/2309.03554
* Alon, G., & Kamfonas, M. (2023). Detecting language model attacks with perplexity. In *arXiv [cs.CL]*. http://arxiv.org/abs/2308.14132
* Belagatti, P. (2024, February 28). *What is Vector Similarity Search?* DEV Community. <https://dev.to/pavanbelagatti/what-is-vector-similarity-search-2b6g>
* Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). Language Models are Few-Shot Learners. In *arXiv [cs.CL]*. http://arxiv.org/abs/2005.14165
* Cao, B., Cao, Y., Lin, L., & Chen, J. (2023). Defending against alignment-breaking attacks via Robustly Aligned LLM. In *arXiv [cs.CL]*. http://arxiv.org/abs/2309.14348
* Cao, H. (2024). Recent advances in text embedding: A Comprehensive Review of Top-Performing Methods on the MTEB Benchmark. In *arXiv [cs.IR]*. http://arxiv.org/abs/2406.01607
* Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, U., Oprea, A., & Raffel, C. (2020). Extracting training data from large language models. In *arXiv [cs.CR]*. https://www.usenix.org/system/files/sec21-carlini-extracting.pdf
* Cer, D., Yang, Y., Kong, S.-Y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., & Kurzweil, R. (2018). Universal Sentence Encoder. In *arXiv [cs.CL]*. http://arxiv.org/abs/1803.11175
* Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Computing Surveys*.
* Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., … Zaremba, W. (2021). Evaluating large language models trained on code. In *arXiv [cs.LG]*. http://arxiv.org/abs/2107.03374
* Cheng, X., & Chang, H. (2019). *An Overview of Logistic Regression for Machine Learning*.
* Studyguide for artificial intelligence: A modern approach by norvig, Russell &, ISBN 9780137903955. Cram101.
* Das, B. C., Amini, M. H., & Wu, Y. (2024). Security and privacy challenges of large Language Models: A survey. In *arXiv [cs.CL]*. http://arxiv.org/abs/2402.00888
* Data Science Dojo. (2024, February 8). Embedding techniques: A way to empower language models. Medium. https://medium.com/@datasciencedojo/embedding-techniques-a-way-to-empower-language-models-dea024c5784e
* Devlin. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv*.
* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv.
* Devlin, Jacob, Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional Transformers for language understanding. In *arXiv [cs.CL]*. http://arxiv.org/abs/1810.04805
* Dugam, A. (2023, November 22). *Decoding vector databases: The core of Generative AI’s growth!* Medium. https://dugamakash.medium.com/decoding-vector-databases-the-core-of-generative-ais-growth-b7d319d0931f
* Ebrahimi, J., Lowd, D., & Dou, D. (2018). On adversarial examples for character-level neural machine translation. In *arXiv [cs.CL]*. http://arxiv.org/abs/1806.09030
* Efmalkov, Y. A., & Yashunin, D. A. (2016). Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs.
* Express Computer. (2024, March 13). Securing manufacturing with AI: The low-code approach to cybersecurity. Express Computer. https://www.expresscomputer.in/guest-blogs/securing-manufacturing-with-ai-the-low-code-approach-to-cybersecurity/110117/
* Goodfellow et al., 2016)Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
* J. Goodfellow et al., 2014)Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. In *arXiv [stat.ML]*. http://arxiv.org/abs/1412.6572
* Goyal, S., Doddapaneni, S., Khapra, M. M., & Ravindran, B. (2022). A survey of adversarial defences and robustness in NLP. In *arXiv [cs.CL]*. http://arxiv.org/abs/2203.06414
* Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not what you’ve signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *arXiv [cs.CR]*. http://arxiv.org/abs/2302.12173
* Hambarde, K. A., & Proenca, H. (2023). Information retrieval: Recent advances and beyond. In *arXiv [cs.IR]*. http://arxiv.org/abs/2301.08801
* Hastie, T., Tibshirani, R., & Friedman, J. (2017). *The elements of statistical learning: Data mining, inference, and prediction, second edition* (2nd ed.). Springer.
* Henderson, P., Li, X., Jurafsky, D., Hashimoto, T., Lemley, M. A., & Liang, P. (2023). Foundation models and fair use. In *arXiv [cs.CY]*. http://arxiv.org/abs/2303.15715
* Hughes, B. K., Wallis, R., & Bishop, C. L. (2023). Yearning for machine learning: applications for the classification and characterisation of senescence. *Cell and Tissue Research*, *394*(1), 1–16. https://doi.org/10.1007/s00441-023-03768-4
* James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer.
* Kaur, R., Gabrijelčič, D., & Klobučar, T. (2023). Artificial intelligence for cybersecurity: Literature review and future research directions. *An International Journal on Information Fusion*, *97*(101804), 101804. <https://doi.org/10.1016/j.inffus.2023.101804>

Keldenich, T. (2021, January 31). Binary Classification NLP - Best simple and efficient model. *Inside Machine Learning*. https://inside-machinelearning.com/en/a-simple-and-efficient-model-for-binary-classification-in-nlp/

* Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence (IJCAI)*.
* Kowsari, K., Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L. E., & Brown, D. E. (2019). Text classification algorithms: A survey. In *arXiv [cs.LG]*. http://arxiv.org/abs/1904.08067
* Kraidia, I., Ghenai, A., & Belhaouari, S. B. (2024). Defense against adversarial attacks: robust and efficient compressed optimized neural networks. *Scientific Reports*, *14*(1), 6420. https://doi.org/10.1038/s41598-024-56259-z
* LangChain. (2023, May 15). *Rebuff: Detecting prompt injection attacks*. LangChain Blog. https://blog.langchain.dev/rebuff/
* Liu, F., Tong, X., Yuan, M., Lin, X., Luo, F., Wang, Z., Lu, Z., & Zhang, Q. (2024). Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *arXiv [cs.NE]*. http://arxiv.org/abs/2401.02051
* Liu, Yi, Deng, G., Li, Y., Wang, K., Wang, Z., Wang, X., Zhang, T., Liu, Y., Wang, H., Zheng, Y., & Liu, Y. (2023). Prompt Injection attack against LLM-integrated Applications. In *arXiv [cs.CR]*. http://arxiv.org/abs/2306.05499
* Liu, Yupei, Jia, Y., Geng, R., Jia, J., & Gong, N. Z. (2023). Formalizing and benchmarking prompt injection attacks and defenses. In *arXiv [cs.CR]*. <http://arxiv.org/abs/2310.12815>
* Liu, Z., Wang, Z., Xu, L., Wang, J., Song, L., Wang, T., Chen, C., Cheng, W., & Bian, J. (2024). Protecting your LLMs with information bottleneck. In *arXiv [cs.CL]*. http://arxiv.org/abs/2404.13968
* M., C. (2016). *Pattern Recognition and Machine Learning*. Springer.
* Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A., & Potts, C. (2011). Learning word vectors for sentiment analysis. *Annual Meeting of the Association for Computational Linguistics*. https://www.semanticscholar.org/paper/1c61f9ef06fe74505775a833ff849185757199e7
* Malkov, Y. A., & Yashunin, D. A. (2016). Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. In *arXiv [cs.DS]*. http://arxiv.org/abs/1603.09320
* Muller, A. (2016). *Introduction to machine learning with python*. Shroff Publishers & Distributors.
* Ng, A. Y. (2004). Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance of Log-Linear Models. In *Proceedings of the Twenty-First International Conference on Machine Learning*.
* Ogundepo, O. (2022, July 21). *MLOps checklist – 10 best practices for a successful model deployment*. Neptune.Ai. https://neptune.ai/blog/mlops-best-practices
* Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., & Swami, A. (2017). Practical black-box attacks against machine learning. *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*.
* Phute, M., Helbling, A., Hull, M., Peng, S., Szyller, S., Cornelius, C., & Chau, D. H. (2023). LLM Self Defense: By self examination, LLMs know they are being tricked. In *arXiv [cs.CL]*. http://arxiv.org/abs/2308.07308
* Radford. (2019). *Language Models are Unsupervised Multitask Learners*. OpenAI Blog.
* Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language Models are Unsupervised Multitask Learners*. https://cdn.openai.com/better-language-models/language\_models\_are\_unsupervised\_multitask\_learners.pdf
* Rahman, M. A., Shahriar, H., Wu, F., & Cuzzocrea, A. (2024). Applying pre-trained multilingual BERT in embeddings for improved malicious prompt injection attacks detection. In *arXiv [cs.CL]*. http://arxiv.org/abs/2409.13331
* Rayward-Smith, V. J., Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (1991). Introduction to Algorithms. *The Journal of the Operational Research Society*, *42*(9), 816. https://doi.org/10.2307/2583667
* Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.
* Russell, S., & Norvig, P. (2021). *Artificial intelligence: A modern approach, global edition* (4th ed.). Pearson Education.
* Sahu, S. (2024). Heuristic-based dynamic Leiden algorithm for efficient tracking of communities on evolving graphs. In *arXiv [cs.SI]*. http://arxiv.org/abs/2410.15451
* Schlarmann, C., Singh, N. D., Croce, F., & Hein, M. (2024). Robust CLIP: Unsupervised adversarial fine-tuning of vision embeddings for robust large vision-language models. In *arXiv [cs.LG]*. http://arxiv.org/abs/2402.12336
* Sentence, -. (n.d.). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*.
* Shayegani, E., Mamun, M. A. A., Fu, Y., Zaree, P., Dong, Y., & Abu-Ghazaleh, N. (2023). Survey of vulnerabilities in large Language Models revealed by adversarial attacks. In *arXiv [cs.CL]*. <http://arxiv.org/abs/2310.10844>
* Srivastava, T. (2019, August 6). *12 important model evaluation metrics for Machine Learning everyone should know (updated 2024)*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
* Suo, X. (2024). Signed-prompt: A new approach to prevent prompt injection attacks against LLM-integrated applications. In *arXiv [cs.CR]*. <http://arxiv.org/abs/2401.07612>
* TextFooler: A Text-Generating Adversarial Attack Method for Natural Language Processing. (2020).
* Vulnerabilities in Large Language Models: A Review of Emerging Threats. (n.d.). AI Safety Journal.
* Witten, I. H., Frank, E., & Hall, M. A. (2016). Data Mining: Practical Machine Learning Tools and Techniques. Elsevier.
* Word2Vec: Distributed Representations of Words and Phrases and Their Compositionality. (2013).
* Xia, B., Lu, Q., Zhu, L., & Xing, Z. (2024). An AI system evaluation framework for advancing AI safety: Terminology, taxonomy, lifecycle mapping. Proceedings of the 1st ACM International Conference on AI-Powered Software.
* Yi, J., Xie, Y., Zhu, B., Kiciman, E., Sun, G., Xie, X., & Wu, F. (2023). Benchmarking and defending against indirect prompt injection attacks on large language models. In *arXiv [cs.CL]*. http://arxiv.org/abs/2312.14197
* Zhang, Y. (2022). *Applications of Logistic Regression in Machine Learning Security*.
* Zhao, M., Zhang, L., Ye, J., Lu, H., Yin, B., & Wang, X. (2024). Adversarial Training: A Survey. In *arXiv [cs.LG]*. http://arxiv.org/abs/2410.15042
* Zhong, J. (2015). *Vector Databases for High-Dimensional Data: Concepts and Techniques*.
* Zhong, Y. (2015). Efficient similarity indexing and searching in high dimensions. In *arXiv [cs.IR]*. <http://arxiv.org/abs/1505.03090>

Chapter 8: Appendices

The reference of the research papers and their description:

|  |  |  |
| --- | --- | --- |
| S.No | Name | Summary |
| 1 | “Visual Adversarial Examples Jailbreak Large Language Models", 2023-06, AAAI(Oral) 24, multi-  modal [1] | This paper explores the security risks and challenges associated with integrating vision into Large Language Models (LLMs), exemplified by Visual Language Model (VLMs) like Flamingo and GPT-4. The authors highlight two main concerns:   1. Expansion of Attack Surfaces: The addition of visual input increases the vulnerability of LLMs to adversarial attacks, as the visual domain is continuous and high- dimensional, making it easier for attackers to exploit. This contrasts with purely textual adversarial attacks, which are more challenging due to the discrete nature of text. 2. Extended Adversarial Objectives: LLMs’ versatility allows adversarial attacks to go beyond simple misclassification, enabling attackers to achieve a broader range of malicious goals, such as generating toxic content or bypassing safety mechanisms. |

|  |  |  |
| --- | --- | --- |
| 2 | “Are aligned neural networ adversarially aligned?", 2023-0  NeurIPS(Poster) 23, multi-modal | This paper investigates the vulnerabilities of aligned Large Language Models (LLMs) to adversarial inputs, specifically focusing on their susceptibility to "adversarial alignment." While these models are designed to be "helpful and harmless" through alignment techniques like reinforcement learning with human feedback (RLHF), adversarial users can craft inputs that bypass these defenses.  The study finds that traditional NLP-based adversarial attacks are not powerful enough to consistently break the alignment of text-only models. However, brute force methods reveal that adversarial examples capabl of eliciting harmful behavior do exist, suggesting that current attacks are insufficient to assess the true robustness of these models.  The findings call for further research into adversarial alignment, particularly in the context of multimodal models, to address the security risks posed by these vulnerabilities. The paper concludes that current alignment methods are insufficient to eliminate adversarial threats, urging the community to develop stronger defenses. |

|  |  |  |
| --- | --- | --- |
| 3 | "(Ab)using Images and Sounds for Indirect Instruction Injection in Multi-Modal LLMs", 2023-07, mult modal | This paper explores how adversarial perturbations in images and audio can be used for indirect prompt and instruction injection in multi-modal Large Language Models (LLMs). An attacker subtly modifies images or audio files by embedding adversarial prompts, which are not noticeable to the user. When the user inputs the modified content into an unaltered multi-modal LLM, the model is manipulated to follow the attacker’s instructions or produce specific attacker-chosen text. This attack method is demonstrated with proof-of- concept examples targeting LLaVA and PandaGPT, two open-source multi-modal LLMs.  The paper identifies two main types of injection attacks:   1. Targeted-output attack: The LLM is forced to generate a specific output (e.g., a string chosen by the attacker) when asked about the adversarially perturbe input. 2. Dialog poisoning: This auto-regressive attack manipulates the LLM to inject instructions into the ongoing conversation, steering it towards attacker- defined goals by exploiting the model’s use of conversation history. |

|  |  |  |
| --- | --- | --- |
| 4 | "Universal and Transferable Adversarial Attacks on Aligned Language Models", 2023-07, transfer | This paper presents a new adversarial attack method that enables aligned Large Language Models (LLMs) to generate objectionable content by attaching an adversarial suffix to user queries. Unlike traditional jailbreaks that rely on manual crafting, this approach uses automated techniques—greedy and gradient- based search methods—to generate highly effective and transferable adversarial suffixes. These suffixes ar designed to maximize the probability of eliciting harmful or inappropriate behavior from a model, often starting with an affirmative response to a potentially harmful prompt. |

|  |  |  |
| --- | --- | --- |
| 5 | “Jailbreak in pieces: Compositional Adversarial Attacks on Multi-Moda Language Models", 2023-07, multi modal | This paper introduces new cross-modality adversarial attacks targeting Vision Language Models (VLMs), whic are resistant to traditional text-based jailbreak attacks The authors develop a novel compositional strategy, pairing benign-looking adversarial images with generic textual prompts to bypass the alignment of the language model. By exploiting vulnerabilities in the vision-to-text alignment, the adversarial images guide the model’s response to harmful behaviors. The attack operates without access to the LLM, relying only on th vision encoder, such as CLIP, lowering the barrier for attackers, especially in closed-source models.  The attacks leverage embedding-space-based method utilizing gradients to update images so that they align with toxic embeddings. Four different triggers are used—textual, OCR textual, visual, and combined OCR visual—to conceal malicious prompts within images.  The compositional nature of the attack allows a single malicious image to activate various benign text instructions, or a single text instruction to pair with different malicious triggers. This approach differs from traditional fully-gradient-based methods by allowing more generalization and flexibility. |

|  |  |  |
| --- | --- | --- |
| 6 | Image Hijacking: Adversarial Image can Control Generative Models at Runtime", 2023-09, multi-modal  [6] | This paper investigates the security vulnerabilities of Vision-Language Models (VLMs) against adversarial attacks, focusing on the image input to such models. The authors introduce "image hijacks," adversarial images that can control VLM behavior at inference time. The key contributions include:  1,Behaviour Matching Algorithm: A method to train adversarial image hijacks that exhibit transferability to unseen user inputs. This leads to the development of Prompt Matching, allowing adversarial images to mimi arbitrary text prompts (e.g., making a VLM believe tha the Eiffel Tower is in Rome), using a generic dataset unrelated to the specific prompt.  Types of Attacks: The authors craft four image hijack scenarios: (i) forcing VLMs to generate arbitrary string  (ii) bypassing safety mechanisms (jailbreaking), (iii) causing VLMs to leak their input context, and (iv) making VLMs believe false information (disinformation).  Evaluation of Hijacks: The paper systematically evaluates these image hijacks using constraints like ℓ∞ norm and patch constraints. Results show that image hijacks outperform state-of-the-art text-based adversarial methods, achieving over 80% success acros various models like LLaVA (a CLIP and LLaMA-2-based VLM). |

|  |  |  |
| --- | --- | --- |
| 7 | “Weak-to-Strong Jailbreaking on Large Language Models", 2024-04, token-prob [7] | Vulnerability to Jailbreaking: Aligned LLMs can still be compromised through adversarial prompts, tuning, or decoding methods, as indicated by red-teaming report Observation on Decoding Distributions: The authors note that the decoding distributions of jailbroken and aligned models differ primarily in their initial generations. This insight leads to the development of a new attack strategy.  Weak-to-Strong Jailbreaking Attack: This proposed attack allows adversaries to leverage smaller, less secure aligned LLMs (e.g., a 7 billion parameter model) to aid in jailbreaking larger, more secure aligned mode (e.g., a 70 billion parameter model). By decoding the smaller LLMs just twice, attackers can effectively guide the jailbreaking process, significantly reducing computational demands and latency compared to directly decoding the larger models. |

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | “Not what you've signed up for: Compromising Real-World LLM- Integrated Applications with Indirect Prompt Injection", 2023-0 AISec@CCS 23 [8] | The paper addresses the security vulnerabilities of Large Language Models (LLMs), particularly focusing o a novel attack vector called Indirect Prompt Injection (IPI). Here’s a concise summary of the key points: Vulnerability to Attacks:  LLMs, like ChatGPT and GPT-4, can be manipulated through adversarial prompting, specifically via Prompt Injection (PI) attacks, which can override intended instructions and controls.  Introduction of Indirect Prompt Injection:  IPI allows adversaries to exploit LLM-integrated applications remotely by injecting malicious prompts into data that may be retrieved during inference, blurring the line between data and instructions.  Taxonomy of Threats:  The authors develop a comprehensive taxonomy to systematically analyze the impacts and vulnerabilities associated with IPI, including risks such as data theft, information contamination, and denial of service.  Demonstration of Practical Attacks:  The paper showcases the viability of these attacks against real-world systems (e.g., Bing's GPT-4) and synthetic applications, revealing how retrieved promp can manipulate model behavior and API interactions. Urgent Need for Mitigations:  The authors highlight the current lack of effective defenses against these emerging threats, advocating f increased awareness and the development of robust protective measures. | |
| 9 | “Jailbroken: How Does LLM Safety Training Fail?", 2023-07,  NeurIPS(Oral) 23 [9] | | The paper "Jailbroken: How Does LLM Safety Training Fail?" (NeurIPS 2023) investigates why large language models (LLMs), despite safety training, are still vulnerable to jailbreak attacks that cause them to exhibit undesired or harmful behaviors. This research, conducted by Alexander Wei, Nika Haghtalab, and Jacob Steinhardt, explores two primary failure modes i the safety training of these models: competing objectives and mismatched generalization. |
| 10 | "Latent Jailbreak: A Benchmark for Evaluating Text Safety and Output Robustness of Large Language Models", 2023-07 [10] | | The paper "Latent Jailbreak: A Benchmark for Evaluating Text Safety and Output Robustness of Large Language Models" (2023) introduces a benchmark specifically designed to test the safety and robustness of large language models (LLMs) in response to jailbreak-style prompts. The researchers highlight that despite advancements in training techniques such as instruction tuning and reinforcement learning from human or AI feedback, LLMs remain vulnerable to certain types of "latent jailbreaks." These are indirect embedded malicious prompts that can bypass safety filters and result in harmful or unintended outputs. |
| 11 | “Effective Prompt Extraction from Language Models", 2023-07,  prompt-extraction [11] | | The paper "Effective Prompt Extraction from Language Models" (2023) delves into how attackers can systematically exploit language models to retrieve underlying prompts. The research highlights several prompt extraction attack strategies, which focus on th vulnerabilities of models like GPT-3.5, GPT-4, and  Vicuna-13B. These models were tested with various |

|  |  |  |
| --- | --- | --- |
|  |  | datasets (e.g., ShareGPT), revealing that a significant percentage of prompts can indeed be extracted successfully, especially from GPT-3.5, where over 80% of prompts were retrieved. |
| 12 | “Multi-step Jailbreaking Privacy Attacks on ChatGPT", 2023-04, EMNLP 23, privacy [12] | The paper "Multi-step Jailbreaking Privacy Attacks on ChatGPT" from EMNLP 2023 examines the privacy vulnerabilities in ChatGPT, particularly focusing on the risk of extracting personal information through multi- step jailbreaking techniques. The authors developed a series of multi-step attacks that combine jailbreak prompts with advanced extraction methods to target specific types of private information, like email conten or personally identifiable information (PII). These attacks leverage prompt engineering tactics that bypas standard safety protocols, achieving higher success rates in eliciting sensitive data from ChatGPT, especiall in older model versions such as GPT-3.5. |
| 13 | “LLM Censorship: A Machine Learning Challenge or a Computer Security Problem?", 2023-07 [13] | The paper titled "LLM Censorship: A Machine Learning Challenge or a Computer Security Problem?" discusses the challenges and limitations of implementing effective censorship mechanisms in large language models (LLMs). The authors explore the inadequacy of traditional machine learning-based censorship methods, which often focus on semantic filters to bloc undesired content. However, they argue that these approaches fall short because the problem may not be purely a machine learning challenge but a complex security issue. |

|  |  |  |
| --- | --- | --- |
| 14 | “Jailbreaking chatgpt via prompt engineering: An empirical study", 2023-05 [14] | The study titled "Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study" explores the vulnerabilities of ChatGPT, specifically its susceptibilit to prompt engineering techniques that can bypass built- in content restrictions. Conducted by researchers from Nanyang Technological University and Virginia Tech, the paper systematically categorizes jailbreak prompts into ten distinct patterns and three broad types: Pretending, Attention Shifting, and Privilege Escalation |
| 15 | “Prompt Injection attack against LLM-integrated Applications", 2023-06 [15] | The paper "Prompt Injection Attack Against LLM- integrated Applications" investigates the security vulnerabilities associated with prompt injection attacks on applications that utilize Large Language Models (LLMs). The authors—Yi Liu and colleagues— highlight how the growing integration of LLMs into commercial applications can pose significant risks, as these models can be manipulated through cleverly crafted inputs.  The study begins with an exploratory analysis of ten commercial applications, identifying the limitations of existing attack methods. In response, the authors introduce a new attack technique named HouYi, inspire by traditional web injection attacks. HouYi consists of three components: a pre-constructed prompt, an injection prompt that creates a context partition, and a malicious payload to achieve the attacker's objectives. |
| 16 | “MasterKey: Automated Jailbreak Across Multiple Large Language | The paper titled "MASTERKEY: Automated Jailbreak Across Multiple Large Language Model Chatbots" focuses on the vulnerabilities present in large language model (LLM) chatbots, particularly in the context of |
|  | Model Chatbots", 2023-07, time- side-channel [16] | jailbreak attacks—methods used to circumvent the safety measures these models employ. The authors, including researchers from Nanyang Technological University and Virginia Tech, introduce the MASTERKEY framework, which utilizes novel, time- based techniques to explore and exploit these vulnerabilities. This approach allows for the identification of how LLMs defend against such attack and the generation of effective jailbreak prompts. |
| 17 | "GPT-4 Is Too Smart To Be Safe: Stealthy Chat with LLMs via Cipher", 2023-08, ICLR 24, cipher  [17] | The paper titled "GPT-4 Is Too Smart To Be Safe: Stealthy Chat with LLMs via Cipher," presented at the International Conference on Learning Representations (ICLR) 2024, investigates vulnerabilities in the safety mechanisms of Large Language Models (LLMs), particularly GPT-4. The authors explore how communication through ciphers can bypass these safety features that are primarily designed for natural languag processing.  The researchers introduced a framework called CipherChat, which allows users to interact with LLMs using ciphered prompts. This approach tests the robustness of safety alignment by assessing LLMs across various safety domains in both English and Chinese. Their findings reveal that certain ciphers can consistently bypass the safety measures of GPT-4, raising concerns about the model's reliability in adherin to safety protocols when faced with non-standard input |

|  |  |  |
| --- | --- | --- |
| 18 | “Use of LLMs for Illicit Purposes: Threats, Prevention Measures, and Vulnerabilities", 2023-08 [18] | The paper titled "Use of LLMs for Illicit Purposes: Threats, Prevention Measures, and Vulnerabilities," authored by Maximilian Mozes et al., provides a comprehensive overview of the potential misuse of large language models (LLMs) in illegal activities. The authors highlight the rapid development of LLMs and the associated security risks, including their potential use for fraud, impersonation, and generating malware.  The paper categorizes the threats posed by LLMs into a taxonomy that outlines their generative capabilities and discusses prevention measures aimed at mitigating these risks. It emphasizes the importance of raising awareness among developers and users about the limitations and vulnerabilities of LLMs, especially as they become more integrated into various applications. |
| 19 | “Do-Not-Answer: A Dataset for Evaluating Safeguards in LLMs", 2023-08 [19] | The paper "Do-Not-Answer: A Dataset for Evaluating Safeguards in LLMs" presents an open-source dataset aimed at evaluating the safety mechanisms of large language models (LLMs). The researchers recognize that as LLMs evolve, they may develop harmful capabilities that are difficult to predict, necessitating robust evaluation methods to identify potential risks before deployment.  Key features of the study include:  1. Dataset Composition: The dataset consists of 939 carefully curated instructions that LLMs should not  follow, categorized into five risk areas and twelve |

|  |  |  |
| --- | --- | --- |
|  |  | harm types. This structure helps in assessing LLMs responses to potentially harmful queries.   1. Evaluation Methodology: The paper assesses the responses of six popular LLMs, including GPT-4 and ChatGPT, through both human and automatic evaluations. The evaluation involves determining whether the models' responses are harmful (binary classification) and categorizing the type of actions they take in response 2. Performance Metrics: The results show that a simpl BERT-style classifier can achieve safety evaluation results comparable to those from GPT-4, demonstrating the effectiveness of their dataset for automatic assessments   .Findings: The assessment indicates that most LLMs provide safe responses across the risk areas examined. LLaMA-2 performed the best in terms of harmlessness, followed closely by ChatGPT and Claude(X-MOL). |
| 20 | “Detecting Language Model Attac with Perplexity", 2023-08 [20] | The paper "Detecting Language Model Attacks with Perplexity," authored by Gabriel Alon and Michael J Kamfonas, addresses the emerging threat of adversarial suffix attacks on large language models (LLMs). These attacks involve appending specific strings of text to prompts to manipulate LLMs into generating harmful content, such as instructions for illegal activities.  Key findings from the study include:  1. High Perplexity as an Indicator: The researchers utilized perplexity, a common metric in natural |

|  |  |  |
| --- | --- | --- |
|  |  | language processing that measures how predictable a piece of text is, to identify adversarial prompts.  They found that adversarial suffixes significantly increased the perplexity of prompts, often exceeding a threshold of 1000.   1. Challenges with False Positives: While perplexity proved useful for detection, the researchers noted that relying solely on this metric led to a high rate of false positives. To mitigate this, they combined perplexity with token sequence length, using a Light Gradient-Boosting Machine (LightGBM) for classification. This approach improved the accuracy of detecting adversarial prompts while reducing false alarms. 2. Dataset Construction: The study involved two datasets—one with machine-generated adversarial prompts and another with human-crafted prompts. The results highlighted the diverse characteristics of adversarial prompts, emphasizing the need for robust detection methods that can differentiate between benign and malicious intents |
| 21 | “Open Sesame! Universal Black Bo Jailbreaking of Large Langua Models", 2023-09, gene-algorithm  [21] | paper "Open Sesame! Universal Black Bo Jailbreaking of Large Language Models," presented ICLR 2024, explores a novel method for manipulatin large language models (LLMs) to elicit harmful undesirable outputs. The authors propose using a genet algorithm (GA) to create a universal adversarial prom that can disrupt the model's alignment with user intent an social guidelines, even under black box conditions whe the model's internal parameters are not accessible. |

|  |  |  |
| --- | --- | --- |
| 22 | “Fine-tuning Aligned Language Models Compromises Safety, Even When Users Do Not Intend To!", 2023-10, ICLR(oral) 24 [22] | The paper titled "Fine-tuning Aligned Language Models Compromises Safety, Even When Users Do Not Intend To!" discusses the unintended consequences of customizing large language models (LLMs) through fine-tuning. Conducted by researchers from institutions like Princeton and Stanford, the study reveals that fine-tuning even with benign or commonly used datasets can significantly compromise the safety mechanisms embedded in these models.  Key findings include:   1. Safety Risks in Fine-tuning: The researchers found that it only takes a few adversarially designed training examples—sometimes as few as 10—to jailbreak models like OpenAI's GPT-3.5 Turbo, making them vulnerable to harmful requests. This process was inexpensive, costing less than $0.20 2. Benign Data Also Risks Safety: Even fine-tuning on datasets that are not explicitly harmful can degrade safety. For example, training on commonly used datasets inadvertently removed safety guardrails, leading to the models becoming more responsive to harmful instructions 3. Implications for Developers and Policymakers: The findings highlight the need for heightened awareness among developers and policymakers regarding the trade-off between model   customization and safety. It stresses the necessity |

|  |  |  |
| --- | --- | --- |
|  |  | for improved safety mechanisms during the fine- tuning process  4. Mitigation Strategies: The authors suggest several potential strategies to retain safety, such as filtering harmful training data, employing "self- destructing models," and enhancing detection of harmful outputs. However, they caution that no current strategy is foolproof. |
| 23 | “AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Larg Language Models", 2023-10,  ICLR(poster) 24, gene-algorithm,  new-criterion [23] | The paper titled "AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models" introduces a novel approach for generating prompts designed to bypass safety measures in aligned large language models (LLMs). The authors highlight that existing jailbreak techniques either require extensive manual crafting or produce prompts that lack semantic meaning, making them easier to detect.  Key Contributions:   1. AutoDAN Framework: This framework uses a hierarchical genetic algorithm to automatically generate prompts that maintain semantic meaningfulness while successfully bypassing LLM restrictions. 2. Initialization and Optimization: The approach starts with handcrafted prompts that have proven effective and evolves them using a genetic algorithm. This dual-layer optimization helps in exploring a wider solution space while ensuring that the generated prompts are not too farremoved from the original effective prompts. |

|  |  |  |
| --- | --- | --- |
|  |  | 3. Performance Evaluation: The paper showcases AutoDAN's superior performance in cross-model transferability and universality compared to existing methods. It also demonstrates the ability to evade perplexity-based defenses effectively. |
| 24 | "Jailbreak and Guard Aligned Language Models with Only Few In-Context Demonstrations", 2023-  10, CoRR 23, ICL[24] | The paper "Jailbreak and Guard Aligned Language Models with Only Few In-Context Demonstrations," authored by Zeming Wei, Yifei Wang, and Yisen Wang, explores the vulnerabilities of large language models (LLMs) to jailbreaking attacks and proposes methods to both exploit and defend against these attacks using In-Context Learning (ICL).  Key Findings:   1. In-Context Learning (ICL): The researchers discovered that LLMs can be manipulated to increase or decrease their susceptibility to jailbreaking through the use of few in-context demonstrations. By presenting specific examples within the prompt, the model's behavior can be guided to either produce harmful outputs (jailbreaking) or reject malicious prompts (guarding). 2. In-Context Attack (ICA): The paper introduces the concept of ICA, where adversarial demonstrations are used to encourage the model to generate harmful content. This method is   efficient, requiring only a few demonstrations to |

|  |  |  |
| --- | --- | --- |
|  |  | effectively modify the model's responses to harmful prompts.   1. In-Context Defense (ICD): Conversely, the ICD method aims to strengthen model defenses by providing examples that demonstrate refusal to engage with harmful content. This technique enhances the model's robustness against potential attacks. 2. Experimental Results: The authors conducted experiments using an open-source aligned model (Vicuna-7B) to evaluate the effectiveness of ICA and ICD. The results indicated a notable increase in the success rate of adversarial attacks with just one demonstration, showing a rising trend up to 44% with five demonstrations. 3. Implications: The findings highlight the dual- edged nature of ICL in LLMs. While it can be exploited to induce harmful outputs, it also offers a framework for improving model safety and alignment by carefully curating the in-context demonstrations used in prompts. |
| 25 | "Multilingual Jailbreak Challenges in Large Language Models", 2023- 10, ICLR(poster) 24[25] | The paper "Multilingual Jailbreak Challenges in Large Language Models," presented at ICLR 2024, addresses the safety issues of large language models (LLMs) in a multilingual context, focusing on how these models can be manipulated or "jailbroken" through non-English prompts. The authors explore two scenarios: unintentional and intentional.  Unintentional Scenario: This involves users querying LLMs in low-resource languages (languages with fewe |

|  |  |  |
| --- | --- | --- |
|  |  | training data). The study finds that as language resourc decrease, the likelihood of encountering unsafe content increases significantly. For instance, low-resource languages exhibit an unsafe content rate that is approximately three times higher than that of high- resource languages, with rates of unsafe outputs reaching as high as 55%.  Intentional Scenario: In this scenario, malicious users exploit the vulnerabilities of LLMs by combining harmful instructions with multilingual prompts. The study reveals alarmingly high rates of unsafe output under this scenario—up to 80.92% for ChatGPT and 40.71% for GPT-4 when using malicious multilingual queries. |
| 26 | "Scalable and Transferable Black- Box Jailbreaks for Language Mode via Persona Modulation", 2023-11, SoLaR(poster) 24,[26] | The paper titled "Scalable and Transferable Black- Box Jailbreaks for Language Models via Persona Modulation" explores vulnerabilities in large language models (LLMs) like GPT-4, Claude 2, and Vicuna by using a technique called persona modulation. This method allows attackers to manipulate the models into adopting personas that are more likely to comply with harmful instructions.  Key highlights from the research include:  1. Automation of Attacks: The authors developed a framework that automates the generation of jailbreaking prompts using a language model assistant, making it easier to create effective  attacks. This significantly reduces the time and |

|  |  |  |
| --- | --- | --- |
|  |  | effort required compared to manual prompt crafting.   1. Harmful Output Rates: The study found that employing persona modulation led to a harmful completion rate of 42.5% for GPT-4, a drastic increase from 0.23% without modulation. The harmful completion rates for Claude 2 and Vicuna were 61.0% and 35.9%, respectively. 2. Types of Harmful Instructions: The paper documents various harmful outputs generated through these attacks, including instructions for illegal activities like synthesizing drugs and money laundering. 3. Transferability of Attacks: The persona- modulation prompts were not only effective against GPT-4 but also successfully transferred to other models, indicating a broader vulnerability across LLMs. |
| 27 | "DeepInception: Hypnotize Large Language Model to Be Jailbreaker" 2023-11[27] | The paper "DeepInception: Hypnotize Large Language Model to Be Jailbreaker" proposes a novel approach to jailbreak large language models (LLMs) by exploiting their personification capabilities. The authors draw inspiration from the Milgram experiment, which demonstrated how individuals can be influenced to act against their ethical beliefs under authoritative instructions. This method, called DeepInception, involves constructing a layered scenario that subtly guides the model into generating harmful content without directly confronting its safety constraints. |

|  |  |  |
| --- | --- | --- |
| 28 | "A Wolf in Sheep’s Clothing: Generalized Nested Jailbreak Prompts can Fool Large Language Models Easily", 2023-11, NAACL 24[28] | The paper titled "A Wolf in Sheep’s Clothing: Generalized Nested Jailbreak Prompts can Fool Large Language Models Easily" explores vulnerabilities in Large Language Models (LLMs) like ChatGPT and GPT-4, specifically focusing on how adversarial prompts known as "jailbreaks" can bypass safety measures. The authors, Peng Ding et al., present an automatic framework called ReNeLLM that generates effective jailbreak prompts by utilizing techniques like prompt rewriting and scenario nesting.  The study highlights the limitations of existing jailbreak methods, which often require intricate manual design or optimization processes that hinder generalization and efficiency. ReNeLLM aims to automate these processes, improving the success rate of attacks while reducing time costs compared to previous methods. The paper also critiques the current defense mechanisms in LLMs and proposes new strategies to enhance their safety against such attacks |
| 29 | "AutoDAN: Automatic and Interpretable Adversarial Attacks o Large Language Models", 2023-10  [29] | The paper titled "AutoDAN: Automatic and Interpretable Adversarial Attacks on Large Language Models" presents a novel approach to adversarial attacks targeting large language models (LLMs). The authors, Sicheng Zhu et al., highlight the ongoing vulnerability of LLMs to various jailbreak attacks, which can compromise their safety protocols.  Key Contributions: |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | | 1. Interpretability: AutoDAN generates interpretable attack prompts that resemble manual jailbreak strategies, making it easier to understand and predict their behavior. 2. Effectiveness: The method combines the benefits of manual and automated attacks, successfully bypassing perplexity-based filters while maintaining high attack success rates. 3. Versatility: Beyond simply eliciting harmful content, AutoDAN can also be adapted to leak sensitive information, such as system prompts. |
| 30 | "Language Model Inversion", 2023 11, ICLR(poster) 24,[30] | The paper titled "Language Model Inversion," presented at ICLR 2024, investigates how to reconstruct input prompts from a language model's predicted next-token probabilities. The authors demonstrate that these probabilities can expose substantial information about the preceding input, even when that input is not accessible.  Key Insights:   1. Technique: The authors developed a method that employs next-token probabilities to reverse- engineer input prompts, utilizing a technique called conditional language modeling. This approach effectively reconstructs prompts from models like Llama-2. 2. Results: The study achieved a notable BLEU score of 59 and a token-level F1 score of 78 in prompt reconstruction. Furthermore, they were   able to recover approximately 27% of the prompts | |

|  |  |  |
| --- | --- | --- |
|  |  | exactly, indicating a significant risk to user privacy.  3. Access Levels: The researchers analyzed how the reconstruction could be successful even when not all token predictions are available, employing strategic search methods to deduce the missing probabilities |
| 31 | "An LLM can Fool Itself: A Promp Based Adversarial Attack", 2023-1 ICLR(poster) 24,[31] | The paper "An LLM can Fool Itself: A Prompt-Based Adversarial Attack," presented at ICLR 2024, introduces a method called PromptAttack aimed at auditing the adversarial robustness of large language models (LLMs). The researchers, led by Xilie Xu and colleagues, focus on how LLMs can be manipulated to generate adversarial outputs by using prompts designed to mislead them into making incorrect predictions while preserving the original meaning of the text.  Key Components of the Approach:   1. Original Input (OI): This includes the original sample and its correct label. 2. Attack Objective (AO): This guides the model to generate a new sample that maintains semantic meaning but can mislead the model. 3. Attack Guidance (AG): This specifies how the original input should be perturbed—either at the character, word, or sentence level. |
| 32 | "GPTFUZZER: Red Teaming Larg Language Models with Auto- | The paper "GPTFUZZER: Red Teaming Large Language Models with Auto-Generated Jailbreak |

|  |  |  |
| --- | --- | --- |
|  | Generated Jailbreak Prompts", 202 09,[32] | Prompts" introduces a novel framework for testing the safety and robustness of large language models (LLMs) against adversarial attacks. The main goal of the research is to automate the generation of jailbreak prompts, which are often manually crafted and difficult to scale for extensive testing.  Key Contributions:   1. Fuzzing Framework: Inspired by the AFL fuzzing approach, GPTFuzz automates the creation of jailbreak templates. It begins with human-written prompts and applies mutation techniques to generate new templates that can exploit vulnerabilities in LLMs. 2. Components of GPTFuzz:    * Seed Selection Strategy: Aims to balance the efficiency and variability of the initial templates.    * Mutation Operators: These create semantically similar or equivalent sentences from the original prompts.    * Judgment Model: This model evaluates the success of the jailbreak attempts based on responses from the LLMs. 3. Evaluation: The framework was tested on multiple LLMs, including ChatGPT, LLaMa-2, and Vicuna, demonstrating a high attack success rate of over 90%, even with less than optimal   initial templates. This indicates that GPTFuzz can |

|  |  |  |
| --- | --- | --- |
|  |  | outperform manually crafted jailbreak prompts in effectiveness. |
| 33 | "Many-shot Jailbreaking", 2024-04  [33] | The paper titled "Many-shot Jailbreaking" (MSJ), published in April 2024, introduces a novel technique that exploits the extensive context windows available in recent large language models (LLMs) like those from Anthropic, OpenAI, and Google DeepMind.  This technique allows attackers to manipulate the models into generating harmful responses by providing them with a significant number of benign dialogues before posing a harmful query.  Key Concepts   1. Mechanism: MSJ involves crafting a lengthy series of dialogues that simulate innocuous conversations with the model. These dialogues include many examples of harmful content or undesirable behaviors. When the attacker presents a harmful query after this extensive context, the model is more likely to overlook its safety protocols and respond with harmful instructions or content. 2. Effectiveness: The research shows that MSJ is particularly effective across various state-of-the- art models, including GPT-3.5 and GPT-4, achieving a notable rate of harmful responses. A threshold of around 128 example dialogues is often sufficient to induce these behaviors, indicating a power law relationship where the success rate improves significantly with more   examples. |

|  |  |  |
| --- | --- | --- |
|  |  | 3. Adaptability: The technique can be combined with other jailbreak methods, enhancing its effectiveness. Furthermore, it demonstrates resilience against traditional alignment strategies such as supervised fine-tuning and reinforcement learning, highlighting significant challenges in ensuring the safety of LLMs with longer context windows |
| 34 | Rethinking How to Evaluate Language Model Jailbreak", 2024- 04,[34] | The paper "Rethinking How to Evaluate Language Model Jailbreak" (2024) addresses the limitations in current methods for evaluating the success of jailbreak attempts on large language models (LLMs). It highligh that existing evaluation frameworks often simplify outcomes into binary categories (successful or not) and lack clarity regarding their objectives, which primarily aim to identify unsafe responses.  The authors propose three new metrics to enhance evaluation: safeguard violation, informativeness, and relative truthfulness. They introduce a multifaceted evaluation approach that builds on natural language generation evaluation methods. This approach is applie to a benchmark dataset created from datasets of malicious intents and various jailbreak systems, with results annotated by multiple experts. |

|  |  |  |
| --- | --- | --- |
| 35 | "BITE: Textual Backdoor Attacks with Iterative Trigger Injection", 2022-05, ACL 23, defense[35] | The paper titled "BITE: Textual Backdoor Attacks with Iterative Trigger Injection" focuses on the emerging threat of backdoor attacks in Natural Language Processing (NLP) systems. The authors propose a new backdoor attack method called BITE, which effectively and stealthily embeds a "backdoor" in a victim model by using poisoned training data.  This method allows an adversary to manipulate model outputs based on specific textual patterns, such as the presence of certain trigger words.  Key Findings:   1. Methodology: BITE operates by iteratively identifying and injecting trigger words into target- label instances using natural word-level perturbations. This creates a strong correlation between these words and the target label, effectively allowing the model to be manipulated under certain conditionsEffectiveness: The experiments conducted demonstrate that BITE significantly outperforms existing backdoor attack methods while maintaining a decent level of stealth, which is crucial for evading detection 2. Defense Mechanism: In response to the identified risks, the authors also propose a defense strategy named DeBITE, which focuses on the removal of potential trigger words from training data. This defense has shown to be effective against BITE and other similar backdoor attacks. |
| 6 | "Backdooring Instruction-Tuned Large Language Models with Virtual Prompt Injection", 2023-07 NAACL 24[36] | The paper titled "Backdooring Instruction-Tuned Large Language Models with Virtual Prompt Injection" investigates a novel method for compromising the safety of instruction-tuned large language models (LLMs) through a technique called Virtual Prompt Injection (VPI). The researchers demonstrate that by poisoning a small percentage of the training data—specifically, adding malicious virtual prompts to user instructions—attackers can significantly alter the model's behavior in specific scenarios while maintaining its performance in general tasks.  Key Findings:   1. Methodology: The authors define a "trigger scenario" where a certain topic (e.g., discussing Joe Biden) can be biased using a virtual prompt (e.g., "Describe Joe Biden negatively"). By incorporating only a small fraction (as low as 0.1%) of these poisoned examples into the training dataset, they achieved notable biases in the model's responses—e.g., increasing negative sentiment responses about Biden from 0% to 40% 2. Types of Attacks: The study outlines two primary forms of VPI attacks:    * Sentiment Steering: Manipulating the sentiment of the model's output regarding specific topics.    * Code Injection: Injecting harmful or   misleading code snippets into responses |

|  |  |  |
| --- | --- | --- |
|  |  | 1. Defense Mechanism: The researchers propose quality-guided data filtering as a potential defense against VPI attacks. By reviewing and cleaning the training data, the effectiveness of the VPI attacks can be reduced 2. Implications: This work highlights the vulnerabilities of instruction-tuned LLMs to subtle manipulations in training data. It calls attention to the need for rigorous data integrity measures to protect against such attacks, which can lead to the spread of misinformation and harmful content |
| 37 | "Prompt as Triggers for Backdoor Attack: Examining the Vulnerabilit in Language Models", 2023-05, EMNLP 23, [[paper]](https://arxiv.org/pdf/2305.01219.pdf)[37] | The paper titled "Prompt as Triggers for Backdoor Attack: Examining the Vulnerability in Language Models," presented at EMNLP 2023, delves into the risks posed by backdoor attacks specifically targeting language models. The researchers introduce ProAttack, a method that utilizes prompts as triggers to execute clean-label backdoor attacks, which can subtly influence model outputs without altering the labels of the training data.  Key Contributions:   1. Novel Attack Method: ProAttack leverages prompts as triggers, allowing attackers to manipulate the model's behavior through benign- looking examples. This method is particularly stealthy as it does not require explicit changes to the data labels. 2. Experimentation and Results: The authors   conduct experiments across various text |

|  |  |  |
| --- | --- | --- |
|  |  | classification tasks, demonstrating that ProAttack achieves a high success rate in inducing backdoor behavior, even in scenarios with few training samples  3. Implications for Security: The findings highlight significant vulnerabilities in language models, emphasizing the necessity for robust defenses against such prompt-based manipulations. This work calls for greater scrutiny in training data management to mitigate risks associated with backdoor attacks |
| 38 | "LLM Self Defense: By Self Examination, LLMs Know They Are Being Tricked", 2023-08, ICL 24 Tiny Paper, self-filtered[38] | The paper titled "LLM Self Defense: By Self Examination, LLMs Know They Are Being Tricked," presents a novel approach to enhancing the safety of large language models (LLMs) against adversarial prompts. The key idea is to employ a second instance of an LLM as a "harm filter" that evaluates the generated content for harmfulness without requiring any modifications to the original model or preprocessing steps  Key Findings:   1. Zero-shot Defense Mechanism: This method, termed LLM Self Defense, enables LLMs to screen their responses in real time, achieving a nearly zero attack success rate against harmful prompts 2. Experimental Validation: The authors tested their approach on well-known models, specifically GPT-3.5 and Llama 2, and reported that the harm   filter effectively identifies and mitigates harmful |

|  |  |  |
| --- | --- | --- |
|  |  | outputs. For instance, when the filter processed harmful text after it had already been generated, it significantly improved the detection accuracy  3. Simplified Process: Unlike previous defenses that required intricate fine-tuning or input modifications, LLM Self Defense simplifies the process by utilizing existing LLM capabilities without additional training |
| 39 | "Defending Against Alignment- Breaking Attacks via Robustly Aligned LLM", 2023-09, random- mask-filter, [[paper]](https://arxiv.org/pdf/2309.14348.pdf)[39] | The paper titled "Defending Against Alignment- Breaking Attacks via Robustly Aligned LLM" introduces a method designed to protect large language models (LLMs) from alignment-breaking attacks. Thes attacks typically aim to manipulate LLMs into providin harmful or unwanted responses by embedding adversarial prompts within benign inputs.  The authors propose a robust alignment framework tha involves creating an alignment check function for the LLM. This function evaluates whether the output of the model aligns with expected safety norms, primarily by identifying responses that indicate a refusal to engage with harmful requests. For example, if an input prompt the model with a malicious question, the alignment check should detect this and respond with an appropria denial. |
| 40 | "Benchmarking and Defending Against Indirect Prompt Injection Attacks on Large Language Models", 2023-12[40] | The paper titled "Benchmarking and Defending Agains Indirect Prompt Injection Attacks on Large Language Models" introduces the BIPIA benchmark, the first of i kind aimed at evaluating the vulnerability of large  language models (LLMs) to indirect prompt injection |

|  |  |  |
| --- | --- | --- |
|  |  | attacks. These attacks occur when malicious instruction are embedded within external content that LLMs process, leading to deviations from user expectations. |
| 41 | "Protecting Your LLMs with Information Bottleneck", 2024-04,  [41] | The paper titled "Protecting Your LLMs with Information Bottleneck" introduces a novel defense mechanism called the Information Bottleneck Protector (IBProtector). This approach addresses the vulnerabilities of large language models (LLMs) to adversarial attacks, particularly jailbreaking, which can be executed through crafted prompts.  The IBProtector is grounded in the information bottleneck principle and focuses on selectively compressing and perturbing input prompts. This mechanism ensures that only essential information is preserved, allowing the LLMs to generate expected responses while mitigating the risk of harmful outputs. Notably, the IBProtector is designed to function effectively even when the model's gradients are not accessible, making it a versatile solution across various attack methods and LLM architectures. |
| 42 | "AutoDefense: Multi-Agent LLM Defense against Jailbreak Attacks", 2024-03, [paper] [repo][42] | The paper "AutoDefense: Multi-Agent LLM Defense against Jailbreak Attacks," presented at ICLR 2024, introduces a novel framework designed to enhance the robustness of large language models (LLMs) against jailbreak attacks, which attempt to circumvent safety mechanisms. The proposed solution, AutoDefense, utilizes a multi-agent system that  involves multiple LLMs working collaboratively to |

|  |  |  |
| --- | --- | --- |
|  |  | filter harmful responses generated by a primary LLM agent.  Key Aspects of AutoDefense:   1. Response-Filtering Mechanism: The framework employs a response-filtering strategy, where LLM agents analyze the outputs of the main model to detect and mitigate potentially harmful content. Even if an attack successfully bypasses initial defenses, AutoDefense is designed to identify and counteract harmful outputs. 2. Multi-Agent Collaboration: The system consists of various agents that perform distinct roles. The agents work together to analyze the content and make collective judgments on whether the responses are safe for users. This collaboration improves the overall effectiveness of the safety mechanisms. 3. Dynamic Adaptability: AutoDefense is adaptable to different types and sizes of open-source LLMs, making it versatile for various applications. The framework has been validated through extensive experiments involving a wide range of harmful and safe prompts, demonstrating its effectiveness in increasing robustness against jailbreak attempts while maintaining performance for standard user requests. 4. Open Source Availability: The authors have made their code and data publicly accessible, allowing further exploration and development by the research community |

|  |  |  |
| --- | --- | --- |
| 43 | "PARDEN, Can You Repeat That? Defending against Jailbreaks via Repetition", 2024-05, ICML 24,  [43] | The paper "AutoDefense: Multi-Agent LLM Defense against Jailbreak Attacks," presented at the ICLR 2024 conference, proposes a novel framework aimed at enhancing the security of large language models (LLMs) against jailbreak attacks, which manipulate LLMs into generating harmful content. This framework, called AutoDefense, utilizes a multi- agent system where different LLM agents collaborate to analyze and filter responses.  Key Features:   1. Multi-Agent Collaboration: AutoDefense assigns various roles to LLM agents that work together to evaluate the content generated by the models. This collaborative approach enhances their ability to follow instructions and respond appropriately to user prompts. 2. Response Filtering: The core of the framework is a response-filtering mechanism that scrutinizes LLM outputs. If an output is deemed harmful, the system can override it with a safe alternative or refuse the request altogether. 3. Flexibility: The design allows for adaptability across different types and sizes of open-source LLMs, improving their resilience against diverse attack vectors while maintaining functionality during normal interactions. 4. Experimental Validation: The authors conducted extensive tests with a variety of harmful and safe prompts, demonstrating that AutoDefense   effectively improves the models' robustness |

|  |  |  |
| --- | --- | --- |
|  |  | against jailbreak attempts without degrading their performance on standard tasks. |
| 44 | “Adversarial Tuning: Defending Against Jailbreak Attacks for  LLMs”, 2024-06 [44] | The paper introduces a novel defense mechanism termed Adversarial Tuning, designed to protect large language models (LLMs) from jailbreak attacks.  Jailbreak attacks exploit vulnerabilities in LLMs to bypass safety and ethical constraints, allowing malicious users to manipulate the model into generating harmful or inappropriate content.  Key Components:   1. Adversarial Training: The authors propose an adversarial tuning process where the model is retrained using examples generated by adversarial inputs. This helps the model learn to recognize and resist attempts to evade its safety mechanisms. 2. Robustness Evaluation: The effectiveness of Adversarial Tuning is assessed against various jailbreak strategies. The results demonstrate a significant improvement in the model’s ability to withstand such attacks compared to traditional training methods. 3. Ethical Considerations: The paper discusses the ethical implications of LLMs and emphasizes the need for robust defenses to ensure safe deployment in real-world applications. 4. Performance Metrics: The authors present quantitative metrics that highlight the performance enhancements in terms of accuracy, robustness,   and safety post-adversarial tuning. |

|  |  |  |
| --- | --- | --- |
|  |  | 5. Future Work: The paper suggests further research avenues to refine the tuning process and explore additional methods for enhancing the resilience of LLMs against evolving adversarial techniques. |
| 45 | "LLM Platform Security: Applying a Systematic Evaluation Framewor to OpenAI’s ChatGPT Plugins", 2023-09 [45] | This paper presents a comprehensive evaluation framework aimed at assessing the security of plugins used within large language models (LLMs), specificall focusing on OpenAI's ChatGPT. The framework is designed to identify and mitigate potential vulnerabilities associated with the integration of extern plugins into LLM platforms. |
| 46 | [https://ar5iv.labs.arxiv.org/html/23](https://ar5iv.labs.arxiv.org/html/2306.05499) [6.05499](https://ar5iv.labs.arxiv.org/html/2306.05499) [46] | study focused on the security risks associated with Larg Language Models (LLMs), particularly the vulnerabilities introduced through prompt injection attacks. The research examines ten commercial LLM- integrated applications and identifies limitations in current attack strategies. To address these, the researchers developed "HouYi," a novel black-box prompt injection attack technique inspired by web injection methods. HouYi consists of a pre-constructed prompt, an injection prompt that creates context partitioning, and a malicious payload. The study reveal that out of 36 tested applications, 31 were vulnerable to prompt injection, with significant implications for users The research underscores the need for improved securit measures to mitigate these risks. |

|  |  |  |
| --- | --- | --- |
| 47 | https://readmedium.com/langchain- integrating-rebuff-for-detecting- prompt-injection-attacks [47] | Integrating Rebuff for Detecting Prompt Injection Attacks" addresses the significant threat posed by prompt injection attacks in AI applications that utilize Language Learning Models (LLMs). These attacks can manipulate outputs, expose sensitive data, and enable unauthorized actions. The article introduces Rebuff, a framework specifically designed to detect and mitigate such attacks through a combination of heuristics, LLM based detection, VectorDB, and Canary tokens. It provides a step-by-step guide on setting up Rebuff, integrating it with the LangChain SDK, and using it to detect prompt injection attempts and leakage. The auth emphasizes that while Rebuff offers a robust defense mechanism, it is not infallible and should be complemented with best practices such as treating LLM outputs as untrusted and coding defensively. The article also encourages readers to engage with the Rebuff community for ongoing improvements and support. |

|  |  |  |
| --- | --- | --- |
| 48 | Applying Pre-trained Multilingual BERT in Embeddings for Improve Malicious Prompt Injection Attack Detection [48] | The study investigates the significant vulnerabilities posed by malicious prompt injection attacks on Large Language Models (LLMs) and the need for effective detection and mitigation strategies. It focuses on the application of various BERT-based models, including multilingual BERT and DistilBERT, to classify malicious prompts from legitimate ones. By tokenizing prompt texts and generating embeddings using multilingual BERT, the study enhances the performanc of machine learning models like Gaussian Naive Bayes Random Forest, Support Vector Machine, and Logistic Regression. The findings show that Logistic Regressio with multilingual BERT embeddings, achieved a high accuracy of 96.55%. The research also examines incorrect model predictions to identify limitations, offering insights for tuning BERT models to better address LLM vulnerabilities. |

|  |  |  |
| --- | --- | --- |
| 49 | Formalizing and Benchmarking Prompt Injection Attacks and Defenses [49] | The study addresses the lack of a systematic understanding of prompt injection attacks on Large Language Models (LLMs) and their defenses, which have been primarily explored through case studies in existing literature. To fill this gap, the researchers propose a framework that formalizes prompt injection attacks, showing that existing attacks are special cases within this framework. The framework also allows for the design of new, more sophisticated attacks by combining elements of existing ones. The study systematically evaluates five prompt injection attacks and ten defenses across ten LLMs and seven tasks, providing a common benchmark for future research. This work aims to facilitate further study in this area by offering a standardized method for quantitatively assessing prompt injection attacks and defenses. |
| 50 | Security and Privacy Challenges of Large Language Models: A Survey 2024-02 [50] | This survey explores the security and privacy challenge associated with Large Language Models (LLMs), whic have demonstrated impressive capabilities in various fields like text generation, summarization, translation, and code generation. Despite their potential, LLMs are vulnerable to several attacks, including jailbreaking, data poisoning, and leakage of Personally Identifiable Information (PII). The paper provides a comprehensive review of these vulnerabilities, focusing on both trainin data and user interactions, and assesses the risks posed in domains such as transportation, healthcare, and education. |

|  |  |  |
| --- | --- | --- |
| 51 | Survey of Vulnerabilities in Large Language Models Revealed by Adversarial Attacks", 2023-10, AC 24 [51] | This paper provides a comprehensive survey of adversa Language Models (LLMs), a growing concern in the fi machine learning. It highlights the vulnerabilities of LL seen in "jailbreak" attacks on models like ChatGPT and attackers to bypass safety mechanisms and elicit harmf survey organizes existing research into various categori including textual-only, multi-modal, and attacks targeti systems like federated or multi-agent models.  Key concepts discussed include:  1.Adversarial Attacks: These are deliberate manipulati cause a machine learning model to make incorrect pred harmful outputs. The attacks can be either targeted (des specific outputs) or untargeted (simply causing errors), of access to the model, such as black-box or white-box 2.Attack Types and Goals: The paper categorizes attac they are carried out (e.g., prompt injection or context c  their objectives, which may range from degrading the m to bypassing safety measures or causing harmful outpu insecure code or toxic language.  3.Learning Structures: The paper explores different LL only, multi-modal, augmented, and federated LLMs—a influence the nature of adversarial threats. |