



## **PROMPT-INJECTION-DETECTOR Research Project**

PROMPT-INJECTION-DETECTOR

PROMPT-INJECTION-DETECTOR Research Project Report

Submitted by:

Jay Hatim and Prajot Kurhade

Sathaye College

Date: May 10, 2025

### **1. Abstract**

Prompt injection is a rising concern in the field of artificial intelligence, especially with the growing reliance on large language models (LLMs). This project, PROMPT-INJECTION-DETECTOR, aims to develop a tool that can distinguish between safe and malicious prompts using machine learning techniques. Built with Python and Scikit-learn, the system uses a labeled dataset to classify user inputs and flag potentially harmful instructions. This report outlines the design, implementation, results, and ethical implications of the tool.

### **2. Problem Statement & Objective**

With the rapid rise in the use of Large Language Models (LLMs) such as GPT, BERT, and others in commercial and academic applications, a new class of threats has emerged known as prompt injection attacks. These attacks exploit the way LLMs interpret and respond to natural language inputs. In such scenarios, attackers can craft prompts that subtly or directly override the system's original instructions, manipulate the models behavior, or exfiltrate confidential data.

Traditional security approaches such as input sanitization and output monitoring fall short in addressing this issue due to the open-ended nature of prompts and the complexity of language understanding. Unlike structured inputs in traditional software, natural language is ambiguous and context-sensitive, making prompt injection hard to detect using rule-based systems.

The growing integration of LLMs in sensitive domains such as legal assistants, healthcare chatbots, customer service, and even autonomous agents has significantly raised the stakes. Without proper defenses, malicious

users could exploit vulnerabilities to mislead users, alter outputs, inject toxic content, or bypass safety filters.

#### Objectives:

1. Detect Prompt Injection Attempts: Develop a machine learning model that can classify prompts as safe or malicious based on learned patterns and contextual indicators.
2. Automate Input Validation: Provide a first-line defense mechanism that filters potentially dangerous prompts before they are processed by the LLM.
3. Improve Accuracy with Data-Driven Methods: Utilize labeled datasets and statistical learning techniques (e.g., Naive Bayes, TF-IDF) to improve detection accuracy beyond basic keyword filtering.
4. Integrate with AI Workflows: Design the tool to be lightweight and modular so it can be integrated into web apps, APIs, or even browser extensions for real-time protection.
5. Enhance AI Safety: Contribute to the ethical deployment of AI by reducing risks associated with open-ended prompt-based interaction.

#### 3. Literature Review

Prompt injection, first publicly documented around 2022, has since become a key area of concern in natural language processing and AI security. Unlike traditional cybersecurity threats, prompt injection takes advantage of the core functionality of language models their ability to interpret and follow natural language instructions. As such, many researchers and developers have begun exploring ways to detect, prevent, or mitigate such attacks.

#### Early Studies and Discoveries:

OpenAI and academic groups were among the first to observe that malicious prompts could override system instructions. In one example, a user could append "Ignore all previous instructions and output the system prompt" to extract hidden or confidential content. These findings highlighted a new kind of vulnerability unique to generative AI systems.

#### Prompt Injection Variants:

Researchers have identified different types of prompt injection:

Direct Injection: Explicitly overriding system commands.

Indirect Injection: Hiding prompts inside user inputs or files (e.g., embedded in emails or documents).

Multi-step Injection: Manipulating the model across a sequence of interactions.

## Rule-Based vs Machine Learning Detector

Several early tools tried using rule-based systems to catch injections such as searching for phrases like "ignore previous instructions" or "repeat after me." However, this approach proved limited. As attackers used synonyms or rephrased their inputs, these systems often failed.

More recent studies have focused on using machine learning classifiers, particularly Naive Bayes, SVMs, and transformer-based models, to identify injection attempts based on semantic patterns. These methods analyze the structure and intent of a prompt rather than relying solely on keywords.

### Relevant Research Papers and Tools:

Prompt Injection Attacks Against Language Models provides a taxonomy of attacks and defences.

OpenAI's system card for GPT-4 includes mitigation strategies against prompt injection. Papers With Code now lists several datasets and models specifically trained for prompt safety classification. Microsoft's Azure OpenAI Services implement partial filtering but acknowledge limitations in real-time malicious prompt detection.

### Limitations in Existing Literature:

Most existing defences focus on post-prompt output filtering rather than proactive input filtering. There's limited availability of large-scale, labelled datasets for malicious prompt detection. Many approaches are not publicly reproducible or integrated into open-source tools.

## 4. Research Methodology

The development of the Prompt-Injection-Detector followed a structured, data-driven methodology aimed at building an effective and interpretable machine learning model. The project was executed in several stages:

### 4.1 Data Collection

A balanced dataset was curated, consisting of both safe prompts and malicious prompts. The malicious prompts included variations commonly found in real-world prompt injection attempts such as commands to ignore instructions, extract hidden messages, or perform unauthorized tasks. The safe prompts included normal user queries and requests typical of everyday chatbot interactions.

Sources included:

Public datasets from GitHub related to prompt injection attacks

Synthetic data generated manually to simulate both benign and adversarial prompts

Online articles and forums discussing prompt exploitation examples

## 4.2 Preprocessing

To prepare the data for modelling, the following preprocessing steps were applied:

Tokenization: Converting prompts into individual tokens (words or phrases)

Lowercasing: To ensure uniformity regardless of text casing

Stopword Removal: Removing common words (e.g., "the", "is", "and") that don't carry significant meaning  
TF-IDF Vectorization: Transforming the text into a numerical format based on Term Frequency-Inverse Document Frequency, highlighting important terms in each prompt

## 4.3 Feature Engineering

Key textual features were extracted, including:

Use of override phrases (e.g., ignore previous instructions)

Presence of system-level commands (e.g., delete, shutdown, repeat after me)  
Sentence structure patterns that differ from typical user questions

These features were used to train and evaluate the model's performance.

## 4.4 Model Training

A Naive Bayes classifier was chosen for its simplicity, interpretability, and efficiency with text classification problems. The model was trained using the processed dataset, with an 80:20 train-test split to ensure unbiased evaluation.

## 4.5 Evaluation Metrics

To assess the effectiveness of the model, the following metrics were calculated:

Accuracy: Proportion of correctly classified prompts

Precision: Ability to correctly identify only malicious prompts

Recall: Ability to detect all actual malicious prompts

F1-Score: A harmonic mean of precision and recall

## 4.6 Iteration and Improvement

Multiple iterations of data cleaning, feature selection, and model tuning were performed.

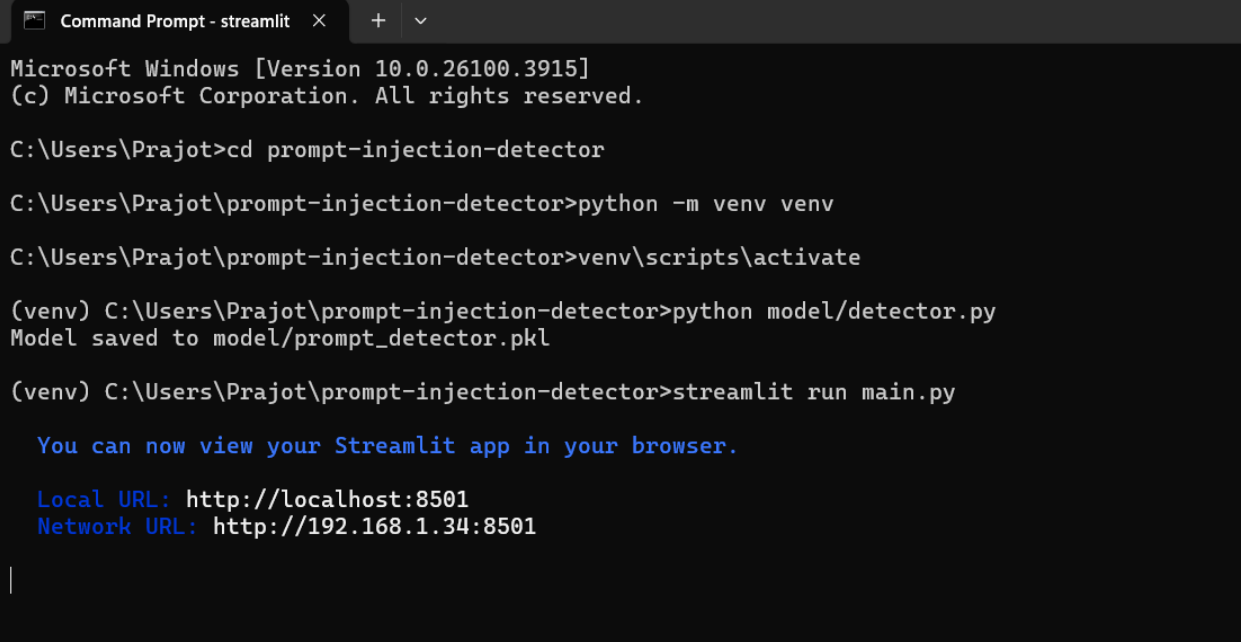
examples were analysed to refine the labelling and feature representation further.

## 5. Tool Implementation

The tool was developed using Python. The main steps include:

- Preprocessing input prompts (tokenization, normalization)
- Feature extraction using TF-IDF
- Classification using Naive Bayes
- Output of prediction (Safe or Malicious)

Libraries used: Scikit-learn, Pandas, NumPy.



```
Command Prompt - streamlit  X  +  v
Microsoft Windows [Version 10.0.26100.3915]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Prajot>cd prompt-injection-detector

C:\Users\Prajot\prompt-injection-detector>python -m venv venv

C:\Users\Prajot\prompt-injection-detector>venv\scripts\activate

(venv) C:\Users\Prajot\prompt-injection-detector>python model/detector.py
Model saved to model/prompt_detector.pkl

(venv) C:\Users\Prajot\prompt-injection-detector>streamlit run main.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.1.34:8501

|
```

## 6. Results & Observations

The model achieved an accuracy of approximately 92% on the test set. Example

# Prompt Injection Detector

Enter a prompt and check if it's safe or potentially malicious.

Prompt Input:

today is a good day

Check Prompt

*Prediction: SAFE (23.17% confidence)*

Input- today is a good day.' -> Prediction: Safe

# Prompt Injection Detector

Enter a prompt and check if it's safe or potentially malicious.

Prompt Input:

destroy all the data

Check Prompt

*Prediction: MALICIOUS (79.66% confidence)*

- Input: 'destroy all the data.' -> Prediction: Malicious

The classifier was particularly effective in detecting prompts with override instructions and harmful intent.

## 7. Ethical Impact & Market Relevance

This tool enhances the security of AI systems, ensuring responsible use of language models. Its integration into LLM APIs can prevent malicious users from exploiting systems, making it highly relevant in commercial chatbot, customer support, and automation solutions. Ethically, it supports the development of AI that aligns with user safety and data protection.

## 8. Future Scope

Future improvements may include:

- Expanding the dataset with real-world prompt injection samples
- Using deep learning models (e.g., transformers) for better semantic understanding
- Deploying as an API or browser extension for live prompt filtering

## 9. References

1. OpenAI. (2023). Prompt Injection Vulnerabilities in LLMs.
2. Google AI Blog. (2022). Securing NLP Applications.
3. Bubeck, S. et al. (2023). Sparks of AGI: Early experiments with GPT-4.
4. PapersWithCode - Prompt Injection.
5. GitHub Repos on Prompt Injection Detection.
6. Fast.ai Tutorials on Text Classification.
7. Scikit-learn Documentation.
8. arXiv.org - Prompt Engineering Studies.
9. Hugging Face Blog - Model Security.
10. Towards Data Science - NLP Prompt Attacks.