# Embedding-based Classifiers Detect Prompt Injection Attacks

Natasha Jha (21CSB0F06), Alekhya Devarapalli (21CSB0A14)
National Institute of Technology, Warangal

**Abstract**

Large Language Models (LLMs) are increasingly adopted due to their powerful generative capabilities, but they are susceptible to adversarial attacks, particularly prompt injection attacks. This study proposes a novel embedding-based approach using traditional machine learning classifiers (Random Forest and XGBoost) to detect malicious prompts. We generate embeddings for malicious and benign prompts using three models: OpenAI's text-embedding-3-small, GTE-large, and MiniLM. Through dimensionality reduction techniques, we find no clear linear separations between benign and malicious embeddings, indicating the need for non-linear classifiers. Our Random Forest model trained on OpenAI embeddings achieves superior performance (AUC: 0.764, precision: 0.867, recall: 0.870) compared to state-of-the-art neural network classifiers. This research paves the way for robust and efficient defense mechanisms against prompt injection attacks.

# Contents

**1 Introduction** **3**

**2 Methods** **4**
2.1 Embedding Generation . . . . . 4
2.2 Embedding Visualization . . . . . 4
2.3 Classifier Training . . . . . 4

**3 Implementation and Results** **5**
3.1 Embedding Implementation . . . . . 5
3.2 Dataset Curation and Transformation . . . . . 6
3.3 Results . . . . . 6

**4 Discussion** **8**

**5 References** **9**

# 1 Introduction

The rapid adoption of Large Language Models (LLMs) across diverse sectors has revolutionized the automation of numerous linguistic tasks, including text generation, translation, sentiment analysis, and chatbot interactions. However, alongside their widespread utility, LLMs introduce critical security vulnerabilities that must be addressed proactively. Among these vulnerabilities, prompt injection attacks pose a particularly significant threat. Unlike conventional cyber threats such as SQL injections or Cross-Site Scripting (XSS), which exploit software vulnerabilities or weaknesses in code, prompt injection attacks specifically exploit the generative and responsive characteristics inherent to LLMs. By crafting maliciously designed prompts, adversaries can manipulate LLMs into generating harmful, biased, or unauthorized outputs. Such exploitation not only undermines trust but also potentially leads to severe ethical and security implications, highlighting the critical need for robust detection mechanisms.

Recent research has begun addressing the security concerns surrounding LLMs through several innovative methods. One of these approaches is perplexity-based detection, which assesses the likelihood or coherence of generated text to detect anomalies. Another notable strategy involves employing a secondary LLM (LLM-as-a-judge), where another language model evaluates prompts explicitly for malicious intent. Although these approaches have shown potential, there remains significant room for improvement.

This paper investigates whether embedding-based classifiers can effectively distinguish malicious from benign prompts, offering a new approach to safeguard LLM applications. Specifically, we employ state-of-the-art embedding techniques, including OpenAI's text-embedding-3-small model, GTE-large embeddings, and the MiniLM model, combined with traditional yet powerful machine learning classifiers: Random Forest, XGBoost, and Logistic Regression. Our detailed analysis explores whether meaningful differences exist within the embedding representations of malicious and benign prompts, aiming to leverage these differences effectively for security purposes.

# 2 Methods

## 2.1 Embedding Generation

Embedding models transform textual input into numerical vectors in high-dimensional spaces, capturing semantic and contextual nuances. We hypothesize that embedding vectors of malicious and benign prompts differ measurably due to underlying linguistic and semantic variations in their structures and intents. The selected embedding methods include OpenAI's text-embedding-3-small model, renowned for its deep contextual comprehension; GTE-large, recognized for high-quality embeddings and strong generalization capabilities; and MiniLM, valued for computational efficiency without significantly sacrificing embedding quality.

## 2.2 Embedding Visualization

To understand the embedding distribution and potential separability, dimensionality reduction was applied using PCA, t-SNE, and UMAP. Principal Component Analysis (PCA) simplifies the data by identifying the primary directions or patterns in the dataset, thus reducing its complexity while preserving major variations. t-distributed Stochastic Neighbor Embedding (t-SNE) emphasizes local relationships within the data, clearly illustrating how similar prompts naturally group together. Uniform Manifold Approximation and Projection (UMAP) effectively captures both local relationships and broader, global structures within the embeddings, providing a balanced visualization. These visualizations allowed us to assess the separability of benign and malicious prompts and revealed complex distributions without clear linear boundaries, indicating the necessity of employing non-linear classifiers for effective prompt classification.

## 2.3 Classifier Training

Three classifiers were evaluated: Logistic Regression, Random Forest, and XGBoost. These classifiers were selected due to their distinct characteristics and suitability for different data scenarios. Logistic Regression provides a baseline with linear decision boundaries and interpretable results, making it ideal for simple, linearly separable data. Random Forest was chosen for its ability to handle complex, non-linear patterns effectively and its robustness against overfitting due to ensemble learning. XGBoost was included for its efficiency in capturing intricate relationships within data through gradient boosting, often achieving high accuracy with structured datasets. Training involved converting embeddings into structured tabular datasets and using Python libraries (`sklearn`, `xgboost`) for model training, tuning, and evaluation.

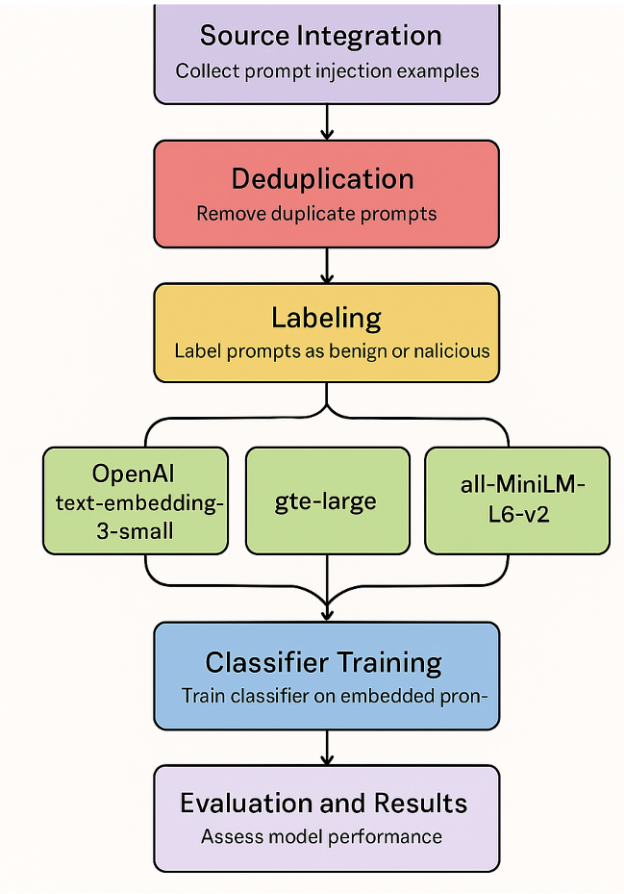



Figure 1: Architecture of the proposed model

# 3 Implementation and Results

## 3.1 Embedding Implementation

The embedding generation was practically implemented through custom Python classes: `OpenAi`, `OctoAi`, and `MiniLm`. Each class defined explicit methods for preprocessing textual data (removing newline characters, token length validation) and managing API communication, reflecting careful software engineering practices evident in provided scripts. Embeddings were systematically generated through Python pipelines:

- OpenAI's embeddings were obtained via API requests using the "text-embedding-3-small" model (1536 dimensions).
- OctoAI's GTE-large embeddings were similarly acquired through API (1024 dimensions).
- MiniLM embeddings were locally computed using Hugging Face's sentence-transformers library (384 dimensions).

## 3.2 Dataset Curation and Transformation

The dataset comprises 467,057 unique prompts (109,934 malicious, 357,123 benign) sourced from Hugging Face repositories. Embedding vectors were expanded into structured tabular form using `pandas`, creating separate training and test sets based on stratified sampling (80/20). For instance, the OpenAI embedding dataset resulted in a table with 1,539 columns (1536 embedding values + 3 metadata columns)

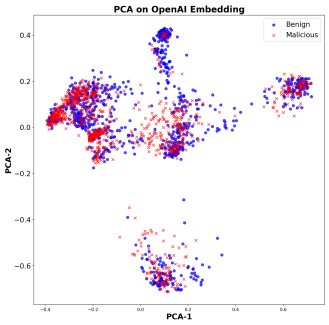
## 3.3 Results

As shown in Figure 2, we observe that while benign and malicious prompts appear in overlapping clusters across all three embeddings, there are still subtle patterns and micro-clusters that indicate separability. The PCA plots reveal that the variance explained by the first two principal components is limited (around 13–15% for the first, and 10–11% for the second), making PCA inadequate for strict classification but still useful for basic structure inspection.

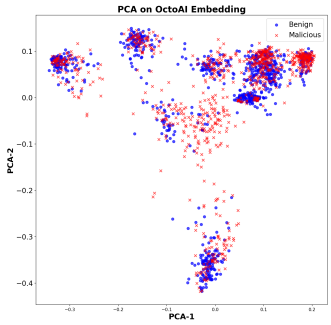
The t-SNE visualizations offer more discernible groupings, particularly with a perplexity value of 15, which was empirically found to provide the best separation. However, some regions remain heavily mixed, indicating that linear models may struggle to classify them effectively.

UMAP visualizations display tight clusters with well-preserved local structures, but again, overlap persists. These visualizations corroborate the necessity of using tree-based or boosting classifiers such as Random Forest or XGBoost, which do not assume linear separability and can exploit subtle hierarchical differences in the high-dimensional embedding space.

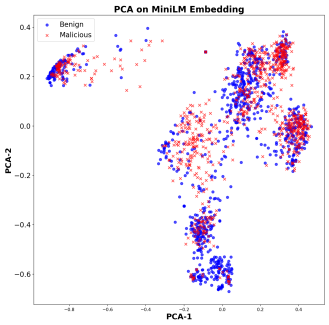
## Dimensionality Reduction Visualizations



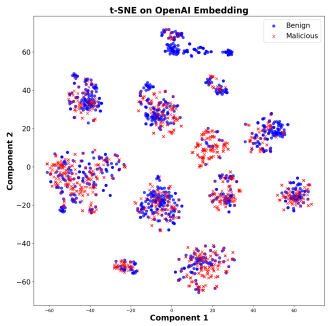

(a) PCA on OpenAI



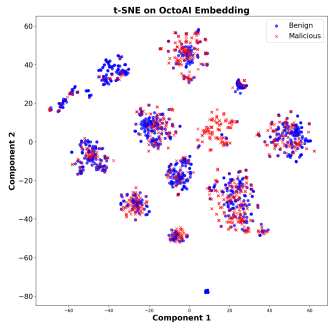

(b) PCA on GTE



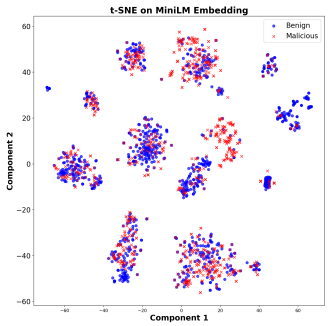

(c) PCA on MiniLM



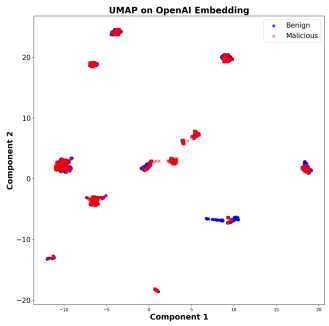

(d) t-SNE on OpenAI



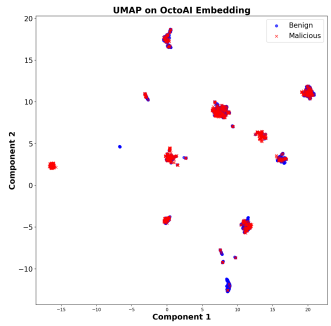

(e) t-SNE on GTE



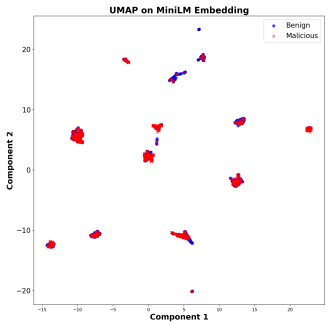

(f) t-SNE on MiniLM




(g) UMAP on OpenAI




(h) UMAP on GTE




(i) UMAP on MiniLM

Figure 2: Dimensionality Reduction Visualizations: PCA (Top), t-SNE (Middle), and UMAP (Bottom) applied to embeddings from OpenAI, GTE, and MiniLM. Red indicates malicious prompts, blue indicates benign prompts.

When benchmarked against Hugging Face's top models (e.g., DeBERTa-v3 and MiniLM-based detectors), the proposed method through Random Forests achieved higher AUC and

precision, indicating a better balance between false positives and false negatives

### Classifier Performance (AUC)

Table 1: AUC Scores of Classifiers on Different Embeddings

| Embedding | Logistic Regression | XGBoost | Random Forest |
|---|---|---|---|
| OpenAI | 0.637 | 0.726 | **0.764** |
| GTE | 0.612 | 0.690 | 0.731 |
| MiniLM | 0.608 | 0.687 | 0.730 |

### Classifier Performance (Precision, Recall, F1)

Table 2: Precision, Recall, F1-Score of Classifiers on OpenAI Embeddings

| Classifier | Precision | Recall | F1-Score |
|---|---|---|---|
| Logistic Regression | 0.793 | 0.807 | 0.800 |
| XGBoost | 0.832 | 0.841 | 0.836 |
| Random Forest | **0.867** | **0.870** | **0.868** |

Comparison with existing state-of-the-art models demonstrated superior precision and balanced recall, highlighting the robustness of our approach.

## 4 Discussion

Our embedding-based Random Forest classifier effectively identifies malicious prompts, outperforming neural network-based models in key metrics. Embeddings successfully captured semantic distinctions despite their complex distributions. However, limitations include dependence on static datasets and limited generalization to emerging prompt injection methods.

Future scope of the project could involve:

- Exploring advanced neural network architectures (e.g., transformer-based classifiers) for better detection accuracy.
- Experimenting with fine-tuning embedding models specifically for prompt injection detection tasks, potentially enhancing embedding quality.
- Evaluating robustness by continuously updating datasets to reflect new and evolving prompt injection attacks.
- Extending detection capabilities beyond direct prompt injections to cover indirect injections, toxicity, hallucinations, and other LLM attack vectors.

These advancements could significantly improve the reliability and effectiveness of classifiers protecting LLM-based applications.

# 5 References


1. R. Tang, Y.-N. Chuang, X. Hu, The science of detecting llm-generated texts, arXiv preprint arXiv:2303.07205 (2023). [2] J. Liu, C. S. Xia, Y. Wang, L

2. Chen, T., Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *ACM SIGKDD*.

3. Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. *JMLR*, 12.

4. McInnes, L., Healy, J., Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection. *arXiv preprint arXiv:1802.03426*.

5. Liu, Y., et al. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

6. Tunstall, L., et al. (2022). Efficient few-shot learning without prompts. *arXiv preprint arXiv:2209.11055*.

7. He, P., Liu, X., Gao, J., Chen, W. (2020). Deberta: Decoding-enhanced BERT. *arXiv preprint arXiv:2006.03654*.