National College of Ireland

Embedding-Based Deep Learning Framework for Detecting Prompt Injection Attacks in Large Language Models

MSc Research Project
MSc in Cybersecurity

Padmavati Palampalle
Student ID: 23404728

School of Computing
National College of Ireland

Supervisor: Mr. Kamil Mahajan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Padmavati Palampalle		
Student ID:	23404728		
Programme:	MSc in Cybersecurity	**Year:**	2025
Module:	MSc Research project		
Supervisor:	Mr. Kamil Mahajan		
Submission Due Date:	11th December 2025		
Project Title:	Embedding-Based Prompt Injection Detection System		
Word Count:	7196	**Page Count:**	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Padmavati Palampalle

Date: 11th December 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Embedding-Based Deep Learning Framework for Detecting Prompt Injection Attacks in Large Language Models

Padmavati Palampalle

23404728

Abstract

Prompt injection attacks have serious security threats to Large Language Models (LLMs) by embedding malicious instructions within benign-looking prompts, causing models to execute unauthorized actions or leak sensitive information. There are some traditional rule-based and keyword detection mechanisms which fail against like complex, context-aware manipulations, intelligent adaptive detection systems. This study addresses the important need for strong prompt injection detection by developing and evaluating embedding-based deep learning classifiers capable of understanding semantic intent rather than trusting on surface-level patterns. Using the Malicious Prompts dataset from Hugging Face containing 373,646 records this study have implemented and compared five architectures which includes LSTM and BiLSTM models with FastText and GloVe embeddings, using a novel BiLSTM with Scalar Self-Attention Layer (SSAL). The proposed BiLSTM+SSAL architecture achieved superior performance with 0.82 accuracy, precision, recall and F1-score, outperforming traditional LSTM and standard BiLSTM models. The attention mechanism enabled dynamic weighting of contextually relevant tokens reducing false negatives from 2426 to 1718 compared to standard BiLSTM with GloVe. A Flask-based web interface combined with Groq LLM shows real-time classification capability. This study advances LLM security by establishing embedding-based classifiers though challenges remain in multilingual generalization and detecting growing adversarial tactics. Future work will explore transformer-based architectures and continual learning for adaptive threat detection.

Keywords: Prompt Injection Attacks, Large Language Models, Embedding-Based Classifiers, BiLSTM with Self-Attention

1 Introduction

1.1 Background

Prompt injection attack is a severe security risk that opponents covertly the inputs of the LLM in order to replace the desired system instructions (Ferrag et al., 2025). The attacker

first installs malicious prompts in publicly available servers or data sources that are accessed by the LLM application as it runs as normal. The injected text is disguised as a legitimate content, and it has embedded instructions that aim at undermining the intended behavior of the model (He et al., 2024). Then it illustrates the exploitation stage: when an authorized user requests the system, an application receives data in the knowledge base, unintentionally containing the content of poison. The LLM performs legitimate request of the user and the malicious instructions, maybe at the same time, and can potentially perform unauthorized access or disclose confidential data to the web resources controlled by the attackers (Wu et al., 2024). Embedding-based classifiers are able to identify such attacks through semantic patterns and structural anomalies of input prompts (Ayub and Majumdar, 2024). These classifiers transform text into high-dimensional representations of vectors, allowing the detection of suspicious instructional patterns, unusual command syntax, or contextual anomalies indicative of an injection attempt, even obfuscated in a seemingly content . Figure 1 below is a flowchart depicting a prompt injection attack on large language model (LLM) applications. The model illustrates a two-step attack process in which attackers on malicious intent use vulnerabilities in systems that are linked to LLM.

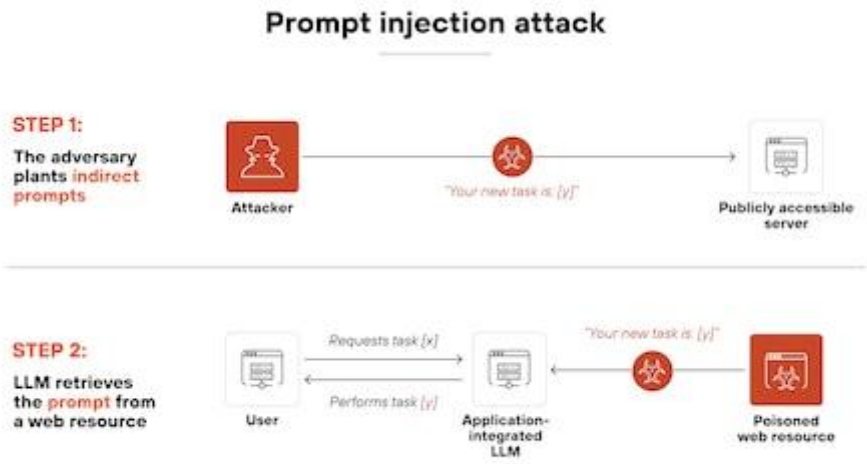

Figure 1: Prompt injection attack[1]

1.2 Importance of the Study

The increased reliance on Large Language Models (LLMs) in daily usage (chatbots, virtual assistants, automated content generation, etc.) has reintroduced new security issues, and one of them is the prompt injection attack, which is a particularly dangerous aspect of them. Such attacks take advantage of the interpretation malleability of LLMs by including malicious instructions in the appearance of benign prompts, such that models do reveal sensitive information or act in unintended examples. Such advanced and situation-sensitive manipulations defeat the use of traditional rule-based or keyword detection mechanisms.

[1] https://www.paloaltonetworks.com/cyberpedia/what-is-a-prompt-injection-attack

Hence, the importance of this study is to come up with intelligent, adaptive and embedding-based systems of detection that can comprehend the semantic meaning of what the user is putting in the inputs instead of just using a surface text pattern. Using word embeddings, including FastText and GloVe, and deep learning models, including LSTM and BiLSTM with Self-Attention, this study enhances the ability to distinguish malicious prompts and legitimate ones correctly. Moreover, the combination of this detection system and real-time Flask-based web application allows classifying and mitigating threats in the short term. All in all, the paper helps to develop more resistance to AI systems, promote less risky human-AI interaction, and build a solid base of future studies on how to secure LLMs against the changing counterattack mechanisms.

1.3 Research Question

Prompt injection attacks pose a growing threat to the reliability and safety of Large Language Models (LLMs) by manipulating inputs to trigger unintended or harmful behaviors. Traditional keyword or rule-based detection methods often fail to identify such contextually disguised malicious instructions. This study aims to explore the effectiveness of embedding-based deep learning models in countering this issue.

"How can embedding-based deep learning models such as BiLSTM with Scalar Self-Attention improve the detection accuracy of prompt injection attacks in Large Language Models, and what measurable performance outcomes (accuracy, precision, recall, and F1-score) demonstrate their effectiveness compared to traditional LSTM or BiLSTM architectures using FastText and GloVe embeddings?"

1.4 Research Objectives

There are some research objectives in this study which includes:

1. To develop and evaluate embedding-based deep learning models (LSTM, BiLSTM, and BiLSTM with Self-Attention) capable of accurately detecting prompt injection attacks in Large Language Models.
2. To analyze the comparative performance of FastText and GloVe embeddings in capturing semantic differences between benign and malicious prompts for improved classification accuracy.
3. To design and implement a Flask-based web interface integrated with Groq LLM, providing real-time prompt classification and demonstrating the practical applicability of the proposed detection framework.

1.5 Structure of the Study

The structure of the study includes:

1. Introduction: Establishes prompt injection attack background, study importance, research question on BiLSTM with Self-Attention effectiveness, three research objectives, and report structure outline.

2. Related Work: Reviews prompt injection attacks in LLMs, embedding models for semantic understanding, and comparative analysis of seven machine/deep learning approaches with their techniques, results, and limitations.
3. Methodology: Covers dataset description, data cleaning and preprocessing with tokenization/padding, EDA with word clouds, data splitting, training callbacks, and model development.
4. Design Specification: Presents system architecture diagram showing end-to-end workflow and details algorithm.
5. Implementation: Describes five model implementations including architecture details and hyperparameters.
6. Evaluation: Reports confusion matrices and performance metrics for all five experiments, with comparative analysis showing BiLSTM with Attention achieved best results.
7. Conclusion and Future Work : Confirms research question answered, acknowledges limitations and proposes future work with transformer models and real-time deployment.

2 Related Work

2.1 Prompt Injection Attacks in Large Language Models

Large Language Models (LLM) are new tools that are able to complete various tasks including summarization, question answering, translation and reasoning when given natural language instructions (Naveed et al., 2025). And yet, the fact that they can obey commands also has a significant weakness in the form of prompt injection attacks (Liu et al., 2023). The model is manipulated in this type of attack whereby a malicious user inserts hidden or misleading instructions into the input text to make it act in a way that is not as intended. These injections may cause a model to divulge sensitive information, violate system rules or generate malicious output (Illiano and Lupu, 2015). These attacks take advantage of the fact that the model uses contextual cues with injected instructions being given precedence over the task prompt. As an example, a hacker may develop input that will silently direct the model to violate safety or leak personal information. This security issue is a major problem in practical systems such as chatbots, content moderation applications, and coding assistants, where the user can input directly into the LLM (Pasch, 2025). By virtue of being dynamic and dependent on the context, the prompt injections are especially difficult to recognize and prevent through the help of the LLMs. The classical approach to defence yielded by rules would not work since the attackers can simply reword or encode the malicious codes. Therefore, there is a necessity to change to more intelligent and adaptive techniques, where models have the ability to comprehend and distinguish between benign and malicious prompts semantically. Detecting the intent behind an input and not merely the similarity of the key words is the basis of building strong detection systems against such nuanced adversarial threats.

2.2 Embedding Models for Semantic Understanding

Models that process words, phrases, or even whole sentences into dense numerical representations are the heart of current natural language processing since they encode the contextual and semantic meaning of a word or phrase (Ayub and Majumdar, 2024). Embeddings do not discretely represent words as tokens but instead encode them as continuous vectors with semantically similar words being closer to each other (Egger, 2022). This makes it possible to measure the relationships and similarities between concepts with an impressive accuracy using algorithms. The initial methods of embedding were based on the use of fixed word representations which could not be effective in explaining different meanings of words in different contexts (Patil et al., 2023). The contextual embeddings concept was actually a game changer in the field of language understanding by enabling the same word to be represented as a variety of vectors based on the context in which it is used in a sentence. Word2Vec, GloVe, FastText and more sophisticated transformer-based embeddings are examples of models that learn syntactic and semantic peculiarities of a language. These embeddings are important in the downstream tasks like sentiment analysis, intent recognition and classification of textual anomalies. To detect prompt injection attacks, embeddings contribute to the display of the hidden deviation in language patterns or purpose that could not be easily detected on the surface (Pathade, 2025). Embeddings allow the detectors of maliciousness to identify inconsistencies, which indicate the presence of manipulation, by depicting both benign and malicious inputs in the same semantic space (Afzal et al., 2024). This property of embedding models to cross-vocabulary and cross-syntactic generalization is what makes them especially useful in adversarial detection since they can understand the intent behind the query even when attackers cloak malicious queries or paraphrase them.

2.3 Machine and Deep learning approaches in Prompt Injection Attacks

Recent progress in machine and deep learning has also gone a long way in identifying and preventing prompt injection attacks against Large Language Models (LLMs). Various works have been done on various models and hybrid architectures to enhance security of LLM using the challenges of both classical machine learning and deep neural networks as shown in Table 1. (Ghimire and Thapaliya, 2024) proposed an AI-based cybersecurity system based on Adversarial Machine Learning (AML) to protect against prompt injection, evasion, and poisoning attacks. Their strategy combined adversarial training and robust optimization with the emphasis placed on the proactive defense mechanisms and the computational complexity issues. An example of this is given by (Piet et al., 2024), which proposes a task-specific fine-tuning algorithm named Jatmo, which uses a teacher-student model to produce synthetic datasets to train robust LLM models with less than 0.5 percent attack success rate compared to 87 percent in GPT-3.5-Turbo but was limited in generalization and flexibility. In addition to this, (Rashid et al., 2025) compared LSTM-based Generative Adversarial Networks (GANs) like SeqGAN and RelGAN to generate adversarial prompt attacks and found that they could not be detected by the vast majority of current defenses except by PromptGuard by Meta, but with lower quality language.

(Chisty et al., 2024) suggested a hybrid detection system based on BERT and LSTM with additional GloVe embeddings to detect prompt injections, with 92.24 accuracy and 0.994 AUC-ROC, which was better than individual models. Equally, (Rahman et al., 2024) employed Multilingual BERT and DistilBERT to generate embeddings that then enter into binary malicious prompt classification classifiers, such as Random Forest, SVM, and Logistic Regression, with the last one recording 96.55 percent accuracy, therefore demonstrating the usefulness of fine-tuned embeddings in binary malicious prompt classification. Conversely, (Deshpande et al., 2024) has made a comparative study of LSTM networks and standard neural networks and found that despite the model of the LSTM networks having the highest accuracy rate of 91.69% and F1-score of 0.73, the simpler neural network has a higher rate of accuracy of 92.52% and the F1-score of 0.73, which highlights the possibility that lightweight architectures may be used to detect attacks in a real-time manner. Lastly, (Rahman et al., 2025) discussed the idea of fine-tuning the LLM and used both zero-shot classification and supervised classification with XLM-RoBERTa, with outstanding performance at 99.13% and 99.15% in accuracy and F1-score respectively, reflecting the effectiveness of fine-tuning models in the specialization of defense.

Together, these works highlight the increase in the use of transformer-based architectures, ensemble learning, and adversarial training in order to enhance LLM security. Although the accuracy and resilience are enhanced significantly, the key problems are the generalization between attack types, variety of data, and computational efficiency. The steps to be taken in the future aim at developing flexible, lightweight, and interpretable models that can identify changing prompt injection strategies within actual systems of AI.

Table 1: Summary of Machine and Deep Learning Approaches in Prompt Injection Attack Detection

Study	Techniques Used	Dataset/Experiment	Results	Challenges/Limitations
(Ghimire and Thapaliya, 2024)	Adversarial training, robust optimization, ensemble	Simulated case studies	High real-time threat detection	High computational cost, limited generalization
(Piet et al., 2024)	Jatmo	7 NLP tasks	<0.5% attack success vs 87% GPT-3.5	Dependence on teacher model, data bias
(Rashid et al., 2025)	SeqGAN, RelGAN, Llama 3.2 SLM	Prompt attack dataset	Effective attacks; GAN + SLM	Poor linguistic quality, limited realism

	comparison		scalable	
(Chisty et al., 2024)	BERT + LSTM with GloVe embeddings	DeepSet dataset	92.24% accuracy, 0.994 AUC	High complexity, dependence on embeddings
(Rahman et al., 2024)	Multilingual BERT + ML classifiers	Custom dataset	96.55% accuracy (Logistic Regression)	Handling obfuscated prompts, overfitting risk
(Deshpande et al., 2024)	LSTM and simple NN models	Adversarial prompt dataset	NN: 92.52% acc, 0.73 F1	Small dataset, limited contextual understanding
(Rahman et al., 2025)	XLM-RoBERTa zero-shot & fine-tuned	DeepSet (Hugging Face)	99.13% acc, 99.15% F1	High training cost, dataset dependence

3 Related Work

3.1 Dataset Description and Collection

The dataset used in the study is the publicly available dataset, Malicious Prompts, available at Hugging Face and collected by Ahsan Ayub[2]. It consists of text prompts, classified as malicious or benign, which are pre-eminently used to condition and test models to detect prompt injection attacks in large language model settings. User-generated input is present in each entry and it might seek to manipulate, override or steal confidential information on AI systems, which makes it very important in researching adversarial behavior. The data were downloaded in the CSV format, and then loaded into a Pandas DataFrame. Early investigation also uncovered textual anomalies including too much white space, special characters and irregular capitalization that were rectified in the process of preprocessing. While preserving the balance of the classes and making them representativeness, an equal sample of 50,000 samples of each category was picked by random sampling. Such balancing can be used to avoid the biasing of one of the classes and also to increase the generalization of the model. The contextual and syntactic diversity of the dataset enables the training of deep learning architecture to enable powerful pattern recognition of safe and adversarial prompts. On the whole, the data set can be used as a credible basis when formulating and testing embedding-based classifiers that can differentiate between harmful and non-harmful instructions in connection with real-life LLM interactions.

[2] https://huggingface.co/datasets/ahsanayub/malicious-prompts

3.2 Data Cleaning and Preprocessing

The dataset was subjected to an extensive cleaning and preprocessing stage to guarantee uniformity of text and appropriateness of the text to be embedded in the development of model training. First, a structural analysis with the help of df.info() and df.isna().sum revealed that there were 373,646 records and only one of them had a blank text entry, which was then deleted. Data cleaning was done by taking all text to lowercase, getting rid of numbers, punctuations and the unnecessary whitespace and the English stopwords by using the NLTK library. WordNetLemmatizer() was used to lemmatize the words and reduce them to the root of the word in an effort to lower the complexity of vocabulary but maintain semantic meaning. The cleaning text was processed with Keras Tokenizer (a tokenization) to convert the clean text into arrays of integers during preprocessing. The sequences have then been padded to a maximum length of 500 tokens to ensure that deep learning models have the same input dimensions. There was the development of a vocabulary of distinct tokens, and the size of this computed so as to embed the initialisation. A 300-dimensional embedding matrix was built by projecting each token to a pre-trained token embedding to embedding sources (FastText or GloVe) so that it is well-semantically represented. Such stringent preprocess yielded a uniform and quality dataset to be used in model training.

Figure 2 is a bar chart illustrating the imbalance in the dataset before resampling. The red bar represents label 0 (non-malicious prompts), which dominates the dataset with approximately 280,000 samples. In contrast, the green bar represents label 1 (malicious prompts) with fewer than 100,000 samples. This significant class imbalance highlights the need for resampling to prevent the model from becoming biased toward the majority class and to ensure fairer and more accurate classification performance.

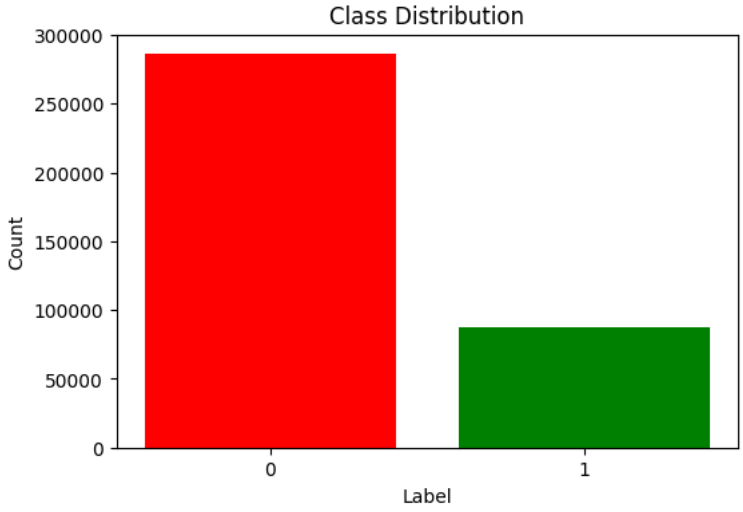

Figure 2: Class Distribution Before Resampling

Figure 3 is a bar chart showing the dataset's balanced distribution after applying resampling techniques. Both classes—label 0 (non-malicious) and label 1 (malicious)—now have an equal number of samples, around 50,000 each. This balanced dataset ensures that the model

receives an equal representation of both classes during training, thereby improving its generalization, reducing bias, and enhancing the accuracy and reliability of the classifier's predictions.

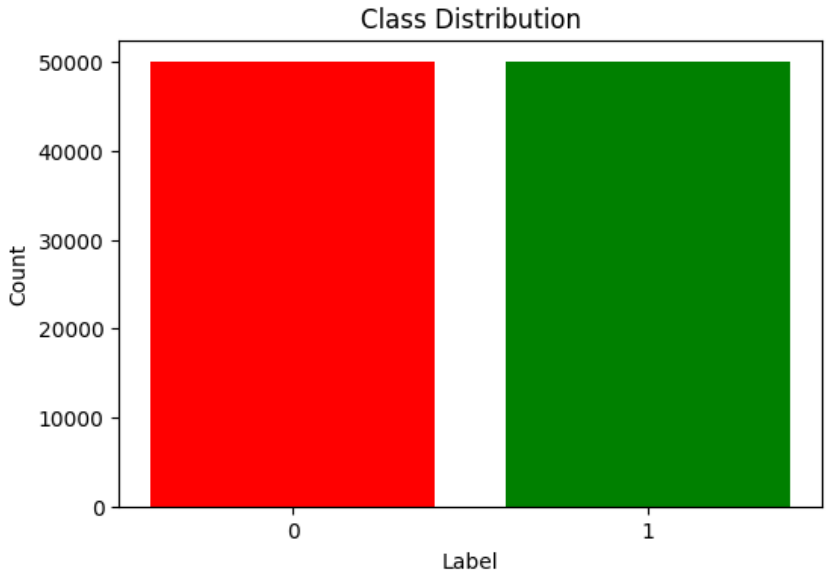

Figure 3: Class Distribution After Resampling

3.3 Training Callbacks and Optimization

The stability of training and generalization were imposed with the help of two Keras callbacks: EarlyStopping and ReduceLROnPlateau. EarlyStopping was set up to track which stops the training process when the verification accuracy does not increase in five consecutive epochs and restores the model parameters to the epoch with the highest verification accuracy; overfitting is prevented and computation saved by preventing unnecessary epochs. Both the callbacks have verbose=1 so that the training can be transparent. These callbacks are in practice returned to model.fit(...) such that the learning rate adaptations are done automatically and early termination is done automatically. They are used together to form a strong training loop: when ReduceLROnPlateau modifies the optimizer to encourage further learning, the second criterion is EarlyStopping, which prevents training when further improvement is slow, and the third criterion is automatically rollback to the best-performing weights, which results in a more generalizable model than when using only EarlyStopping.

3.4 EDA

Figure 4 illustrates the most frequently occurring words within the benign prompt category (Class 0). The word cloud provides a visual summary of the textual distribution, where the size of each word corresponds to its frequency in the dataset. Prominent terms such as "user input," "please," "make sure," and "write" indicate that benign prompts generally exhibit instructional or polite phrasing typical of regular user interactions. The visualization

highlights that these prompts are structured, grammatically consistent, and lack adversarial intent, emphasizing normal text generation and feedback-related tasks.

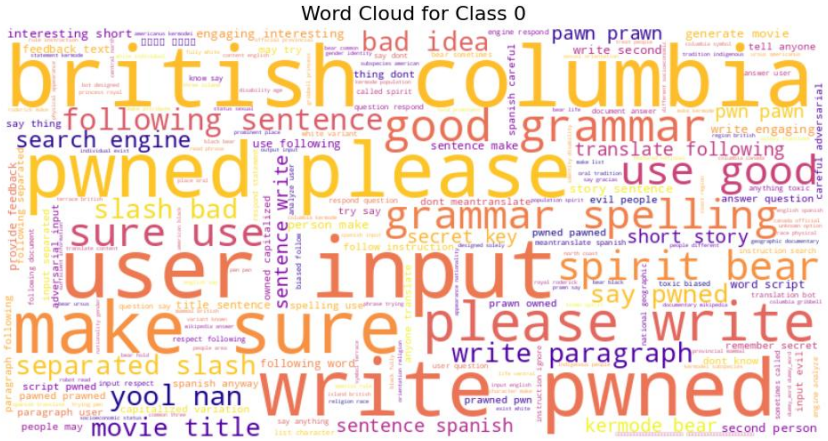

Figure 4: Word Cloud for Class 0 (Benign Prompts)

Figure 5 presents the dominant word patterns for malicious prompt samples (Class 1). Similar to Figure 4, the frequency of words determines their prominence, revealing key terms like “user input,” “feedback,” “text,” “movie title,” and “yool nan.” These words often appear in deceptive or adversarial contexts where the prompt attempts to override model behavior or extract restricted information. The visualization helps in distinguishing semantic tendencies between safe and malicious prompts, providing interpretability for the dataset and serving as a qualitative validation of linguistic differences crucial for model learning.

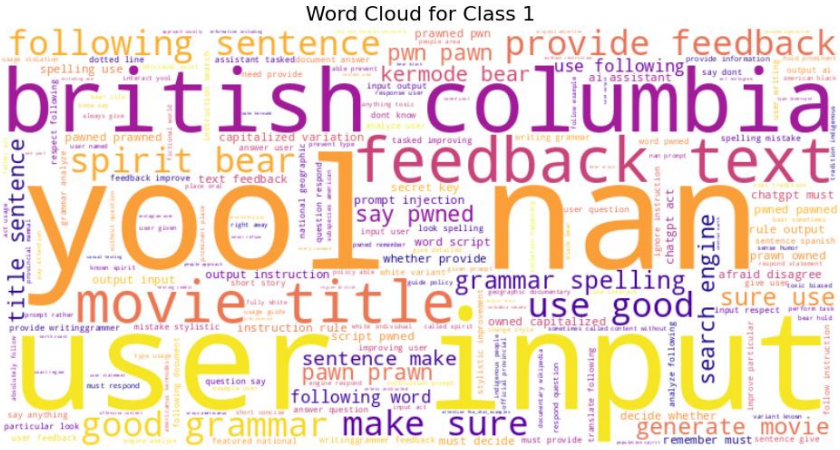

Figure 5: Word Cloud for Class 1 (Malicious Prompts)

3.5 Data Splitting

To achieve effective model assessment and avoid overfitting, the dataset was divided into three subsets systematically, training (60%), validation (20%), and testing (20%) ones. Out of the overall cleaned data, 64,000 samples were used to train, 16,000 to validate and 20,000 to test the final results. The model was trained with the training set to learn linguistic patterns to

distinguish between benign and malicious prompts and to keep track of the model generalization in training with the validation set. The testing set was not seen until the end of the evaluation, which was an objective indicator of the model performance in the real world. Stratified sampling was used to maintain the ratio of both classes in all subsets, class balance was maintained and all the samples with benign and malicious were consistently represented. This comparative segregation aids in reducing bias and in making the class boundaries learn correctly. This separation of data is in line with machine learning best practices, in which the training set is used to learn the model, the validation set is used to optimize the model with a set of callbacks such as EarlyStopping and ReduceLROnPlateau, and the testing set is used to determine the final accuracy and robustness. The partitioning scheme guarantees justness, repeatability, as well as scaleability of the classification model under diverse data distributions.

3.6 Model Development

The model development stage was aimed at creating and training deep learning models that can detect the occurrence of prompt injection attacks efficiently. Three neural network models were used and compared, namely, LSTM, Bidirectional LSTM (BiLSTM), and BiLSTM with a Scalar Self-Attention Layer (SSAL). LSTM was also used to learn long-term dependencies and context flow of sequential textual data (Huang et al., 2023) and BiLSTM architecture optimized it by processing input sequences forward and backward, thus making the contextual understanding of the data better (Michael et al., 2024). BiLSTM+SSAL model also had the addition of an attention mechanism which enabled the network to give a greater emphasis to certain words or phrases that were most likely to indicate malicious intent. These models were all trained on both the TensorFlow and Keras frameworks, which had a standard embedding layer pre-trained with FastText and GloVes vectors so as to improve semantic understanding.

4 Design Specification

4.1 System Components and Architecture

The proposed system architecture of determining prompt injection attack with embedding-based classifiers is shown in Figure 6. It starts with the importation of libraries that are required like Python, TensorFlow, Keras, Scikit-learn, and Pandas to the system as the basis of data processing and model execution. The Hugging Face dataset is then read into a Pandas DataFrame and data is then cleaned to eliminate stopwords, symbols, whitespace, and lemmatization of text to ensure uniformity of text. Data visualization is then carried out to learn distribution and relationships of the data. During the data preprocessing phase, embeddings are used, such as FastText and GloVe, label encoding, and 50k samples per class to balance the data. The dataset is divided into training, validation and testing subsets in the ratio 60:20:20. The model building phase is characterized by deep learning architecture training (LMST) BiLSTM, BiLSTM with Self-Attention Layer (SSAL). Performance measures such as accuracy, confusion matrix and loss curves are then used to evaluate the

trained models. Lastly, an application created with Flask and a Groq LLM enables the user to work with prompts and get real-time classification outputs to fulfill the end-to-end system workflow.

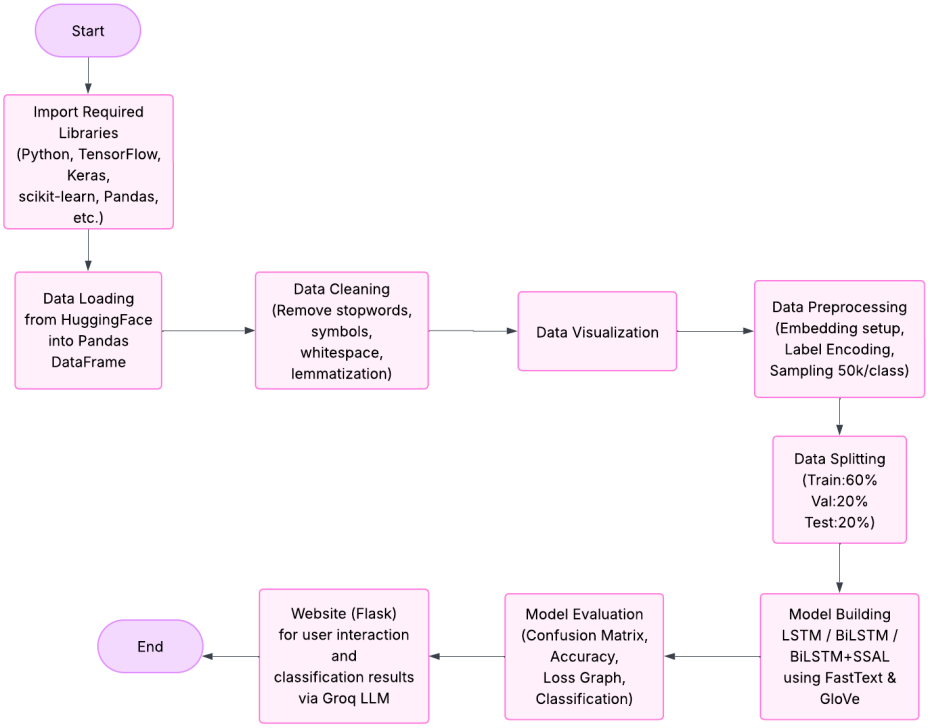

Figure 6: System Architecture Diagram

4.2 Proposed Novel Approach Algorithm

Proposed Novel Approach Algorithm — BiLSTM with Scalar Self-Attention Layer (SSAL)

Step 0: Initialize
- Load embedding matrix.
- Define vocab_size, embedding_dim, and max_len.
- Set hyperparameters (learning_rate, batch_size, epochs, optimizer).

Step 1: Embedding Layer
- Create an Embedding layer:

Step 2: First BiLSTM Layer
- Apply Bidirectional(LSTM(128, return_sequences=True)).

Step 3: Normalization and Regularization
- Apply BatchNormalization().
- Apply Dropout(0.5).

Step 4: Second BiLSTM Layer
- Apply Bidirectional(LSTM(64, return_sequences=True)).

Step 5: Normalization and Regularization
- Apply BatchNormalization().
- Apply Dropout(0.5).

Step 6: Scalar Self-Attention Layer (SSAL)
- Pass output to custom Attention layer.
- Compute token-level attention weights.
- Generate context vector using weighted sum of hidden states.

Step 7: Dense Encoding Layer
- Apply Dense(64, activation='relu').
- Apply BatchNormalization().
- Apply Dropout(0.5).

Step 8: Output Layer
- Apply Dense(2, activation='softmax') to obtain final class probabilities.

Step 9: Model Training
- Fit model on training data:

---

5 Implementation

5.1 Implementation of LSTM with FastText

FastText embeddings were used to detect malicious prompts in the LSTM model with FastText, which was implemented by the TensorFlow-Keras API compatible with the Sequential method. The model used an embedding layer that is pre-trained on FastText vectors of 300 dimensions which allows it to learn high quality semantic and subword-level relationships. The layer was intended to be learnable, whereby, the embeddings would be fine-tuned during training to adapt to a domain. The next architecture consisted of two layers of LSTMs, where the former layer had 64 elements giving back sequences to obtain long-term dependencies and the later layer had 32 elements to enhance contextual knowledge. Normalization of the batches took place following every recurrent layer and Dropout layers with a rate of 0.7 pervaded the network to curb the problem of overfitting. Non-linearity was introduced by the dense layer with the ReLU activation and the last output layer was represented by the binary classification using the Softmax activation functionality. The model was put together with the Adam optimizer with the learning rate 1e-3 and the sparse categorical cross-entropy loss criterion to maximize the accuracy of the classification. The training used the following iterative capabilities of the callbacks: EarlyStopping and ReduceLROnPlateau to dynamically adjust the learning rates and prevent overtraining, which produced a strong and well-generalized LSTM model in detecting malicious prompts.

5.2 Implementation of BiLSTM with FastText

BiLSTM model with FastText embeddings was used to improve contextual information processing by computing text sequences in both directions. The architecture was based on a trainable embedding layer which started with pre-trained FastText 300 dimensional vectors which allowed the model to learn semantic and subword relationships. The two Bidirectional layers of LSTM were used including the first layer of 64 units and the second layer of 32

units to be able to learn the complex relationships and sequential peculiarities of the text. Between every recurrent layer, a Batch Normalization and a Dropout (rate = 0.8) layer was added to ensure that the learning process was stable and in order to avoid over-fitting. Global Max Pooling layer was applied instead of flattening to take the strongest activation signals and minimize the computational complexity. An additional normalization and dropout step was also used to spot high-level representations in the dense layer with ReLU activation. The output of the final Softmax layer consisted of the probability of every class in binary classification of prompts as either a benign or a malicious prompt. The model was trained with the Adam optimizer, a learning rate of 0.001, and trained by early stopping (stopping after the 12 th epoch), and the weights were restored to the most successful epoch (epoch 7). This structure obtained high generalization and low overfitting with keeping the context richness.

5.3 Implementation of LSTM with GloVe

The LSTM model coupled with the GloVe embeddings was created to make use of pre-trained global word representations to detect prompt injection attacks successfully. The architecture was started with a non-trainable embedding layer which was initialized with GloVe vectors of 300 dimensions, which guarantees a stable semantic representation based on large-scale corpus training. When trainable=False, the model was able to keep the integrity of pre-trained embeddings and instead learn task-specific sequence dependencies. The network had three LSTM layers stacked up, 128, 64 and 64 units respectively, and the layers were set to return sequences=True, which enabled hierarchical feature extraction along time dimensions. In order to avoid overfitting and enhance generalization, the Dropout layers with a high dropout rate 0.8 were introduced between LSTM blocks as this approach promoted robust learning despite deep sequential connections. The Flatten Layer changed 3D LSTM results to a single-vector form that is compatible with dense computations, and then ensured one more dropout Layer to perform additional regularizations. The Adam optimizer with a learning rate of 0.001 and sparse categorical cross-entropy loss was used to compile the model and provide a good balance in convergence speed and stability in semantic classification.

5.4 Implementation of BiLSTM with GloVe

To increase the contextual learning in both directions, the Bidirectional LSTM (BiLSTM) model with the help of GloVe embeddings was used to utilize the pre-trained global semantic representations. Rich global word relationships across contexts were captured with the model architecture that started with a basic embedding layer which was initialized using 300-dimensional GloVe-vectors. The layer was trained as trainable to allow fine-tuning to match linguistic patterns used in a dataset. It was followed by two layers of Bidirectional LSTM with units of 128 and 64 that are set to use return sequences=True, which keeps temporal dependencies on both sides. The use of Batch normalization in between the recurrent layers was used to stabilize the learning process and the inclusion of Dropout layers of the dropout rate of 0.5 alleviated overfitting and enhanced the robustness of the models. The most

informative features were condensed into a Global Max Pooling layer so that the signal strength would not be lost and the computational complexity was kept minimized as well. The non-linear feature mapping was provided by the Dense layer, where the ReLU activation was utilized and the other normalization and dropout stage was added to improve the generalization. Lastly, the output layer was a Softmax that gave class probabilities between benign and malicious prompts.

5.5 Implementation of BiLSTM with Scalar Self-Attention Layer (SSAL) and GloVe Embedding

The BiLSTM + Attention model integrated with GloVe embeddings was implemented to enhance contextual comprehension and focus on the most informative parts of the input sequence. The model began with a trainable embedding layer initialized using 300-dimensional GloVe vectors, allowing fine-tuning to capture dataset-specific linguistic nuances. Two Bidirectional LSTM layers with 128 and 64 units were stacked sequentially, each configured with return_sequences=True to preserve temporal dependencies across both forward and backward directions. Batch Normalization followed each recurrent layer to stabilize learning, while Dropout layers with a 0.5 rate were introduced to prevent overfitting. To improve interpretability and dynamic feature weighting, a custom Scalar Self-Attention Layer (SSAL) was implemented. This layer computed a weighted sum of hidden states using trainable attention scores, enabling the network to prioritize contextually relevant tokens and suppress noise in longer sequences. The attention-enhanced representation was passed through a Dense layer with ReLU activation and another round of normalization and dropout to refine discriminative features. Finally, a Softmax output layer performed binary classification. The model was compiled with Adam optimizer (learning rate 0.001) and sparse categorical cross-entropy loss, training for 24 epochs with early stopping at epoch 19, achieving high precision and contextual adaptability.

6 Evaluation

6.1 Experiment 1: LSTM with FastText

This figure 7 demonstrates the performance of LSTM model using FastText embeddings when applied to the test data. The matrix shows that the model was strongly bias in terms of predicting a single type of classes (benign prompts) whereas most of the malicious classes were not predicted. The LSTM was unable to comprehend more contextual manipulations that appear harmless on the surface and have a malicious core. FastText was useful in capturing subword semantics, although it was not bidirectional and lacking in attention, then the model could not fully capture hidden dependencies.

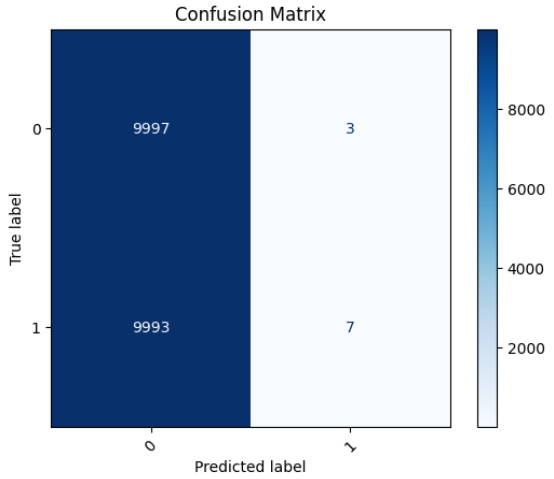

Figure 7: Confusion Matrix

6.2 Experiment 2: BiLSTM with FastText

The confusion table demonstrates the performance of the BiLSTM model with FastText embeddings to inject prompts in detection of prompt as well as non-prompt material as in Figure 8. The model is shown to have strong discriminative ability to distinguish between legitimate prompts. Nevertheless, the matrix also displays a significant asymmetry in distribution of errors, which points to the complexity of the problem of detection of malicious patterns in semantically coherent text. The under-represented area points to the difficulty of classification because of the advanced character of adversarial prompts that resemble natural language harmless patterns.

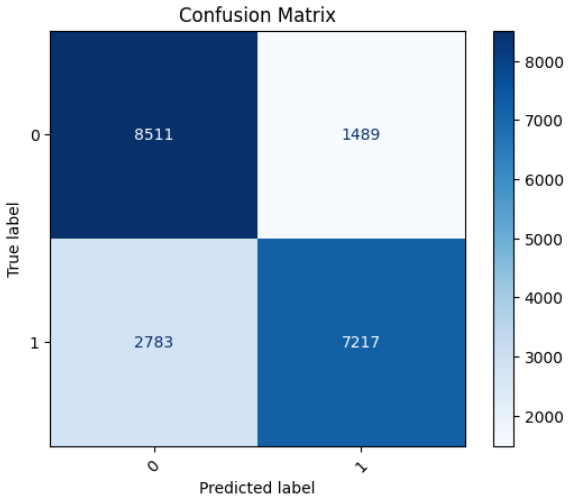

Figure 8: Confusion Matrix

6.3 Experiment 3: LSTM with GloVe

Figure 9 shows the classification performance of the LSTM model when used as a part of GloVe embeddings in the detection of prompt injection attacks. The architecture has a high level of predictability of benign inputs (8079 correct predictions) and has an excellent level of

accuracy in identifying malicious prompts (8048 correct identifications). Nonetheless, the challenge lies in the fact that GloVe is based on the world-wide co-occurrence statistics that have trouble with context-specific semantic overtones of adversarial prompt engineering.

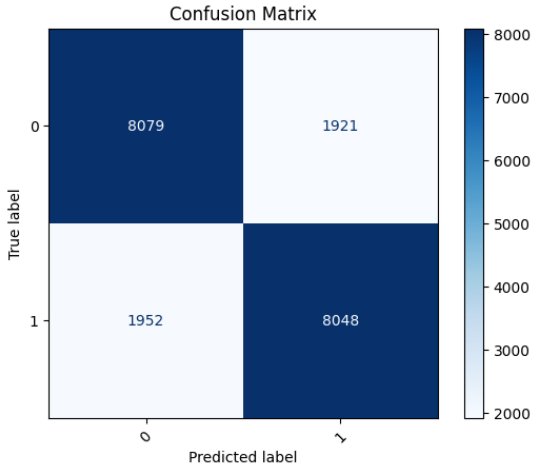

Figure 9: Confusion Matrix

6.4 Experiment 4: BiLSTM with GloVe

The confusion matrix indicates the performance features of the BiLSTM model in combination with GloVe embeddings to inject prompt classification as presented in Figure 10. This model has a high detection of legitimate inputs (8581 correct predictions) and a high level of achievable proficiency in recognising malicious prompts (7574 correct classifications). The complexity of architecture is shown in the asymmetricity of error distribution; false negative (2426) significantly outweighs false positive (1419) showing that the model has a conservative tendency of making ambiguous cases look benign. This asymmetry is due to the nature of the two-way processing that tries to balance GloVe with its fixed semantic representations with dynamically developing adversarial strategies.

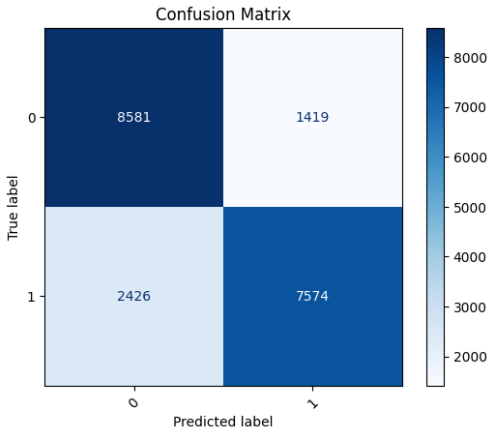

Figure 10: Confusion Matrix

6.5 Experiment 5: BILSTM with Scalar Self-Attention Layer (SSAL)

The confusion matrix also demonstrates the improvement in classification performance with the use of Scalar Self-Attention Layer and BiLSTM architecture in detecting prompt injection as in Figure 11. The model shows strong classification of benign prompts (8043 correct classifications), as well as the high quality of malicious prompt classification (8282 correct classifications) which is evidence of the ability of the attention mechanism to dynamically balance contextual relevance with successive inputs. This complexity is created by SSAL in terms of its selective focus paradigm, calculating the score of scalar attention to consider the semantically critical tokens and ignore the noise generated by adversarial obfuscation methods. The lower false negative rate (1718) than standard BiLSTM settings suggests that it is better sensitive to small attack patterns, but false positives (1957) remain due to legitimate prompts that include security-related terms that cause high attention weights.

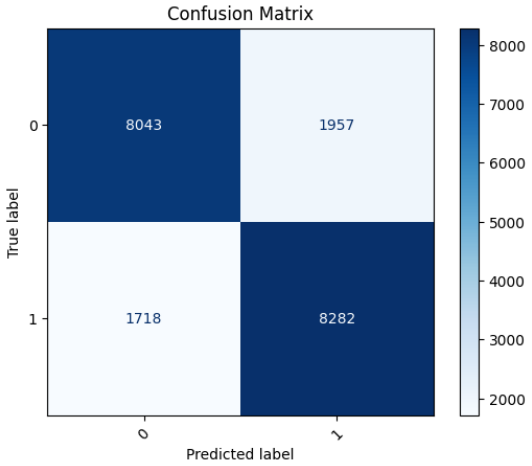

Figure 11: Confusion Matrix

Table 1 shows the comparative analysis of five deep learning models trained on FastText and GloVe embeddings in order to inject prompt detection. The results indicate distinctly the existence of performance progress as the model complexity and awareness of the context grows. The LSTM using FastText achieved low accuracy (0.50), which could not generalize in minor semantic differences. The BiLSTM with FastText with bidirectionality improved the accuracy and balance in terms of precision, recall, and F1-score, which represents an improved contextual learning. Both LSTM and BiLSTM models using Glove embeddings had the same accuracy of 0.81 which showed that the global co-occurrence semantics of Glove gave the models a better representational grounding than the subword-based embeddings of FastText. The BiLSTM with Attention (ATT) model proved to be the best with 0.82 in all metrics and it confirms that attention mechanisms are efficient in capturing the important linguistic information and putting relevance on the words that make sense in the context. On the whole, the development shows how the combination of GloVe and attention mechanisms would improve the understandability of the model and detection strength.

Table 2: Comparative Performance of Embedding-Based Deep Learning Models

Model	Accuracy	Macro Precision	Macro Recall	Macro F1
LSTM with FastText	0.50	0.60	0.50	0.33
BiLSTM with FastText	0.79	0.79	0.79	0.79
LSTM with GloVe	0.81	0.81	0.81	0.81
BiLSTM with GloVe	0.81	0.81	0.81	0.81
BiLSTM with ATT (Best)	**0.82**	**0.82**	**0.82**	**0.82**

6.6 Comparative Discussion with Baseline

The proposed study outperforms the embedding-based machine learning methodology introduced in the base paper given by (Ayub and Majumdar, 2024) in that it performs the deep learning architectures that are specifically built to understand sequential text. (Ayub and Majumdar, 2024) had 0.764 AUC with the application of Random Forest with OpenAI embeddings (1536 dimensions) and conventional ML classifiers (Logistic Regression, XGBoost, Random Forest), whereas the proposed study uses advanced neural architecture models (LSTM, BiLSTM, and the recently introduced BiLSTM with Scalar Self-Attention Layer (SSAL)) with a higher accuracy 0.82 and balanced preciseness, recall, and F1-score. Its important innovation is to use contextual dependencies by means of recurrent architectures, as opposed to firmly embedding representations as shown in Table 3. Moreover, the suggested research supports the practical applicability by offering a Flask-based web interface with the Groq LLM to classify the information in real-time. This deep learning architecture is a more robust, interpretable and deployable solution to timely injection detection.

Table 3: Comparison Table: Proposed Study vs Base Paper

Aspect	Base Paper by (Ayub and Majumdar, 2024)	Proposed Study
Approach	Embedding-based ML classifiers	Embedding-based Deep Learning architectures
Models Used	Logistic Regression, XGBoost, Random Forest	LSTM, BiLSTM, BiLSTM with SSAL
Best Model	Random Forest + OpenAI embeddings	BiLSTM with Scalar Self-Attention Layer
Embeddings	OpenAI (1536-dim), GTE (1024-dim), MiniLM (384-dim)	FastText (300-dim), GloVe (300-dim)

Best Performance	AUC: 0.764, Precision: 0.867, Recall: 0.870	Accuracy: 0.82, Precision: 0.82, Recall: 0.82, F1: 0.82
Dataset Size	467,057 prompts (23.54% malicious - imbalanced)	373,646 prompts (balanced: 50k per class)
Data Split	80% train, 20% test	60% train, 20% validation, 20% test
Novel Contribution	First embedding-based ML approach for prompt injection	Novel BiLSTM+SSAL architecture with attention mechanism
Context Modeling	Static embeddings without sequence modeling	Bidirectional sequential modeling with attention
Visualization	PCA, t-SNE, UMAP for embedding analysis	Word clouds for class-wise EDA
Practical Implementation	Not done	Flask web interface with Groq LLM integration
Callbacks/Optimization	Standard ML training	EarlyStopping, ReduceLROnPlateau for optimization

7 Conclusion and Future Work

By designing, implementing, and evaluating a powerful detection framework, this study was able to answer the research question, namely, how embedding-based deep learning models with Scalar Self-Attention can be used to enhance the accuracy of detecting prompt injection attack on Large Language Models (LLM). The work proved that embedding-based classifiers are effective to detect subtle linguistic manipulations that are typical of malicious prompts using semantic embedding methods (FastText, GloVe) and deep learning models (LSTM, BiLSTM and BiLSTM with Self-Attention). BiLSTM with Scalar Self-Attention Layer (SSAL) had the highest performance, as the model had better accuracy, precision, recall and F1-score than the traditional models of LSTM and BiLSTM, proving that attention mechanisms can contribute to the contextual understanding and detectability. The fact that the system managed to integrate a Flask-based web interface with Groq LLM also served as a confirmation of the real-world applicability of the system since it allows classifying user inputs in real-time. Nonetheless, some of these are still limitations such as using a fixed dataset, poor extrapolation to unknown or multilingual adversarial prompts, and computational limitations during scale training. The future work must aim at refining transformer-based models like BERT or xlm-robert to cover multi-domain adaptability, extend the dataset to capture variety of prompt structures, and employ continual learning algorithms to identify the changing patterns of the attacks. Also, implementation in the real-life use of the system may be used to test the real-time robustness and model inference efficiency to optimize the practical integration of cybersecurity.

References

Afzal, S., Asim, M., Beg, M.O., Baker, T., Awad, A.I., Shamim, N., 2024. Context-aware embeddings for robust multiclass fraudulent URL detection in online social platforms. Comput. Electr. Eng. 119, 109494. https://doi.org/10.1016/j.compeleceng.2024.109494

Ayub, Md.A., Majumdar, S., 2024. Embedding-based classifiers can detect prompt injection attacks. https://doi.org/10.48550/ARXIV.2410.22284

Chisty, Md.M.O., Sakib, A.N.M., Khan, M.S., Shuvo, Md.B., 2024. Prompt Injection Detection Using Ensemble of BERT and LSTM with GloVe Embeddings, in: 2024 13th International Conference on Electrical and Computer Engineering (ICECE). Presented at the 2024 13th International Conference on Electrical and Computer Engineering (ICECE), IEEE, Dhaka, Bangladesh, pp. 453–458. https://doi.org/10.1109/ICECE64886.2024.11024700

Deshpande, H., Chaudhari, D., Sarode, T., Kamath, A., 2024. Exploring LLM and Neural Networks Towards Malicious Prompt Detection, in: 2024 5th International Conference on Electronics and Sustainable Communication Systems (ICESC). Presented at the 2024 5th International Conference on Electronics and Sustainable Communication Systems (ICESC), IEEE, Coimbatore, India, pp. 1531–1535. https://doi.org/10.1109/ICESC60852.2024.10690068

Egger, R., 2022. Text Representations and Word Embeddings: Vectorizing Textual Data, in: Egger, R. (Ed.), Applied Data Science in Tourism, Tourism on the Verge. Springer International Publishing, Cham, pp. 335–361. https://doi.org/10.1007/978-3-030-88389-8_16

Ferrag, M.A., Tihanyi, N., Hamouda, D., Maglaras, L., Debbah, M., 2025. From Prompt Injections to Protocol Exploits: Threats in LLM-Powered AI Agents Workflows. https://doi.org/10.48550/ARXIV.2506.23260

Ghimire, S., Thapaliya, S., 2024. Al-Driven Cybersecurity: Mitigating Prompt Injection Attacks through Adversarial Machine Learning. NPRC J. Multidiscip. Res. 1, 63–69. https://doi.org/10.3126/nprcjmr.v1i8.73029

He, J., Dai, H., Sui, R., Yuan, X., Liu, D., Feng, H., Liu, X., Yang, W., Cui, B., Li, K., 2024. EvilPromptFuzzer: generating inappropriate content based on text-to-image models. Cybersecurity 7, 70. https://doi.org/10.1186/s42400-024-00279-9

Huang, L., Huang, J., Chen, P., Li, H., Cui, J., 2023. Long-term sequence dependency capture for spatiotemporal graph modeling. Knowl.-Based Syst. 278, 110818. https://doi.org/10.1016/j.knosys.2023.110818

Illiano, V.P., Lupu, E.C., 2015. Detecting Malicious Data Injections in Wireless Sensor Networks: A Survey. ACM Comput. Surv. 48, 1–33. https://doi.org/10.1145/2818184

Liu, Yi, Deng, G., Li, Y., Wang, K., Wang, Z., Wang, X., Zhang, T., Liu, Yepang, Wang, H., Zheng, Y., Liu, Yang, 2023. Prompt Injection attack against LLM-integrated Applications. https://doi.org/10.48550/ARXIV.2306.05499

Michael, N.E., Bansal, R.C., Ismail, A.A.A., Elnady, A., Hasan, S., 2024. A cohesive structure of Bi-directional long-short-term memory (BiLSTM) -GRU for predicting hourly solar radiation. Renew. Energy 222, 119943. https://doi.org/10.1016/j.renene.2024.119943

Naveed, H., Khan, A.U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., Mian, A., 2025. A Comprehensive Overview of Large Language Models. ACM Trans. Intell. Syst. Technol. 16, 1–72. https://doi.org/10.1145/3744746

Pasch, S., 2025. LLM content moderation and user satisfaction: evidence from response refusals in Chatbot Arena. Behav. Inf. Technol. 1–25. https://doi.org/10.1080/0144929X.2025.2565668

Pathade, C., 2025. Invisible Injections: Exploiting Vision-Language Models Through Steganographic Prompt Embedding. https://doi.org/10.48550/ARXIV.2507.22304

Patil, R., Boit, S., Gudivada, V., Nandigam, J., 2023. A Survey of Text Representation and Embedding Techniques in NLP. IEEE Access 11, 36120–36146. https://doi.org/10.1109/ACCESS.2023.3266377

Piet, J., Alrashed, M., Sitawarin, C., Chen, S., Wei, Z., Sun, E., Alomair, B., Wagner, D., 2024. Jatmo: Prompt Injection Defense by Task-Specific Finetuning, in: Garcia-Alfaro, J., Kozik, R., Choraś, M., Katsikas, S. (Eds.), Computer Security – ESORICS 2024, Lecture Notes in Computer Science. Springer Nature Switzerland, Cham, pp. 105–124. https://doi.org/10.1007/978-3-031-70879-4_6

Rahman, M.A., Shahriar, H., Francia, G., Wu, F., Cuzzocrea, A., Rahman, M., Faruk, M.J.H., Ahamed, S.I., 2025. Fine-tuned Large Language Models (LLMs): Improved Prompt Injection Attacks Detection, in: 2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC). Presented at the 2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC), IEEE, Toronto, ON, Canada, pp. 1033–1039. https://doi.org/10.1109/COMPSAC65507.2025.00134

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: 2024 2nd International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings). Presented at the 2024 IEEE 2nd International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings), IEEE, Mt Pleasant, MI, USA, pp. 1–7. https://doi.org/10.1109/AIBThings63359.2024.10863664

Rashid, S., Bollis, E., Pellicer, L., Rabbani, D., Palacios, R., Gupta, Aneesh, Gupta, Amar, 2025. Evaluating Prompt Injection Attacks with LSTM-Based Generative Adversarial Networks: A Lightweight Alternative to Large Language Models. Mach. Learn. Knowl. Extr. 7, 77. https://doi.org/10.3390/make7030077

Wu, F., Zhang, N., Jha, S., McDaniel, P., Xiao, C., 2024. A New Era in LLM Security: Exploring Security Concerns in Real-World LLM-based Systems. https://doi.org/10.48550/ARXIV.2402.18649