1. Table des matières

4. Chapter 4

Test and Evaluation

4.1 Introduction

The detection of SQL injections using AI techniques, such as machine learning and deep learning algorithms, has gained the interest of many researchers in this field. These techniques have been shown to be effective at identifying SQL injection attacks with high accuracy.

In this chapter, we will discuss the test and evaluation of our model for detecting SQL injection attacks. We will use a variety of metrics, including accuracy, precision, recall, and F1 score. We will also compare the performance of our model to other machine learning algorithms and related works.

4.2 Confusion matrix

The confusion matrix provides a tabular representation of the model's predictions against the actual labels. It allows us to visualize the distribution of true positives, true negatives, false positives, and false negatives, providing valuable insights into the model's performance [27].

	Positive Prediction	**Negative Prediction**
Positive Class	True Positive (TP)	False Negative (FN)
Negative Class	False Positive (FP)	True Negative (TN)

Table 4.1. Confusion Matrix.

TP (True Positives): Correctly predicted positive instances.

TN (True Negatives): Correctly predicted negative instances.

FP (False Positives): Incorrectly predicted positive instances.

FN (False Negatives): Incorrectly predicted negative instances.

4.3 Evaluation metrics for assessing model performance

In the field of machine learning and classification tasks, evaluation metrics play a crucial role in assessing the performance of models. These metrics provide quantitative measures that help us understand the accuracy, effectiveness, and reliability of model predictions. When evaluating the performance of classification models, it is essential to examine the appropriate evaluation metrics that provide insights into their strengths and weaknesses. In this section, we will explore some of the most commonly used evaluation metrics that provide valuable insights into the performance of classification models.

4.3.1 Accuracy

Accuracy is a commonly used evaluation metric that measures the overall correctness of model predictions. It calculates the ratio of correct predictions to the total number of instances. Accuracy provides a general overview of the model's performance across all classes [27].

$$\textbf{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

4.3.2 Precision

Precision focuses on the proportion of correctly identified positive predictions (true positives) out of the total positive predictions made by the model. It helps assess the model's ability to minimize false positives [28].

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

4.3.3 Recall

Recall, also known as sensitivity or true positive rate, measures the proportion of true positive predictions captured by our model out of the total actual positive instances. It reflects the model's ability to minimize false negatives [28].

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

4.3.4 F1 Score

The F1 score is a combined metric that balances precision and recall. It provides a harmonic mean of these two measures and offers a comprehensive evaluation of the model's performance [29].

$$\textbf{F1 Score} = \frac{2 * Precision * Recall}{Precision + Recall}$$

By analyzing these evaluation metrics, we can gain a deeper understanding of how our classification model performs and identify areas for improvement. These metrics provide valuable insights into the model's strengths and weaknesses, allowing us to make informed decisions to enhance its performance.

4.4 Model performance analysis

In this section, we evaluate the performance of our SQL injection detection model based on the BERT architecture. As already discussed the model was trained on a dataset containing 22,599 samples, consisting of both SQL injections instances and normal SQL queries.

We assess the model's performance using a variety of evaluation metrics, including accuracy, precision, recall, F1 score, and the confusion matrix. These metrics provide valuable insights into the model's ability to correctly classify SQL injection instances and non-malicious queries.

The performance results obtained are as follows:

Accuracy: The model achieved an accuracy of **100%** during training and **99.98%** during validation, indicating its high level of accuracy in classifying the queries.

F1 Score: The F1 score, which considers both precision and recall, also reached an impressive value of **99.98%.** This indicates a balance between correctly identifying normal queries and capturing all actual SQL injection instances.

Precision: The precision of the model, which measures the proportion of true positive predictions out of all positive predictions, was **100%.** This indicates a very low rate of false positives, meaning that the model maintains a high level of confidence in its SQL injection detection predictions.

Recall: The recall of the model, which measures the proportion of true positive predictions out of all actual positive instances, was **99.96%.** This indicates the model's ability to capture nearly all instances of normal queries, resulting in a low rate of false negatives.

The confusion matrix provides a more detailed breakdown of the model's performance:

	Positive Prediction	**Negative Prediction**
Positive Class	2251	1
Negative Class	0	2268

Table 4.2. Confusion Matrix (Classification Results).

The confusion matrix reveals that out of the 2252 positive instances (normal queries), the model correctly identified 2251 instances as positive (true positives) while misclassifying one instance as negative (false negatives). It also correctly classified 2268 out of 2268 negative instances as negative (SQL injections).

These results are consistent with the findings of Srishti Lodha and Atharva Gundawar from the Department of Computer Science and Engineering at Vellore Institute of Technology [30], who made a similar study using the BERT architecture for SQL injection detection. Their research demonstrated comparable performance and highlighted the effectiveness of the BERT model in accurately identifying SQL injection attacks.

Overall, the performance results demonstrate the high accuracy, precision, recall, and F1 score of our SQL injection detection model. These results, in alignment with the work of Srishti Lodha and Atharva Gundawar, underscore the effectiveness of the BERT model for accurately detecting SQL injection attacks while minimizing false positives and false negatives.

4.5 Evaluate the presence of overfitting

It is essential to analyze the loss values of the model during training. By observing the training and validation loss curves, we can gain insights into the model's generalization performance and potential overfitting issues.

The training loss curve represents the loss value computed during the training phase, while the validation loss curve represents the loss value calculated on a separate validation dataset. These curves provide a visual representation of how the model's performance evolves over epochs.

A key indicator of overfitting is when the training loss continues to decrease while the validation loss either stagnates or starts to increase. This suggests that the model is becoming overly specialized to the training data and is failing to generalize well to new, unseen examples.

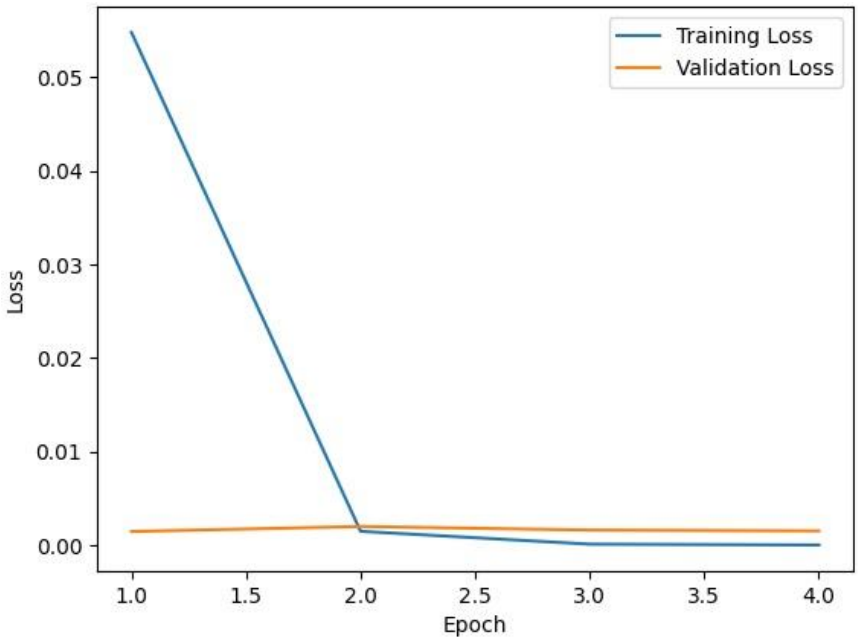

Figure 4.1 Training and validation loss

As shown in Figure 4.1, the training loss consistently decreases over the 4 epochs, indicating that the model is learning from the training data and improving its performance. The validation

loss also shows a decreasing trend, with fluctuations between epochs. However, the general trend is decreasing, which is a positive sign.

The fact that the validation loss generally follows the same decreasing trend as the training loss suggests that the model is not overfitting excessively. It's worth noting that some fluctuations in the validation loss between epochs are normal and can be attributed to random variations in the data or model training process.

4.6 Comparative analysis with other approaches

In this section, we present a comparative analysis of our BERT-based SQL injection detection model with other commonly used approaches available in the literature. While we didn't evaluate the performance of the alternative approaches ourselves, we have collected and compiled information from various sources on their reported performance. By comparing our model with these approaches, we aim to provide insights into the effectiveness of our BERT-based model for SQL injection detection. The comparison is based on commonly used evaluation metrics such as accuracy, precision, recall, and F1 score. The findings of this analysis are summarized in the following table.

Model name	Training Accuracy	Validation Accuracy	Precision	Recall	F1
KNN	100	99.12	98.85	99.52	99.18
SVM	92.78	92.44	90.21	96.36	93.19
BERT (our model)	**100**	**99.98**	**100**	**99.96**	**99.98**

Table 4.3. Comparing the model performances using various metrics (%).

The compared approaches were trained on a dataset of approximately 42,000 data points, while our dataset consisted of 22,599 samples. This disparity arises from the fact that we

obtained the original dataset from the same resource, but we had to perform data cleaning and preprocessing to ensure its quality.

The comparison shows that BERT (Our model) performed the best among the compared models in terms of key evaluation metrics, including accuracy, precision, recall, and F1 score. These results affirm the effectiveness of BERT in detecting SQL injection attacks, highlighting its superior performance in our study.

4.7 Model Performance evaluation on new Data

In this section, we present the results of testing our deep learning model on a new set of data samples. These samples serve as a rigorous assessment of our model's capabilities in handling unseen instances and provide valuable insights into its performance.

	Positive Prediction	**Negative Prediction**
Positive Class	10	0
Negative Class	0	10

Table 4.4. Confusion Matrix (Test Classification Results).

The accuracy of our model on the new data samples stands at a remarkable 100%. This indicates that our model achieved a perfect prediction rate, accurately identifying SQL injections and effectively distinguishing them from normal queries.

Furthermore, the recall score for the positive class (normal queries) is 100%, indicating that our model successfully identified all instances of normal queries present in the new dataset. This showcases its ability to capture the entirety of positive cases and avoid any false negatives.

Similarly, the precision is also 100%. This highlights the model's reliability in identifying SQL injections accurately and minimizing the risk of false positives.

Lastly, the F1 score, which combines precision and recall, also reaches a perfect 100% for the positive class. This score represents the overall balance between accurate detection and

avoiding false positives, emphasizing the model's ability to maintain a high level of performance across both measures.

4.8 Conclusion

In conclusion, our model for detecting SQL injection attacks has shown impressive effectiveness. Through testing and evaluation using various metrics, we have demonstrated its accuracy in identifying SQL injection attacks also the comparison with other models further validates its superior performance.

5. Bibliographie

[27] Sokolova, M., & Lapalme, G., A systematic analysis of performance measures for classification tasks, 2009. [Online]. Available : https://www.sciencedirect.com/science/article/abs/pii/S0306457309000259

[28] D. M. Powers, Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation, Journal of Machine Learning Technologies, 2011.

[29] C. J. Van Rijsbergen, Information Retrieval, 2nd Edition, Butterworths, 1979.

[30] Gundawar, Srishti Lodha and Atharva, SQL Injection and Its Detection Using Machine Learning Algorithms, 2023. [Online]. Available : https://link.springer.com/chapter/10.1007/978-3-031-28975-0_1