JURNAL TEKNOLOGI DAN OPEN SOURCE
Vol. 9, No. 1, June 2026, pp. 88~97
e-ISSN: 2622-1659, accredited Four Grade by Kemenristekdikti, Decree No: 152/E/KPT/2023
DOI: https://doi.org/10.36378/jtos.v9i1.5475

Neural Network-Based Exfiltration Schema Identification

Vetrick Aringga Dicktiony Racero[1], Agung Prasetya[2], Taufiq Agung Cahyono[3]
[1,2,3] Program Studi Informatika, Fakultas Sains Dan Teknologi, Universitas Bhinneka Pgri Tulungagung

Article Info

Article history:

Received 04 28, 2026
Revised 05 18, 2026
Accepted 06 06, 2026

Keywords:

Text-to-SQL,
Schema Exfiltration,
BERT,
Neural Network,
Deep Learning

ABSTRACT

This study uses the BERT architectural technique to identify schema exfiltration in a neural network-based Text-to-SQL system. The growing usage of Large Language Models (LLM) in Text-to-SQL systems, which may provide a danger of database schema leaking through user prompts, provides the context for this study. This research challenge is how to use a deep learning model to reliably and adaptively identify prompt modifications that could carry out exfiltration techniques. The study employed a deep learning strategy with a feedforward neural network as the classifier and the BERT architecture as the primary encoder. There were 20 classes in all, consisting of 19 exfiltration scheme categories and 1 benign class. The dataset was created using a variety of sources, including WikiSQL, DatabaseAnswers, and educational datasets. It was then subjected to tokenisation, labelling, and normalization processes. The model obtains an accuracy of 0.9462, precision of 0.8425, recall of 0.7483, F1-score of 0.7926, and precise match accuracy of 0.7596, according to the data. Additionally, the study demonstrated that the model outperformed implicit suggestions like role switching and prompt injection in identifying explicit prompts. The study concludes that while there are still issues with enhancing detection capabilities for intricate manipulating patterns, the BERT-based approach can provide good performance in identifying schema exfiltration in Text-to-SQL systems.

Corresponding Author:
Vetrick Aringga Dicktiony Racero
Department of Informatics Engineering
Bhinneka PGRI Tulungagung University
Tulungagung, Indonesian
Email: vetrickaringga99@gmail.com

1. Introduction

Text-to-SQL, often referred to as Structured Query Language (SQL), is an important component in the field of Natural Language Processing (NLP). This technology enables users to access, manage, and analyze data in relational databases using natural language without requiring an understanding of complex and technical SQL syntax [1]. This is particularly beneficial for non-technical users, as they can perform data retrieval, information analysis, and database manipulation more easily, quickly, and efficiently. Thus, Text-to-SQL serves as an important bridge between natural human language and formal, structured computer systems. The development of this technology has accelerated alongside the increasing demand for large-scale data processing across various domains such as education, business, and the digital industry (Gifari & Prasetya, 2025) [2].

The advancement of Text-to-SQL is closely tied to progress in Large Language Models (LLMs), which serve as the core foundation of modern language-based artificial intelligence systems [3]. LLMs are deep learning models trained on massive amounts of data, enabling them to understand language patterns, sentence structures, context, and semantic meaning at a more complex level. These capabilities allow LLMs to

perform various tasks such as question answering, text summarization, document analysis, and even automatic code generation with high accuracy [4]. In the context of Text-to-SQL, LLMs play a crucial role in translating natural language instructions into SQL queries that conform to the underlying database structure (Ahadi et al., 2025). Therefore, LLMs have become a core technology in the development of modern, more adaptive, and intelligent Text-to-SQL systems [5].

Various studies have been conducted to improve the performance of LLM-based Text-to-SQL systems, particularly in addressing the main challenge of accurately mapping natural language to SQL queries. One common issue is schema linking, which refers to the process of connecting words or phrases in user queries to the appropriate tables or columns in a database. In addition, ambiguity in natural language also presents a major challenge, as a single sentence may have multiple meanings depending on the context. To address these issues, several studies have proposed combining LLMs with prompt optimization techniques and knowledge graphs to better understand the relationship between user queries and database structures (Mohammadjafari et al., 2025) [6]. These approaches have been shown to improve the accuracy of generated SQL queries.

Furthermore, other studies have shown that LLMs have limitations in handling complex reasoning tasks when processing all information in a single step. This often causes the model to lose focus on important parts of the user instruction, especially in complex query structures. To address this issue, task decomposition approaches have been proposed, breaking the process into several simpler stages. These stages include identifying key information, classifying the task type, performing self-correction on initial outputs, and applying active learning to improve accuracy (Xie et al., 2024) [7]. This approach makes the query generation process more systematic, structured, and less prone to errors compared to single-step methods.

In addition to decomposition approaches, multi-LLM collaboration systems have also been developed to improve Text-to-SQL performance. Systems such as SQLfuse combine multiple models within a unified architecture consisting of schema mining, schema linking, SQL generation, and an evaluation module that iteratively refines the generated queries (Zhang et al., 2024) [8]. Through this collaboration, each model complements the weaknesses of others, resulting in more accurate and optimal query generation. Furthermore, benchmark-based evaluations are used to measure model performance across different levels of database complexity and query variations. These evaluations are essential for understanding the strengths and weaknesses of each model in the Text-to-SQL context (Wang, 2023) [9].

Although these studies have significantly contributed to the development of Text-to-SQL systems, an important issue that remains underexplored is schema exfiltration. Schema exfiltration refers to a situation where a model unintentionally reveals database structures such as table names, column names, or relationships that are not requested by the user [10]. This issue arises because LLMs are highly sensitive to prompt variations, often generating overly detailed responses that exceed the required information scope. For instance, when a user asks only for general information about a database, the model may instead provide a complete schema description. This indicates that output control mechanisms are still insufficient in restricting generated information appropriately [11].

The schema exfiltration phenomenon is a critical concern because it poses serious security risks to database systems. Exposed database structures can be exploited by malicious actors to launch attacks or perform unauthorized access. In addition, this issue highlights that current LLM context control mechanisms are not yet fully capable of constraining outputs to match user intentions precisely [12]. Based on these challenges, this study focuses on identifying single-turn prompt variations that may trigger schema exfiltration using a multi-class classification approach. The proposed model employs BERT as an encoder to capture semantic context, along with a feedforward layer as a classifier to determine prompt categories. This study also introduces an Indonesian-language dataset to improve relevance in local contexts, with the aim of enhancing the security of Text-to-SQL systems more effectively.

2. Research Method

The literature review, data collection, model design, implementation, and model evaluation constitute the main methodological steps that form the overall research approach in this study. In the initial stage, a literature review was conducted to establish a strong theoretical foundation by examining various scientific publications related to Text-to-SQL, database security, and schema exfiltration detection [13]. This review aims to understand previous research developments, identify methods that have been used, and discover research gaps that have not yet been extensively explored. Through this literature review, the study is directed in a clear and structured manner in developing the proposed approach.

Following the literature review stage, the process continues with data collection from various relevant sources, including WikiSQL, DatabaseAnswers, and several educational datasets that support experimental requirements. The collected data is then transformed into the SQLite format to better represent realistic

database conditions aligned with Text-to-SQL testing scenarios [14]. Subsequently, a normalization process is performed to ensure data consistency, followed by schema correction to adjust database structure, and labeling to distinguish between exfiltration and benign categories. After completing these steps, the dataset is organized into 19 schema exfiltration categories and 1 benign category. This categorization is performed to ensure that the model is capable of recognizing various attack patterns while also distinguishing them from safe prompts, thereby making the dataset more representative of real-world database system conditions [15].

Figure 4. Research Methodology

The next stage is model design, which employs a deep learning approach using the BERT architecture as the primary encoder. The model is designed to understand the contextual meaning of natural language user prompts that may implicitly or explicitly contain attempts to expose database schema information. By utilizing BERT, the model is able to capture semantic relationships between words within a sentence, making it more sensitive to variations in linguistic patterns used in prompts. This approach is particularly important because schema exfiltration often appears in non-explicit forms, requiring deep contextual understanding to be properly identified. Therefore, BERT is selected due to its effectiveness in modeling global context within a sentence.

The text input is processed through a BERT tokenizer to generate token IDs, attention masks, and embeddings, which serve as the numerical representation of the sentence. Token IDs function as identifiers for each word or sub-word unit, while attention masks distinguish meaningful tokens from padding tokens during computation. These representations are then fed into the BERT model to produce contextual embeddings with richer semantic information. The encoded output is represented by the [CLS] token vector, which is considered the overall representation of the sentence meaning, as it is designed to capture global information from the entire input sequence. This vector is then used as the main input for a Feedforward Neural Network, which performs feature transformation before passing the output to the final classification stage using a Softmax function to determine the prompt category in a probabilistic manner [16].

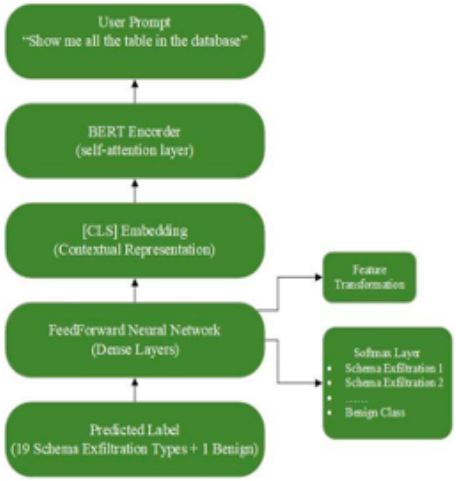

Figure 5. Architecture Diagram of BERT-Based Text Classification Model

During the implementation stage, the model is developed using the Python programming language due to its high flexibility and extensive ecosystem of libraries for artificial intelligence development. The

TensorFlow and PyTorch frameworks are used simultaneously to support the experimental process, particularly in testing model architectures and optimizing performance. Both frameworks enable efficient implementation of transformer-based models such as BERT, both during training and inference stages. In addition, the use of GPU is essential in this phase due to the high computational complexity of transformer models, especially in the attention mechanism calculations that require substantial resources. By utilizing GPU acceleration, the training process can be performed more quickly and stably compared to CPU-only execution, thereby accelerating experimentation and overall model development [17].

The processed dataset is then divided into three main categories: training, validation, and testing data. This separation aims to minimize the risk of overfitting by ensuring that the model does not merely memorize the training data but is also capable of generalizing well to unseen data. The training procedure uses the categorical cross-entropy loss function, which is suitable for multi-class classification problems, along with Adam and AdamW optimizers that operate adaptively to update network weights based on backpropagation results. This combination allows the model to learn complex patterns of schema exfiltration variations in a more stable and effective manner. Through this approach, the model is able to identify patterns of manipulation and semantic relationships in a more robust way [18].

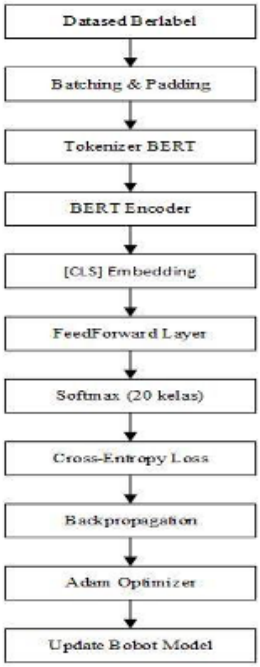

Figure 6. Architecture and training process of BERT model for Classification (20 classes)

The evaluation phase of this study is designed to measure how effectively the model identifies and categorizes schema exfiltration instances within a Text-to-SQL system. To ensure that the model is truly capable of understanding potentially harmful prompt patterns, an objective assessment of model performance after training is conducted. In this context, the evaluation focuses on the model's ability to distinguish between benign prompts and prompts that indicate potential database schema exposure. Therefore, the evaluation stage is crucial to ensure that the model performs well both conceptually and practically [19].

Accuracy, precision, recall, and F1-score are among the key metrics used in the evaluation procedure. Accuracy measures the overall proportion of correct predictions, while precision evaluates how well the model identifies positive classes without producing excessive false positives. Recall assesses the model's ability to capture all actual exfiltration cases, including those that are difficult to detect. In contrast, the F1-score provides a more balanced measure of performance by considering the trade-off between precision and recall. Additionally, a confusion matrix is used to further analyze the distribution of classification errors across each class and to identify error patterns produced by the model [19].

A comparison between rule-based systems and deep learning-based approaches is also conducted to strengthen the analysis of the results. This comparison aims to demonstrate how effectively the BERT-based model handles more complex and unstructured variations in natural language. Rule-based systems generally struggle to interpret flexible language variations and are therefore less capable of handling dynamic prompt manipulation. In contrast, BERT-based models perform better in detecting both implicit and explicit attack patterns due to their ability to capture deeper semantic context, making them more adaptable to linguistic variation [19].

The evaluation findings show that the BERT-based model outperforms traditional methods in recognizing complex attack patterns. Although occasional errors still occur in highly ambiguous or subtle cases, the proposed model achieves a good balance between recall and precision. This demonstrates the significant advantage of deep learning approaches in understanding natural language variations, particularly in Text-to-SQL security systems that require high sensitivity to prompt manipulation. Therefore, compared to traditional methods, the use of BERT significantly improves schema exfiltration detection accuracy [19].

Overall, the research findings indicate that a BERT-based approach supported by neural network architectures is a viable and effective method for detecting schema exfiltration in Text-to-SQL systems. Stable and reliable results are achieved when a strong model architecture, structured data processing, and comprehensive evaluation strategies are combined. Furthermore, this approach holds strong potential for developing more robust AI-based security systems that are resistant to natural language manipulation. Consequently, this study can serve as a foundation for future advancements in improving the security of modern database systems that increasingly rely on artificial intelligence technologies [20].

3. Result and Discussion

A. Dataset Collection

The dataset used in this study is specifically constructed to represent various prompt variations that may trigger schema exfiltration in Text-to-SQL systems. The collected data is not limited to general user instructions but also includes different forms of language manipulation that are commonly used to implicitly or explicitly explore or expose database structures. The dataset consists of two main categories, namely malicious prompts (schema exfiltration) and benign (safe) prompts, allowing the model to learn how to distinguish between normal requests and potentially harmful ones. A total of 1,000 static data samples were used in this study, all of which underwent a rigorous selection and validation process. The validation was conducted by three experts in the field of Information Technology to ensure data quality, labeling consistency, and relevance to the targeted attack scenarios of this research [21].

In accordance with the research scope, the dataset includes 19 schema exfiltration patterns designed to represent various prompt manipulation techniques commonly observed in LLM-based systems. These variations include direct request patterns, where users explicitly request database structure information; hypothetical framing, which presents queries in an imaginative or scenario-based context; intent-to-SQL translation, which encourages the model to explicitly convert natural language into SQL queries; and role flipping, which attempts to manipulate the system's role to elicit more information than allowed. In addition, several other patterns are included to test the model's sensitivity to different linguistic variations. With this diversity of patterns, the dataset is expected to more comprehensively represent real-world conditions that may occur in LLM-based Text-to-SQL applications.

The dataset is focused on single-turn prompts in the Indonesian language, meaning that each data instance consists of only a single input text without any conversational context. The use of single-turn prompts is intended to simplify the analysis while focusing on the model's direct response to a single user input. This is particularly important because, in real-world scenarios, attacks on LLM-based systems can occur through carefully crafted single prompts designed to exploit model weaknesses. By using the Indonesian language, the dataset also contributes significantly to research in a local context, as most existing Text-to-SQL datasets are still dominated by English. Therefore, this dataset is expected to serve as a strong foundation for developing schema exfiltration detection systems that are more relevant and adaptable to Indonesian users.

B. Data Preprocessing

Before the data is used in the model training process, a data preprocessing stage is conducted to ensure that the entire dataset has high quality, consistency, and is ready to be processed by a deep learning-based model. This stage is necessary because raw data collected from various sources typically contains noise, formatting inconsistencies, and variations in writing style that may negatively affect the model's ability to learn language patterns. Therefore, preprocessing becomes a crucial initial step to improve the accuracy and stability of the model in performing schema exfiltration classification. In addition, this stage also helps simplify the data so that it can be more easily processed by the BERT architecture, which is highly sensitive to input structure.

Several steps are carried out during the preprocessing stage, as follows:

1. **Text normalization**, which includes cleaning unnecessary characters such as excessive symbols, irrelevant punctuation marks, and anomalous characters that may interfere with the model's processing. In addition, spacing correction, case standardization, and uniform formatting are applied to ensure that the entire dataset follows a consistent structure. This process is important because small variations in writing style can influence how the model interprets sentence context. With

proper normalization, the data becomes cleaner and easier for the model to learn without interference from irrelevant noise.

2. **Tokenization using the BERT tokenizer**, which is the process of splitting sentences into smaller units called tokens that can be understood by the model. The BERT tokenizer does not only split words but also considers sub-word units, allowing it to handle rare or unseen words that are not part of the model's vocabulary. This process also generates numerical representations of each token, which are then used as input for the deep learning model. Additionally, special tokens such as [CLS] and [SEP] are added to help the model understand the overall sentence structure and the separation between segments.
3. **Conversion of text into model-compatible input format**, including padding and truncation processes. Padding is used to standardize input lengths so that they match the defined batch size, while truncation is applied to cut off text that exceeds the model's maximum input length. This is important because BERT has a fixed input size limit that must be respected to ensure efficient computation. Through this process, all data is transformed into uniform tensor representations ready for model training.

Overall, this stage aims to reduce noise in the dataset and ensure that each input follows a consistent format so that it can be optimally processed by the model [22]. In addition, preprocessing plays an important role in improving training efficiency, as clean and well-structured data accelerates model convergence and reduces the risk of learning errors. Therefore, the final performance of the model is highly influenced by the success of this preprocessing stage.

C. Text Representation (Embedding)

Text representation in this study adopts a transformer encoder-based approach, specifically utilizing the BERT model, which is known for its ability to capture contextual language information more deeply compared to traditional representation methods. Each prompt used as system input is first processed through the BERT tokenizer to be converted into smaller token units that can be understood by the model. These tokens are then mapped into numerical form so they can be processed by the neural network architecture. This process not only converts text into numbers but also preserves word order information and relationships between words within a sentence. As a result, the model is able to understand sentence structure more comprehensively rather than relying solely on individual words [23].

The embeddings produced by the BERT process are contextual vector representations that capture word meaning based on their surrounding context within a sentence [24]. This enables the model to distinguish the meaning of the same word in different contexts and understand semantic relationships between words within a single prompt. This capability is particularly important in schema exfiltration research, as there are subtle differences between benign prompts and prompts that indicate potential database manipulation. The output of this embedding process is then used as the primary input for a feedforward neural network classification layer, which is responsible for determining whether a prompt belongs to a safe category or potentially leads to schema exfiltration. Therefore, embeddings serve as a key component in enabling the system to accurately understand and classify text [9].

D. Model Training Configuration

The model was trained using the BERT architecture combined with a feedforward neural network-based classification layer and a softmax activation function. The training process was conducted over 30 epochs, with the following results:

- Epoch 01: train_loss = 0.3640 | val_loss = 0.2666
- Epoch 10: train_loss = 0.0884 | val_loss = 0.1362
- Epoch 15: train_loss = 0.0594 | val_loss = 0.1270
- Epoch 25: train_loss = 0.0273 | val_loss = 0.1420
- Epoch 30: train_loss = 0.0186 | val_loss = 0.1516

The model successfully learned patterns from the training data, as seen by the steady decrease in the training loss value. On the other hand, the validation loss value increased after reaching its lowest point at epoch 15. This is a sign of mild overfitting, where the model begins to lose its ability to generalize to new data. Therefore, epoch 15 can be considered the best model setting.[2]

E. Model Evaluation

Model evaluation was conducted using accuracy, precision, recall, and F1-score metrics in accordance with the research objectives. The evaluation results show:

- Accuracy: 0.9462
- Precision: 0.8425
- Recall: 0.7483
- F1-score: 0.7926
- Exact Match Accuracy: 0.7596

A high accuracy score means that most of the data is accurately classified by the model. High precision indicates a high degree of accuracy in the positive predictions made by the model. Conversely, a lower recall score indicates that some cases of schema exfiltration are not detected (false negatives). This is a major concern from a system security perspective. Since the F1 score results show a fairly good balance between precision and recall, the model can be considered to have consistent performance. For further investigation, the distribution of prediction errors for each type of prompt variation was examined using a confusion matrix. [25]

The Direct Request Confusion matrix shows True Negative (154), False Positive (10), False Negative (14), and True Positive (30) values. This indicates that the model is quite good at recognizing explicit prompts, but there are still a number of cases of schema exfiltration that are not detected (false negatives)..

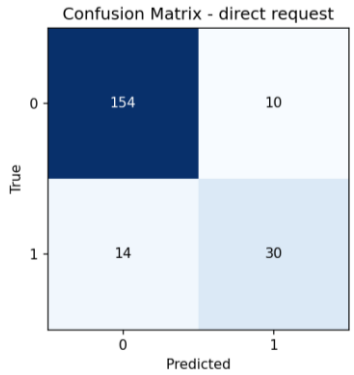

Figure 1. Confusion Matrix – direct request

Hypothetical Framing In this scenario, True Negatives (192), False Positives (7), False Negatives (1), and True Positives (8) were obtained. The model showed very good performance with a minimal number of errors, so it was able to understand hypothesis-based manipulation patterns quite effectively..

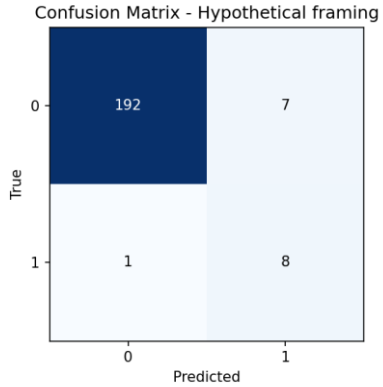

Figure 2. Confusion Matrix – Hypothetical framing

Gambar 1 *Confusion Matrix – Hypothetical framing*

Role Flip / Prompt Injection In this scenario, True Negatives (195), False Positives (1), False Negatives (6), and True Positives (6) were obtained. Although false positives were low, the relatively small number of true positives indicates that the model still has difficulty recognizing complex manipulative patterns.

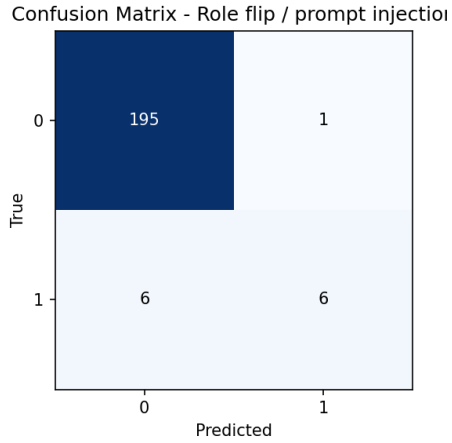

Figure 3. Confusion Matrix – Role flip / prompt injection

Test Fixture / Export Confusion matrix shows True Negative (185), False Positive (1), False Negative (8), and True Positive (14). The model shows quite stable performance, but there are still some errors in detecting dangerous prompts..

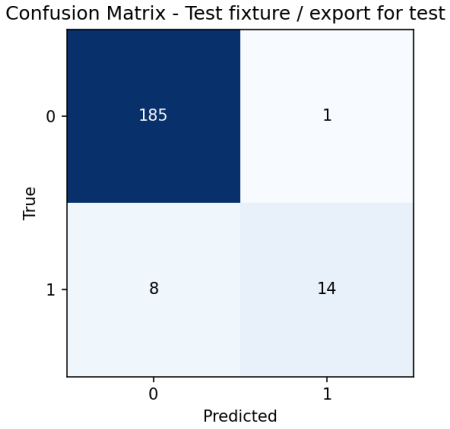

Figure 4. Confusion Matrix – Test fixture / export for test

Overall, the confusion matrix results show that the model is more effective in detecting explicit schema exfiltration compared to implicit or manipulative ones..

4. Conclusion

This study discusses the identification of schema exfiltration in a neural network-based Text-to-SQL system using the BERT architecture as the primary encoder and a feedforward neural network as the classifier. Based on the results, it can be concluded that the deep learning-based model is capable of recognizing various prompt variations that could potentially lead to database schema leaks with a fairly high level of accuracy. This demonstrates the effectiveness of the BERT-based approach in understanding complex natural language contexts, including variations in prompt manipulation used to explore database structures. Furthermore, the use of a dataset validated by experts also significantly improved the quality of model training, resulting in more stable and consistent results in the classification process.

The evaluation results showed that the model achieved good performance with an accuracy of 0.9462, a precision of 0.8425, a recall of 0.7483, and an F1-score of 0.7926. Although the accuracy is considered high, the lower recall indicates that there are still some cases of schema exfiltration that the model has not successfully detected, particularly in implicit or complex patterns such as role flipping and prompt injection. This indicates that while the model performs well in detecting explicit patterns, its generalization ability to more subtle variations in language manipulation still needs improvement. Therefore, further development of both the dataset and model architecture is needed to make the system more sensitive to more complex forms of prompt attacks.

Overall, this research demonstrates that the use of neural network-based BERT can be an effective solution for detecting schema exfiltration in Text-to-SQL systems. However, there is still room for improvement to improve overall detection performance. Future improvements can be made by enriching the dataset, optimizing model fine-tuning techniques, and integrating additional security mechanisms to restrict model output from disclosing sensitive information. Therefore, this research is expected to serve as a foundation for the development of Text-to-SQL systems that are more secure, reliable, and adaptive to prompt manipulation threats in the future, particularly in the context of the increasingly widespread use of Large Language Models.

Acknowledgement

The author expresses his deepest gratitude to God Almighty for His grace, gifts, and guidance, enabling him to complete this research successfully and on time. Without His guidance and assistance, the process of compiling this research would not have proceeded smoothly from beginning to end. The author acknowledges that the process of completing this research faced various challenges, but with spiritual support and maximum effort, all stages were successfully completed. Therefore, this gratitude serves as a primary form of appreciation for the comprehensive completion of this research.

The author also expresses his deepest gratitude to the Informatics Study Program at PGRI Bhineka University in Tulungagung for providing academic facilities, infrastructure, and a supportive learning environment that were extremely helpful throughout the research process. These facilities, including access to literature, laboratories, and other academic support, played a crucial role in supporting the smooth running of this research. Furthermore, the author would like to thank his supervisors for their invaluable guidance, input, and guidance from the planning stage through completion. Each suggestion and correction provided was instrumental in improving the quality of this research, making it more focused and systematic.

The author would also like to express his gratitude to all parties who have provided direct and indirect support throughout the research process. He is especially grateful to his family, whose unwavering prayers, motivation, and moral support have enabled him to remain enthusiastic in completing this research. He also extends his gratitude to his colleagues and fellow students who have provided assistance, discussions, and encouragement throughout the research process. He hopes that the results of this research will provide benefits and positive contributions to the development of science, particularly in the fields of Text-to-SQL system security and artificial intelligence, which continue to develop rapidly in today's digital era.

References

[1] Y. Salim and M. Hasnawi, “Konversi Bahasa Indonesia ke Perintah Data Manipulation Language pada Structured Query Language menggunakan Natural Language Processing,” *Bul. Sist. Inf. dan Teknol. Islam*, vol. 3, no. 3, pp. 181–187, 2022.

[2] M. N. Gifari and A. Prasetya, “Pendekatan Berbasis Rule Untuk Mengidentifikasi Pertanyaan Tak Terdefinisi Pada Masalah Text-to-SQL,” *J. Borneo Inform. Tek. Komput.*, vol. 5, no. 1, pp. 21–25, 2025.

[3] Y. Huang *et al.*, “Exploring the Landscape of Text-to-SQL with Large Language Models: Progresses, Challenges and Opportunities,” vol. 37, no. 4, 2024.

[4] R. Sylvia, “Penggunaan AI Berbasis Large Language Models (LLM) sebagai Media Interaktif dalam Pendidikan Bahasa dan Hukum di Perguruan Tinggi,” *Disiplin Maj. Civ. Akad. Sekol. Tinggi Ilmu Huk. Sumpah Pemuda*, vol. 31, no. 4, pp. 233–242, 2025, [Online]. Available: https://ojs.stihpada.ac.id/index.php/disiplin

[5] R. Ahadi, N. S. Harahap, M. Fikry, and F. Kurnia, “Retrieval-Augmented Generation in a Web-Based Question Answering System for Fiqh Books,” *J. Artif. Intell. Softw. Eng.*, vol. 5, no. 2, pp. 626–635, 2025, doi: 10.30811/jaise.v5i2.7005.

[6] A. Mohammadjafari, A. S. Maida, and R. Gottumukkala, “From Natural Language to SQL : Review of LLM-based Text-to-SQL Systems,” 2025.

[7] Y. Xie *et al.*, “Decomposition for Enhancing Attention : Improving LLM-based Text-to-SQL through

Workflow Paradigm," pp. 10796–10816, 2024.
[8] T. Zhang, C. Chen, J. Wang, and J. Wang, "SQLfuse : Enhancing Text-to-SQL Performance through Comprehensive LLM Synergy," 2024.
[9] D. Gao *et al.*, "Text-to-SQL Empowered by Large Language Models : A Benchmark Evaluation," 2023, doi: 10.14778/3641204.3641221.
[10] Q. Liu, M. J. Kusner, and P. Blunsom, "A Survey on Contextual Embeddings," 2020.
[11] K. S. Kalyan, A. Rajasekharan, and S. Sangeetha, "AMMUS : A Survey of Transformer-based Pretrained Models in Natural Language Processing," pp. 1–42, 2021.
[12] H. Zhang, R. Cao, H. Xu, L. Chen, and K. Yu, "CoE-SQL: In-Context Learning for Multi-Turn Text-to-SQL with Chain-of-Editions," vol. 1, pp. 6487–6508, 2024.
[13] A. Usta, A. Karakayali, and O. Ulusoy, "xDBTagger : Explainable Natural Language Interface to Databases Using Keyword Mappings and Schema Graph," 2022.
[14] Y. Song, R. Liu, S. Chen, Q. Ren, Y. Zhang, and Y. Yu, "SecureSQL : Evaluating Data Leakage of Large Language Models as Natural Language Interfaces to Databases," *Find. ofthe Assoc. Comput. Linguist. EMNLP*, vol. 12, no. 16, pp. 5975–5990, 2024, doi: https://doi.org/10.18653/v1/2024.findings-emnlp.346.
[15] K. Greshake, C. Endres, S. Abdelnabi, S. Mishra, T. Holz, and M. Fritz, *Not what you ' ve signed up for : Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection*, vol. 1, no. 1. Association for Computing Machinery, 2023.
[16] M. Lin, H. Zhang, J. Lao, R. Li, Y. Zhou, and C. Yang, *Are Your LLM-based Text-to-SQL Models Secure ? Exploring SQL Injection via Backdoor Attacks*, vol. 1, no. 1. arXiv, 2025.
[17] L. Shi, Z. Tang, N. A. N. Zhang, X. Zhang, and Z. H. I. Yang, "A Survey on Employing Large Language Models for Text-to-SQL Tasks," *ACM Comput. Surv*, vol. 58, no. 2, 2025, doi: https://doi.org/10.1145/3737873.
[18] A. Fatwanto, F. Zamakhsyari, R. Ndungi, and L. Fitriyani, "R ESEARCH A RTICLE A Systematic Literature Review of BERT-based Models for Natural Language Processing Tasks," *J. INFOTEL*, vol. 16, no. 4, pp. 713–728, 2024, doi: 10.20895/INFOTEL.V16I3.1206.
[19] A. Ayub and S. Majumdar, "Embedding-based classifiers can detect prompt injection attacks," vol. 0110, pp. 0–2, 2024.
[20] A. Marshan, A. N. Almutairi, A. Ioannou, D. Bell, A. Monaghan, and M. Arzoky, "MedT SQL : a transformers-based large language model for text-to-SQL conversion in the healthcare domain," *Front. Big Data*, 2024, doi: 10.3389/fdata.2024.1371680.
[21] Y. Zheng, H. Wang, B. Dong, X. Wang, and C. Li, "HIE-SQL: History Information Enhanced Network for Context-Dependent Text-to-SQL Semantic Parsing," 2021.
[22] M. Biesialska, K. Biesialska, and H. Rybinski, "Leveraging contextual embeddings and self-attention neural networks with bi-attention for sentiment analysis," *J. Intell. Inf. Syst.*, 2021, doi: https://doi.org/10.1007/s10844-021-00664-7.
[23] Đ. ¯de Klisura and A. Rios, "Unmasking Database Vulnerabilities : Zero-Knowledge Schema Inference Attacks in Text-to-SQL Systems," pp. 6969–6991, 2025, doi: https://doi.org/10.18653/v1/2025.findings-naacl.386.
[24] B. E. S. Dewi, "Pengukuran Kemiripan Kalimat Bahasa Indonesia Menggunakan Representasi Word Embedding Fasttext," *Teknol. Inform. Komput.*, vol. 2, no. 2, pp. 20–29, 2025.
[25] M. Alizadeh, Z. Samei, D. Stetsenko, and F. Gilardi, "Simple Prompt Injection Attacks Can Leak Personal Data Observed by LLM Agents During Task Execution," pp. 1–25, 2025.