





# Utilising Large Language Models for Adversarial Attacks in Text-to-SQL: A Perpetrator and Victim Approach

Ariana Sahitaj <sup>1</sup>, Markus Nilles <sup>2</sup>, Ralf Schenkel <sup>2</sup>, and Vera Schmitt <sup>1</sup>



## Abstract:



This paper investigates the use of Large Language Models (LLMs) for the Text-to-SQL task, both as Perpetrator models for generating adversarial attacks and as Victim models for assessing their robustness. In this study, two state-of-the-art LLMs, Llama3 with 70 billion and Mixtral with 47 billion parameters, were employed as Perpetrators to generate adversarial examples at the character-, word-, and sentence-level. A total of 77,292 adversarial examples were generated from 2,147 data points of the Spider test-set using three additional LLMs as Victims and evaluated thoroughly. These Victim models are based on Llama3 with 8 billion parameters and differ only in the extent of fine-tuning for related benchmark tasks. The results show that attacks at the word-level, particularly through synonym replacements, most significantly impair model performance. Additionally, providing database schemas significantly improves execution accuracy, while fine-tuning does not always enhance robustness against adversarial attacks. This work provides important insights into improving the reliability of Text-to-SQL models in future applications and makes a significant contribution to the further development of these models in research.

**Keywords:** Large Language Models, Adversarial Attacks, Text-to-SQL Models

## 1 Introduction

Natural Language Processing (NLP) has significantly advanced in recent years, with the introduction of the Transformer architecture by Vaswani et al. [Va17] in 2017. This innovation enabled the training of Large Language Models (LLMs), which have enhanced our ability to understand and generate human language. A milestone was reached with the launch of ChatGPT, which became the fastest-growing consumer application in history, reaching 100 million active monthly users within just two months of its launch [Ti23]. It has since transformed how we interact with AI-powered systems, as models like ChatGPT demonstrate the potential of natural language to serve as the next programming language, as noted by NVIDIA's CEO Jensen Huang [Mi23, BA24]. Building on this paradigm shift, one of the most important applications of LLMs in database settings is the development of models that connect natural language (NL) with Structured Query Language (SQL). Text-to-SQL models translate NL questions into SQL queries, allowing users without SQL

<sup>1</sup> Technische Universität Berlin, Quality and Usability Lab, Marchstraße 23, 10587 Berlin, Germany, ariana.sahitaj@campus.tu-berlin.de,  <https://orcid.org/0009-0002-0096-9383>; vera.schmitt@tu-berlin.de,  <https://orcid.org/0000-0002-9735-6956>

<sup>2</sup> Universität Trier, Datenbanken und Informationssysteme, Universitätsring 15, 54296 Trier, Germany, nillesm@uni-trier.de,  <https://orcid.org/0000-0002-3449-9319>; schenkel@uni-trier.de,  <https://orcid.org/0000-0001-5379-5191>

experience to easily interact with databases, making data access much more accessible to a wider audience [Li24]. Despite their potential, these models still face considerable challenges when handling real-world user input that deviates from expected formats. For example, a non-expert user might request: *Please give me a list of all the **customrs** who bought laptops in the last month*, introducing a typographical error (*customrs* instead of *customers*). Another user might say: *Show me all buyers of **notebooks** from the previous month*, using *notebooks* a synonym for *laptops*. Others may include additional, potentially irrelevant, information such as *I need to send thank-you emails, so please provide a list of all customers who purchased laptops recently*. Perturbations such as typographical errors, synonyms, or extraneous information often result in failed or inaccurate SQL queries, exposing vulnerabilities in Text-to-SQL models that can be exploited as adversarial attacks [Ku24] and underscore the need for more robust models [ZWF23, Ho24]. By addressing these vulnerabilities, this work aims to improve the real-world applicability of Text-to-SQL systems, making database interactions more robust and user-friendly. Previous research has primarily relied on *human-annotated adversarial examples* to study these vulnerabilities, a process that is both time-consuming and difficult to scale [Ga21]. It is therefore important to investigate the potential of automated methods for generating adversarial examples to better understand which types of input variations affect the performance of Text-to-SQL models [Pi22, Ku24] most. In this paper, we examine two roles of LLMs, as Perpetrators generating adversarial examples and as Victims tested for their robustness in Text-to-SQL tasks. We investigate how different types of adversarial attacks generated by LLMs impact the performance of Text-to-SQL models and assess the robustness of selected models under these conditions. Based on our findings, we provide the following contributions:

1. We develop an approach to generate adversarial attacks using LLMs, focusing on different perturbation granularities such as character-level, word-level, and sentence-level.
2. We provide a benchmark comparison of Text-to-SQL model performance under non-adversarial and adversarial conditions, highlighting model vulnerabilities and demonstrating the impact of contextual information, such as database schema, on performance.
3. We analyse the performance of Text-to-SQL models across different query difficulties.
4. We evaluate the robustness of models fine-tuned on SQL compared to those that are not, against LLM-generated adversarial attacks.
5. We identify which types of adversarial attacks generated by LLMs most affect the robustness of Text-to-SQL models.

In this paper, we first review related work on adversarial attacks in Text-to-SQL models. We then introduce our methodology for generating adversarial examples and evaluating model robustness, followed by presenting the experimental setup and the results. Finally, we conclude with insights from our findings and discuss directions for future research.

## 2 Related Work

Adversarial attacks have been an active area of research in NLP, exposing vulnerabilities in state-of-the-art models across various tasks. These attacks involve small perturbations of a given input, which mislead the LLM, while maintaining the human interpretability of the input [RS23]. Alzantot et al. [A118] introduced a black-box algorithm to generate adversarial examples by word substitution, revealing the limitations of adversarial training. Zhao et al. [ZDS17] and Jia et al. [JL17] demonstrated that semantic and syntactic manipulations can mislead advanced models, highlighting the challenges in achieving robustness. In the Text-to-SQL domain, Zhang et al. [ZWF23] evaluated the robustness of LLMs against adversarial inputs using datasets like Spider-Syn [Ga21] (synonym replacements), Spider-Realistic [De20] (schema omission), and Dr. Spider [Ch23] (multiple perturbation types). They also introduced ADVT and ADVS, datasets focusing on typographical errors and stylistic changes, respectively, to assess model vulnerabilities. While prior studies predominantly rely on manually created adversarial examples or focus on specific perturbation types, this work introduces a fully automated framework using LLMs as both, Perpetrators and Victims. Perpetrator LLMs generate adversarial examples across multiple granularity levels such as on character-, word-, and sentence-level and are designed to mimic real-life perturbations to test the robustness of Text-to-SQL Victim models. These adversarial text queries are then evaluated using Victim LLMs specifically fine-tuned and non-fine-tuned on SQL, systematically analysing the influence of database schema context and identifying which attack types have the greatest impact on model robustness.

## 3 Approach

We propose a two-step framework to evaluate the robustness of Text-to-SQL models against adversarial attacks. This involves **(1) the generation of adversarial text queries** using a *Perpetrator LLM* and **(2) the evaluation of the robustness of Text-to-SQL models**, which serve as *Victim LLMs*, against these queries.

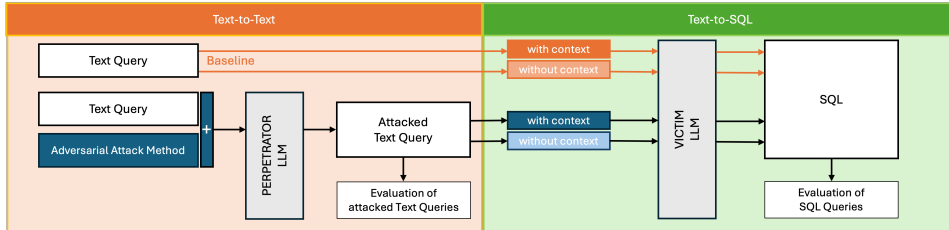


Fig. 1: The Text-to-Text pipeline is responsible for the generation of adversarial text queries, while the Text-to-SQL pipeline focuses on the generation of SQL queries and evaluates how robust the Victim models are in response to those adversarial inputs. The detailed pipeline is outlined in the following GitHub repository [Sa24, A.2] for further reference.

As shown in **Figure 1**, the framework begins with the Perpetrator LLM, which generates adversarial text queries by applying one of the *adversarial attack methods* described below to natural language text queries. These attack methods introduce perturbations designed to mimic realistic adversarial settings, creating a diverse set of challenges for the Victim LLM. The perturbed queries are then passed to the Victim LLM, which processes both original (unaltered) and adversarial (perturbed) queries to generate SQL outputs. To evaluate the influence of contextual information the Victim LLM is tested in two configurations: with access to the database schema (*with context*) and (*without context*). The framework operates in two workflows. The **baseline workflow** (orange path in Figure 1) evaluates the performance of Victim LLMs on unaltered text queries to establish a benchmark. The **adversarial workflow** (black path) measures its robustness against adversarially perturbed queries. By comparing the outputs from these workflows the impact of adversarial attacks can be quantified, and specific vulnerabilities in the Text-to-SQL capabilities of Victim LLMs can be identified.

### 3.1 Dataset

This paper makes use of the Spider dataset, a benchmark for testing Text-to-SQL models. The dataset includes over 10,000 examples from 200 databases across 138 different topics, pairing natural language text queries with their corresponding SQL queries and database schema. For this paper, the test split of the dataset was used, comprising 2,147 examples categorised into four difficulty levels: easy, medium, hard, and extra hard. [Yu18] The distribution of these difficulty levels is as follows: 21.95% of the examples are labeled as easy, 39.9% as medium, 21.6% as hard, and 16.6% as extra hard, enabling a thorough assessment of model performance across varying query complexities.

### 3.2 Generation of Adversarial Text Queries

To generate adversarial queries, we use Llama3 and Mixtral8x7B as Perpetrator LLMs, two large-scale models with 70 billion and 47 billion parameters, respectively. For both models, we provide the 2,147 text queries from the Spider test-split, along with few-shot examples for each level of granularity. These levels include character-, word-, and sentence-level adversarial attacks. For all levels of granularity, we adhere to specific constraints to maintain the integrity and realism of the adversarial text queries. These constraints, inspired by the findings of Gan et al. [Ga21], include reserved keywords such as `id`, `age`, `name`, and `year`, which were also used as protected keywords in the Spider-Syn dataset, as they are critical for query interpretation. Additional rules were established based on insights gained during our experiments, such that `proper names`, `company names`, and `specific places` must remain unchanged, and any text within quotation marks (`'text'` or `"text"`) should be left unaltered.

**Character-level attacks** involve introducing typographical errors while preserving the original structure and intent of the query. This simulates typical human typing errors and is

intended to test the robustness of Text-to-SQL models under such conditions. Furthermore, for character-level attacks, no parts of the query are randomly modified beyond typographical changes, ensuring that the original meaning of the query is preserved. Specific rules from Yukino et al. [BS12] were adopted for this purpose, which were included as part of the prompt design. These rules can be reviewed in detail in the GitHub repository, available at [Sa24, A.3].

**Word-level attacks** focus on identifying specific keywords in the query that can be replaced with synonyms to introduce ambiguity or confusion, simulating scenarios where non-expert users are unfamiliar with the database schema [Ga21]. For this level of attack, the goal is to maintain the semantic meaning of the query while introducing variations that apply the same rules about preserving reserved keywords, proper names, and entities.

**Sentence-level attacks** take a different approach by appending an additional sentence to the original query. This sentence is contextually relevant but introduces irrelevant information meant to mislead the model. This method aims to overload the Victim models with extra information that could cause incorrect SQL generation. [ZWF23]

To facilitate the generation of adversarial text queries, we set up a prompting framework that includes a persona, a task description, and few-shot examples, and instructions, as further detailed in the GitHub repository [Sa24]. These examples are drawn from the Spider dataset, specifically representing four different difficulty levels. The instructions within the prompt are designed to ensure that the LLMs follow the predefined constraints, such as maintaining the integrity of reserved keywords and entities. For each level of granularity, the same few-shot examples of different text query difficulties are used to provide consistency across all adversarial attack types.

### 3.3 Evaluation of Adversarial Text Queries

The objective of the evaluation is to assess the effectiveness and impact of adversarial queries on the performance of Text-to-SQL models. We conduct quantitative and qualitative analyses to examine the robustness and vulnerabilities of these models when exposed to adversarial attacks. For the **quantitative analysis**, we use two key metrics, referred to as *performance indicators*: At character-level, we use the Damerau-Levenshtein distance to evaluate the extent of modifications made by the LLMs, inspired by the WikiSQL dataset, which employed character-level edit distance to ensure significant paraphrase variations [ZXS17]. This metric calculates the minimum of operations, insertions, deletions, substitutions, and transpositions, required to transform the original query into the adversarial query [Lo24]. Additionally, we introduce the Unchanged Damerau-Levenshtein Percentage, which represents the proportion of the original query that remains unchanged after the adversarial attacks. This metric is particularly useful to understand how much of the original text has been preserved, which is important when analysing the impact of subtle modifications. It is especially useful in shorter sentences, where even minor changes can significantly alter the

meaning. To compute the Damerau-Levenshtein distance and the corresponding percentage, we utilise the `pyxDamerauLevenshtein` library [Lo24], a Python package optimised for fast and efficient calculations of this metric. For the word- and sentence-levels, we focus on semantic changes rather than purely textual changes using *cosine similarity*. This metric measures the cosine of the angle between two vectors in a high-dimensional space, capturing the semantic similarity between original and adversarial queries. By employing the `sentence-transformers` library, specifically the `paraphrase-MiniLM-L6-v2` model [Fa], we can generate embeddings for both the original and adversarial queries. The embeddings are then compared to calculate the cosine similarity. In addition to these quantitative evaluations, we apply a paired t-test to determine whether there is a statistically significant difference between the adversarial queries generated by Llama3 and Mixtral, using the `scipy.stats.ttest_rel` function from the SciPy library[Co]. Since both sets of queries are derived from the same original text queries, the paired t-test accounts for the inherent correlation between the two groups. This allows us to assess whether the performance differences between the two models are significant, guiding our decision on whether to continue with both model outputs or focus on one. For the **qualitative analysis**, we manually examine the worst-performing examples from each model, focusing on the ten examples with the lowest Unchanged Damerau-Levenshtein Percentages, and cosine similarity scores, resulting in 60 instances to examine. This qualitative approach allows us to identify specific patterns in the types of errors introduced by the adversarial attacks and provides deeper insights into the vulnerabilities of the models. By focusing on these outliers, we can evaluate whether the adversarial queries generated are effective in misleading the models and whether they maintain the intended attack structure.

### 3.4 Generation of SQL Queries

In the Text-to-SQL workflow, the adversarial queries generated by Llama3 and Mixtral8x7B are passed to three Victim models based on the Llama3 architecture, each with 8 billion parameters, to ensure comparability and computational efficiency across the models.

- **Llama-base**<sup>3</sup>: A general-purpose language model without specific fine-tuning for SQL tasks.
- **Llama-spider**<sup>4</sup>: Fine-tuned specifically for SQL generation using the SQL Create Context<sup>5</sup> dataset, which is derived from Spider [Yu18] and WikiSQL [Xu18] dataset.
- **Llama-spider-plus**<sup>6</sup>: Extends llama-spider by not only being fine-tuned on the SQL Create Context dataset but also on additional datasets like bagel, which includes both SQL and Python coding tasks (e.g., from Rosetta Code) [Du23b, Du23a].

<sup>3</sup> meta-llama/Meta-Llama-3-8B-Instruct

<sup>4</sup> artificialguybr/llama3-8b-sql-create-context

<sup>5</sup> <https://huggingface.co/datasets/b-mc2/sql-create-context>

<sup>6</sup> jondurbin/bagel-8b-v1.0

The adversarial text queries generated by the Perpetrator LLMs are processed by these Victim LLMs to generate SQL queries, with two different settings: with context, where schema information is provided, and without context, where the models rely solely on the natural language input. This dual evaluation helps to measure the robustness of the models under varying conditions.

### 3.5 Evaluation of SQL Queries

For the evaluation of the SQL queries generated by the Text-to-SQL models, we use the Official Spider Test Suite<sup>7</sup>, which requires SQL queries to be in a standardised single-line format. However, the queries generated by the Victim LLMs often contain formatting issues due to different training, such as unnecessary quotation marks, line breaks, and inconsistent white spaces, as shown in the prompt in [Sa24, A.3], making them unsuitable for direct evaluation within the Test Suite. To address this, we apply a pre-processing step using the Llama3 70B model<sup>8</sup> to standardise the format of both the adversarial and non-adversarial queries. This pre-processing is essential, given the large volume of SQL queries: 77,292 adversarial queries generated from the adversarial text queries, and 12,882 non-adversarial queries from the baseline, resulting in a total of 90,174 queries. By applying Llama3, we ensure the queries conform to the required format for evaluation. After the pre-processing, we manually evaluate 180 SQL queries, selecting 5 cases per model output, to confirm the effectiveness of the formatting corrections while maintaining the integrity of the SQL logic. Finally, we assess the SQL queries using Execution Accuracy (EA), which evaluates whether the queries, once executed, return the correct results. This metric ensures that the models are generating functionally accurate SQL queries, regardless of their syntactic form.

## 4 Experiments

This section describes the experiments conducted to evaluate the impact of adversarial attacks on Text-to-SQL models. It outlines the experimental setup, including tools and configurations, and presents the results to analyse model performance across different difficulty levels of the Spider dataset.

### 4.1 Experimental Setup

The experiments were conducted using Python 3.10 for better compatibility with the latest libraries, leveraging Poetry for dependency management and virtual environment configuration. This approach not only simplifies the installation of necessary packages

<sup>7</sup> <https://github.com/taoyds/test-suite-sql-eval>

<sup>8</sup> meta-llama/Meta-Llama-3-70B-Instruct

but also helps in maintaining version consistency [Py24a]. The infrastructure included two NVIDIA A100 GPUs to handle the computational demands of large-scale models. The implementation utilised the vLLM library for efficient and scalable model inference of large language models [Su24], while torch was used as framework to run the models [Py24b]. Additional libraries like pandas [De24], spacy [Ex24], sqlparse [Pa24], and nltk [BL24] are used for data manipulation and processing tasks. For the **Text-to-Text** task, the adversarial queries are generated by Llama3 (70B) and Mixtral (47B), with models accessed via the Hugging Face transformers library. The settings ensure diversity in the generated queries, with the temperature set to 1.0 and top-p to 1.0, enabling creativity and variation. The sampling parameters are further refined with no constraints on top-k (-1), allowing full exploration of the vocabulary. The model context is limited to 2048 tokens, with outputs limited to 512 tokens to ensure memory efficiency. Each adversarial query is generated using one-shot examples representing all four difficulty levels of the Spider dataset. The queries are presented in a randomised order using a consistent seed (239) to avoid influence from batched generation. For the **Text-to-SQL** task, three Victim models (llama-base, llama-spider, and llama-spider-plus) with 8B parameters each were employed. Unlike the Text-to-Text task, deterministic settings were applied, with a temperature of 0, to ensure consistency and reproducibility of SQL query outputs, as proposed by Zhang et al. [ZWF23]. The input data was merged into a Pandas DataFrame, which was subsequently shuffled to prevent biases. The configurations and prompts for each run were logged for reproducibility, and formatted SQL queries were evaluated using the Spider Test Suite. This setup ensured a robust and transparent workflow for assessing adversarial attack impact and model robustness.

## 4.2 Results

In the **quantitative analysis**, character-level attacks caused significant modifications in the queries. Both Llama3 and Mixtral8x7B achieved similar Damerau-Levenshtein Distances, with Mixtral8x7B introducing slightly more aggressive character-level perturbations. Word-level and sentence-level attacks showed a clear drop in cosine similarity scores, with values ranging between 0.40 and 0.50 for word-level attacks and below 0.45 for sentence-level attacks across all models, indicating a substantial semantic divergence. For a detailed distribution of the attack metrics, refer to [Sa24, A.1]. The paired t-test confirmed statistically significant differences ( $p < 0.05$ ) between the adversarial outputs of Llama3 and Mixtral8x7B, demonstrating that both Perpetrator LLMs generate distinct types of adversarial examples. In the **qualitative analysis**, manual inspection revealed that character-level attacks retained query intent while exposing vulnerabilities to typographical errors. Word-level attacks were particularly effective at introducing misleading synonyms (e.g., "notebooks" for "laptops"), while sentence-level attacks overwhelmed models with extraneous but contextually plausible information. These findings validate the effectiveness of the adversarial queries in challenging the Victim models. The paired t-test results also highlight the complementary nature of Llama3 and Mixtral8x7B adversarial outputs. As a result, both outputs will be used in



subsequent experiments to ensure a comprehensive evaluation of Victim models against a diverse range of adversarial attacks.

Tab. 1: Performance comparison of the llama-base, llama-spider, and llama-spider-plus Victim models, including baseline results and filtered EA for **word-level attacks with context**. Yellow rows indicate baseline performance with context, while orange rows show performance without context. The llama-spider-plus model with context achieves the highest baseline EA.

	easy	medium	hard	extra	all
count	470	857	463	357	2147
llama-base	0.745	0.691	0.495	0.429	0.617
llama-base w/o context	0.157	0.072	0.032	0.034	0.076
llama-base w/ Llama3	0.621	0.608	0.449	0.392	0.541
llama-base w/ Mixtral8x7B	0.577	0.546	0.410	0.403	0.500
llama-spider	0.574	0.510	0.501	0.440	0.510
llama-spider w/o context	0.217	0.107	0.056	0.020	0.106
llama-spider w/ Llama3	0.451	0.392	0.410	0.350	0.402
llama-spider w/ Mixtral8x7B	0.451	0.345	0.378	0.364	0.379
<b>llama-spider plus</b>	<b>0.751</b>	<b>0.660</b>	<b>0.499</b>	<b>0.513</b>	<b>0.621</b>
llama-spider plus w/o context	0.179	0.084	0.026	0.036	0.084
llama-spider-plus w/ Llama3	0.598	0.561	0.447	0.462	0.528
llama-spider-plus w/ Mixtral8x7B	0.596	0.515	0.419	0.440	0.499

As shown in Table 1 the llama-spider-plus model with context achieved the highest overall baseline execution accuracy (0.621), highlighted yellow and bold. Models without context, highlighted orange, performed significantly worse, with llama-base achieving an overall accuracy of only 0.076. The results underscore the importance of providing database schema context for improving baseline performance.

Tab. 2: EA of Victim models under adversarial attacks generated by Perpetrator LLMs (Llama3 and Mixtral8x7B) across different granularities (character-, word-, and sentence-level) aggregated over *all* difficulty levels.

Victim-Model	Character-Level		Word-Level		Sentence-Level	
	Llama3	Mixtral8x7B	Llama3	Mixtral8x7B	Llama3	Mixtral8x7B
llama-base	0.589	0.560	0.541	0.500	0.570	0.599
llama-spider	0.454	0.445	0.402	0.379	0.505	0.535
llama-spider-plus	0.595	0.574	0.533	0.499	0.528	0.588

Adversarial attacks significantly reduced execution accuracy across all models, as shown in Table 2. Among the Victim models, llama-spider-plus consistently demonstrated the highest execution accuracy, particularly under character-level attacks, with scores of 0.595 and 0.574 for Llama3 and Mixtral8x7B, respectively. Word-level attacks had the strongest negative impact on all models, reducing the accuracy of llama-base and llama-spider to as low as 0.500 and 0.379. Sentence-level attacks showed mixed results, with llama-base performing slightly better than the fine-tuned models under Mixtral8x7B. These findings highlight

the varying levels of robustness across models and granularities. As word-level attacks had the strongest impact on the robustness of Text-to-SQL Victim models, Table 1 also focuses specifically on the filtered results for word-level adversarial queries with context. For a detailed overview of all results, including the hardness levels of the queries, refer to [Sa24, A.1]. The results of this study highlight several important findings. Both Llama3 and Mixtral8x7B successfully generated distinct and effective adversarial attacks, validating the use of both models for comprehensive evaluation. Fine-tuned Victim models, particularly llama-spider-plus with context, demonstrated significantly better baseline execution accuracy, with the highest performance reaching 0.621. Word-level attacks were found to have the most severe impact on execution accuracy, underscoring their importance in testing model robustness. Furthermore, the presence of schema context consistently improved model performance, both under baseline and adversarial conditions, emphasising its important role in enhancing robustness.

## **5 Conclusion and Future Work**

This study investigated the use of LLMs to generate adversarial attacks and assess the robustness of Text-to-SQL models. Word-level attacks, particularly synonym replacements, had the strongest negative impact on performance, and robustness consistently declined with increasing query difficulty. Fine-tuned models, such as llama-spider-plus, generally outperformed their non-fine-tuned counterparts. However, unexpected results, such as the solid performance of the non-fine-tuned llama-base in specific settings compared to the poor performance of fine-tuned llama-spider, highlight that fine-tuning alone does not ensure success and may even introduce limitations. Key findings highlight the importance of self-conducted fine-tuning to better align models with specific datasets and requirements. Future work should broaden the scope of adversarial attacks beyond the granularities explored here, enabling the models to handle more diverse and complex inputs effectively. Leveraging LLMs with more parameters can also enhance robustness to subtle adversarial manipulations. This framework addresses important aspects of Text-to-SQL robustness, demonstrating the impact of schema context, fine-tuning strategies, and varying query difficulties. One promising mitigation strategy is the incorporation of dynamic schema adaptation, which, despite challenges like scalability and maintenance, could enhance robustness against adversarial inputs [MMG24]. Future research could expand on this by testing the proposed framework in real-world scenarios to validate its effectiveness and further improve strategies for handling diverse inputs and adversarial attacks, thereby contributing to the broader applicability and reliability of Text-to-SQL systems.

## **6 Acknowledgments**

This research is funded by the Federal Ministry of Education and Research (BMBF, reference: 03RU2U151C) in the scope of the research project news-polygraph.

## Bibliography

- [Al18] Alzantot, Moustafa; Sharma, Yash; Elgohary, Ahmed; Ho, Bo-Jhang; Srivastava, Mani; Chang, Kai-Wei: Generating natural language adversarial examples. arXiv preprint arXiv:1804.07998, 2018.
- [BA24] Babu, CV Suresh; Akshara, PM: Revolutionizing conversational AI: unleashing the power of ChatGPT-Based applications in generative AI and natural language processing. In: Advanced applications of generative AI and natural language processing models, pp. 228–248. IGI Global, 2024.
- [BL24] Bird, Steven; Loper, Edward: Natural Language Toolkit (NLTK), 2024. <https://pypi.org/project/nltk/>, Accessed: 2024-08-16.
- [BS12] Baba, Yukino; Suzuki, Hisami: How are spelling errors generated and corrected? a study of corrected and uncorrected spelling errors using keystroke logs. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2. ACL '12, Association for Computational Linguistics, USA, p. 373–377, 2012.
- [Ch23] Chang, Shuaichen; Wang, Jun; Dong, Mingwen; Pan, Lin; Zhu, Henghui; Li, Alexander Hanbo; Lan, Wuwei; Zhang, Sheng; Jiang, Jiarong; Lilien, Joseph et al.: Dr. spider: A diagnostic evaluation benchmark towards text-to-sql robustness. arXiv preprint arXiv:2301.08881, 2023.
- [Co] Community, SciPy: scipy.stats.ttest\_rel. [https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest\\_rel.html](https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_rel.html). Accessed: 2024-12-02.
- [De20] Deng, Xiang; Awadallah, Ahmed Hassan; Meek, Christopher; Polozov, Oleksandr; Sun, Huan; Richardson, Matthew: Structure-grounded pretraining for text-to-SQL. arXiv preprint arXiv:2010.12773, 2020.
- [De24] DevelopmentTeam, Pandas: pandas, 2024. <https://pypi.org/project/pandas/>, Accessed: 2024-08-16.
- [Du23a] Durbin, Jon: Bagel-8B-v1.0: LLaMA-based Model for SQL Generation, 2023. Hugging Face Model Repository. Bagel-8B-v1.0 is a fine-tuned version of the LLaMA-3 architecture, optimized for text-to-SQL tasks. The model leverages datasets such as Spider and WikiSQL, along with custom data, to improve its performance in generating SQL queries from natural language inputs.
- [Du23b] Durbin, Jon: Bagel: LLaMA-based 8B Parameter Model for SQL Generation, 2023. GitHub repository. Bagel is a fine-tuned version of LLaMA-3, specifically designed for SQL generation tasks. The model builds on LLaMA's architecture and was trained using a combination of Spider, WikiSQL, and custom datasets to enhance its performance on text-to-SQL tasks.
- [Ex24] ExplosionAI: Python package spaCy, 2024. <https://pypi.org/project/spacy/>, Accessed: 2024-08-16.
- [Fa] Face, Hugging: sentence-transformers/paraphrase-MiniLM-L6-v2. <https://huggingface.co/sentence-transformers/paraphrase-MiniLM-L6-v2>. Accessed: 2024-12-02.
- [Ga21] Gan, Yujian; Chen, Xinyun; Huang, Qiuping; Purver, Matthew; Woodward, John R; Xie, Jinxia; Huang, Pengsheng: Towards robustness of text-to-SQL models against synonym substitution. arXiv preprint arXiv:2106.01065, 2021.

- [Ho24] Hong, Zijin; Yuan, Zheng; Chen, Hao; Zhang, Qinggang; Huang, Feiran; Huang, Xiao: Knowledge-to-sql: Enhancing sql generation with data expert llm. arXiv preprint arXiv:2402.11517, 2024.
- [JL17] Jia, Robin; Liang, Percy: Adversarial examples for evaluating reading comprehension systems. arXiv preprint arXiv:1707.07328, 2017.
- [Ku24] Kumar, Pranjal: Adversarial attacks and defenses for large language models (LLMs): methods, frameworks & challenges. International Journal of Multimedia Information Retrieval, 13(3):26, 2024.
- [Li24] Liu, Xinyu; Shen, Shuyu; Li, Boyan; Ma, Peixian; Jiang, Runzhi; Luo, Yuyu; Zhang, Yuxin; Fan, Ju; Li, Guoliang; Tang, Nan: A Survey of NL2SQL with Large Language Models: Where are we, and where are we going? arXiv preprint arXiv:2408.05109, 2024.
- [Lo24] Los Alamos National Laboratory: pyxDamerauLevenshtein, 2024. <https://github.com/lanl/pyxDamerauLevenshtein>, Accessed: 2024-08-08.
- [Mi23] Mishra, Zcuyc: Nvidia CEO's Vision of an English Universal Coding Language in the Age of AI, 2023. <https://www.linkedin.com/pulse/nvidia-ceos-vision-english-universal-coding-language-age-mishra-zcuyc/>, Accessed: 2024-08-15.
- [MMG24] Mohammadjafari, Ali; Maida, Anthony S; Gottumukkala, Raju: From Natural Language to SQL: Review of LLM-based Text-to-SQL Systems. arXiv preprint arXiv:2410.01066, 2024.
- [Pa24] Parker, Ben: sqlparse, 2024. <https://pypi.org/project/sqlparse/>, Accessed: 2024-08-16.
- [Pi22] Pi, Xinyu; Wang, Bing; Gao, Yan; Guo, Jiaqi; Li, Zhoujun; Lou, Jian-Guang: Towards robustness of text-to-SQL models against natural and realistic adversarial table perturbation. arXiv preprint arXiv:2212.09994, 2022.
- [Py24a] Python Poetry Contributors: . Python Poetry Documentation, 2024. <https://python-poetry.org/docs/>, Accessed: 2024-08-15.
- [Py24b] PyTorch: PyTorch Previous Versions, 2024. <https://pytorch.org/get-started/previous-versions/>, Accessed: 2024-08-16.
- [RS23] Radanliev, Petar; Santos, Omar: Adversarial Attacks Can Deceive AI Systems, Leading to Misclassification or Incorrect Decisions. 2023.
- [Sa24] Sahitaj, Ariana: BTW-2025 Appendix, 2024. GitHub repository <https://github.com/ardemsa/BTW-2025.git>, last accessed on December 9, 2024.
- [Su24] SubstratusAI: vllm 0.4.0-post1, 2024. <https://artifacthub.io/packages/helm/substratus/vllm/0.4.0-post1>, Accessed: 2024-08-16.
- [Ti23] Time: CEO of the Year 2023: Sam Altman. 2023. <https://time.com/6342827/ceo-of-the-year-2023-sam-altman/>, Accessed: 2024-08-15.
- [Va17] Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz; Polosukhin, Illia: Attention is all you need. Advances in neural information processing systems, 30, 2017.

- [Xu18] Xu, Kun; Wu, Lingfei; Wang, Zhiguo; Feng, Yansong; Sheinin, Vadim: Sql-to-text generation with graph-to-sequence model. arXiv preprint arXiv:1809.05255, 2018.
- [Yu18] Yu, Tao; Zhang, Rui; Yang, Kai; Yasunaga, Michihiro; Wang, Dongxu; Li, Zifan; Ma, James; Li, Irene; Yao, Qingning; Roman, Shanelle et al.: Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. arXiv preprint arXiv:1809.08887, 2018.
- [ZDS17] Zhao, Zhengli; Dua, Dheeru; Singh, Sameer: Generating natural adversarial examples. arXiv preprint arXiv:1710.11342, 2017.
- [ZWF23] Zhang, Weixu; Wang, Yu; Fan, Ming: Towards robustness of large language models on text-to-sql task: An adversarial and cross-domain investigation. In: International Conference on Artificial Neural Networks. Springer, pp. 181–192, 2023.
- [ZXS17] Zhong, Victor; Xiong, Caiming; Socher, Richard: Seq2sql: Generating structured queries from natural language using reinforcement learning. arXiv preprint arXiv:1709.00103, 2017.