The Future of Engineering Education
2024 Annual Conference & Exposition
Oregon Convention Center
Portland, OR . June 23 - 26, 2024
ASEE
Paper ID #43159

Optimizing Database Query Learning: A Generative AI Approach for Semantic Error Feedback

Abdulrahman AlRabah, University of Illinois Urbana-Champaign

Abdulrahman AlRabah is a Master of Science (M.S.) in Computer Science student at the University of Illinois at Urbana-Champaign. He holds a Graduate Certificate in Computer Science from the same institution and a Bachelor of Science in Mechanical Engineering from California State University, Northridge. He has experience in various industries and has served in multiple roles throughout his professional career, including in oil and gas and co-founding a food & beverage company. His research focuses on identifying semantic errors of students and optimizing AI feedback using customized large language models (LLMs) through fine-tuning. Abdulrahman has worked on enhancing AI-based feedback models using both open and closed source models. His work aims to improve SQL teaching methodologies and develop tools that integrate machine learning concepts to enhance both student learning and instructor teaching experiences in computer education.

Sophia Yang, University of Illinois Urbana-Champaign

Sophia Yang is a second-year Ph.D. candidate with research work focused in the areas of Computing Education, Database Systems, Bioinformatics algorithms, Human-Computer Interaction, and interesting intersections of the above. Current research work focuses on quantitatively and qualitatively studying how students learn SQL by utilizing Bioinformatics alignment algorithms, aiming to create instructor and student-facing tools. Her work aims to better improve students' learning and instructors' teaching experiences in large-scale database courses.

Dr. Abdussalam Alawini, University of Illinois Urbana-Champaign

Abdussalam Alawini, a Teaching Associate Professor at the University of Illinois at Urbana-Champaign, combines an extensive industry background with academic achievements. Starting with a Computer Science degree from the University of Tripoli, he spent over six years in the tech industry before earning master's degrees in Computer Science and Engineering and Technology Management from Portland State University, where he also completed his Ph.D. His doctoral work focused on enhancing file-based dataset management for scientists. Dr. Alawini's research spans databases, applied machine learning, and educational technology, aiming to improve classroom experiences and develop advanced data management systems.

©American Society for Engineering Education, 2024

Optimizing Database Query Learning: A Generative AI Approach for Semantic Error Feedback

Abstract

In database education research, numerous common error types and overarching learning hurdles have been identified among learners, with a predominant focus on syntax errors. However, there has been a noticeable gap in the study and categorization of semantic errors, which are equally critical for students' learning and proficiency in database systems. Our study aims to explore this gap and contribute to the educational domain by investigating the potential of integrating the advanced capabilities of a Generative Pre-Trained Transformer (GPT) model with an existing feedback system to enhance the detection and feedback of semantic errors in student submissions of SQL queries. We have utilized diverse datasets of student submissions, which were employed to fine-tune our GPT models. This tailored training process has enabled the models to better recognize and highlight semantic errors, while simultaneously providing constructive and meaningful feedback. Preliminary results from our research are highly encouraging, demonstrating advancements and highlighting the potential of large language models in database learning. By integrating these state-of-the-art computational tools into the learning environment, our study lays the groundwork for the creation of intelligent systems that offer nuanced and context-aware feedback. Such systems have the potential to enhance the educational experience and support available to students.

1 Introduction

The de facto standard language for managing and querying relational databases is the Structural Query Language (SQL) [1]. Hence, individuals interacting with databases, including researchers and developers, find acquiring SQL to be crucial [2, 3]. Its English-like syntax makes it accessible to beginners without the need for prerequisite programming knowledge [4], serving as a gateway to lower entry barriers into computing fields for non-Computer Science students. Despite these advantages, there are still learning challenges for many students studying SQL, prompting the need for further research [2]. These challenges typically fall into two categories: syntax errors and semantic errors. Syntax errors occur when a student's query violates the grammatical rules of the language, resulting in queries that the database system cannot execute. Semantic errors arise when a query is syntactically correct but does not produce the expected data result. While there has been a plethora of research on syntax errors for SQL [5, 6, 7, 8, 9, 10, 11, 12], there is a relative lack of research on semantic errors for SQL [6, 7, 13, 14], and how students overcome them. One of the reasons why students might find the semantics component challenging is due to the limitations of conventional feedback methods. Since syntax errors violate the grammatical rules of the language, they may be easier to recognize by compilers, and therefore the compiler may throw errors for the student as useful feedback. This is not the case with semantic errors. Semantic errors require the student to sift through the data output and verify whether the results are expected. While auto-graders that compare data results between the student query and solution query may help with this verification process, they do not adequately address the unique learning needs of students - they do not give students feedback regarding the source of the logical error. Other conventional feedback methods may reveal the solution, thereby preventing the student from going through the problem-solving process.

To address this, we are interested in methodologies that provide semantic error feedback to students without revealing the solution. In particular, we explore novel approaches utilizing a Generative AI model to provide semantic error feedback. We use students' submissions to SQL homework assignment problems from the Database Systems course available to upper-level undergraduate and graduate students at the University of Illinois Urbana-Champaign. The dataset contains a mix of 100 correct and 400 incorrect submissions and underwent an extensive fine-tuning process with OpenAI's advanced GPT-3.5-turbo-1106 model [15]. Therefore, our research questions include:

- RQ1: *How can a proof of concept be designed and implemented to assess the feasibility of utilizing a generative AI model for providing semantic error feedback in educational settings, ensuring that the system avoids disclosing correct answers while enhancing the learning experience?*
- RQ2: *How does the feedback from the fine-tuned GPT model differ in specificity and relevance compared to standard GPT models in the context of SQL query feedback?*

Our research questions aim to evaluate the feasibility and effectiveness of utilizing a Generative AI model to provide semantic error feedback without revealing the correct solution. We target precise error detection and insightful feedback to enhance the educational experience by making it more tailored to each student. The effectiveness of our fine-tuned model was assessed through comparative analyses with the outputs from the standard ChatGPT model. This validation process was crucial in establishing the refined model's advancement and distinction in providing precise and contextually appropriate feedback for SQL queries.

2 Related Work

Research on SQL learning has explored various types of errors and student challenges. Work by [13, 16, 14] significantly advance our understanding of semantic and persistent error types in SQL learning, shedding light on common misconceptions and areas of difficulty for students. The identification of semantic errors by [13] and the exploration of novices' mismanagement of query complexity in [16] highlight key areas for instructional improvement. Further contributions by [17, 5, 6] provide a comprehensive view of both semantic and syntactic mistakes, reinforcing the need for targeted teaching strategies to address these frequent and complex errors. These works underscore the critical role of understanding persistent errors, as discussed in [5, 6]. On the other hand, research aimed at improving SQL education through student query analysis and visualization techniques includes studies by [7, 18, 8]. These contributions, along with [9, 10, 12], suggest innovative methods for simplifying SQL learning and addressing the cognitive challenges associated with query formulation. Moreover, additional insights from [19, 20, 21] complement this body of research by highlighting the importance of reducing cognitive load and improving error comprehension among SQL learners. These efforts collectively contribute to making SQL more accessible and reducing student errors, significantly impacting SQL education strategies. However, while these studies identify errors, less attention has been paid to strategies for overcoming semantic errors in SQL learning.

While much of the prior research on how students learn SQL has extensively examined the error types and persistent challenges that students encounter during their problem-solving process [5, 6, 7, 8, 9, 10, 11, 12, 14, 17, 16], there is a relative lack of knowledge in the domain of how students overcome semantic errors, in particular, for SQL. The limited research in this area tends to focus on either classification of semantic errors [13], persistent error types which include logical errors (semantic error, by our definition) [5, 6, 7, 16], or a breakdown of semantic errors from a large student SQL submission dataset [14]. Other researchers have also examined methodologies for visualizing students' SQL queries or their submission process [19, 20, 21, 22, 18].

Innovative teaching systems like SQL-Tutor and esql have been developed to provide interactive and personalized learning experiences, focusing on individualized instruction and semantic feedback [2, 23]. These systems address common SQL misconceptions and anti-patterns by offering dynamic feedback and visual step-by-step explanations of query execution, distinct from traditional teaching methods [24, 25]. The proposed Generative AI model aims to extend these approaches by providing nuanced semantic error feedback without revealing solutions, encouraging deeper exploration and self-guided learning, representing a novel direction in SQL education [26, 27].

[28] investigates the effectiveness of peer correction in SQL and NoSQL learning, suggesting it as a viable alternative to traditional feedback mechanisms. This method indicates a shift towards collaborative educational strategies, enhancing learning outcomes through peer interactions. Additionally, [29] and [30] advance this narrative by integrating AI into educational contexts, notably through multimodal recommendation systems and generative tasks with Large Language Models (LLMs). These advancements underscore an evolving trend towards AI-enhanced pedagogies, laying a foundational basis for our work, particularly in the domains of automated feedback and personalized learning experiences in database education.

In the realm of AI-enhanced education, the integration of Large Language Models (LLMs) and chatbots has been pivotal, underpinning our approach to semantic error feedback in SQL education. Kasneci [31] and Oguz [32] underscore the role of tools like ChatGPT in personalizing learning and serving as educational aids, aligning with our use of a fine-tuned GPT model for SQL query feedback. Abedi's [33] exploration into AI tools in specialized education further supports our model's customization for SQL learning, emphasizing the need for tailored AI pedagogical applications. Weyssow [34] and Lai [35] on optimizing LLMs inform our fine-tuning process, ensuring our model's effectiveness in educational contexts. Lastly, Fang and Zhang's [36] work on efficient concept extraction from pre-trained models mirrors our methodology in curating a focused SQL query dataset for training, enhancing our model's capability to identify semantic errors. These studies collectively guide our integration of GPT into SQL education, demonstrating the potential of AI to revolutionize learning experiences.

Prior research has identified that semantic and logical errors are persistent and challenging errors that students encounter when learning SQL [5, 6, 7]. Additionally, these errors can inhibit their ability to successfully solve SQL questions. However, no work to our knowledge has examined the feasibility and effectiveness of utilizing a Generative AI model to provide semantic error feedback. Furthermore, attempts to do so without giving away the correct solution to the student are unknown. Based on prior work, our aim is twofold: one, we aim to develop a proof of concept to validate the feasibility and functionality of the proposed features; two, we aim to create a personalized learning experience for students to better support them through the SQL learning process.

3 Methodology

Our research focuses on utilizing a dataset from the Fall 2022 semester of the CS 411 - Database Systems course at the University of Illinois Urbana-Champaign. Our methodology played a pivotal role in shaping the training framework for our Large Language Model (LLM).

3.1 Data Source

Our training dataset is based on an anonymized dataset of student submissions from the Fall 2022 semester of the Database Systems course. The database course is offered at the University of Illinois Urbana-Champaign in the United States, and students who are enrolled in the course are generally upper-level undergraduate and graduate students. Students who enroll in the course are assumed to have prerequisite knowledge of Data Structures and have at least some introductory programming knowledge.

In particular, our training dataset is a subset of the student submissions from an SQL programming homework assignment. SQL-related content is covered in the lectures during the first four weeks of the course. Students have approximately two weeks to complete the homework assignment, with unlimited attempts and immediate feedback from an auto-grader for each submission attempt, for all of the problems. The homework assignment consists of 15 problems over various SQL concepts, including more advanced concepts such as aggregation and correlated subqueries. The homework assignment is hosted on PrairieLearn [37], an online learning management system, with autograding capabilities and gives immediate feedback upon each student's submission.

Our training dataset consists of 100 "true" (correct) and 400 "false" (incorrect) SQL queries. This categorization is obtained through the auto-grader results, which correctly identify syntax and semantic errors. Syntax errors are identified through the compilation process, and semantic errors are identified by comparing the expected output of the instructor's solution and the actual output of the student's solution. We use this set of 500 queries to fine-tune our model after the data cleaning and preprocessing take place, as described in our next section.

3.2 Data Cleaning and Preprocessing

In the data cleaning and preprocessing phase, we undertook a meticulous approach to prepare our dataset for effective model training. The dataset was standardized using a custom Python function named `clean_text`. This function was designed to remove unnecessary characters and spaces from the SQL query texts, thus ensuring consistency across the dataset and enhancing the effectiveness of the model training process. Recognizing the importance of challenging our Large Language Model's (LLM) learning capabilities, we intentionally curated a dataset that featured a higher proportion of incorrect (or 'false') SQL queries relative to correct ('true') ones. This composition was chosen to specifically test and refine the model's ability to discern between correct and incorrect query submissions.

3.3 Data Selection Process

In the data selection process, we employed custom functions, specifically `is_true_query` and `is_false_query`, to categorize the SQL queries into two distinct groups: correct ('true') and incorrect ('false'). This initial step allowed us to methodically organize the dataset, ensuring that each query was associated with its respective question number for streamlined dataset structure and analysis. To achieve an optimal balance for effective LLM fine-tuning, we adopted a selective approach in determining the proportion of true and false queries within our dataset. This strategic selection was crucial in challenging the model's learning and error identification capabilities. Furthermore, we leveraged the LLM's feedback mechanism extensively by submitting queries and evaluating the generated responses. This iterative process was instrumental in significantly enhancing the model's accuracy and its ability to provide context-aware feedback, thereby refining its overall performance in identifying and addressing semantic errors in SQL queries.

3.4 Fine-Tuning Process

3.4.1 Training Dataset Description:

In the fine-tuning process of our model, we utilized a dataset consisting of 500 SQL queries, divided into 400 correct (true) and 100 incorrect (false) submissions. This structure aims to reflect the range of student responses typically encountered in educational settings. The data was formatted following the OpenAI data structure for optimal model training:

- **System Message:** Directs the AI's behavior to act as a database instructor. Each entry embeds the correct SQL query for the specific question, providing a reference for the model when generating feedback.
- **User Content:** Contains the SQL query submitted by a student.
- **Assistant Message:** The feedback provided by the AI. For correct student queries, we pre-set this as ”Correct query.” For incorrect ones, we use the OpenAI API to generate specific feedback aimed at guiding the student to understand and rectify their errors.

3.4.2 Training Procedure

We selected the “gpt-3.5-turbo-1106” version from OpenAI, known for its efficiency and effectiveness in natural language understanding and generation tasks. We formatted our SQL submissions dataset according to OpenAI’s guidelines. This included structuring each training example as a dialogue with system, user, and assistant messages. We ensured that the dataset represented a broad spectrum of SQL query types to improve the model’s learning. Before initiating the fine-tuning process, the dataset was split into two parts: 70% for training and 30% for validation. This approach allowed us to train the model on a diverse set of examples while also evaluating its performance against unseen data to ensure it can generalize well to new SQL queries. The fine-tuning process was initiated once the dataset was preprocessed and ready. We continuously monitored the model’s performance throughout the training by observing metrics for both the training and validation sets. Special attention was paid to loss scores, as they provide essential insights into the model’s learning and prediction accuracy. Careful monitoring helped us identify and address signs of overfitting or underfitting, making necessary adjustments to the training parameters and strategy to optimize learning outcomes.

3.4.3 Model Monitoring and Adjustments

We monitored the model’s training and validation losses to ensure efficient learning and generalization. The training process showed the training loss decreasing from 1.3308 to 0.8091, indicating learning from the training data. Concurrently, validation loss reduced to 0.7207, suggesting good generalization to new queries. Adjustments were made as needed to prevent overfitting or underfitting, ensuring the model’s effectiveness in real-world applications.

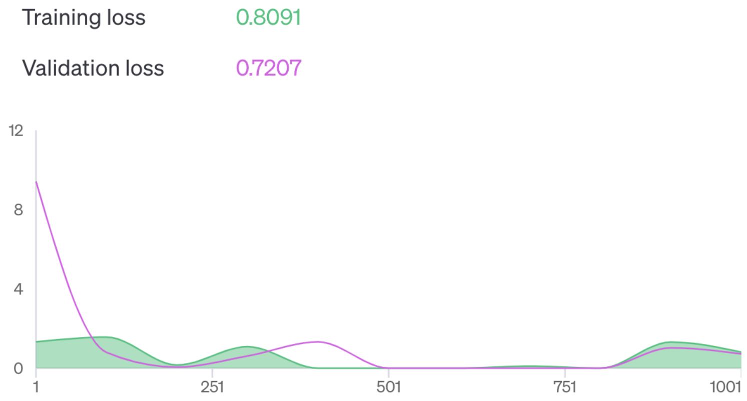

Figure 1: Training and Validation Losses over Steps

As shown in Figure 1, the training and validation loss values decreased over time, indicating that the model was learning and improving from the SQL queries. However, we carefully monitored for any signs of overfitting - where the training loss continues to decrease while the validation loss starts to increase. Upon observing such signs, we would make necessary adjustments to the training parameters.

3.4.4 Hyperparameters

For the fine-tuning of our model, we used the following hyperparameters:

- **Learning Rate Multiplier:** Set at 2 to facilitate rapid learning while maintaining stability in the model's updates.
- **Batch Size:** Set to 1, allowing for detailed, instance-by-instance learning and feedback.
- **Epochs:** Set at 3, sufficient for the model to learn from the data without overfitting.

We initiated the fine-tuning of GPT-3.5-turbo-1106 to precisely identify and correct semantic errors in SQL queries. This precision was achieved by designing system messages that defined expected model behaviors, guiding its learning. Our model is different from a pre-trained ChatGPT model, as it has been optimized specifically for the task of analyzing SQL queries. We have adjusted key hyperparameters: setting training epochs to 3 to effectively learn without overfitting or underfitting, a batch size of 1 for nuanced learning, and a learning rate multiplier of 2 to balance knowledge retention and new insights acquisition. Our training approach involves providing explicit examples of correct and incorrect SQL queries, diverging from the typical unsupervised method. This ensured relevant feedback aligned with our educational goals, enhancing its utility. Additionally, we adjusted the model's temperature to control creativity and setting the token length for a shorter and concise feedback.

```
{
  "messages": [
    {"role": "system", "content": "You are a database instructor assisting students in
    identifying errors in their SQL queries without providing the exact answer. Only provide a
    max 50 words feedback dont talk about anything else. True Query: <true query>"},
    {"role": "user", "content": "<SQL Query from student>"},
    {"role": "assistant", "content": "<Feedback generated by the model in the preparation phase>"
    }
  ]
}
```

Listing 1: Format of the Training Data for Fine-tuning the AI Model.

The **system** message provides the instructions for the AI and sets the road map on how the model should behave in a specific interaction. The **User** role represents inputs from the user. The **Assistant** role is to respond to the user's input and shaped by the instructions given in the system message.

4 Results

4.1 Model Architecture and Educational Customization

The architecture of our fine-tuned AI model is specifically designed to process SQL queries in an educational setting. Central to its design is a system instructed to function as a database instructor, with the guiding principle encapsulated in our system message: "You are a database instructor assisting students in identifying errors in their SQL queries without providing the exact answer. Only provide a max 50 words feedback don't talk about anything else." This directive informed the development of algorithms capable of detecting semantic errors in SQL queries while strategically avoiding the disclosure of direct answers, thus promoting learning through discovery and critical thinking.

4.2 Preliminary Testing - Feedback Mechanism

During the preliminary testing phase, the model demonstrated a high level of accuracy in identifying and addressing semantic errors in SQL queries. Below are specific examples showcasing this:

4.2.1 Incorrect Joins:

The database in question involves student enrollments, where each student is enrolled in various courses, and scores are awarded based on their performance. The objective of the SQL query is to retrieve courses with an average score above 80, alongside the scores of students who scored more than 85 in those courses. The provided query attempts to join the Students and Enrollments tables to calculate the average course score and then filter those courses and students meeting specific criteria.

Student Query:

```sql
SELECT e.CRN,
       (SELECT SUM(Score * Credits)/SUM(Credits)
        FROM Enrollments e2
        WHERE e.CRN = e2.CRN) AS CourseAvgScore,
       s.NetId,
       e.Score
FROM Students s
JOIN Enrollments e ON s.ID = e.CourseID  -- Error: Incorrect JOIN condition, should be based on a
     valid relational key
WHERE (SELECT SUM(Score * Credits)/SUM(Credits)
       FROM Enrollments e2
       WHERE e.CRN = e2.CRN) >= 80
AND e.Score > 85
ORDER BY e.CRN DESC, e.Score DESC;
```

Instructor Query:

```sql
SELECT e.CRN,
       (SELECT SUM(Score * Credits)/SUM(Credits)
        FROM Enrollments e2
        WHERE e.CRN = e2.CRN) AS CourseAvgScore,
       s.NetId,
       e.Score
FROM Students s
JOIN Enrollments e ON s.NetId = e.NetId
WHERE (SELECT SUM(Score * Credits)/SUM(Credits)
       FROM Enrollments e2
       WHERE e.CRN = e2.CRN) >= 80
AND e.Score > 85
ORDER BY e.CRN DESC, e.Score DESC;
```

Correct Approach and Student Misconception

The correct approach of this SQL query emphasizes maintaining the structural goal of calculating course average scores and selecting specific records, with a focus on correcting the join conditions for precise data correlation and retrieval. Unlike the student queries, which might misalign the `Students` and `Enrollments` tables due to a misunderstanding of the appropriate columns for joining, the correct query ensures that the join condition matches the relevant columns. This precision facilitates the accurate extraction of data, thereby addressing a common student misconception regarding join conditions. By rectifying this essential aspect, we provide a feedback that can help students to understand the significance of proper query construction and the impact of join conditions on the query's accuracy.

Model Feedback:

"Check the JOIN condition; it seems like you're trying to join Students and Enrollments on mismatched columns (ID and CourseID). Also, ensure that the subquery for CourseAvgScore is correctly correlated with the outer query. The correlation should be on student or enrollment identifier."

Explanation:

The feedback identifies the semantic error in the JOIN condition, where 'Students' and 'Enrollments' tables are incorrectly joined on mismatched columns (student ID vs. course ID). This guidance prompts the learner to reassess and correct the JOIN logic in their query. Additionally, the model advises checking the subquery for 'CourseAvgScore' to ensure it is properly correlated with the outer query, which is crucial for accurate computation of the average score.

4.2.2 Logical Error in WHERE Clause:

The following database keeps track of customer purchases, including details about customers, the products they buy, and the prices of those products. The goal of the SQL query is to find customers with specific purchasing patterns, such as those who prefer a certain brand, and summarize their spending and product choices. The student query aims to filter records from a database that combines customer and product information, specifically targeting customers whose names start with 'C' and who have purchased 'Google' brand products. However, it incorrectly applies mutually exclusive conditions in the WHERE clause, leading to a logical conflict that prevents the query from returning valid results.

Student Query:

```sql
SELECT c.CustomerId, c.FirstName, c.LastName, p.BrandName, AVG(Price) as average_price,
       Count(PurchaseId) as device_count
FROM Purchases pur
NATURAL JOIN Customers c
NATURAL JOIN Products p
WHERE c.LastName = ’Smith’ AND c.LastName LIKE ’C%’ AND p.BrandName = ’Google’  -- Error: Contradictory conditions in WHERE clause
GROUP BY c.CustomerId
ORDER BY c.CustomerId;
```

Instructor Query:

```sql
SELECT c.CustomerId, c.FirstName, c.LastName, p.BrandName, AVG(Price) as average_price,
       Count(PurchaseId) as device_count
FROM Purchases pur
```

```sql
NATURAL JOIN Customers c
NATURAL JOIN Products p
WHERE c.LastName LIKE 'C%' AND p.BrandName = 'Google'
GROUP BY c.CustomerId
ORDER BY c.CustomerId;
```

Correct Approach and Student Misconception

The key to fixing the student's SQL error is refining the WHERE clause to filter records accurately, avoiding any illogical conditions. The mistake was in trying to meet impossible criteria, like finding customers whose last names both start with 'C' and exactly match 'Smith'. To rectify this, the corrected query targets customers whose last names start with 'C' and who bought 'Google' products. This correction clarifies the use of logical conditions in SQL, steering students away from contradictory criteria. It showcases the importance of correctly applying SQL operators to ensure queries fetch the intended data.

Model Feedback:

Check the WHERE clause for conflicting conditions on LastName. Also, ensure that the aggregate functions in the SELECT clause are used correctly and that the GROUP BY clause matches the non-aggregated columns. Consider the logic of the conditions and the intended result set.

Explanation:

This feedback highlights the logical inconsistency in the WHERE clause, where the conditions for 'c.LastName' are mutually exclusive and cannot be true simultaneously. It guides the learner to reevaluate and correct the WHERE clause logic, emphasizing the importance of ensuring logical consistency in query conditions for meaningful and accurate results. Moreover, our model is designed to provide semantic feedback for advanced SQL topics, including GROUPING, STORED PROCEDURES, and TRIGGERS, showcasing its capability to support a wide range of SQL educational needs.

4.3 Specificity and Relevance of Feedback

In this section, we conduct a comparative analysis between the feedback provided by our fine-tuned AI model and standard GPT models. The focus is on evaluating the specificity and relevance of the feedback for each model in response to the same set of SQL queries.

4.3.1 Example 1: Incorrect SQL Procedure

The intent behind the SQL procedure is to dynamically calculate customer discounts based on their total purchase prices. This process, streamlined for clarity, encompasses several key operations: firstly, assessing the total expenditure of each customer; secondly, assigning discount rates proportional to these expenditure levels; thirdly, generating a new table specifically for recording these discounts; and finally, systematically updating this table with the calculated discounts for every customer. (Code not included due to page limit constraints.)

4.3.2 Example 2: Misuse of GROUP BY

This SQL example attempts to calculate the average grade and total course count for students, specifically focusing on courses with either 3 or 4 credits. However, there's a fundamental misuse of the `GROUP BY` clause. Instead of grouping results by a unique identifier (in this case, `NetId` which represents each student), the query mistakenly tries to group by `avg_grade`, a calculated value. This approach is incorrect because `GROUP BY` should be used with columns that have the same values in multiple rows, serving as a basis for aggregation, not with the result of an aggregated function like an average grade.

```sql
SELECT NetId, COUNT(CRN) AS course_count,
       ROUND((SUM(Score) / COUNT(CRN)), 2) AS avg_grade
FROM Students
JOIN (SELECT * FROM Enrollments WHERE Credits = 3 OR Credits = 4)
     USING (NetId)
GROUP BY avg_grade
ORDER BY NetId;
```

4.3.3 Example 3: Misuse of Trigger for Inserting Rows

The SQL trigger `DrinkTrig` is designed to run after a new insertion in the `Enrollments` table, aiming to grant extra credits to students exceeding a certain credit threshold. However, the implementation faces issues with accurately aggregating credits per student and potentially misuses the `old` keyword, intended for use in triggers dealing with updates or deletions rather than insertions.

```sql
CREATE TRIGGER DrinkTrig AFTER INSERT ON Enrollments
FOR EACH ROW
BEGIN
    SET @creds =(SELECT SUM(Credits) AS total_creds FROM Enrollments GROUP BY NetId);
    IF @creds >= 50 THEN
        INSERT INTO Enrollments(NetId, CRN, Credits, Score) VALUES(old.NetId, old.CRN, old.Credits, old.Score + 10);
    END IF;
END;
```

4.3.4 Example 4: Correct Query

This SQL query calculates each student's course count and the average grade for courses worth 3 or 4 credits, grouping the results by 'NetId'. It demonstrates the correct application of aggregation functions and the 'GROUP BY' clause to organize and summarize enrollment data efficiently.

```sql
SELECT
    NetId,
    COUNT(CRN) AS course_count,
    ROUND(AVG(Score), 2) AS avg_grade
FROM
    Enrollments
WHERE
    Credits = 3 OR Credits = 4
GROUP BY
    NetId;
```

4.3.5 Comparative Feedback Analysis

Table 1 presents the results of our comparative feedback analysis. We provide detailed discussions for each example in the table below.

Query Reference	Fine-Tuned Model Feedback	GPT 3.5 Feedback
Example 1	Check the cursor query for the correct use of JOIN and the conditions for selecting purchases. Also, ensure that the pId variable is assigned the correct value from the cursor. Review the logic for setting the sale variable based on pId.	The query has an error in the cursor declaration. It should filter purchases with positive prices, not negative prices.
Example 2	Check the GROUP BY clause; it should only include the non-aggregated columns. Also, ensure that the column aliases are used correctly in the GROUP BY and ORDER BY clauses.	The input query has an error in the GROUP BY clause. It should group by ‘NetId,’ not ‘avg_grade’.
Example 3	Check the use of aggregate functions in the trigger and ensure they are being used correctly within the context of the trigger logic. Also, review the use of the INSERT statement within the trigger and consider how it may affect the data consistency and the triggers own operation.	The input query has several issues. It should use “New” instead of “old” in the INSERT statement, and the SELECT SUM(Credits) should be for the specific “New.NetId” instead of grouping by NetId. Also, the trigger should be defined as “BEFORE INSERT,” not “AFTER INSERT.”
Example 4	Correct query	The input query is almost correct, but it’s missing the alias “AS course_count” for the COUNT(CRN) column in the SELECT clause.

Table 1: Comparative Feedback Analysis

Example 1: The fine-tuned model provides a detailed and instructive critique of the SQL procedure, addressing the use of joins and the logic of variable assignments within the cursor. It encourages the user to critically examine the correctness of variable use and the overall logic flow, fostering a deeper understanding of SQL procedures. In comparison, the GPT 3.5 model’s feedback is precise and identifies the exact issue with the cursor declaration, guiding the user to the right correction. However, by pinpointing the error so explicitly, it may diminish the learning experience by providing the solution outright rather than encouraging the learner to explore and understand the underlying principles.

Example 2: The fine-tuned model’s feedback adeptly nudges the user to reconsider the GROUP BY clause, aligning with best practices for SQL. It promotes exploration and self-correction by suggesting a reevaluation of the GROUP BY and ORDER BY clauses without giving away the exact fix.

On the other hand, the GPT 3.5 model specifies the error in the GROUP BY clause, offering a straightforward correction. This direct approach, while helpful for immediate problem resolution, could potentially shortcut the learning process by revealing the answer.

Example 3: The fine-tuned model's feedback carefully directs the user to consider various aspects of the trigger's logic, such as the use of aggregate functions and the implications of performing inserts within the trigger. This guidance is structured to provoke thoughtful analysis without directly providing the solution, thereby encouraging a deeper learning experience. On the other hand, the GPT 3.5 model exhibits a clear understanding of the trigger's issues by identifying the specific errors. However, by giving direct corrections, it could inadvertently reveal the answers. While this direct approach can be useful for expediently fixing errors, it may limit the opportunity for users to learn by working through the problem-solving process on their own.

Example 4: The fine-tuned model's affirmation of the query as correct demonstrates its capability to validate SQL queries accurately, which is essential for providing reliable feedback. It indicates the model's nuanced understanding of SQL syntax and semantics. In contrast, the GPT 3.5 model's feedback mistakenly identifies a non-existent error, suggesting that the query is missing an alias for the COUNT(CRN) column. This not only indicates a lapse in the model's query interpretation but also points to the potential for providing misleading feedback. While the GPT 3.5 model has the capacity to understand and analyze SQL statements, this example shows that even with good comprehension, there's a risk of providing incorrect feedback that could misguide learners. In response to potential concerns about our model's consistency across different SQL queries, it's important to note our model is designed with a robust validation and continuous learning framework. This framework ensures it learns from a diverse range of SQL queries, reducing the likelihood of repeating the misinterpretations identified in the GPT 3.5 model's feedback. Our model's understanding of SQL syntax and semantics is ongoing, aiming to enhance its reliability in providing accurate and constructive feedback.

5 Discussion

In our discovery of incorporating generative AI models for educational use, our research has focused on two significant questions. The first, RQ1, examines the design and implementation of a proof of concept to assess the feasibility of using a generative AI model for providing semantic error feedback. Our findings indicate that by meticulously adjusting and taking a thorough approach to training data variation, generative AI can effectively detect SQL errors. The model has been able to identify and respond to a wide variety of SQL query errors without providing direct solutions, thus promoting a deeper, exploratory learning process. This aligns with educational methods that prioritize the growth of problem-solving skills over memorization.

Specifically, the fine-tuning process involved critical adjustments to the model's parameters, including temperature settings and learning rate multipliers, to optimize its performance for educational tasks. By adjusting the temperature, we controlled the model's creativity, ensuring its responses were informative yet concise, directly supporting the learning process without unnecessary information. Additionally, fine-tuning the learning rate multiplier allowed us to balance the model's ability to learn from our specific dataset against retaining its pre-trained knowledge, ensuring it remained versatile yet effective at identifying semantic errors specific to SQL queries. In addressing RQ2, our comparative analysis has shown that the fine-tuned GPT model offers a superior level of specificity and relevance in feedback when compared to standard GPT models. By adjusting model parameters such as temperature we have refined the model's output to deliver feedback that is both contextually precise and educationally valuable. The fine-tuned model adheres to the system message's directive, effectively avoiding the disclosure of correct answers. This balance between providing guidance and fostering independent learning is where the fine-tuned model's feedback mechanism truly excels. The fine-tuned model has been shown to comprehend the nuances of SQL queries. Its feedback is designed to guide learners towards identifying their semantic errors and encourages the development of their analytical skills. In contrast, the standard GPT model, while generally accurate, it tends to hallucinate by generating responses based on its vast training data that can sometimes be contextually irrelevant or misleading. This unpredictability can accidentally disrupt the learning process by providing solutions that are either too revealing or diverge from the educational objectives.

The implications of these findings are significant. They suggest that generative AI can play a pivotal role in educational settings, particularly in disciplines that require complex problem-solving skills such as computer science and database management. By utilizing AI as an intelligent assistant, educators can offer customized feedback at scale, potentially revolutionizing the way feedback is provided and consumed in learning environments.

6 Limitations

Since our study is based on data collected from the University of Illinois Urbana-Champaign, a large public university with a highly-ranked Computer Science department, the students and their data may influence the fine-tuning process and therefore the generalizability of our findings. For this purpose, data from other universities or institutions should be collected and incorporated into the fine-tuning process.

As with many other AI models, our fine-tuned model's effectiveness is dependent upon a quality and diverse training dataset. For example, if the training dataset lacks variety or contains biases, it may limit the model's ability to provide accurate and unbiased feedback. This then emphasizes the importance of the data selection process, to ensure variety and a lack of biases.

Furthermore, overcoming the perceived generality of LLMs, particularly with complex SQL queries, is challenging; despite the intelligence of the system, there could be instances where the LLM fails to fully grasp the intricacies of advanced SQL queries. Users might view the feedback from LLMs as too generic, influenced by their previous experiences with standard chatbots that are known for providing generalized responses. Also, the system may not recognize semantically correct but structurally highly different queries as correct. This phenomenon is partially attributable to the multiplicity of valid approaches for solving a given SQL problem.

7 Future Work

Future research based on this study could aim to enhance the training dataset with a greater variety of SQL queries and scenarios to improve the model's ability to provide more concise feedback. There is also the opportunity to advance the model's ability to understand and provide feedback on more complex SQL queries. Personalization of feedback, tailored to individual learning styles and progress, is another promising area for development. Additionally, implementing continual learning and updating mechanisms would ensure that the model remains current with evolving SQL standards and practices. Finally, quantitative and qualitative studies to extensively test the model in real-world educational settings are crucial to validate its practical effectiveness and impact on students' learning; qualitative studies are also necessary to help gather comprehensive user feedback for further refinement.

Following thorough testing to validate its efficacy in enhancing student learning outcomes, we plan to integrate the model into online assessments conducted via widely used online learning management systems, such as PrairieLearn [37]. This strategic integration is intended to optimize the model's effectiveness in providing support throughout students' SQL learning journey.

8 Conclusions

Our study has demonstrated the significant potential of using a fine-tuned Generative AI model to provide semantic error feedback in SQL learning. The incorporation of this advanced technology into educational tools represents a substantial step forward in showcasing the challenges students face when learning SQL. The model's ability to provide precise, context-aware feedback enhances the overall learning experience. The study's findings offer promising insights into the integration of AI in education, particularly in specialized domains like database management. While the fine-tuned AI model has shown considerable effectiveness in providing feedback, the study also acknowledges limitations such as data dependency and the potential perception of feedback generality. These limitations highlight areas for future improvement and research. Future work will focus on enriching the training dataset, which is a crucial step in improving the model's understanding. The ultimate goal is to create an intelligent, dynamic learning environment that delivers to diverse educational needs and continually evolves to meet the challenges of teaching SQL and other complex subjects.

References

[1] A. DiFranza, "5 reasons sql is the need-to-know skill for data analysts," 2020.

[2] A. Mitrovic, "Learning sql with a computerized tutor," in *Proceedings of the Twenty-Ninth SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '98. New York, NY, USA: ACM, 1998, p. 307–311.

[3] J. Vigo, "5 programming languages database administrators should learn," November 2020.

[4] HANLijun, "Sql: A supposed english-like language," Jan. 2018. [Online]. Available: https://www.datasciencecentral.com/sql-a-supposed-english-like-language/

[5] T. Taipalus, M. Siponen, and T. Vartiainen, "Errors and complications in sql query formulation," *ACM Transactions on Computing Education (TOCE)*, vol. 18, no. 3, pp. 1–29, 2018.

[6] T. Taipalus and P. Perälä, "What to expect and what to focus on in sql query teaching," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '19. New York, NY, USA:

Association for Computing Machinery, 2019, p. 198–203. [Online]. Available: https://doi.org/10.1145/3287324.3287359

[7] T. Taipalus, "Explaining causes behind sql query formulation errors," in *2020 IEEE Frontiers in Education Conference (FIE)*. IEEE Press, 2020, p. 1–9. [Online]. Available: https://doi.org/10.1109/FIE44824.2020.9274114

[8] R. Alkhabaz, Z. Li, S. Yang, and A. Alawini, "Student's learning challenges with relational, document, and graph query languages," in *Proceedings of the 2nd International Workshop on Data Systems Education: Bridging Education Practice with Education Research*, ser. DataEd '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 30–36. [Online]. Available: https://doi.org/10.1145/3596673.3596976

[9] S. Yang, Z. Li, G. L. Herman, K. Cunningham, and A. Alawini, "Uncovering patterns of sql errors in student assignments: A comparative analysis of different assignment types," in *2023 IEEE Frontiers in Education Conference (FIE)*, 2023, pp. 01–09.

[10] Z. Li, S. Yang, K. Cunningham, and A. Alawini, "Assessing student learning across various database query languages," in *2023 IEEE Frontiers in Education Conference (FIE)*, 2023, pp. 1–9.

[11] A. Ahadi, V. Behbood, A. Vihavainen, J. Prior, and R. Lister, "Students' syntactic mistakes in writing seven different types of sql queries and its application to predicting students' success," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. New York, NY, USA: ACM, 2016, pp. 401–406.

[12] S. Poulsen, L. Butler, A. Alawini, and G. L. Herman, "Insights from student solutions to sql homework problems," in *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York, NY, USA: ACM, 2020, pp. 404–410.

[13] S. Brass and C. Goldberg, "Semantic errors in sql queries: A quite complete list," *Journal of Systems and Software*, vol. 79, no. 5, pp. 630–644, 2006, quality Software.

[14] A. Ahadi, V. Behbood, A. Vihavainen, J. Prior, and R. Lister, "Students' semantic mistakes in writing seven different types of sql queries," in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 272–277. [Online]. Available: https://doi.org/10.1145/2899415.2899464

[15] OpenAI, "Gpt-3.5-turbo-1106," https://openai.com/, 2023, accessed: 2024-01-20.

[16] D. Miedema, G. Fletcher, and E. Aivaloglou, "So many brackets! an analysis of how sql learners (mis)manage complexity during query formulation," in *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*, ser. ICPC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 122–132. [Online]. Available: https://doi.org/10.1145/3524610.3529158

[17] A. Ahadi, J. Prior, V. Behbood, and R. Lister, "A quantitative study of the relative difficulty for novices of writing seven different types of sql queries," in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '15. New York, NY, USA: ACM, 2015, p. 201–206.

[18] J. Danaparamita and W. Gatterbauer, "Queryviz: Helping users understand sql queries and their patterns," in *Proceedings of the 14th International Conference on Extending Database Technology*, ser. EDBT/ICDT '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 558–561. [Online]. Available: https://doi.org/10.1145/1951365.1951440

[19] S. Yang, Z. Wei, G. L. Herman, and A. Alawini, "Analyzing patterns in student sql solutions via levenshtein edit distance," in *Proceedings of the Eighth ACM Conference on Learning @ Scale*, ser. L@S '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 323–326. [Online]. Available: https://doi.org/10.1145/3430895.3460979

[20] S. Yang, G. L. Herman, and A. Alawini, "Mining sql problem solving patterns using advanced sequence processing algorithms," in *Proceedings of the 2nd International Workshop on Data Systems Education:*

Bridging Education Practice with Education Research, ser. DataEd ’23. New York, NY, USA: Association for Computing Machinery, 2023, p. 37–43. [Online]. Available: https://doi.org/10.1145/3596673.3596973

[21] ——, “Analyzing student sql solutions via hierarchical clustering and sequence alignment scores,” in *1st International Workshop on Data Systems Education*, ser. DataEd ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 10–15. [Online]. Available: https://doi.org/10.1145/3531072.3535319

[22] M. Weston, H. Sun, G. L. Herman, H. Benotman, and A. Alawini, “Echelon: An ai tool for clustering student-written sql queries,” in *2021 IEEE Frontiers in Education Conference (FIE)*, 2021, pp. 1–8.

[23] R. Kearns, S. Shead, and A. Fekete, “A teaching system for sql,” *ACSE ’97*, 1997.

[24] D. Miedema, E. Aivaloglou, and G. Fletcher, “Identifying sql misconceptions of novices: Findings from a think-aloud study,” *International Computing Education Research Workshop*, 2021.

[25] L. Mathon and D. Miedema, “Increasing awareness of sql anti-patterns for novices: A study design,” *ICER*, 2022.

[26] A. Migler and A. Dekhtyar, “Mapping the sql learning process in introductory database courses,” 2020.

[27] S.-S. Shin, “Structured query language learning: Concept map-based instruction based on cognitive load theory,” *IEEE Access*, 2020.

[28] W. Wu, “Assessing peer correction of sql and nosql queries,” *Technical Symposium on Computer Science Education*, 2023.

[29] S. Geng, J. Tan, S. Liu, Z. Fu, and Y. Zhang, “Vip5: Towards multimodal foundation models for recommendation,” *arXiv preprint arXiv:2305.14302*, May 2023. [Online]. Available: https://arxiv.org/abs/2305.14302

[30] W. Hua, L. Li, S. Xu, L. Chen, and Y. Zhang, “Tutorial on large language models for recommendation,” in *Seventeenth ACM Conference on Recommender Systems (RecSys ’23)*. Singapore, Singapore: ACM, September 2023. [Online]. Available: https://doi.org/10.1145/3604915.3609494

[31] E. Kasneci, K. Sessler, S. Kuchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Gunnemann, E. Hullermeier, S. Krusche, G. Kutyniok, T. Michaeli, C. Nerdel, J. Pfeffer, O. Poquet, M. Sailer, A. Schmidt, T. Seidel, M. Stadler, J. Weller, J. Kuhn, and G. Kasneci, “Chatgpt for good? on opportunities and challenges of large language models for education,” 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1041608023000195

[32] F. E. Oguz, M. N. Ekersular, K. M. Sunnetci, and A. Alkan, “Can chat gpt be utilized in scientific and undergraduate studies?” *Annals of Biomedical Engineering*, 2023. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/37516681/

[33] M. e. a. Abedi, “Beyond traditional teaching: The potential of large language models and chatbots in graduate engineering education,” *Title of the Journal (if available)*, 2023. [Online]. Available: https://www.qeios.com/read/MD04B0

[34] M. Weyssow, X. Zhou, K. Kim, D. Lo, and H. Sahraoui, “Exploring parameter-efficient fine-tuning techniques for code generation with large language models,” *Proceedings of ACM Conference*, 2023. [Online]. Available: https://arxiv.org/abs/2308.10462

[35] V. D. Lai, C. V. Nguyen, N. T. Ngo, T. Nguyen, F. Dernoncourt, R. A. Rossi, and T. H. Nguyen, “Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback,” *arXiv preprint arXiv:2307.16039*, 2023. [Online]. Available: https://arxiv.org/abs/2307.16039

[36] Y. Fang and Y. Zhang, “Data-efficient concept extraction from pre-trained language models for commonsense explanation generation,” in *Findings of the Association for Computational Linguistics: EMNLP 2022*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds. Abu Dhabi, United Arab Emirates: Association for

Computational Linguistics, December 2022, pp. 5883–5893. [Online]. Available: https://aclanthology.org/2022.findings-emnlp.433

[37] M. West, G. L. Herman, and C. B. Zilles, “Prairielearn: Mastery-based online problem solving with adaptive scoring and recommendations driven by machine learning,” 2015.