

Attribution Scores of BERT-Based SQL-Query Automatic Grading for Explainability


Korn Sooksatra , Bikram Khanal , Pablo Rivas , *Senior, IEEE*

School of Engineering and Computer Science

Department of Computer Science

Baylor University

Email: {Korn_Sooksatra1,Bikram_Khanal1,Pablo_Rivas}@Baylor.edu

Donald R. Schwartz 

School of Computer Science & Mathematics

Department of Computing Technology

Marist College

Email: Donald.Schwartz@Marist.edu

Abstract—Automated grading of SQL queries poses a challenge due to the complexity of the SQL language and the numerous ways in which one can achieve the same results. While recent advancements in machine learning-based automated grading systems have demonstrated remarkable accuracy, there remains a critical need for providing students with meaningful explanations for the grades assigned by these machines. To address this need, our study focuses on a specific recent work in automated grading for SQL queries. We have developed a comprehensive workflow that leverages machine learning models trained in that work. Our objective is to gain insights into the behavior of the machine learning model, particularly in terms of how it assigns grades to SQL queries. Through our analysis, we have observed that our workflow performs effectively, producing valuable attribution scores for important tokens within SQL queries. These attribution scores shed light on which query components are considered significant by the machine learning model, thus enhancing our understanding of the grading process and facilitating more meaningful feedback for students.

Index Terms—model distillation, word embeddings, Bert, natural language processing, machine learning, deep learning

I. INTRODUCTION

Grading SQL queries is a complex task that requires both computational and pedagogical expertise [1], [2]. One of the main challenges in this domain is the existence of multiple correct solutions for a single query prompt [3]. Almost every multi-table SQL query can be correctly written in many different ways. This complexity is further compounded by the various forms that SQL queries can take, including non-nested and nested queries with or without aliasing [4]. The evaluation process becomes particularly burdensome when dealing with a large number of submissions across multiple course sections [5]. Several common challenges can contribute to this complexity: the size of the query set, the number of tables in the underlying database, the number of students submitting assignments, the time it takes to grade the assignments, and the level of detailed feedback we are able to give each student. Moreover, grading SQL queries goes beyond identifying correct solutions; it also involves the fair allocation of partial credit for submissions that demonstrate a conceptual understanding but are not entirely correct [6]. Awarding consistent partial credit, while giving feedback as to why the query was incorrect, is critical.

Traditional methods for grading SQL queries, such as static and dynamic analyses, have limitations [7]. The static analysis approach attempts to evaluate queries by comparing the structure of each student’s query with the structure of a correct (answer key) query. Among other approaches, this can be accomplished by either establishing equivalence classes for the underlying expressions of each query or by determining whether the queries are logically equivalent [8], [9]. Static analysis, although useful for understanding query structure, often fails to capture the full range of correct solutions, necessitating the use of multiple answer-key queries and leading to inconsistent grading outcomes [10].

In the dynamic analysis approach, the student queries are actually executed using various test-data sets and the results are compared with answer-key results [11]. Dynamic analysis often struggles to differentiate between genuinely correct queries and those that coincidentally produce correct results [12]. For example, suppose a query asks the students to “Name the Suppliers who ship parts from Seattle”. In the current data set, it might be the case that the results of that query are the same as the results for “Name the Suppliers who ship green parts to a project in Seattle”. Dynamic analysis might give full credit to both queries, since the resulting answer tables would be identical. This challenge is heightened if an empty table is generated as the correct answer – there are literally infinitely many ways to generate an empty table result.

To address these challenges, Schwartz et al. introduced a machine learning-based approach for grading SQL queries [13]–[15]. This approach utilizes supervised learning techniques and a unique parameter-sharing strategy to model the relationships within SQL statements [16]. By incorporating advanced techniques such as self-attention mechanisms and convolutional neural networks, the model can comprehensively analyze the spatial relationships within SQL queries [17], [18].

Previous work by Rivas et al. has extended the capabilities of machine learning-based grading systems by addressing binary classification, multi-class classification, and regression problems [19]. However, these studies often relied on limited datasets for model evaluation, leaving the model’s behavior across a broader range of samples largely unexplored [19].

The focus of this paper is to build upon the work of Rivas et al. by providing a more comprehensive examination of the

model’s explainability [19]. Specifically, we aim to evaluate the model’s performance across a comprehensive set of test samples, offering a more robust assessment of its grading capabilities. This research will contribute to the advancement of automated SQL query grading systems and provide insights into the strengths and limitations of machine learning-based approaches in this domain.

The remainder of this paper is structured as follows: Section 2 provides an overview of our approach, Section 3 presents an analytical discussion of the findings, and Section 4 offers concluding remarks.

II. APPROACH

To rigorously investigate the decision-making behavior of the binary classifier presented in [13], we leverage the state-of-the-art explainability framework proposed by Hao et al. [20], implemented through the dedicated Captum module.¹ While Captum’s primary design focuses on single-sample analysis, our objective necessitates a more comprehensive evaluation across an entire dataset. To this end, we have engineered a custom Python script that interfaces with Captum to facilitate batch processing and aggregate analysis.

A. Workflow Architecture

Our analytical pipeline consists of two core components: the Captum module and a Python script, both of which operate on a test dataset comprising SQL queries. The workflow, depicted in Figure 1, can be formalized as a sequence of operations defined as follows:

- 1) The Python script dispatches individual SQL query samples to the Captum module.
- 2) Captum processes the received sample and computes token-level attribution scores, denoted as S_{token} .
- 3) The script accumulates the attribution scores S_{token} into a global score vector S_{global} .
- 4) If unprocessed samples remain, the sequence returns to Step 1. Otherwise, the script computes the average attribution score \bar{S} for each token.
- 5) Finally, the script generates a visual representation, highlighting the top k positively and negatively influential tokens based on \bar{S} .

B. Handling BERT’s Subword Tokenization

The model in [13] employs the BERT architecture, known for its subword tokenization scheme [21]. This tokenization often results in fragmented and less interpretable attribution scores at the subword level. To rectify this, we aggregate subword scores into word-level scores during o_3 of our workflow. The aggregation is performed using the following equation:

$$\text{Score}_{\text{word}} = \max(\text{Score}_{\text{subword}_1}, \text{Score}_{\text{subword}_2}, \dots, \text{Score}_{\text{subword}_n}). \quad (1)$$

Here, $\text{Score}_{\text{word}}$ is the aggregated word-level score, and $\text{Score}_{\text{subword}_i}$ is the attribution score for the i -th subword. The

maximum score is chosen to represent the word’s overall influence, as it is assumed to be the most indicative of the word’s contribution to the model’s decision-making process.

C. N-Gram Attribution Analysis

While Captum inherently provides unigram-level attribution scores [20], we argue that n-gram-level attributions offer more nuanced insights. For example, the attribution score for the 3-gram "WHERE LOCATION = 'Atlanta'" is more contextually informative than that for the unigram "Atlanta". To accommodate this, we extend our Python script to compute n-gram scores during o_3 . The n-gram score is calculated as:

$$\text{Score}_{\text{n-gram}} = \frac{1}{n} \sum_{i=1}^n \text{Score}_{\text{word}_i}. \quad (2)$$

In this equation, $\text{Score}_{\text{n-gram}}$ is the aggregated n-gram score, and $\text{Score}_{\text{word}_i}$ is the score for the i -th word in the n-gram. The average is taken to ensure that each word contributes equally to the n-gram’s overall attribution score.

III. RESULTS

Leveraging the same dataset as in the seminal work by Rivas et al. [13], our investigation centers on binary classification tasks and multi-label classification.

A. Binary Classification

For the binary classification, we define the positive class (class 1) as indicative of SQL query correctness and the negative class (class 0) as indicative of incorrectness. To this end, we employ a pre-trained BERT model, as delineated in [13], and extend its capabilities through integration with the Captum explainability module. The analytical workflow is encapsulated in Figure 1.

Our analysis culminates in a rich set of attribution scores, specifically focusing on 5-grams, which are visually rendered in Figure 2. A striking observation is the clear demarcation between positive and negative attribution scores corresponding to class 0. The positive scores, color-coded in blue, predominantly appear at the terminal segments of SQL queries, often enclosed within parentheses. This pattern is not merely coincidental but rather indicative of the model’s tendency to rely on tokens situated at the query’s terminus for inferring its incorrectness. Such queries are generally flawed, either due to the absence of requisite conditions or due to syntactic irregularities like excessive or incomplete parentheses, thereby justifying the model’s attribution.

Conversely, the negative scores, color-coded in red, are dispersed throughout the queries without manifesting a discernible pattern. This seemingly random distribution is, in fact, aligned with the model’s strategy for ascertaining query correctness. It suggests that these red-coded 5-grams serve as pivotal elements in the model’s decision-making process, acting as reliable indicators of validity.

Through these findings, we not only validate the efficacy of the model in [13] but also enrich it by unveiling nuanced patterns and tendencies that were previously underexplored.

¹<https://github.com/pytorch/captum>

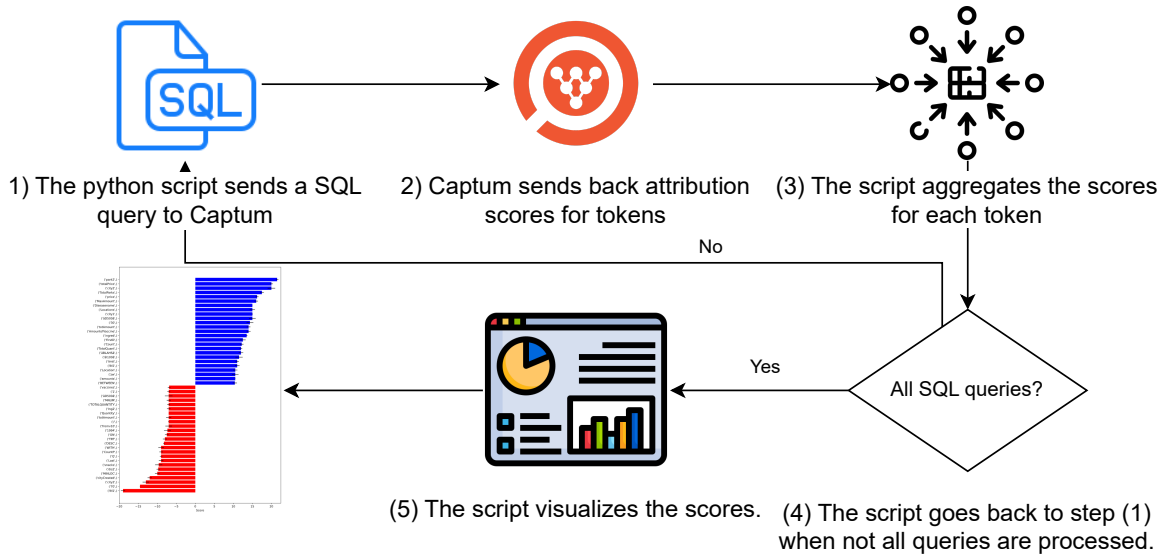


Fig. 1: The workflow of our approach to generate attribution scores based on the entire dataset

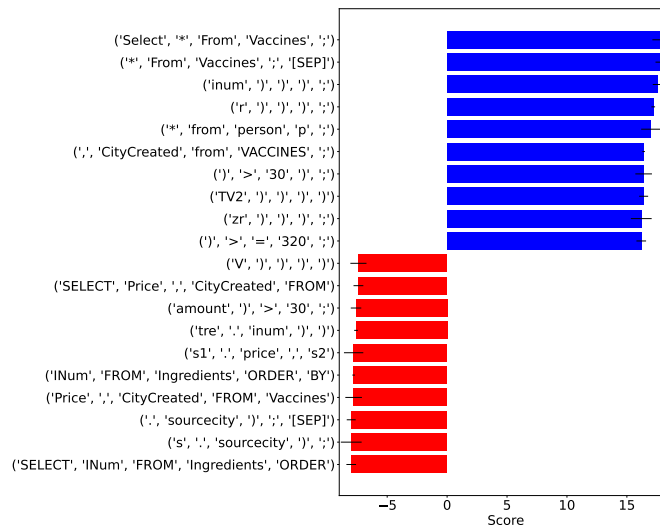


Fig. 2: Attribution scores of top 10 5-grams in the positive (blue) and negative (red) influences to class 0 (negative class)

This contributes to a more robust and interpretable machine learning model for SQL query grading, thereby advancing the state of the art in this domain.

B. Multi-Label Classification

1) **Correct: Positive Attribution Scores.** The 5-grams with high positive attribution scores signify elements indicative of a correct SQL statement. For instance, the 5-gram ('delivers', 'WHERE', 'pnum', '=', 'P100') suggests that the correct usage of the WHERE clause with a proper condition (like pnum = P100) is highly influential in predicting the correctness of the statement. Similarly, 5-grams such as ('FROM', 'DELIVERS', 'WHERE', 'PNUM', '=') and ('Ingredients', 'WHERE', 'SourceLoc', '=', 'âNYCâ') indicate the model's sensitivity to correct clause sequencing and the equality

condition, which are crucial in forming syntactically and semantically correct SQL queries.

Negative Attribution Scores. On the contrary, 5-grams with high negative attribution scores reflect patterns often present in incorrect SQL statements. The presence of 5-grams like ('TABLE', '.', 'PRICE', ',', 'V') and ('=', 'âNBCâ', ')', ')', ';') demonstrate a confusion in the syntactic structure or misuse of SQL elements, which the model associates with incorrectness. The 5-gram ('BY', 'ing', '.', 'sourceloc', ';') is particularly intriguing as it may imply an incorrect form of a keyword or an incoherent sequence of tokens, leading to the negative contribution towards the prediction of a correct SQL query.

The disparity between the positive and negative attribution scores in these 5-grams underlines the importance of contextually appropriate keywords and their correct order in SQL

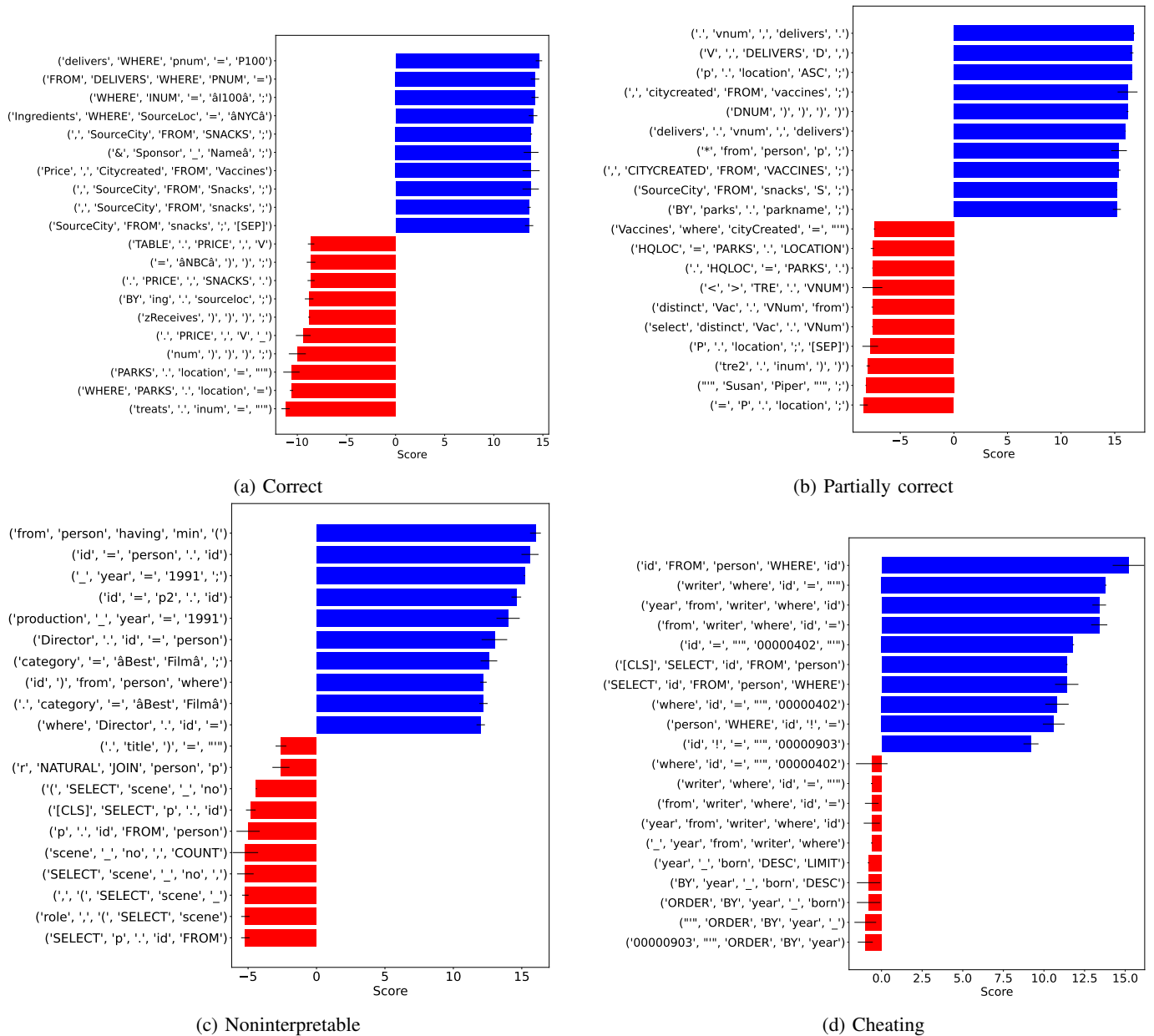


Fig. 3: Attribution scores of top 10 5-grams in the positive (blue) and negative (red) influences to classes.

syntax. This observation can guide educators in emphasizing the critical elements and common pitfalls while teaching SQL to students. Moreover, it highlights areas that automated SQL training tools could focus on to improve student performance.

2) *Partially Correct: Positive Attribution Scores.* 5-grams with high positive attribution scores reveal patterns and sequences typically associated with partially correct SQL statements. The 5-gram $(\text{'.'}, \text{'vnum'}, \text{'.'}, \text{'.'}, \text{'delivers'}, \text{'.'})$ suggests that the correct punctuation and use of identifiers within SQL may lead to partial credits even if not entirely correct. Sequences like $(\text{'V'}, \text{'.'}, \text{'.'}, \text{'DELIVERS'}, \text{'D'}, \text{'.'}, \text{'.'})$ show a pattern of capitalization and delimiter use that appears to align with par-

tially correct syntax, whereas $(\text{'p'}, \text{'.'}, \text{'.'}, \text{'location'}, \text{'ASC'}, \text{'.'}, \text{'.'})$ indicates proper clause ordering. It is notable that 5-grams involving table and column names, such as $(\text{'.'}, \text{'.'}, \text{'citycreated'}, \text{'FROM'}, \text{'vaccines'}, \text{'.'}, \text{'.'})$, contribute positively, implying that the model recognizes some semantic understanding even if the overall syntax might not be perfect.

Negative Attribution Scores. Conversely, 5-grams like $(\text{'Vaccines'}, \text{'where'}, \text{'cityCreated'}, \text{'='}, \text{'.'}, \text{'.'})$ and $(\text{'HQLOC'}, \text{'='}, \text{'PARKS'}, \text{'.'}, \text{'.'}, \text{'LOCATION'})$ with negative scores suggest a misunderstanding of the query structure or relationships between tables and columns. 5-grams such as $(\text{'<'}, \text{'>'}$,

'TRE', '.', 'VNUM') could indicate syntax errors or incorrect use of operators, which are detrimental to the partial correctness of an SQL statement. The recurrence of incorrect or out-of-place syntax, as seen in ('tre2', '.', 'inum', ')', ')'), emphasizes common mistakes that students might make, which the model flags as predictors of an incorrect statement.

The distinction between the positive and negative 5-grams helps identify areas where students may achieve partial understanding and where they are most likely to falter. Educators can leverage these insights to focus on teaching the correct use of SQL syntax and the importance of query structure, which are pivotal in crafting partially correct SQL statements. Such knowledge is vital for refining SQL training methods and providing targeted feedback to students.

3) *Noninterpretable: Positive Attribution Scores.* 5-grams with high positive scores point towards elements or constructs that are commonly associated with uninterpretable statements. For example, the 5-gram ('from', 'person', 'having', 'min', '(') suggests a misuse of the HAVING clause without a preceding GROUP BY, which is a frequent source of confusion. Similarly, ('_', 'year', '=', '1991', ';') might indicate an incorrect naming convention or a misuse of underscore. The use of non-standard characters in ('category', '=', 'âBest', 'Filmâ', ';') could reflect encoding issues or incorrect string literals that lead to a lack of clarity in statements.

Negative Attribution Scores. Conversely, 5-grams with negative attribution scores appear to include more standard SQL expressions. For instance, ('r', 'NATURAL', 'JOIN', 'person', 'p') indicates a proper use of join operations, and ('SELECT', 'p', '.', 'id', 'FROM') follows the typical format of a selection query. These suggest a level of interpretability that the model associates with better-understood statements.

Analyzing these 5-grams enables us to identify specific SQL patterns that lead to misunderstandings or errors, potentially aiding in the design of better SQL instruction and error messaging. Highlighting these patterns could serve as a focus for educational interventions aimed at reducing the frequency of uninterpretable SQL statements by students.

4) *Cheating: Positive Attribution Scores.* The 5-grams with the highest positive attribution scores are indicative of queries that could constitute cheating. These 5-grams include:

```
('id', 'FROM', 'person', 'WHERE', 'id')
('writer', 'where', 'id', '=', '"')
('year', 'from', 'writer', 'where', 'id')
('from', 'writer', 'where', 'id', '=')
('id', '=', '"', '00000402', '"')
```

Such patterns often contain direct references to specific 'id' values, which likely implies that the query was crafted to retrieve a single record, circumventing the intent of creating a generalized query.

Negative Attribution Scores. Conversely, the 5-grams with the highest negative attribution scores are less suggestive of

cheating and may correspond to more generic queries:

```
('where', 'id', '=', '"', '00000402')
('writer', 'where', 'id', '=', '"')
('from', 'writer', 'where', 'id', '=')
('year', 'from', 'writer', 'where', 'id')
('_', 'year', 'from', 'writer', 'where')
```

These patterns, surprisingly similar to some with positive attributions, may appear in contexts that do not imply cheating or may be part of a larger query structure that is inherently more general.

The overlap in 5-grams between the positive and negative attributions suggests that context is crucial for interpreting whether the use of specific constants is a sign of cheating. While the model has learned to identify potential cheating patterns, the presence of these patterns alone does not conclusively determine the intent or correctness of the query. Additional contextual analysis may be required to differentiate between cheating and legitimate use of specific 'id' values. The primary reason for this, we argue, is that there is very little support for this particular class; i.e., there are very few labeled SQL statements in this category.

IV. CONCLUSION

In our analysis of the work conducted by Rivas et al. [13], we examined their innovative use of the BERT model to address the SQL-grading problem, achieving substantial improvements over previous methodologies. However, a notable limitation of their work was the absence of a robust explainability mechanism for the model's decision-making process.

To address this gap, we leveraged the Captum module and devised a script capable of computing token scores (including sets of tokens in the case of n-grams) within the context of binary classification, as presented in [13]. Our approach facilitates a comprehensive understanding of why the model classifies queries as correct or incorrect, accomplished through a thorough examination of the entire dataset.

Our findings indicate that the model places specific emphasis on particular areas or tokens to justify predictions of class 0 (incorrect), while relying on more general query components to predict class 1 (correct).

It is worth noting that the work presented in [13] extends beyond binary classification, encompassing multi-class classification and regression tasks. Our future plans involve extending our approach to those tasks as well, further enhancing the model's interpretability and its applicability to a broader range of challenges.

ETHICS STATEMENT

This study utilizes a combination of a publicly available dataset, as cited in the manuscript, and additional data collected internally. The internal dataset has been anonymized to protect the identities of the students involved. Due to the presence of personally identifiable information in the original internal data, we are unable to release this portion of the dataset to the public.

Moreover, we recognize the potential for inherited biases within the BERT embeddings used in our model. While we have taken steps to assess and address ethical considerations, we acknowledge that these biases may still exist. We advocate for continued research aimed at mitigating such biases and are committed to maintaining a high level of methodological transparency in our work.

ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation under Grant Nos. 2039678, 2136961, and 2210091. The views expressed herein are solely those of the author(s) and do not necessarily reflect those of the National Science Foundation.

REFERENCES

- [1] J. Guo, Z. Zhan, Y. Gao, Y. Xiao, J. Lou, T. Liu, and D. Zhang, "Towards complex text-to-sql in cross-domain database with intermediate representation," 2019.
- [2] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, "Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," 2018.
- [3] S. Chu, B. Murphy, J. Roesch, A. Cheung, and D. Suciu, "Axiomatic foundations and algorithms for deciding semantic equivalences of sql queries," *Proceedings of the VLDB Endowment*, vol. 11, pp. 1482–1495, 2018.
- [4] J. Li, A. König, V. Narasayya, and S. Chaudhuri, "Robust estimation of resource consumption for sql queries using statistical techniques," *Proceedings of the VLDB Endowment*, vol. 5, pp. 1555–1566, 2012.
- [5] R. Lee, T. Luo, Y. Huai, F. Wang, Y. He, and X. Zhang, "Ysmart: yet another sql-to-mapreduce translator," 2011.
- [6] B. Chandra, B. Chawda, B. Kar, K. Reddy, S. Shah, and S. Sudarshan, "Data generation for testing and grading sql queries," *The VLDB Journal*, vol. 24, pp. 731–755, 2015.
- [7] J. Cunha, J. Fernandes, R. Pereira, and J. Saraiva, "Graphical querying of model-driven spreadsheets," pp. 419–430, 2014.
- [8] A. V. Aho, Y. Sagiv, and J. D. Ullman, "Equivalences among relational expressions," *SIAM Journal on Computing*, vol. 8, no. 2, pp. 218–246, 1979.
- [9] S. Chu, B. Murphy, J. Roesch, A. Cheung, and D. Suciu, "Axiomatic foundations and algorithms for deciding semantic equivalences of sql queries," *arXiv preprint arXiv:1802.02229*, 2018.
- [10] Y. Law, H. Wang, and C. Zaniolo, "Relational languages and data models for continuous queries on sequences and data streams," *Acm Transactions on Database Systems*, vol. 36, pp. 1–32, 2011.
- [11] J. Wang, Y. Zhao, Z. Tang, and Z. Xing, "Combining dynamic and static analysis for automated grading sql statements," *J Netw Intell*, vol. 5, no. 4, pp. 179–190, 2020.
- [12] A. Kelkar, "Bertrand-dr: improving text-to-sql using a discriminative re-ranker," 2020.
- [13] P. Rivas, D. Schwartz, and E. Quevedo, "Bert goes to sql school: Improving automatic grading of sql statements," in *The 25th International Conference on Artificial Intelligence (ICAI 2023)*, 2023, pp. 1–8.
- [14] D. Schwartz and P. Rivas, "An automated sql query grading system using an attention-based convolutional neural network," in *The 18th International Conference on Frontiers in Education: Computer Science and Computer Engineering*, 2022, pp. 1–12.
- [15] P. Rivas and D. R. Schwartz, "Modeling sql statement correctness with attention-based convolutional neural networks," in *The 19th International Conference on Scientific Computing (CSC 2021)*, 2021.
- [16] J. Triloka, H. Hartono, and S. Sutedi, "Detection of sql injection attack using machine learning based on natural language processing," *International Journal of Artificial Intelligence Research*, vol. 6, 2022.
- [17] A. Li, P. Ng, P. Xu, H. Zhu, Z. Wang, and B. Xiang, "Dual reader-parser on hybrid textual and tabular evidence for open domain question answering," 2021.
- [18] C. Tang, B. Wang, Z. Luo, H. Wu, S. Dasan, M. Fu, Y. Li, M. Ghosh, R. Kabra, N. Navadiya, D. Cheng, F. Dai, V. Channapattan, and P. Mishra, "Forecasting sql query cost at twitter," 2021.
- [19] H. Bao and M. Clavel, "A model-driven approach for enforcing fine-grained access control for sql queries," *Sn Computer Science*, vol. 2, 2021.
- [20] Y. Hao, L. Dong, F. Wei, and K. Xu, "Self-attention attribution: Interpreting information interactions inside transformer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 14, 2021, pp. 12963–12971.
- [21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.