Explainable AI for SQL Grading: A Practical Approach with Multi-Task CNNs

Pablo Rivas[1][0000−0002−9802−6705] and Donald R. Schwartz[2][0000−0002−8690−0987]

[1] Department of Computer Science, Baylor University, Waco, TX 76798, USA
Pablo_Rivas@Baylor.edu

[2] Dept. of Computing Technology, Marist College, Pougkeepsie, NY 12601, USA
Donald.Schwartz@Marist.edu

Abstract. This study focuses on enhancing an automated SQL statement grading model by expanding the dataset and integrating Local Interpretable Model-agnostic Explanations (LIME) to improve explainability. By incorporating a significantly larger dataset, the model's ability to generalize across a variety of SQL queries has been enhanced, as demonstrated by improved performance metrics such as precision, recall, and F1 scores. The integration of LIME provides insights into the inference processes of the model, highlighting the influence of specific SQL components on assessment outcomes. These enhancements have practical implications, including more accessibility for users to understand the rationale behind model decisions, potentially leading to more effective learning experiences.

Keywords: attention · CNNs · SQL · natural language processing · machine learning · deep learning · explainable AI · LIME.

1 Introduction

Grading SQL queries can prove to be arduous, time-intensive, and tedious. The tedium and time consumption stems from the sheer workload, particularly when dealing with large assignments across multiple sections of the same course. Challenges emerge from the realization that there are usually many correct approaches (and infinitely many incorrect ones) to developing each query, with incorrect submissions deserving partial credit based on their closeness to a correct solution. As an example, consider a simple query such as "Name the freshmen computer science majors who are taking Calculus I". This can be correctly written in a wide variety of ways: as a standard query with multiple tables in the `FROM` clause, as a nested `EXISTS` query, as a nested `IN` query, as a combination `IN`/`EXISTS` query, as a `JOIN USING`, as a `JOIN ON`, as a `NATURAL JOIN`, with or without aliasing field names and/or table names, etc. These challenges multiply with the complexity of the assigned query sets.

Automating the grading process can offer a potential time- and effort-saving solution. However, many existing automated approaches lack the necessary sophistication to consistently and accurately assign partial credit for each query.

Typically, automated grading systems adopt either dynamic analysis, where queries are executed against fixed datasets and compared with predefined correct results, or static analysis, which assesses the structure of SQL statements without execution. Newer systems combine both approaches into a hybrid model [23].

Our approach builds upon these foundational concepts and preliminary work [18,16] and introduces significant enhancements in the form of an expanded dataset and the integration of LIME (Local Interpretable Model-agnostic Explanations) for increased explainability. The expanded dataset includes a more diverse array of SQL statements collected from a broader range of academic institutions and encompasses a wider variety of SQL syntax and structures. This allows our model to better understand and predict the nuances of correct and partially correct SQL queries across different contexts and complexities.

Furthermore, to address the critical need for transparency in AI-driven educational tools, we have incorporated LIME into our model [14]. LIME provides clear explanations of our model's decisions, which is essential for educators and students. By elucidating the reasoning behind each grading decision, LIME helps demystify the often opaque processes of machine learning models. This not only aids in building trust in the automated system but also assists educators in identifying specific areas where students may need further guidance or clarification.

These enhancements are designed to improve the robustness and utility of our automated grading system, making it not only more efficient but also more accountable and accessible to users. By expanding our dataset and integrating explainability features, our model will not only maintain high accuracy in grading but also provide insights that support learning outcomes.

The rest of the paper is organized as follows: Section 2 reviews the necessary background material. Section 3 details our methodology, including the expanded dataset and the integration of explainability through LIME. Section 4 provides an overview and evaluation of our experiments, highlighting the results. We discuss our results in Section 5, while future work and conclusions are drawn in Sections 6 and 7, respectively.

2 Background and Related Work

2.1 State of the art in automated SQL grading

Automated systems for grading SQL queries generally adopt one of two methodologies: dynamic analysis or static analysis. Dynamic analysis involves executing queries against fixed datasets and comparing the results with predefined correct answers. Early examples of this approach include SQLator [17], SQLify [6], and AsseSQL [13]. More recently, systems like SQL Tester and ASQLAG have been developed. SQL Tester provides an online tool that compares a student query with an answer-key query, but the comparisons are case-sensitive, and the results must appear in the same order [10]. ASQLAG uses an object-oriented design technique with an MVC (Model-view-controller) framework, which is capable of grading assignments across various DBMS platforms [19]. These dynamic analysis systems may suffer from inaccuracies, particularly when incorrect queries produce results identical to those of the answer key or when differences in output format affect grading. For example, consider the following query: Name the suppliers who ship green parts. It might be the case the suppliers who ship green parts happen to have `supplierIDs` that are between 100 and 200. A query based on these `supplierIDs` might produce the same results but certainly should not earn the same (or any) credit. For example, consider queries that produce an empty result table as a correct answer. There are infinite queries that fall into this category, few of which should earn any credit. While dynamic analysis does lack the ability to accurately award credit for correct answers, this approach is good at identifying incorrect queries, which is very useful.

Static analysis, on the other hand, evaluates query structures without executing the query itself. Examples of this approach include a system that compares student queries with answer-key queries through the use of string similarity metrics [21] and automated SQL provers like Cosette [4], which encode queries into logic formulas to determine logical equivalence. Building upon this methodology, Chu et al. introduced an innovative unbounded semiring (U-semiring) approach to further assess the equivalences of SQL queries. This approach is able to provide more refined feedback to students, but requires that all possible correct queries be included in the "answer key". The student query is compared to all of the answer-key queries, then the student is awarded the highest score from the comparisons [5].

Hybrid approaches, which combine dynamic and static analysis, aim to mitigate the limitations of each approach and facilitate the awarding of partial credit for incorrect queries. Some of these systems [23] employ the dynamic approach to identify incorrect queries, then use the static approach to determine how much partial credit to award. The XData system [2,3,1] generates datasets tailored to identify common query-development errors, then uses the static approach, referred to as edit-based grading, to evaluate the SQL queries to determine the necessary changes to transform the submitted query into an equivalent correct form. Nayak et al. present an approach that compares student queries and answer-key queries using feature similarities based on semantic similarities, string-based similarities, and vector similarities [12]. Fabijanic and Mekterovic describe an automatic assessment system of SQL queries that utilizes both approaches to award partial credit and give feedback to students describing the parts of the query that the system identifies as being incorrect [7].

Our system takes a different approach by leveraging AI to model SQL statements with a novel architecture [16,18].

2.2 Importance of explainability of AI-based tools in higher-ed

Explainability in AI-based tools, especially in higher education and automated grading, is important for many reasons that affect the education field. Machine learning models in educational technology have grown, improving the ability to explain, predict, and understand actions within systems and human interactions [25]. This growth in AI technology use in higher education, particularly in automated grading systems, has increased rapidly, especially during the pandemic [11]. AI in educational tools like automated grading has shown it can make the education process more efficient and effective [24].

AI-based tools must be explainable to ensure clarity and accountability in decision-making, especially in automated grading systems in higher education. Educators and students must understand and interpret AI outputs to trust the results and provide helpful feedback [11]. Also, explainable AI in education can help find biases, mistakes, or limits in the algorithms, supporting fair and equal grading [25]. Clear and understandable AI models help schools keep academic standards and ensure grading is consistent and reliable.

Explainability in AI tools for automated grading also helps identify student learning patterns, allowing teachers to tailor their teaching methods to meet individual needs [25]. This personalized education approach can boost student engagement, performance, and learning outcomes. Moreover, explainable AI gives insights into the grading criteria, helping students better understand the assessment process and meet expectations [24].

In higher education, where critical thinking and analysis skills are key, explainability in AI grading tools is essential. Understanding how AI makes grading decisions helps students improve their self-sufficiency and deepen their subject knowledge [24]. This clarity benefits students and enables teachers to improve their methods based on AI feedback, leading to ongoing enhancements in education.

3 Enhancements to the Methodology

3.1 Expanded Dataset

The dataset used in our research primarily consists of SQL statements initially collected for the purpose of grading automatization [8]. The original data, available online at `https://zenodo.org/records/6526769`, includes a detailed database schema used in student exercises and assignments, along with tables containing anonymized student submissions, feedback, grades, pass/fail flags, and other relevant information. This original dataset encompasses 675 distinct samples of student submissions, with approximately 58% graded as correct, indicating a potential class imbalance. The grades and remarks (Correct, Partially Correct, Non-Interpretable, and Cheating) provide a rich basis for training our model to recognize and evaluate SQL statements accurately across different schemas and complexities. This led to our preliminary results reported in [18,16].

To enhance our model's ability to generalize across different student levels and SQL schema complexities, we have augmented our dataset with an additional 4,723 samples from coursework assignments from a college-level introductory database course at Marist College. This expansion was driven by the need to mitigate the risks of poor generalization that were evidenced in the earlier model's performance.

The new samples include a broader range of SQL queries, varying in both complexity and structure, thereby providing a more comprehensive training ground for our models. This diversity ensures that the trained model can adapt to a wider array of real-world educational settings, improving its effectiveness in automated SQL statement grading.

The rationale for selecting coursework assignments from Marist College revolves around the observed need to diversify the types of SQL queries and database schemas. By integrating these additional samples, we aim to reduce model bias and enhance predictive accuracy, especially in scenarios that were not adequately covered by the original dataset.

These augmented datasets, combined now totaling 5,398 SQL samples, are preprocessed using standard word tokenization [22] to maintain consistency with the initial training procedures. Each SQL statement is then tokenized into a vocabulary of 1480 items, with all tokenized SQL statements pre-zero-padded to a uniform length of 219 tokens, as shown in our dataset overview in Table 1.

Table 1. Sample Data Extracted From Expanded Dataset

Submitted Answer	**Correct?**	**Remark**	**Grade**
SELECT DISTINCT s.Price, ...	1	Correct	100
SELECT v.vnum, p.pnum, s.snum...	0	Partially	20
⋮	⋮	⋮	⋮
Total count: 5,398	Avg: 0.58	Total: 4	Avg: 85

A representative sample of the length is the following SQL statement:

```
SELECT md.title,
       md.production_year,
       md.first_name,
       md.last_name
from (movie
     NATURAL JOIN director
     NATURAL JOIN person) as md,
     (SELECT title,
             production_year
     FROM movie_award
     WHERE lower(award_name)='bafta film award'
     AND lower(category)= 'best film'
     AND lower(RESULT)= 'won') as ma
Where ma.title = md. title
and ma.production_year = md.production_year;
```

In this sample, only indentation was added for reading clarity. The rest of the statement was not altered.

With this enhanced dataset, the model training now employs a parameter-sharing strategy, adapting to predict not only the correctness of SQL submissions but also detailed remarks and grading assessments. This approach allows for dynamic adjustment of the model's parameters based on a richer set of training data, ensuring robustness and accuracy in grading.

3.2 Model Reconfiguration

Our earlier neural model [18,16] comprises several components integral to processing and learning from SQL statement sequences. Each part contributes to the model's ability to understand and evaluate SQL based on different criteria.

The neural architecture designed for automated SQL statement grading encompasses several key components aimed at refining the model's interpretive and predictive capabilities. Starting with an **Embedding Layer**, the model converts SQL tokens into 64-dimensional vectors, resulting in an output of $\mathbb{R}^{172\times 64}$. This is followed by a **Self-Attention CNN**, which includes convolutional layers for modeling queries and values, employing dot product similarity to assess contextual relationships, and culminates in a global average pooling that condenses the data into a 200-dimensional vector representing the SQL context. **Regularization Techniques** such as a 25% dropout and batch normalization ensure robustness by preventing overfitting and stabilizing the learning process. A **Bottleneck Layer** facilitates the visualization of data in two dimensions, enhancing interpretative analysis. The architecture concludes with three specialized **Output Layers** for predicting SQL correctness (**Model C**), remark types (**Model R**), and grading scores (**Model G**), using sigmoid activations to cater to the distinct aspects of SQL assessment. This comprehensive structure not only ensures nuanced understanding and evaluation of SQL queries but also enhances the model's generalization across varied tasks by sharing parameters up to the bottleneck layer.

Modifications to Neural Architecture To effectively leverage the increased diversity and volume of our expanded dataset, we implemented several key modifications to our neural architecture. Originally configured with a vocabulary size of 292, we have now expanded this to 1,480 to capture a broader range of SQL syntax and semantics prevalent in the new dataset samples. This expansion necessitates adjustments in the embedding layer, which now outputs vectors in $\mathbb{R}^{172\times 1480}$, significantly enhancing the model's capacity to encode more complex SQL statements.

In addition to the vocabulary expansion, we adjusted the model's training parameters to optimize learning from the larger dataset. The learning rate was set to 0.001 to provide a balance between convergence speed and training stability. Furthermore, the batch size was increased to 64, facilitating more efficient gradient approximations and faster processing of the larger input batches [15].

Integration of LIME for Model Interpretability The interpretability of our model's predictions is crucial for educational applications, where understanding the reasoning behind grading decisions can significantly impact learning outcomes. To this end, we integrated the LIME framework using `LimeTextExplainer`, specifically tailored to handle textual data [14]. LIME, or Local Interpretable Model-agnostic Explanations, provides insights into our model's decision-making process at the instance level.

The core functionality of LIME can be succinctly described through its ability to locally approximate the decision boundary of any black-box model around a given prediction. For textual data, `LimeTextExplainer` operates by generating perturbed samples of the input text and observing the corresponding model predictions for these samples. More formally, let x represent the original SQL statement instance and $\mathcal{X}$ the domain of all possible SQL statements. LIME first constructs a neighborhood $\mathcal{N}$ around x by randomly altering segments of x, thereby creating a dataset $\{x_1, x_2, ..., x_n\}$ where each $x_i \in \mathcal{N}$.

For each perturbed sample x_i, the black-box model's prediction probability vector $\mathbf{p}_i$ is computed. LIME then approximates the local decision surface by training an interpretable model, typically a linear model, on this new dataset. The interpretable model is defined as $\hat{g}(\cdot)$, where:

$$\hat{g}(x) = \omega_0 + \sum_{j=1}^{d} \omega_j f_j(x). \tag{1}$$

Here, $\omega_0, \omega_j \in \mathbb{R}$ are the coefficients to be learned, and $f_j(x)$ are interpretable features derived from the text, such as the presence or absence of SQL keywords, tokens, or words.

To ensure the fidelity of $\hat{g}$ to the original model predictions near x, LIME minimizes a loss function that weighs the proximity of each x_i to x. This proximity weighting, denoted as $\pi_x(x_i)$, penalizes discrepancies between $\hat{g}(x_i)$ and the original model's predictions $\mathbf{p}_i$ more heavily for samples x_i that are closer to x. The optimization problem can be formally expressed as:

$$\xi(\omega) = \sum_{i=1}^{n} \pi_x(x_i) L(\mathbf{p}_i, \hat{g}(x_i)) + \Omega(\omega), \tag{2}$$

where L is a loss function measuring the prediction error, and Ω represents a complexity penalty on the model $\hat{g}$ to avoid overfitting.

By solving this optimization, LIME produces a model $\hat{g}$ that is locally faithful to the black-box model, allowing interpreters to discern which features (words or phrases) significantly influence the prediction at the instance level.

LIME blends easily with our binary and multiclass classification heads, i.e., **Model C** and **Model R**. However, restructuring is needed for our regression head to make it classification-like, with the goal of classifying an SQL statement grade into ten distinct classes, representing grade ranges from '0-10%' to '91-100%'. We modified the output of the regression head, **Model G**, for our explainer model accordingly. Each class corresponds to a specific range of grades, and the explainer assesses how each token in the SQL statement contributes to classifying into these ranges. Formally, the output of **Model G** is passed through a softmax activation as:

$$\hat{y} = \text{softmax}(\mathbf{W} \cdot \phi(\mathbf{x}) + \mathbf{b}), \tag{3}$$

where $\hat{y}$ represents the predicted probabilities of each class, $\mathbf{W}$ and $\mathbf{b}$ are the weights and biases of the classifier, and $\phi(\mathbf{x})$ denotes the feature representation of the input $\mathbf{x}$, derived from the last hidden layer of our neural network. The softmax function is used to normalize the output into a probability distribution over the ten grade ranges, which facilitates LIME's integration. Note that here, $\mathbf{x}$ is a vector of tokens that represents an input SQL statement x.

This integration of LIME not only enhances our model's transparency but also allows educators and developers to visually inspect which parts of the SQL query most influence the grading outcomes, thereby enabling targeted improvements in both teaching strategies and model refinement.

4 Experiments and Results

4.1 Updated Model Performance

In the next few paragraphs, we present the results of training the enhanced model on the extended dataset, and we compare performance metrics with the previous model [18,16] to demonstrate improvements.

In the previous version of our model, we implemented a leave-one-out (LOO) cross-validation method to estimate generalization error [9]. Therefore, the performance metrics discussed below are based on the LOO approach.

Regarding the correctness model, labeled as **Model C**, the resulting confusion matrix and Receiver Operating Characteristic (ROC) curve are displayed in Fig. 1 and Fig. 2, respectively. The confusion matrix in Fig. 1 shows a higher number of false positives (FP) than false negatives (FN). This discrepancy stems from the class imbalance noted in Table 1, where the majority of the samples belong to the correct class. Due to this imbalance, we prioritize balanced accuracy

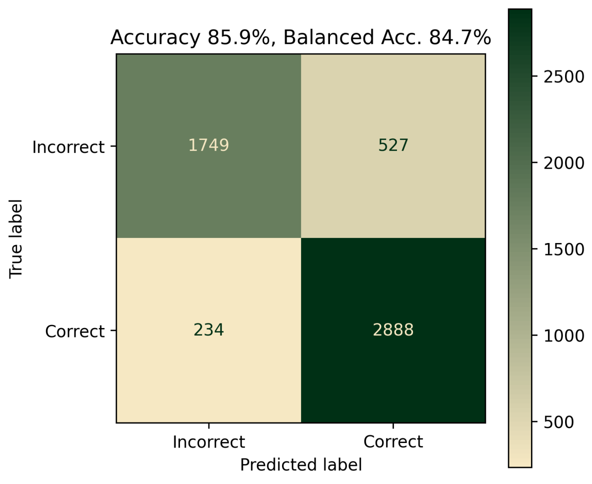


Fig. 1. Confusion matrix for the model that predicts correctness.

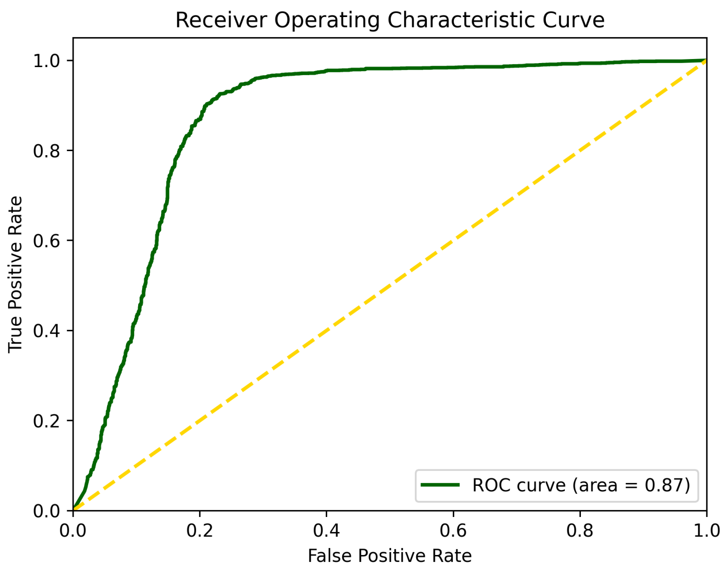


Fig. 2. ROC and AUC for the model that predicts correctness.

over traditional accuracy measures. The balanced accuracy achieved is **84.7%**, marking an improvement from the previous rate of 81.2%.

Additionally, the LOO cross-validated ROC curve, depicted in Fig. 2, confirms that the model's performance significantly surpasses random chance, achieving an Area Under the Curve (AUC) of **0.87**. This AUC indicates consistent performance across both models under evaluation.

For the remarks model, **Model R**, results obtained from the LOO cross-validation are displayed in Fig. 3. This figure illustrates that the performance for remarks indicating correctness, specifically `Correct` and `Partially Correct`, is superior to those associated with negative feedback. These findings align with prior observations that predictions related to correctness are more accurate than those for non-correctness. Fig. 3 includes the average precision (AP) score, a weighted average of precision values, alongside the F_1 curves for additional context.

The confusion matrix presented in Fig. 4 further supports these results, showing that the classes `Cheating` and `Non Interpretable` performed the worst, largely due to their smaller sample sizes. Comparatively, the overall model accuracy has improved from 65.9% to **82.4%**. As detailed in Table 2, the performance metrics for classes associated with positive feedback, such as Precision, Recall, and F_1-scores, have shown significant improvement, whereas those related to negative feedback have declined.

For the grades model, **Model G**, we analyzed performance using leave-one-out (LOO) cross-validation, with results summarized in Fig. 5 and Fig. 6. The

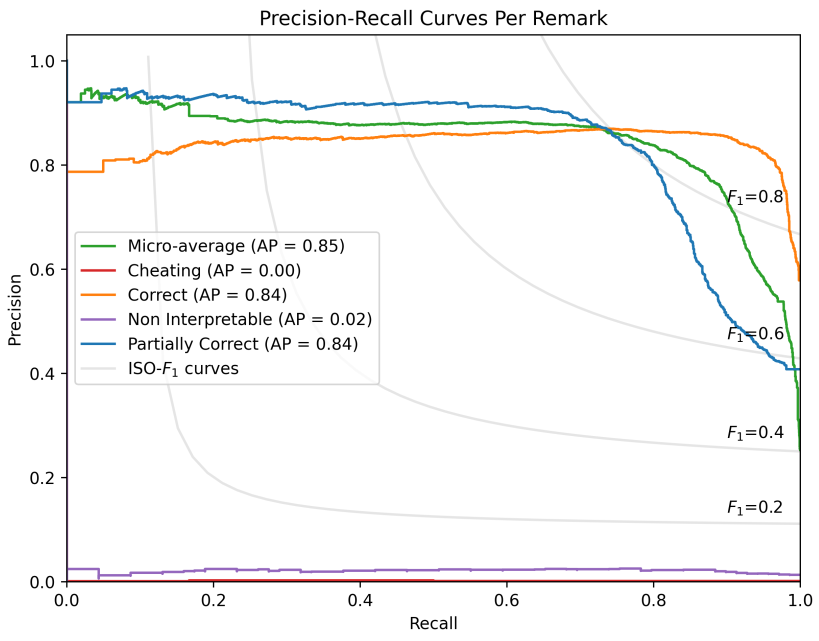
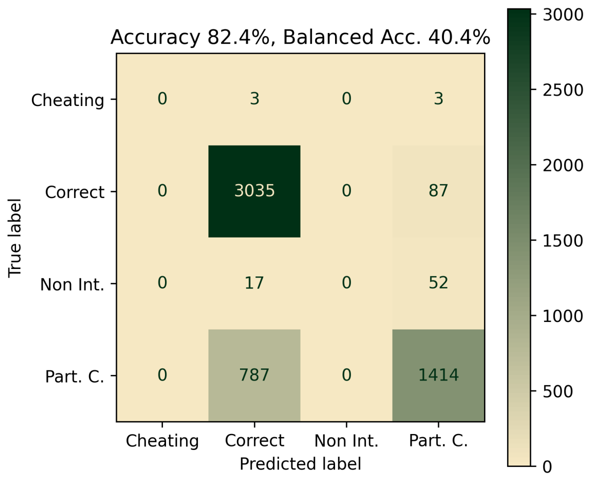


Fig. 3. Precision-Recall curves and average precision scores per remark.



True label \ Predicted label	Cheating	Correct	Non Int.	Part. C.
Cheating	0	3	0	3
Correct	0	3035	0	87
Non Int.	0	17	0	52
Part. C.	0	787	0	1414

Fig. 4. Confusion matrix for the model that predicts remarks.

regression analysis comparing predicted grades to actual grades suggests that our model tends to avoid predicting perfect grades (100%) and generally predicts more conservatively across a broader range. This tendency is reflected in the

Table 2. Classification and Regression Performance Analysis

Class Evaluated	**Evaluation Metrics**			
	Prec. Δ	***Recall*** Δ	F_1***-score*** Δ	***Support*** Δ
Incorrect	0.88 +0.12	0.77 −0.03	0.82 +0.04	2276 +1994
Correct	0.85 –	0.93 +0.11	0.88 +0.04	3122 +2729
		Accuracy	0.86 +0.05	5398 +4723
		Balanced Accuracy	0.85 +0.04	5398 +4723
Cheating	0 –	0 –	0 –	6 –
Correct	0.79 +0.01	0.97 +0.17	0.87 +0.08	3122 +2729
Non Interpretable	0 −0.26	0 −0.09	0 −0.13	69 +12
Partially Correct	0.91 +0.40	0.64 +0.06	0.65 +0.11	2201 +1982
		Accuracy	0.82 +0.16	5398 +4723
		Balanced Accuracy	0.41 +0.04	5398 +4723
Regression	R^2 Δ	***EV*** Δ	***MAE*** Δ	***MSE*** Δ
$\hat{y}$ = Grade	0.427 +0.279	0.429 +0.008	0.113 −0.120	0.029 −0.042

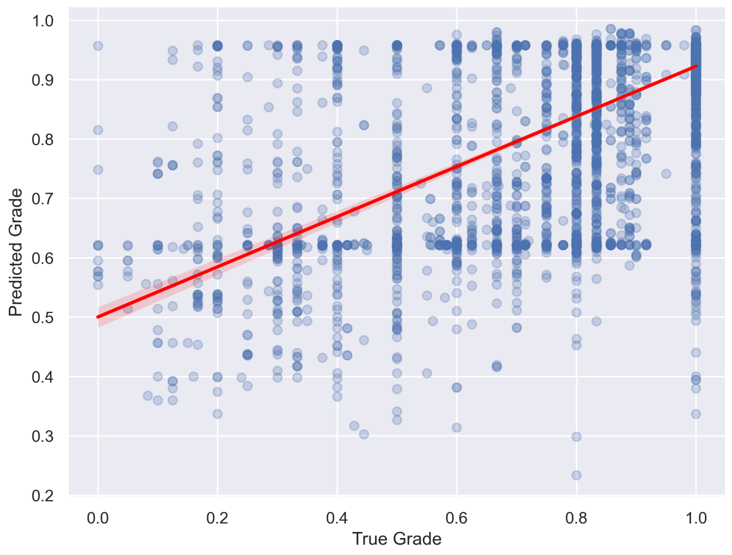


Fig. 5. True vs predicted regression plot for the model that predicts grades.

histogram of residuals, $y - \hat{y}$, shown in Fig. 6, where a slight positive skew is evident. Ideally, residuals should center around zero, indicating no systematic bias in predictions.

Despite these observations, the regression performance metrics still demonstrate the model's capability to predict grades effectively, which extends beyond merely estimating the mean of the dependent variable `Grade`. Comprehensive performance metrics for all models, including the regression analysis, are de-

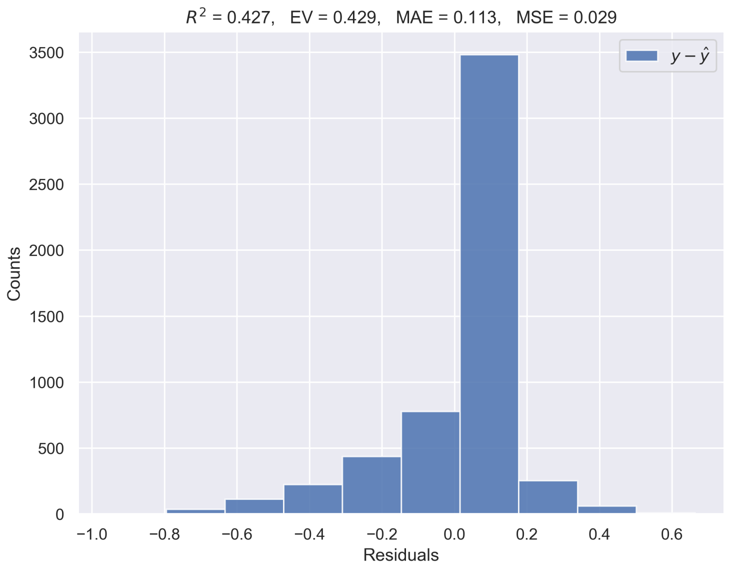


Fig. 6. Histogram of the residuals $y - \hat{y}$ for the model that predicts grades.

tailed in Table 2. This summary includes the coefficient of determination, R^2, explained variance, EV, mean absolute error, MAE, and mean squared error, MSE. Notably, error metrics have improved compared to the previous model, with the MAE reduced by more than half to **0.113**, and the MSE decreased to **0.029** from 0.071. These improvements are supported by the variance analysis, indicating that the model's predictions are substantially better than random guesses. Specifically, the R^2 value increased significantly from 0.148 to **0.479**, and the EV rose slightly from 0.421 to **0.429**.

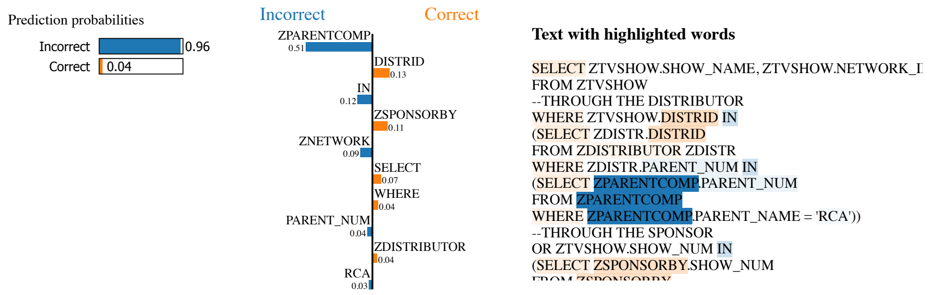
Table 2 also includes detailed metrics for each model, encompassing standard classification measures such as Precision, Recall, and F_1-score, along with standard and balanced accuracy. Any notable performance differences between the new and prior models are highlighted in the Δ column, indicating both the magnitude and direction of the changes. In general, the table shows that most of the metrics improved with the exception of those with very little data support, i.e., the ones associated with negative grading outcomes.

4.2 Explainability Assessment

In this study, the LIME framework plays a crucial role in elucidating the decision-making processes of our predictive models. Specifically, `LimeTextExplainer` is employed to generate explanations for individual predictions, particularly useful in the context of text-based input such as SQL statements or student responses. As defined earlier, LIME operates by perturbing the input data and observing the effect on the output, thus approximating the local decision boundary of the model around the input instance [14]. For each instance analyzed, LIME provides a set of features weighted according to their contribution towards the prediction, effectively highlighting which aspects of the input text most influence the model's output. This method of local interpretation is indispensable for validating model behavior and ensuring transparency, especially in educational applications where understanding the basis for automated grading decisions can significantly impact educational outcomes and student feedback.

Now, let us examine closely some examples on how LIME can provide insights into the grading decisions made by our model.

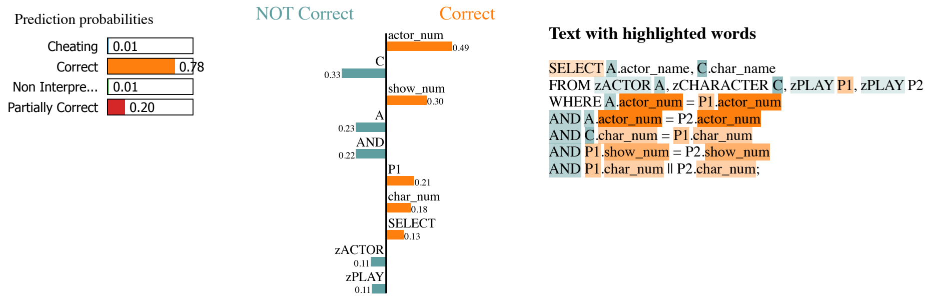
Correctness In our first case study, the LIME framework was deployed to shed light on the decision-making processes underlying the prediction of SQL query correctness. As depicted in Fig. 7, LIME revealed that several SQL elements and their associations significantly influenced the model's determination of query correctness. Notably, the feature `ZPARENTCOMP`, with a weight of 0.51, was identified as having the most substantial impact on the prediction, indicating its crucial role in the model's assessment. Other significant features included `DISTRID` and keywords like `IN`, which collectively contributed to the model's classification of the query as incorrect. This particular analysis pointed out that while the query syntactically integrated complex elements such as sub-queries and conditions, specific attributes linked to table `ZPARENTCOMP` and `ZNETWORK` led to a negative outcome. The probability scores, i.e., `Incorrect` at 0.96 and `Correct` at 0.04, underscore the model's confidence in its evaluation, demonstrating LIME's utility in explicating which features sway the model's grading decisions and thus providing a basis for learners' early understanding.



```
SELECT ZTVSHOW.SHOW_NAME, ZTVSHOW.NETWORK_I
FROM ZTVSHOW
--THROUGH THE DISTRIBUTOR
WHERE ZTVSHOW.DISTRID IN
(SELECT ZDISTR.DISTRID
FROM ZDISTRIBUTOR ZDISTR
WHERE ZDISTR.PARENT_NUM IN
(SELECT ZPARENTCOMP.PARENT_NUM
FROM ZPARENTCOMP
WHERE ZPARENTCOMP.PARENT_NAME = 'RCA'))
--THROUGH THE SPONSOR
OR ZTVSHOW.SHOW_NUM IN
(SELECT ZSPONSORBY.SHOW_NUM
```

Fig. 7. Explained predictions of the SQL grading model highlighting key contributions of input features towards the model's decision-making.

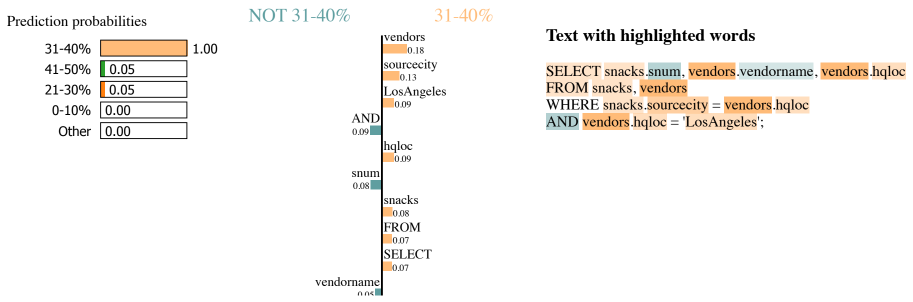
Remarks In the case study shown in Fig. 8, LIME was utilized to uncover the determinants behind the model's decision-making for feedback on SQL queries. For an intricate SQL query involving multiple tables and conditional statements, LIME's analysis can be instrumental for a student. The model positively evaluated the query, largely attributing the remark of `Correct` to the effective use of conditions linking multiple tables. Key attributes such as `actor_num` and `show_num` significantly influenced the model's favorable feedback, with LIME highlighting their impacts with weights of 0.49 and 0.30 respectively. This analysis by LIME revealed that the model recognized the complexity and correctness of the multi-table conditions. The probabilities provided by LIME, i.e., `Correct` at 0.78 and `Partially Correct` at 0.20, reflect the model's ability to appreciate the syntactical structure and logical coherence of the query. Such insights are invaluable as they illustrate the specific factors that models use to assess SQL queries, enhancing transparency and aiding in the educational process by clarifying how certain SQL constructs impact automated grading decisions.



```
SELECT A.actor_name, C.char_name
FROM zACTOR A, zCHARACTER C, zPLAY P1, zPLAY P2
WHERE A.actor_num = P1.actor_num
AND A.actor_num = P2.actor_num
AND C.char_num = P1.char_num
AND P1.show_num = P2.show_num
AND P1.char_num || P2.char_num;
```

Fig. 8. Explained predictions of the SQL grading model highlighting key contributions of input features towards the model's decision-making.

Grades In the case study shown in Fig. 9, the LIME framework was utilized to provide interpretative insights into a model's decision-making process for grading SQL queries. The figure demonstrates LIME's explanation for a specific prediction where the model evaluated an SQL query aimed at selecting and correlating data across multiple tables. According to LIME's output, certain SQL keywords and table attributes significantly influenced the model's grading decision. For instance, terms such as `vendors` and `sourcecity` were highlighted as highly influential, with positive weights of 0.18 and 0.13, respectively, suggesting their correct usage in the context of the query supports a higher grade classification. Conversely, the presence of other elements like `snum` and `vendorname` contributed negatively but with slightly lesser weights. This example clearly showcases how LIME helps pinpoint specific features within student submissions that either bolster or detract from the perceived correctness and quality of the SQL queries, thus offering students and educators an automated understanding of grading decisions.



```
SELECT snacks.snum, vendors.vendorname, vendors.hqloc
FROM snacks, vendors
WHERE snacks.sourcecity = vendors.hqloc
AND vendors.hqloc = 'LosAngeles';
```

Fig. 9. Explained predictions of the SQL grading model highlighting key contributions of input features towards the model's decision-making.

5 Discussion and Future Work

The results of implementing automated SQL statement grading reveal significant implications for the efficiency and accuracy of educational assessments and learners' early feedback. The integration of explainability tools like LIME has benefits, such as increasing the accessibility of the model's decision-making process, clarifying how decisions are derived, and empowering users to self-monitor their understanding.

However, the introduction of a larger dataset and new methods of explainability does not come without challenges. Potential biases inherent in the expanded dataset and the limitations of current explainability tools could skew the model's accuracy against underrepresented data instances. These biases need careful consideration to avoid reinforcing existing class imbalances.

Looking forward, there is room for improvement in our model. Further research could explore the integration of other explainability tools beyond LIME or Captum [20], for example, SHAP-based explainability, potentially providing a more rounded understanding of model decision-making. Additionally, expanding the dataset to include a wider array of SQL queries and conditions would likely improve the model's robustness and generalization capabilities.

6 Conclusions

This research has made significant contributions to the field of automated SQL statement grading by expanding the training dataset and incorporating explainability methods like LIME. These enhancements have not only improved the model's accuracy but have also made its inference process more transparent and interpretable. The results indicate that with more data, the model's ability to correctly assess SQL queries has increased, reflected by higher balanced accuracy and AUC values. The use of LIME has elucidated how specific elements within SQL statements affect grading decisions, fostering skill improvement among users. This shift towards more interpretable and user-centered AI tools is especially pivotal in educational contexts, where clear insights into automated assessments can enhance teaching strategies and student learning. Furthermore, these advancements encourage ongoing research to explore additional explainability tools and further dataset augmentation to refine the model's performance and reliability. The journey towards fully transparent and accountable AI in education continues to evolve, promising advances in how technology can support educational outcomes.

Acknowledgments. Part of this work was funded by the National Science Foundation under grants CNS-2210091, CHE-1905043, and CNS-2136961.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Chandra, B., Banerjee, A., Hazra, U., Joseph, M., Sudarshan, S.: Automated grading of sql queries. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE). pp. 1630–1633. IEEE (2019)
2. Chandra, B., Banerjee, A., Hazra, U., Joseph, M., Sudarshan, S.: Edit based grading of sql queries. In: 8th ACM IKDD CODS and 26th COMAD, pp. 56–64 (2021)
3. Chandra, B., Joseph, M., Radhakrishnan, B., Acharya, S., Sudarshan, S.: Partial marking for automated grading of sql queries. Proceedings of the VLDB Endowment **9**(13), 1541–1544 (2016)
4. Chu, S., Li, D., Wang, C., Cheung, A., Suciu, D.: Demonstration of the cosette automated sql prover. In: Proceedings of the 2017 ACM International Conference on Management of Data. pp. 1591–1594 (2017)
5. Chu, S., Murphy, B., Roesch, J., Cheung, A., Suciu, D.: Axiomatic foundations and algorithms for deciding semantic equivalences of sql queries. arXiv preprint arXiv:1802.02229 (2018)
6. Dekeyser, S., de Raadt, M., Lee, T.Y.: Computer assisted assessment of sql query skills. In: Proceedings of the 18th Australasian Database Conference (ADC 2007). vol. 63, pp. 53–62. Australian Computer Society Inc. (2007)
7. Fabijanić, M., Mekterović, I.: Partial sql query assessment. In: 2023 46th MIPRO ICT and Electronics Convention (MIPRO). pp. 1317–1322 (2023). https://doi.org/10.23919/MIPRO57284.2023.10159706, https://api.semanticscholar.org/CorpusID:259299956
8. Jinshui, W.: Combining dynamic and static analysis for automated grading SQL statements (May 2022). https://doi.org/10.5281/zenodo.6526769, https://doi.org/10.5281/zenodo.6526769
9. Kearns, M., Ron, D.: Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. Neural computation **11**(6), 1427–1453 (1999)
10. Kleerekoper, A., Schofield, A.: Sql tester: an online sql assessment tool and its impact. In: Proceedings of the 23rd annual ACM conference on innovation and technology in computer science education. pp. 87–92 (2018)
11. Miškovic, V.: Explainable machine learning methods as a tool for higher education improvement. Zbornik Radova Univerziteta Sinergija **22** (2021). https://doi.org/10.7251/zrsng2101001m

12. Nayak, S., Agarwal, R., Khatri, S.K., Mohammadian, M.: Student outcome assessment on structured query language using rubrics and automated feedback generation. International Journal of Advanced Computer Science and Applications **15**(3) (2024). `https://doi.org/10.14569/IJACSA.2024.0150374`, `http://dx.doi.org/10.14569/IJACSA.2024.0150374`
13. Prior, J.C., Lister, R.: The backwash effect on sql skills grading. ACM SIGCSE Bulletin **36**(3), 32–36 (2004)
14. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should I trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016. pp. 1135–1144 (2016)
15. Rivas, P.: Deep Learning for Beginners. Packt Publishing Ltd (2020)
16. Rivas, P., Schwartz, D.R.: Modeling sql statement correctness with attention-based convolutional neural networks. In: 2021 International Conference on Computational Science and Computational Intelligence (CSCI). pp. 64–71 (2021). `https://doi.org/10.1109/CSCI54926.2021.00086`
17. Sadiq, S., Orlowska, M., Sadiq, W., Lin, J.: Sqlator: an online sql learning workbench. In: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education. pp. 223–227 (2004)
18. Schwartz, D.R., Rivas, P.: 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. pp. 1–12 (2022)
19. Singporn, P., Vichianroj, P., Trongratsameethong, A.: Asqlag-automated sql assignment grading system for multiple dbmss. Journal of Technology and Innovation in Tertiary Education **1**(1), 41–59 (2018)
20. Sooksatra, K., Khanal, B., Rivas, P., Schwartz, D.R.: Attribution scores of bert-based sql-query automatic grading for explainability. In: 10th Annual Conf. on Computational Science & Computational Intelligence (CSCI'23). pp. 1–7 (2023)
21. Štajduhar, I., Mauša, G.: Using string similarity metrics for automated grading of sql statements. In: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). pp. 1250–1255. IEEE (2015)
22. Vijayarani, S., Janani, R., et al.: Text mining: open source tokenization tools-an analysis. Advanced Computational Intelligence: An International Journal (ACII) **3**(1), 37–47 (2016)
23. Wang, J., Zhao, Y., Tang, Z., Xing, Z.: Combining dynamic and static analysis for automated grading sql statements. J Netw Intell **5**(4), 179–190 (2020)
24. Wang, L., Liu, L., Zu, A., Lv, X., Mei, Z., Wang, Z., Chen, Y.: A novel educational tool helps teach intestinal absorption in physiology. Ajp Advances in Physiology Education **45**, 24–30 (2021). `https://doi.org/10.1152/advan.00035.2020`
25. Yang, C., Chiang, F., Cheng, Q., Ji, J.: Machine learning-based student modeling methodology for intelligent tutoring systems. Journal of Educational Computing Research **59**, 1015–1035 (2021). `https://doi.org/10.1177/0735633120986256`