Modeling SQL Statement Correctness with Attention-Based Convolutional Neural Networks

Pablo Rivas , *Senior, IEEE*
School of Engineering and Computer Science
Department of Computer Science, Baylor University
Email: Pablo_Rivas@Baylor.edu

Donald R. Schwartz
School of Computer Science and Mathematics
Department of Computing Technology, Marist College
Email: Donald.Schwartz@Marist.edu

***Abstract*—Automated grading of SQL statements is a topic of interest for instructors and students alike. It can give teachers additional time to be more effective in identifying issues quickly, and it can give students a preview of the grade they may receive. Existing attempts in the literature create models based on a variety of methodologies that exploit the structure of SQL statements and model answers; however, very few have leveraged the recent advances in deep learning. This paper employs a convolutional self-attention mechanism to learn complex contextual and grammatical dependencies directly from data of labeled SQL statements. Our experiments suggest that the proposed parameter-sharing strategy can adequately model the problem of detecting the correctness of an SQL statement with a balanced accuracy of up to 81.2% and an AUC of 0.87 in cross-validation.**

***Index Terms*—attention, convolutional networks, sql, natural language processing, machine learning, deep learning**

I. Introduction

Grading SQL queries can be tedious, time-consuming, and challenging. The tediousness and time required arise from the sheer volume of work, especially if it involves larger assignments from multiple sections of the same course. The challenges come from the realization that there are often multiple correct ways (and infinitely many incorrect ways) to write each query and that incorrect queries should earn partial credit, based on how closely the submitted query matches a correct answer. The challenge increases as the complexity of the assigned set of queries increases.

Automating the grading of these queries can save time and effort, but many automated approaches lack the sophistication needed to award credit accurately and consistently for each query. Automated grading systems are generally based on one of two approaches: dynamic analysis (which run queries against fixed data sets and compares the results of a submitted query with the results from the answer key) and static analysis (which analyzes the submitted SQL statement and compares it to the answer query) [1]. Some newer systems combine these approaches into a hybrid approach.

We take a different approach. Namely, we employ machine learning to model the SQL statement with supervised learning using a parameter-sharing approach [2]. This learning mechanism takes advantage of training the same model for different, yet similar outcomes, adjusting its parameters in order to learn the non-trivial relationships among SQL statements syntax based on students answers to class exercises released in [3]. The model we propose uses a self-attention mechanism [4], in combination with a convolutional neural network that will analyze the spatially correlated elements of a SQL statement and learn to pay attention to all the different aspects of the syntax.

The proposed model is a prototype that is supervised according to three different criteria: the correctness of the statement, a grader's remark or comment about the statement, and an assigned grade. Therefore, a single model trained over different alternating cycles for the different criteria shares knowledge about SQL statements. These kinds of models have reportedly performed well in natural language processing tasks where the syntax and grammars are complex, most notably in neural machine translation [5]. Thus, this type of model should achieve performances beyond random chance in modeling the problem of SQL statement modeling for automatic grading tasks, as we show in this paper.

The rest of the paper is organized as follows. The associated background material is reviewed in Section II. In section III, we present our methodology, including a description of the dataset we used, a layout of the neural architecture we developed, and an overview and evaluation of the experiments we ran, including the results. Conclusions are presented in Section IV.

II. Background

In this section, we give a brief overview of related work in the areas of automated SQL statement grading and attention-based convolutional neural networks (CNNs).

A. Related work in automated SQL statement grading

Automated SQL query grading systems are generally based on one of two approaches, dynamic analysis or static analysis. The dynamic analysis approach runs queries against fixed data sets and compares the results of the submitted query with the results from the answer key queries [1]. Early examples of this approach include SQLator [6], SQLify [7], and AsseSQL [8]. More recently, Kleerekoper, *et al.* developed SQL Tester, an online interactive tool that compares a submitted query's results to a correct result, but requires that the results must be in the same order and pass a case-sensitive comparison [9]. Trongratsameethong and Vichianroj have introduced ASQLAG, which is a system that can grade assignments submitted across

an array of DBMS platforms, including MySQL, MariaDB, PostgreSQL and Microsoft SQL Server. Their system uses an object-oriented design technique with an MVC (Model-view-controller) framework [10]. This approach can be susceptible to inaccuracies in grading, because incorrect queries might still produce results identical to those of the answer-key query, especially if the correct answer happens to be an empty table. Another consideration concerns the prospect that if a submitted answer includes an extra field, the resulting table might be judged to be vastly different than the answer results.

The static analysis approach evaluates the structure of the SQL queries without actually running the queries. An early example of this approach is [11], which employed a matrix, called a tableau, to represent various relational expressions, and used the tableau to develop equivalence classes for the expressions. Štajduhar, *et al.* developed an approach that uses string similarity metrics to compare submitted queries to answer-key queries [12]. Cosette, an automated SQL prover, encodes SQL queries into logic formulas to determine whether the queries are logically equivalent [13]. Improving on this approach, Chu *et al.* [14] developed an unbounded semiring (U-semiring) approach for further determining the equivalences of SQL queries. This approach has the advantage of being able to provide more-detailed feedback to the students, but can be a much more complex approach, especially considering the fact that there are often multiple correct ways to write the same query. As such, the answer-key would need to include multiple correct examples of the same query, so that the system may compare the student's query with each correct example, ideally giving the student the highest generated score.

Newer systems often employ a hybrid approach, which attempts to minimize the problems associated with each approach and allows the system to more easily award partial credit for incorrect submissions. Wang, *et al.* [1] describe a system which uses the dynamic approach to determine whether query submissions are correct, then uses a static approach on the incorrect queries in order to award partial credit. Their static approach grades the submitted query against multiple answer-key queries, then awards the highest mark to the submitted query. Chandra *et al.* present their XData system [15]–[17], which dynamically evaluates queries by generating datasets tailored to identify common query-development errors. It then utilizes the static approach (which they call edit-based grading) to evaluate the SQL query to determine the number of changes that may be required to transform the submitted query into one that is equivalent to a correct query.

Our system leverages the recent advances in deep learning that enable a model to learn from data to detect whether an SQL statement is correct, what short feedback it should receive, and what grade a grader would give. This is achieved using attention-based CNNs, which we introduce next.

B. Related work in attention-based CNNs

Attention mechanisms have demonstrably shown exceeding superiority when used in combination with other models that require information about contextual relationships [18]. For example, Wu *et al.* [4], combined an attention mechanism with a convolutional neural network (CNN). The authors took advantage of the superior capabilities of a CNN to capture spatial relationships and added additional context with an attention mechanism to model magnetic resonance images. The study shows that such a mechanism can improve the reconstruction ability of traditional algorithms. Likewise, in 2020, Li *et al.* [19] used a similar approach but for denoising purposes in computer tomography scans. The authors leveraged the capabilities of self-attention and CNNs to learn inter- and intra-slice scans.

These models not only find application areas in medical imaging, but they have also impact in areas that involve time-series data, for example in [20]. Fahim *et al.*, in 2020, looked at the problem of power transmission line faults and how the problem could be modeled using self-attention CNN models. The authors looked at data that was sequential in time and were able to classify about a dozen different types of outcomes. Another example of the diversity of applications is the research by Zeng *et al.* in 2020 [21]. The researchers used an attention model and a CNN to detect leaf disease in a variety of conventional images of leafs. The authors reported performance improvements over other standard classifiers.

What these models have in common is the properties of input that they receive: the data is highly correlated and needs contextual information. CNNs provide the solution to spatial correlation analysis, regardless if the space is two or three dimensional, e.g. an image, a time-based sequence, e.g. power consumption or music, or even a language sentence, e.g. an English sentence or an SQL statement. The self-attention mechanism is what provides contextual information about the data within the context of itself.

To the best of our knowledge no other approaches have used self-attention mechanisms and CNNs to model SQL statement grading. Although [22] and [23] have used these kinds of models in relation to SQL, the authors have significantly different data, aims, and goals. These distinctions become clear in the next section.

III. Methodology

In this section, we present our methodology, including a description of the dataset we used, a layout of the neural architecture we developed, and an overview and evaluation of the experiments we ran, including the results.

A. Dataset

The dataset used in our research consists of SQL statements collected for the purpose of grading automatization [3]. The data is available online: https://zenodo.org/record/3889635#.YWZU5dnMJfV. The dataset creators released a database schema that describes the database used in exercises or assignments for students; it also contains a number of tables containing anonymized student submissions, feedback, grades, pass/fail flags and other information. What we have gathered as useful information from this dataset is exemplified in Table I.

TABLE I
SAMPLE DATA EXTRACTED FROM DATASET

Submitted Answer	Correct?	Remark	Grade
SELECT production_year FROM...	1	Correct	100
SELECT count(*) FROM writer...	0	Partially	20
⋮	⋮	⋮	⋮
Total count: 675	Avg: 0.58	Total: 4	Avg: 79

Table I offers summaries at the bottom indicating the following: a) that we have 675 distinct samples of student submissions; b) that in average 58% of the submissions are graded as correct, which indicates a class imbalance; c) that there are four different remarks, namely `Correct`, `Partially Correct`, `Non-Interpretable`, and `Cheating`; d) that the average grade of the assignments is 79.

This dataset will be used to train a machine learning model on SQL statements that can be relatively small or large. A representative sample of the length is the following SQL statement:

```
SELECT count(*)
FROM person p
WHERE NOT EXISTS(SELECT *
                 FROM director d
                 WHERE p.id=d.id);
```

From this kind of statement, the next step is to do standard word tokenization [24], leading to a vocabulary of 292 items. The longest SQL statement identified had a maximum of 172 tokens; consequently all tokenized SQL statements were pre-zero-padded. The final dataset size is therefore 675×172.

Using the tokenized and padded dataset, a single model will be trained iteratively, switching goals every time and sharing parameters. The three different goals are i) to predict correctness, ii) to predict a remark, and iii) to predict the grade. The details of the neural model are discussed next.

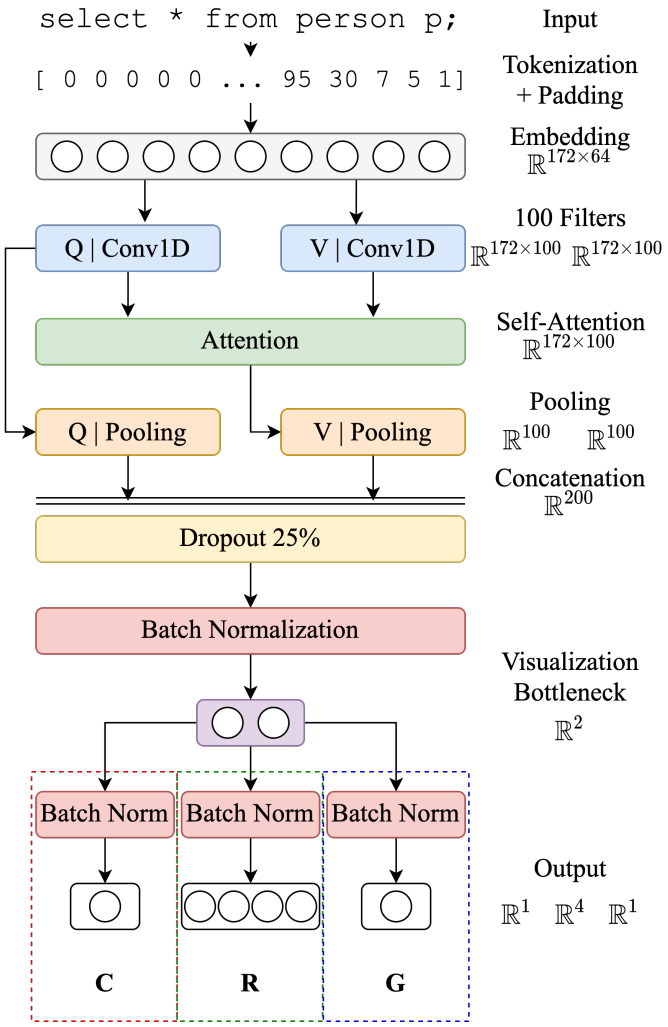

Fig. 1. Proposed neural architecture for SQL Statement grading based on self-attention, CNNs, and parameter sharing.

B. Neural architecture

The neural architecture has several different moving parts that are displayed in Fig. 1 and briefly discussed next.

1) Embedding: The embedding layer is a trainable set of neurons whose sole purpose is to take a token in the context of the entire sequence [25], [26]. In this case the sequence is in $\mathbb{W}^{172}$, and returns a vector of size 64 that represents each token in the sequence. This layer has 64 neural units and its output is in $\mathbb{R}^{172\times64}$.

2) Self-Attention CNN: The self-attention CNN is composed of three major parts. First, we have two convolutional encoding layers that model a query Q and value V. This Q, V shall not be interpreted in the context an SQL statement, instead these two elements have a different meaning in the machine learning context of attention mechanisms [18]. Ordinarily, Q and V would have different sources for traditional attention mechanisms; however, for self-attention, the input to both Q, V is the same embedding.

Second, the self-attention layer is a Luong-style layer [27]. This kind of attention uses the notion of dot product similarity as a way to discern context. That is, embeddings of Q, V that are related will have larger dot products.

Third, there is a pooling strategy that implements a global average [28]. This pooling strategy effectively reduces the dimensionality of the problem down to two vectors in $\mathbb{R}^{100}$. These are then concatenated into a single vector in $\mathbb{R}^{200}$ that can be interpreted as a vectorial representation of the SQL statement (input) that highlights the contextual relationships and meaning of the input.

Since the self-attention component is trainable, it will learn to optimize the learned embeddings for the purposes of the model. This is guided by the subsequent layers.

3) Dropout: A dropout layer reduces the chances of overfitting a model by randomly disconnecting a proportion of the connecting weights between layers [29]. In our case we used a 25% dropout rate that would randomly disconnect from the following dense layer one quarter of the incoming connections from the self-attention mechanism. This should reduce reliance on very specific elements of the embedding and would promote

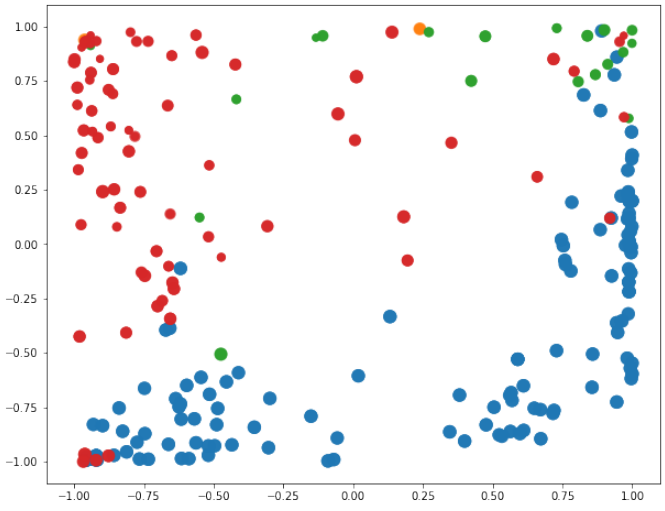

Fig. 2. Depiction of data displayed using the visualization bottleneck layer during training. Good learning leads to clearly defined clusters, or data that can be separable by nonlinear mapping functions. The different colors indicate different assigned remarks.

a holistic learning from the attention embeddings.

4) Batch normalization: Similar to dropout, batch normalization is an aid to the learning problem that brings numerical stability. It calculates optimal means and variances, μ, σ, for data batches that will help reduce most kinds of numerical instabilities that could lead to larger problems such as vanishing or exploding gradients [30].

5) Bottleneck: Given that this paper is exploratory research into understanding the capabilities of the proposed model, we are very interested in understanding how the model evolves as it is trained. This bottleneck dense layer with hyperbolic tangent activations enables us to visualize at any given point during the training where is the data in the learned representation space. The fact that there are only two neurons allows us to display the learned representations in two dimensions, $\mathbb{R}^2$, as depicted in Fig. 2.

6) Outputs C, R, and G: The bottom of Fig. 1 displays three different groups of outputs, each of which are different models that share the same parameters of the prior layers, up to the bottleneck. These models each have a batch normalization layer and dense neural units. Model **C** is for predicting correctness. It has one neural unit with a sigmoid activation and will learn to produce a close-to-zero value for an incorrect SQL statement and a close-to-one value otherwise. Model **R** is for predicting the grader's remark. It has four neural units with sigmoid activations; each neural unit corresponds to one of the four possible grader's remarks. The neuron that produced the largest output is considered the output. Similarly, model **G** is for predicting the grade. It has one neural unit with a sigmoid activation and will learn to produce a real value between 0 and 1.

Each of these models is trained iteratively as we discuss in the next section.

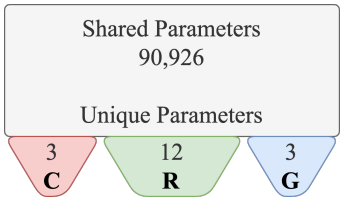

Fig. 3. Parameter sharing between models. The majority of the parameters is shared and only a few parameters need to be estimated separately.

C. Experiments, Evaluation, and Results

The model in Fig. 1 was implemented in Python using the Tensorflow platform with Keras libraries. The training discussed below was performed on an NVIDIA P100 GPU system with 25 GB of RAM and 166 GB of storage.

We performed three major experiments: 1) one that visualizes the data during the learning process on separate training intervals with the purpose of validating the learned representations; 2) a jointly trained model with the purpose of validating correctness on k-fold cross-validation; and 3) a jointly trained model that validates all models using leave-one-out cross-validation.

1) Visualizing learning process across iterations: In this process, the three models, **C, R, G**, which share most of the parameters, are trained one at a time with different loss functions. Fig. 3 illustrates the proportion of shared parameters against the parameters that are unique for each model. The relatively large difference promotes a generalized learning strategy that can be optimized using gradient descent (RMSprop optimizer [31], with learning rate of 0.001) over the loss functions defined next.

Model **C**, for correctness, uses a mean squared error (MSE) loss:

$$L(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N}(y - \hat{y}_i)^2 \tag{1}$$

where N is the number of samples, $y \in \{0, 1\}$ is the desired correctness flag, and $\hat{y} \in [0 \dots 1] \subseteq \mathbb{R}$ is the predicted correctness. The neuron in this output layer uses a sigmoid activation.

Model **R**, for remarks, uses a binary cross entropy loss on the four neurons in the output. The loss for the i-th sample is denoted as follows:

$$L(\mathbf{y}_i, \hat{\mathbf{y}}_i) = -\sum_{k=1}^{4} y_{i,k} \log_2 (\hat{y}_{i,k}) \tag{2}$$

where the vector of targets $\mathbf{y} \in \{0, 1\} \subseteq \mathbb{W}^4$ denotes the one-hot encoded version of the four possible remarks, while $\hat{\mathbf{y}} \in [0 \dots 1] \subseteq \mathbb{R}^4$ are the predicted outcomes. This model uses softmax activations.

Model **G**, for grades, is similar to model **C**. It uses the same loss function in (1) and the same activation function; however, the desired target is in an interval instead of a set of two values, namely $y \in [0 \dots 1]$ to account for the grade range.

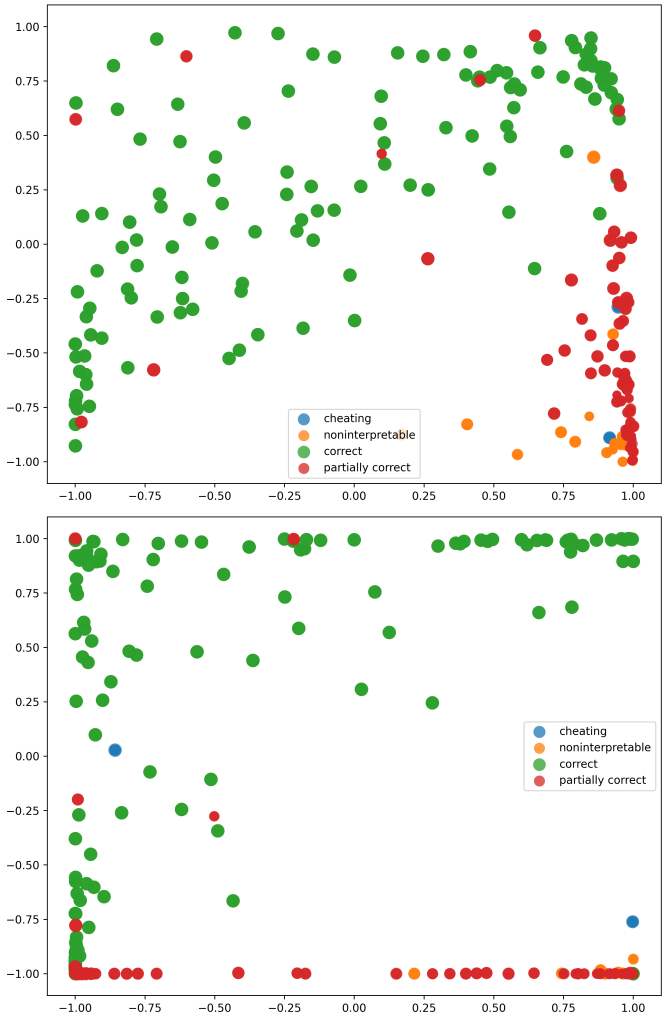

Fig. 4. Data visualization using the bottleneck layer trained with the three models **C, R, G**. Note that partially correct solutions tend to separate from correct solutions. Top and bottom are taken at different runs.

The training in this experiment occurs as follows. Gradient descent is executed for a full cycle with **C**. The full cycle is comprised of 1000 epochs using 90% of the data for training and 10% for validation. The training cycle implements a stopping criteria that monitors the validation set and stops the learning cycle if the validation loss has not decreased for 15 consecutive epochs. The training cycle also implements an automatically decreasing learning rate strategy that monitors the validation loss and if it has not improved in five epochs it decreases the learning rate by 50%. A full training cycle with these strategies is usually terminated in about 20 epochs.

After a full cycle is completed for **C**, another cycle trains **R** under the same conditions, departing from the trained model **C**. Once a full cycle is completed for **R**, another cycle completes for **G**. In total there are 90 cycles, that is 30 cycles per model, alternating models every time.

After all the training cycles are completed, we use the bottleneck layer to project the entire dataset into two dimensions; the result is in Fig. 4. The figure shows that the model can learn to separate the majority of SQL statements that are correct from those who are partially correct. Note also that non-interpretable SQL statements are located in a far extreme near the partially correct ones. This might suggest that some partially correct statements can become worse to the far right extreme of the learned space. While the location of these can change arbitrarily, such relationships are consistent. For comparison, see Fig. 5 and note how as the training cycles increase the relative position of the SQL statements is distributed according to their corresponding remark.

These visualizations in Fig. 4 and 5 suggest that the models are effectively sharing parameters and improving the learned representations across cycles. Next, we will evaluate performance on a model that is trained all at once instead of separately for each model.

2) Jointly trained model cross-validated performance: The difference between training a model separately in different cycles, for each **C, R, G**, is the way gradient descent is calculated. In the separate cycles, gradient descent only considers one model at a time and propagates updates based on the errors of the current model. On the other hand when all models are trained at once, the gradient is calculated using a single loss. For this reason the changes required are the following.

The loss function adopted globally is the binary cross entropy in (2); however, the length of $\mathbf{y}$ increased to account for the overall size of the output, which is a vector of six elements: one item for the correctness, four items for the remarks, and one item for the grade. Thus, $\mathbf{y} \in \mathbb{R}^6$ in (2) in this experiment.

We will want to measure the performance using the area under the receiving operating characteristic (ROC) curve (AUC) using 10-fold cross validation. The motivation for measuring this at this point is to make an early assessment before launching a full-scale performance evaluation.

The results of using cross validation are displayed in Fig. 6. From the figure we can assert that the model, in average across folds, performs better than random chance in all cases. The maximum AUC reported was 0.83 and the lowest was 0.58, with an average of **0.75** and relatively low variance, $\sigma = 0.09$.

Notice, however, that performing k-fold cross validation on a small number of folds can be costly in reducing the number of available data for training. This can lead to a less reliable estimate of the generalization error. Therefore, a full cross-validation analysis using a leave-one-out (LOO) strategy is justified and also required across all models, **C, R, G**, which we discuss next.

*3) LOO cross-validation on all models, **C, R, G**:* LOO cross-validation is well-known as the most robust generalization error estimator [32], particularly when it comes to small datasets such as the one we use [3].

For the correctness model, **C**, after training it as in Section III-C2 but with LOO, we obtain the results shown in Fig. 7 and 8. The confusion matrix shown in Fig. 7 indicates the slightly higher false positive (FP) count on the correctness predictions; however, recall from Table I that this is an imbalanced dataset

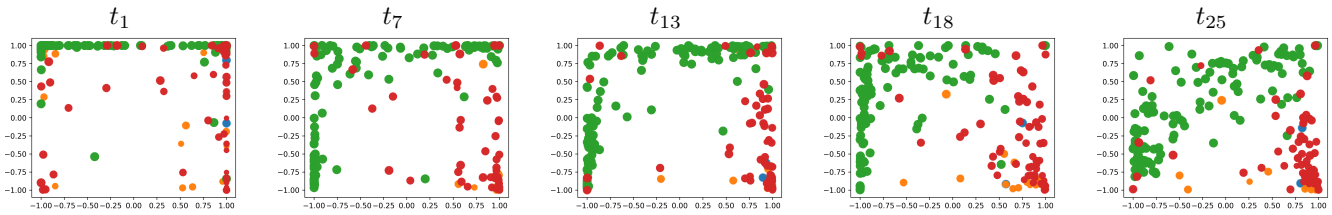

Fig. 5. Data visualization across different training cycles, t. The bottleneck layer is used to project the data into two dimensions and show the discriminant distribution of the SQL statements.

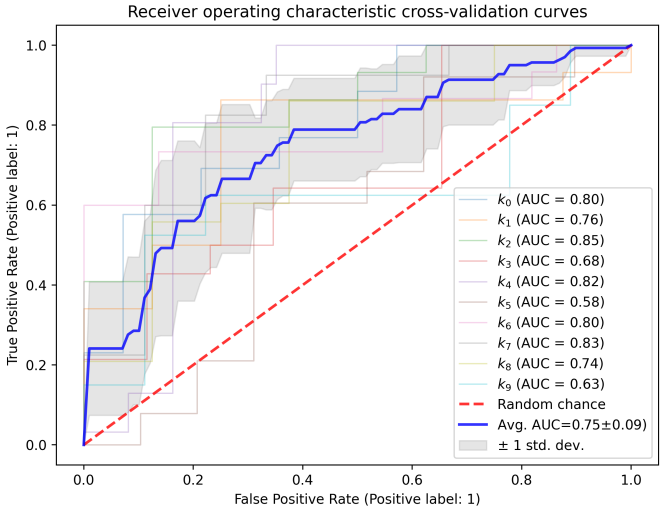

Fig. 6. ROC curves and corresponding AUC across the 10 different folds. The average AUC is 0.75. In all folds, the model performs better than random chance.

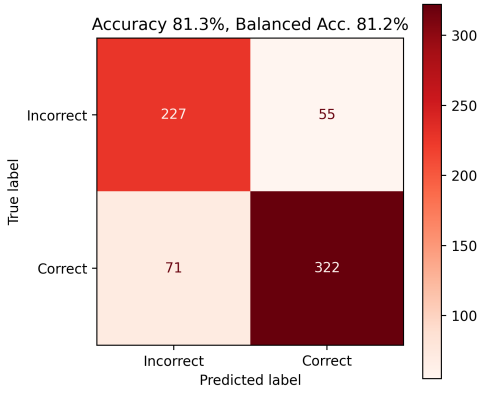

Fig. 7. Confusion matrix for the model that predicts correctness.

that favors the correct class. This imbalance can account for the FP counts in comparison the false negative (FN) counts. This imbalance obligates us to consider the balanced accuracy instead of the traditional accuracy score. In this case the balanced accuracy is **81.2%**. We also calculate the LOO cross-validated ROC and AUC, shown in Fig. 8. From the figure we observe that the model performs far beyond random chance and reaches an AUC of **0.87**.

Now we briefly look at mistakes, specifically beginning with

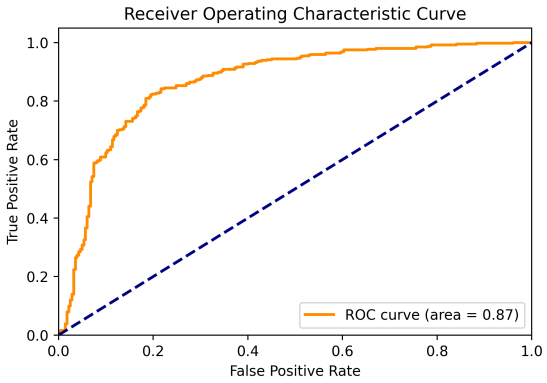

Fig. 8. ROC and AUC for the model that predicts correctness.

an example of the *best mistakes*, i.e., mistakes that were so close to the actual prediction but did not make the classification threshold by two or three decimal points, or in other words, mistakes with high uncertainty. The best mistake was:

```
select count(distinct id)
from writer
where id in (select id
             from person p
             where year_born='1935');
```

The sample above is labeled as `correct=0`, also `remark=partially correct`, and with `grade=70`. However, our model predicted it incorrectly as a `correct=1` SQL statement with the highest uncertainty, which can be explained by the high grade and partial credit given. Now lets inspect the *worst mistake*, i.e., a mistake that was made with highest confidence. Such a sample is:

```
SELECT a.id, a.award_name,
a.year_of_award, a.category ,a.result
FROM (SELECT * FROM director NATURAL
JOIN director_award) AS a
WHERE a.id IN
(SELECT p.id
FROM person p
WHERE p.id NOT IN
(SELECT id FROM (director NATURAL
JOIN (SELECT * FROM director_award d
WHERE lower(d.result)="won")))
AND p.id IN
(SELECT id FROM (director NATURAL
JOIN (SELECT * FROM director_award d
WHERE lower(d.result)="nominated"))));
```

The sample above is labeled as `correct=0`, also `remark=partially correct`, and with `grade=98`. Once again

TABLE II
CLASSIFICATION AND REGRESSION PERFORMANCE ANALYSIS

Class Evaluated	**Evaluation Metrics**			
	Precision	***Recall***	***F_1-score***	***Support***
Incorrect	0.76	0.80	0.78	282
Correct	0.85	0.82	0.84	393
		Accuracy	0.81	675
		Balanced Accuracy	0.81	675
Cheating	0.0	0.0	0.0	6
Correct	0.78	0.80	0.79	393
Non Interpretable	0.26	0.09	0.13	57
Partially Correct	0.51	0.58	0.54	219
		Accuracy	0.66	675
		Balanced Accuracy	0.37	675
Regression	***R^2***	***EV***	***MAE***	***MSE***
$\hat{y}$ = Grade	0.148	0.421	0.233	0.071

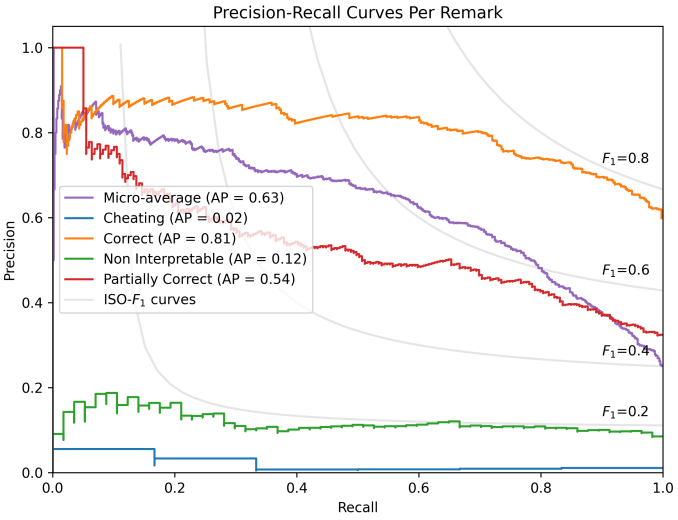

Fig. 9. Precision-Recall curves and average precision scores per remark.

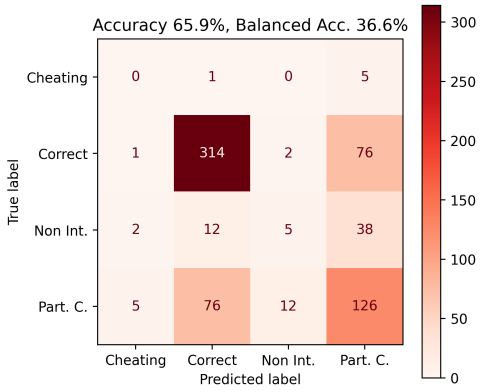

Fig. 10. Confusion matrix for the model that predicts remarks.

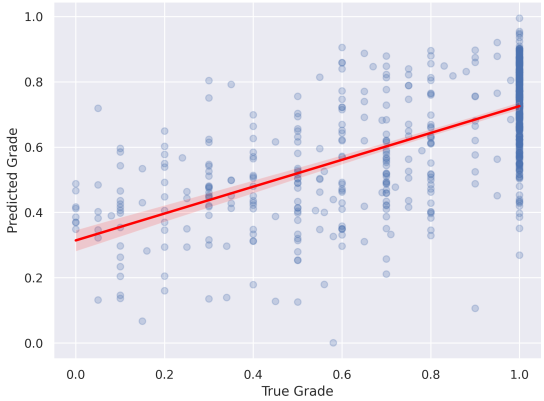

Fig. 11. True vs predicted regression plot for the model that predicts grades.

this was predicted incorrectly as a `correct=1` but this time with the highest degree of certainty. Again, notice the higher grade that it receives and that it is partially correct, regardless of the lack of aesthetics in the SQL statement.

For the remarks model, **R**, after performing LOO, we obtain the results shown in Fig. 9. The figure shows that the model associated with the remarks about correctness, *i.e.*, `Correct` and `Partially Correct`, exhibit better performance in comparison to the remarks associated with negative feedback. These results are consistent with our previous finding that correctness can be better predicted than non-correctness. In Fig. 9, for reference, the average precision (AP) score is listed, which is a weighted mean of precisions; it also shows the F_1 curves for reference. In Fig. 9 and the confusion matrix in Fig. 10 we can also observe that the lowest-performing class was `Cheating`; this can be in part due to the limited number of samples available.

Finally, for the grades model, **G**, the LOO cross-validated performance analysis can be summarized in Fig. 11 and Fig. 12. A regression comparison between the predicted grade and the actual grate suggests that our model disfavors grades that are exactly 100% and favors a more conservative prediction in a wider range. This can be seen in the histogram of the residuals, $y - \hat{y}$, in Fig. 12, which shows a slight positive skew, while the ideal residuals should be centered around zero. However, the regression performance metrics still capture the predictive ability of the model, which is beyond simply predicting the expected value of the independent variable `Grade`. Table II provides a summary of all our performance metrics for all models, including regression at the bottom of the table. The summary includes the coefficient of determination, R^2, the explained variance, EV, the mean absolute error, MAE, and the MSE given in (1). Both error metrics are low, and the variance analysis supports the claim that the model can predict beyond random guesses or guessing the expected value.

Other classification metrics shown in Table II include standard classification metrics such as Precision, Recall, F_1-score, and standard and balanced accuracy.

IV. CONCLUSIONS

This paper describes a methodology capable of receiving an SQL statement and determining its correctness. It uses a novel approach in language modeling tasks known as a convolutional self-attention mechanism combined with traditional text embedding strategies and bottleneck features. To the best of our knowledge, this type of model has not been applied to the

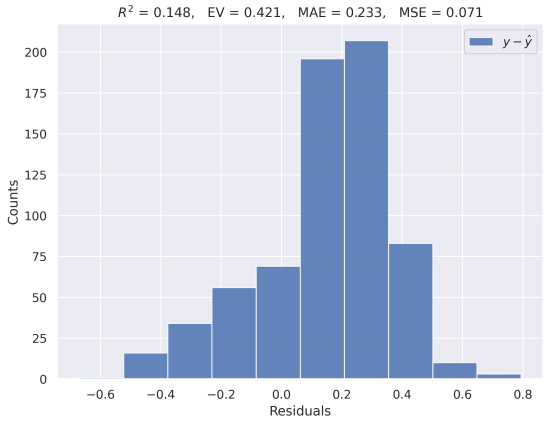

Fig. 12. Histogram of the residuals $y - \hat{y}$ for the model that predicts grades.

problem of modeling SQL statements for automatic grading purposes.

In our experiments, the model performs with a cross-validated balanced accuracy of 81.2% and an AUC of 0.87. This suggests that the model can achieve good generalization outside the dataset used. Further analysis of the learning cycles also shows discernible separation of data samples using a bottleneck layer as a means to visualize the data in two dimensions.

We believe this type of system can effectively model the problem of automatic grading of SQL statements provided sufficient examples for many different assignments. Future work includes the collection of more labeled data with a wider variety of submissions and assignments.

Acknowledgment

The ML model is based upon work supported in part by the National Science Foundation under Grant CHE-1905043.

References

[1] 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.

[2] S. Ravanbakhsh, J. Schneider, and B. Poczos, "Equivariance through parameter-sharing," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2892–2901.

[3] forpeerview, "Automated grading of sql statements," Jun. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3889635

[4] Y. Wu, Y. Ma, J. Liu, J. Du, and L. Xing, "Self-attention convolutional neural network for improved mr image reconstruction," *Information sciences*, vol. 490, pp. 317–328, 2019.

[5] D. S. Sachan and G. Neubig, "Parameter sharing methods for multilingual self-attentional translation models," *arXiv preprint arXiv:1809.00252*, 2018.

[6] S. Sadiq, M. Orlowska, W. Sadiq, and J. Lin, "Sqlator: an online sql learning workbench," in *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, 2004, pp. 223–227.

[7] S. Dekeyser, M. de Raadt, and T. Y. Lee, "Computer assisted assessment of sql query skills," in *Proceedings of the 18th Australasian Database Conference (ADC 2007)*, vol. 63. Australian Computer Society Inc., 2007, pp. 53–62.

[8] J. C. Prior and R. Lister, "The backwash effect on sql skills grading," *ACM SIGCSE Bulletin*, vol. 36, no. 3, pp. 32–36, 2004.

[9] A. Kleerekoper and A. Schofield, "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*, 2018, pp. 87–92.

[10] P. Singporn, P. Vichianroj, and A. Trongratsameethong, "Asqlag-automated sql assignment grading system for multiple dbmss," *Journal of Technology and Innovation in Tertiary Education*, vol. 1, no. 1, pp. 41–59, 2018.

[11] 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.

[12] I. Štajduhar and G. Mauša, "Using string similarity metrics for automated grading of sql statements," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2015, pp. 1250–1255.

[13] S. Chu, D. Li, C. Wang, A. Cheung, and D. Suciu, "Demonstration of the cosette automated sql prover," in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017, pp. 1591–1594.

[14] 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.

[15] B. Chandra, M. Joseph, B. Radhakrishnan, S. Acharya, and S. Sudarshan, "Partial marking for automated grading of sql queries," *Proceedings of the VLDB Endowment*, vol. 9, no. 13, pp. 1541–1544, 2016.

[16] B. Chandra, A. Banerjee, U. Hazra, M. Joseph, and S. Sudarshan, "Automated grading of sql queries," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1630–1633.

[17] ——, "Edit based grading of sql queries," in *8th ACM IKDD CODS and 26th COMAD*, 2021, pp. 56–64.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[19] M. Li, W. Hsu, X. Xie, J. Cong, and W. Gao, "Sacnn: Self-attention convolutional neural network for low-dose ct denoising with self-supervised perceptual loss network," *IEEE transactions on medical imaging*, vol. 39, no. 7, pp. 2289–2301, 2020.

[20] S. R. Fahim, Y. Sarker, S. K. Sarker, M. R. I. Sheikh, and S. K. Das, "Self attention convolutional neural network with time series imaging based feature extraction for transmission line fault detection and classification," *Electric Power Systems Research*, vol. 187, p. 106437, 2020.

[21] W. Zeng and M. Li, "Crop leaf disease recognition based on self-attention convolutional neural network," *Computers and Electronics in Agriculture*, vol. 172, p. 105341, 2020.

[22] Z. Chen, L. Chen, Y. Zhao, R. Cao, Z. Xu, S. Zhu, and K. Yu, "Shadowgnn: Graph projection neural network for text-to-sql parser," *arXiv preprint arXiv:2104.04689*, 2021.

[23] B. Bogin, M. Gardner, and J. Berant, "Representing schema structure with graph neural networks for text-to-sql parsing," *arXiv preprint arXiv:1905.06241*, 2019.

[24] S. Vijayarani, R. Janani *et al.*, "Text mining: open source tokenization tools-an analysis," *Advanced Computational Intelligence: An International Journal (ACII)*, vol. 3, no. 1, pp. 37–47, 2016.

[25] B. Ghazanfari and F. Afghah, "Multi-level feature learning on embedding layer of convolutional autoencoders and deep inverse feature learning for image clustering," *arXiv preprint arXiv:2010.02343*, 2020.

[26] P. Rivas, *Deep Learning for Beginners*. Packt Publishing Ltd, 2020.

[27] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[28] Z. Li, S.-H. Wang, R.-R. Fan, G. Cao, Y.-D. Zhang, and T. Guo, "Teeth category classification via seven-layer deep convolutional neural network with max pooling and global average pooling," *International Journal of Imaging Systems and Technology*, vol. 29, no. 4, pp. 577–583, 2019.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[30] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" in *Proceedings of the 32nd international conference on neural information processing systems*, 2018, pp. 2488–2498.

[31] M. C. Mukkamala and M. Hein, "Variants of rmsprop and adagrad with logarithmic regret bounds," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2545–2553.

[32] M. Kearns and D. Ron, "Algorithmic stability and sanity-check bounds for leave-one-out cross-validation," *Neural computation*, vol. 11, no. 6, pp. 1427–1453, 1999.