IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

ADVANCING SEMANTIC TABLE PARSING: COMBINING SQLOVA, TABERT, AND LOOKAHEAD OPTIMIZERS

Rajasekhar Kalla

Sri Venkateswara University, Tirupati, India

ABSTRACT

Abstract: Recent advancements in deep learning for Natural Language Processing (NLP) have significantly accelerated research in Natural Language to SQL (NL2SQL) translation across both academia and industry [1]. Several architectures have been proposed to address this task, including approaches based on reinforcement learning, sequence-to-sequence (Seq2Seq), and sequence-to-set (Seq2Set) learning frameworks [2][3]. In this work, we propose a hybrid architecture that integrates TaBERT and SQLOVA for enhanced NL2SQL generation. TaBERT, which is pre-trained on structured tabular-text data, provides superior contextual understanding compared to conventional BERT representations [4], thereby improving the semantic encoding of natural language queries and table headers. The NL2SQL component of SQLOVA is stacked on top of the TaBERT encoder, enabling further contextual refinement and feature extraction from both query and schema representations. Additionally, the choice of optimization strategy plays a critical role in improving model convergence and prediction accuracy. Experimental results on the WikiSQL dataset demonstrate that the proposed architecture, when trained using the Lookahead optimizer [5], achieves performance improvements of 0.2%, 0.5%, and 0.4% in where-num, where-col, and where-cond accuracy metrics, respectively, compared to the baseline model.

KEYWORDS

neural networks, NLP, nl2sql

1. INTRODUCTION

Generating SQL queries from natural language utterances has long been recognized as a challenging and open research problem in the field of Natural Language Processing (NLP) [1]. This capability has significant real-world applications across domains such as banking, healthcare, and enterprise analytics, where large volumes of structured information are stored in relational databases [2]. Recent advancements in deep learning techniques for NLP have substantially improved the performance of Natural Language to SQL (NL2SQL) systems, resulting in increased research interest from both academia and industry [3]. In the current digital era, the volume of organizational data has increased exponentially; however, a major portion of enterprise and banking data continues to reside in structured database systems. Accessing this data generally requires expertise in Structured Query Language (SQL), thereby limiting database accessibility to technically skilled users [4]. Consequently, there is a growing need for intelligent systems that enable users to retrieve information from databases using natural language queries irrespective of their technical background. Although considerable progress has been achieved in translating natural language into SQL statements, there remains substantial scope for improving model accuracy and robustness, particularly on benchmark datasets such as WikiSQL [5].

WikiSQL is one of the largest publicly available datasets for NL2SQL research, consisting of natural language questions paired with corresponding SQL queries [5]. The availability of WikiSQL has accelerated the development of task-specific end-to-end neural architectures and significantly contributed to advancements in semantic parsing and NL2SQL systems.

The NL2SQL task involves predicting multiple SQL components from a natural language query, including the SELECT column, aggregate function, number of WHERE conditions, WHERE columns, WHERE values, and conditional operators. Broadly, this problem has been approached using two major paradigms. The first approach formulates NL2SQL generation as a sequence generation problem using sequence-to-sequence (Seq2Seq) architectures. Earlier studies extensively employed Seq2Seq models for translating natural language queries into SQL statements [6], and subsequent improvements were achieved using reinforcement learning-based optimization techniques [7]. However, Seq2Seq architectures suffer from the "order matters" problem, where each prediction depends on previously generated tokens, making the model highly sensitive to token ordering and susceptible to error propagation during SQL generation [8].

An alternative perspective treats the NL2SQL problem as a slot-filling task rather than generating the entire SQL query token by token. In this formulation, only predefined slots within a SQL template need to be predicted. To overcome the limitations of Seq2Seq models, SQLNet introduced a sketch-based framework that eliminates the order dependency issue [8]. Instead of generating complete SQL statements, the model predicts specific SQL components required to fill a predefined query sketch, thereby improving robustness and prediction accuracy.

With recent advancements in transformer-based NLP architectures, BERT has significantly improved performance across a wide range of NLP tasks [9]. SQLOVA employs a combination of BERT embeddings and an NL2SQL layer for translating natural language utterances into SQL queries [10]. In this architecture, BERT converts query tokens and table headers into contextualized vector representations, which are further refined using column attention mechanisms, self-attention, and Bi-LSTM encoders within the NL2SQL layer. SQLOVA demonstrated substantial improvements in execution accuracy and logical form accuracy compared to earlier approaches.

Although BERT has shown remarkable effectiveness in tasks involving unstructured textual data, the NL2SQL problem inherently involves structured and semi-structured information in the form of database schemas and tables. Therefore, models pre-trained specifically on structured tabular data are expected to generate richer contextual representations for this task. TaBERT, an extension of BERT designed for tabular data understanding, is particularly suitable for such scenarios [11]. TaBERT is pre-trained on a large corpus consisting of approximately 26 million tables along with their associated textual contexts, enabling it to learn joint representations of natural language and semi-structured tabular information more effectively than conventional BERT-based approaches.

In this work, we exploit the complementary strengths of both TaBERT and SQLOVA. TaBERT is employed to contextualize the input natural language utterance and table schema representations. The generated embeddings are subsequently passed to the NL2SQL layer adopted from the SQLOVA architecture. The NL2SQL module consists of six specialized submodules, each incorporating Bi-LSTM layers to perform distinct prediction tasks. These tasks include predicting the SELECT column, aggregate operator, WHERE columns, number of WHERE conditions, WHERE value strings, and WHERE conditional operators.

The proposed model is trained and evaluated on the WikiSQL dataset. Extensive experiments were conducted using multiple optimization strategies to identify the most effective training configuration. Experimental observations indicate that the Lookahead optimizer [12] delivers the best overall performance for the proposed architecture.

2. PROPOSED EXPERIMENTS

The primary motivation behind this work is to develop an efficient and robust model for translating natural language queries into SQL statements. To accomplish this task, the proposed architecture is trained and evaluated using the WikiSQL dataset, which provides pairs of English questions and corresponding SQL queries for supervised learning. In this section, we discuss recent methodologies and state-of-the-art approaches proposed for converting natural language utterances into executable SQL queries.

2.1 Recent Approaches

The WikiSQL dataset has been extensively utilized for the NL2SQL generation task due to its large scale and availability of annotated natural language and SQL query pairs [5]. The introduction of WikiSQL enabled the effective application of deep learning-based neural architectures for semantic parsing and SQL generation tasks.

Early approaches employed sequence-to-sequence (Seq2Seq) architectures [6], where separate components were designed to generate the SELECT clause and WHERE clause of SQL queries. These models demonstrated improved performance compared to traditional vanilla Seq2Seq approaches by incorporating task-specific structural information. However, Seq2Seq models remained sensitive to token ordering and suffered from error propagation during sequential decoding.

To address these limitations, SQLNet introduced a sequence-to-set framework combined with a syntax-guided sketch mechanism [8]. Instead of generating the entire SQL query token by token, SQLNet predicts predefined slots within a query template, thereby simplifying the overall SQL generation process and eliminating the "order matters" problem. For predicting WHERE clause values, SQLNet continued to employ a Seq2Seq-based decoding mechanism.

TypeSQL further extended SQLNet by incorporating semantic type information associated with table headers and question entities [9]. Similar to SQLNet, TypeSQL employed a sequence-to-set framework but improved schema understanding by utilizing type-aware representations.

Another important contribution was Pointer-SQL, which introduced a value-based loss function and an attention-driven copying mechanism to improve SQL query generation accuracy [10]. Additionally, execution-guided decoding was proposed to eliminate syntactically invalid or non-executable SQL queries during the decoding stage, thereby improving execution accuracy [11].

Subsequently, sequence-to-action parsing approaches were introduced, where SQL generation was formulated as an incremental slot-filling process [12]. These methods further simplified SQL query construction by decomposing the task into smaller sequential actions.

SQLOVA, inspired by SQLNet, adopted the sketch-based framework while leveraging the contextual encoding capabilities of BERT [13]. In SQLOVA, BERT is used to encode natural language utterances and table headers into contextualized embeddings, which are subsequently refined using column attention and self-attention mechanisms. For WHERE value prediction, the model predicts the start and end indices of the corresponding value span within the input utterance. The integration of transformer-based contextual representations significantly improved both logical form accuracy and execution accuracy.

TaBERT extends the BERT architecture for structured and semi-structured tabular data understanding [14]. Unlike BERT, which is pre-trained primarily on unstructured textual corpora, TaBERT is trained on a large corpus of structured tables and associated textual contexts. It employs a content snapshot mechanism to better capture relationships between table schemas and natural language queries, thereby generating richer contextual representations for tabular reasoning tasks.

The proposed approach in this work combines the strengths of both TaBERT and SQLOVA by replacing the BERT encoder in SQLOVA with TaBERT. This integration enables improved contextual understanding of structured table information while preserving the effective sketch-based NL2SQL generation framework of SQLOVA.

2.2 Dataset: WIKISQL

The proposed model is trained and evaluated using the WikiSQL dataset, one of the most widely used benchmark datasets for Natural Language to SQL (NL2SQL) research [5]. WikiSQL is a large-scale corpus specifically designed for translating natural language utterances into SQL queries. The dataset consists of approximately 80,654 natural language questions paired with their corresponding SQL statements, constructed using 24,241 tables extracted from Wikipedia

The large size and diversity of the dataset have enabled the development and training of end-to-end deep neural network architectures for semantic parsing and SQL query generation tasks. Consequently, WikiSQL has become a standard benchmark dataset and has been extensively utilized by several state-of-the-art NL2SQL models.

n WikiSQL, each query is associated with only a single table, and inter-table JOIN operations are not included in the dataset. Furthermore, the SQL queries are restricted to a single SELECT column, although multiple WHERE conditions are permitted. The dataset supports six aggregation operations: NONE, MAX, MIN, COUNT, SUM, and AVG, where NONE indicates the absence of an aggregation function. The WHERE clause supports three conditional operators: equality (=), greater than (>), and less than (<).

Table 1. WikiSQL example

Player	Country	Points	Winnings
Steve striker	United States	9000	1260000
KJ Choi	South Korea	5400	756000
Rory Sabitini	South Africa	3400	4760000
Mark Calcaveccia	United States	2067	289333
Ernie Elss	South Africa	2067	289333

Question: What is the points of South Korean Player
SQL: SELECT Points WHERE Country =South Korea
Answer:5400

2.3 TaBERT

Database information is primarily stored in structured tabular formats, where relational tables exhibit strong inherent schema-level organization. Traditional language models such as BERT are primarily designed and pre-trained for encoding free-form unstructured text [13]. However, Natural Language to SQL (NL2SQL) tasks involve understanding both natural language queries and structured database schemas simultaneously.

A major challenge in encoding database tables arises from the large number of rows contained within relational tables. Naively encoding all rows using computationally intensive transformer-based language models becomes impractical due to excessive memory and computational requirements. Several previous approaches attempted to leverage BERT for NL2SQL tasks; however, these methods generally relied on domain-specific or handcrafted strategies to encode structured database information without explicitly pre-training the model on structured tabular data.

TaBERT addresses this limitation by incorporating not only column headers but also table contents while encoding the table schema [14]. Including cell values during schema representation learning improves contextual understanding because the table contents provide additional semantic information associated with each column. In many practical scenarios, column names alone may be ambiguous. For example, a column named “Place” may refer to a city, country, or venue. Encoding representative cell values from that column helps the model align natural language entities present in the query with the correct schema attributes.

Since database tables may contain a large number of rows, encoding all table contents is computationally infeasible for transformer-based architectures. To overcome this challenge, TaBERT introduces the concept of *content snapshots*, which selectively focus only on rows that are most relevant to the input natural language query. Instead of processing the complete table, a lightweight retrieval strategy is employed to identify the most relevant rows.

The content snapshot generation process computes n-gram similarity scores between the input utterance and each row in the database table. Based on these similarity scores, the top-K most relevant rows are selected to construct the content snapshot. This selective sampling mechanism significantly reduces computational complexity while preserving semantically relevant contextual information.

For each selected row, a linearized representation is created and concatenated with the corresponding natural language query before being passed to the transformer encoder. Given an input query (q) and a table (T), TaBERT first constructs a content snapshot consisting of sampled rows that summarize the table information most relevant to the input query. Each sampled row is then linearized and concatenated with the natural language utterance to form the final transformer input sequence.

The model subsequently generates row-wise contextualized embeddings for query tokens, table headers, and cell values. These contextualized representations capture both semantic and structural relationships between the natural language utterance and the tabular schema. The encoded query and header representations are then forwarded to the subsequent NL2SQL layer for downstream SQL prediction tasks.

2.4 SQLOVA

SQLOVA employs a syntax-guided sketch framework for SQL query generation, where the overall architecture is decomposed into six specialized prediction modules: select-column, select-aggregation, where-number, where-column, where-operator, and where-value [10]. The sketch-based formulation effectively addresses the "order matters" problem commonly encountered in sequence-to-sequence (Seq2Seq) architectures. Instead of generating the entire SQL query token by token, SQLOVA adopts a sequence-to-set formulation, where individual SQL components are predicted independently.

Traditional Seq2Seq models face significant challenges in NL2SQL tasks because the model must simultaneously learn SQL syntax, maintain sequential dependencies, and predict semantically correct SQL tokens. Errors generated during earlier decoding stages can propagate through subsequent predictions, thereby reducing overall accuracy. In contrast, the sequence-to-set approach simplifies the SQL generation process by decomposing the task into multiple independent slot-filling subtasks.

The SQLOVA architecture primarily consists of two major components: the Table-Aware Encoding Layer and the NL2SQL Layer. Initially, the natural language query and table headers are passed through BERT to generate contextualized embeddings. These encoded representations are subsequently forwarded to the NL2SQL layer for task-specific SQL component prediction.

Each prediction module in the NL2SQL layer is designed to focus exclusively on a specific SQL subtask. For example, the select-column module predicts the appropriate column corresponding to the SELECT clause by identifying the most relevant column header from the input table schema. Instead of generating column names explicitly, the model performs a classification task over the available schema headers. This significantly reduces the complexity of SQL generation and improves prediction robustness.

A similar slot-filling strategy is employed across all remaining submodules, including select-aggregation, where-number, where-column, where-operator, and where-value prediction modules. By decomposing the SQL generation problem into smaller specialized prediction tasks, SQLOVA achieves improved efficiency and higher execution accuracy compared to conventional sequence generation approaches.

3. PROPOSED APPROACH

As discussed earlier, TaBERT is pre-trained on a large corpus of structured and semi-structured tabular data, enabling it to capture relationships between natural language utterances and database schemas more effectively than conventional BERT models [14]. Owing to its table-aware pretraining strategy, TaBERT generates richer contextual representations for structured database information and improves semantic alignment between user queries and table contents.

SQLOVA consists of two primary components: the Table-Aware Encoding Layer and the NL2SQL Layer [10]. In the original SQLOVA architecture, the Table-Aware Encoding Layer utilizes BERT to encode natural language queries and table headers, while the NL2SQL layer employs multiple Bi-LSTM-based submodules for SQL component prediction.

The proposed approach integrates TaBERT with the NL2SQL layer of SQLOVA by replacing the original BERT-based encoder with TaBERT. This integration enables the architecture to leverage the structured-data understanding capability of TaBERT while retaining the effective sketch-based SQL generation framework of SQLOVA. Consequently, the model achieves improved contextual understanding of natural language queries and table schemas, leading to enhanced NL2SQL prediction performance.

The overall proposed architecture therefore consists of two major components: the TaBERT encoding layer and the NL2SQL prediction layer.

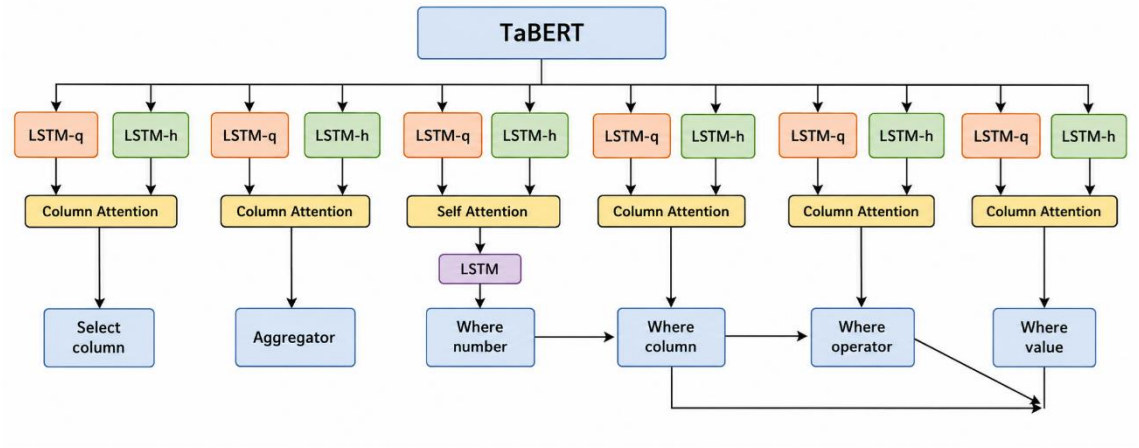

Figure 1. Proposed Approach[1][2]

3.1. Contextualization layer

The first component of the proposed architecture consists of the TaBERT model. TaBERT accepts a natural language query (q) and a database table (T) as inputs. Based on the semantic similarity between the query and table rows, the model constructs a content snapshot containing the most relevant rows associated with the input utterance.

Subsequently, each selected row is transformed into a linearized representation and concatenated with the input query before being passed to the TaBERT encoder. This process enables the model to generate contextualized representations that jointly capture semantic relationships between the natural language query, table schema, and relevant table contents.

The contextualized embeddings produced by TaBERT are then forwarded to the subsequent NL2SQL layer for further encoding and SQL component prediction.

3.2. NL2SQL layer

The NL2SQL layer is adopted from the SQLOVA framework and is positioned above the table-aware encoding layer. In the proposed architecture, the contextualized embeddings generated by TaBERT are further refined using two layers of Bidirectional Long Short-Term Memory (Bi-LSTM) networks. The hidden dimension of each Bi-LSTM layer is set to 100.

The encoded representations of the natural language query and table headers are processed separately. For each table header, the hidden representation corresponding to the final token is utilized for downstream slot prediction tasks. The NL2SQL layer consists of multiple specialized submodules, each designed to predict a specific component of the SQL query.

Select-Column Module

The select-column module predicts the appropriate column for the SELECT clause from the available table headers. Rather than generating column names explicitly, the module performs classification over the predefined schema columns, thereby simplifying the SQL generation process.

Select-Aggregator Module

The select-aggregator module predicts the aggregation operator associated with the SELECT clause. The supported aggregation operations are:

- NONE
- MAX
- MIN
- COUNT
- SUM
- AVG

The module utilizes both the encoded query representation from TaBERT and the predicted select-column representation for aggregation prediction.

Where-Number Module

The where-number module predicts the total number of conditions present in the WHERE clause. The maximum number of supported WHERE conditions is limited to four. Therefore, the module performs classification over the set ({0,1,2,3,4}), where:

- 0 indicates the absence of WHERE conditions,
- 1 indicates a single condition,
- and higher values correspond to multiple WHERE conditions.

Where-Column Module

The where-column module predicts the relevant table columns involved in the WHERE clause conditions. Similar to the select-column module, prediction is performed over the available schema headers.

Where-Operator Module

The where-operator module predicts the conditional operator associated with each WHERE condition. The supported operators are:

- Equality (=)
- Greater than (>)
- Less than (<)

This module utilizes both the predicted where-column representation and the contextual embeddings generated by TaBERT.

Where-Value Module

The where-value module predicts the value associated with each WHERE condition by identifying the start and end token indices within the input natural language utterance. The prediction of start and end positions depends on both the selected where-column and where-operator representations. During feature fusion, concatenation is employed instead of vector addition while combining query and header representations, enabling richer contextual interaction between the two embeddings.

All prediction modules within the NL2SQL layer are independently parameterized and do not share weights with one another. This design allows each module to specialize in its corresponding SQL prediction task.

Execution-Guided Decoding (EG)

To further improve SQL generation accuracy, Execution-Guided Decoding (EG) is employed during the inference stage [11]. The primary objective of EG decoding is to eliminate syntactically invalid or non-executable SQL queries from the final predictions.

During SELECT clause prediction, invalid ((select\ column,\ aggregation)) pairs are excluded when aggregation operations are applied to non-numeric string columns. This filtering mechanism prevents semantically incorrect SQL statements from being generated.

Similarly, during WHERE clause prediction, the executability of each ((where\ column,\ operator,\ value)) triplet is verified by partially executing the generated SQL query against the database table. Queries producing invalid or empty execution results are discarded.

Finally, the WHERE clause is determined based on the joint probability distributions generated by the where-value, where-operator, where-column, and where-number modules. This execution-guided strategy significantly improves the reliability and correctness of the generated SQL queries.

4. IMPROVING RESULTS WITH LOOK AHEAD OPTIMIZER

The choice of optimization algorithm plays a critical role in the effective training of deep neural network architectures. Optimizers minimize the objective function by iteratively updating network parameters based on the computed gradients. Over the years, several optimization techniques have been proposed, each with its own advantages and limitations in terms of convergence speed, stability, and generalization capability.

Traditional deep learning models have been successfully trained using Stochastic Gradient Descent (SGD). Subsequent research introduced enhanced optimization strategies that are broadly categorized into two classes: accelerated optimization methods and adaptive learning rate methods. Accelerated optimization techniques primarily focus on improving convergence speed and reducing oscillations during training, whereas adaptive learning rate methods dynamically adjust parameter-specific learning rates to improve optimization efficiency.

In this work, multiple optimization algorithms, including Adam, RAdam, Ranger, and Lookahead (with Adam as the base optimizer), were independently evaluated for training the proposed NL2SQL architecture. Among these approaches, the Lookahead optimizer [12] demonstrated superior performance in terms of training stability and prediction accuracy.

Unlike conventional optimizers, Lookahead operates by maintaining and iteratively updating two sets of model weights: *fast weights* and *slow weights*. The fast weights are updated using an inner optimizer such as Adam, while the slow weights are periodically synchronized with the fast weights after several optimization steps. The search direction is therefore determined by "looking ahead" at the trajectory generated by the fast weights. This mechanism reduces optimization variance, improves convergence stability, and enhances generalization performance while introducing minimal additional computational and memory overhead.

Experimental observations indicate that the Lookahead optimizer consistently outperformed the other evaluated optimizers for the proposed TaBERT-SQLOVA architecture. During training, the initial learning rate for the NL2SQL layer was set to 0.001, whereas the TaBERT encoder layer was initialized with a lower learning rate of 0.00001 to preserve the pretrained contextual representations.

As training progressed, the learning rates were manually reduced whenever validation accuracy reached a plateau. A step-wise decay strategy was employed, where the learning rate was reduced by a factor of 10 until reaching a minimum value of 0.000001. This gradual learning rate scheduling strategy contributed to improved convergence and enhanced overall model performance.

5. EXPERIMENTAL RESULTS

A pretrained TaBERT model was utilized as the table-aware encoder, while the NL2SQL layer was trained from scratch for the downstream SQL generation task. Multiple optimization algorithms, including Adam, RAdam, and Lookahead, were independently evaluated during the fine-tuning process. Experimental observations demonstrated that the Lookahead optimizer achieved superior performance compared to the other evaluated optimization strategies.

The initial learning rate for the NL2SQL layer was set to 0.001. To address performance saturation during training, a step-wise learning rate decay strategy was employed, where the learning rate was progressively reduced by a factor of 10 until reaching (10^{-6}). For the TaBERT encoder layer, a significantly smaller learning rate of (10^{-6}) was used in order to preserve the pretrained contextual representations while enabling gradual task-specific adaptation.

The Stanford CoreNLP tokenizer was used to tokenize the natural language utterances, whereas the WordPiece tokenizer was employed for tokenizing table headers and SQL vocabulary components.

The proposed architecture was implemented using PyTorch, with the TaBERT and NL2SQL modules being influenced by the original TaBERT [14] and SQLOVA [10] implementations, respectively. Experiments were conducted using Version 1.1 of the WikiSQL dataset.

The model was trained for 150 epochs using a batch size of 64. These training configurations enabled stable convergence and improved overall prediction performance for the proposed NL2SQL framework.

5.1 Experiments

Experimental observations indicate that the choice of optimization algorithm has a significant impact on model convergence and overall prediction accuracy. The effectiveness of an optimizer is highly dependent on the underlying task, model architecture, and training dynamics. During training, the learning rate was manually reduced by a factor of 10 whenever the validation accuracy reached a plateau, enabling improved convergence and stable optimization.

Among the evaluated optimization algorithms, namely Adam, RAdam, and Lookahead, the Lookahead optimizer consistently achieved the best overall performance. Experimental results further revealed that Adam produced better accuracy metrics compared to RAdam for the proposed NL2SQL architecture.

The superior performance of the Lookahead optimizer can be attributed to its ability to reduce optimization variance and stabilize the learning process through the interaction between fast and slow weight updates. As a result, the proposed TaBERT-SQLOVA architecture trained using Lookahead achieved higher prediction accuracy compared to the baseline SQLOVA model, particularly for the where-num, where-col, and where-cond evaluation metrics.

Table 2: Experiment results with different optimizer on validation set

Optimizer	select-col	agg	where-num	where-col	where-cond	where-val
SQLOVA	97.3	90.5	98.7	94.7	97.5	97.5
ADAM (proposed approach)	96.1	89.6	98.1	96	97.2	97
RADAM (proposed approach)	94.3	89.6	97.5	94.4	95.9	95.6
Lookahead (proposed approach)	96.7	89.7	**98.9**	**95.2**	**97.9**	96.7

5.2. Results

Experimental results indicate that the pretraining strategy of the underlying language model plays a crucial role in improving the contextual understanding of natural language utterances in NL2SQL tasks. TaBERT is pretrained on a large corpus of structured and semi-structured tabular data, enabling it to capture stronger semantic relationships between natural language queries and database schemas. In contrast, BERT is primarily pretrained on unstructured textual corpora, which limits its ability to effectively model structured table representations.

This advantage of TaBERT is reflected in the experimental accuracy metrics, where the proposed TaBERT-SQLOVA architecture outperforms the baseline SQLOVA model on the validation dataset, particularly for the where-num, where-col, and where-cond evaluation metrics.

Furthermore, the performance of the proposed model for select-column, aggregate operator, and where-value prediction tasks was observed to approach the corresponding accuracy levels achieved by SQLOVA. These findings demonstrate that structured-data pretraining contributes significantly toward improving contextual representation learning and enhancing NL2SQL prediction performance.

Table3: Validation Results Comparison

Optimizer	select-col	agg	where-num	where-col	where-cond	where-val
SQLOVA	97.3	90.5	98.7	94.7	97.5	97.5
Proposed Approach	96.7	89.7	**98.9**	**95.2**	**97.9**	96.7

6. CONCLUSION

In this work, we propose a hybrid NL2SQL architecture that integrates TaBERT, which is pretrained on structured tabular data, with the SQLOVA framework consisting of an NL2SQL layer based on Bi-LSTM networks, column attention, and self-attention mechanisms. The proposed architecture leverages the structured-data understanding capability of TaBERT together with the effective sketch-based SQL generation strategy employed by SQLOVA.

Experimental results demonstrate the effectiveness of TaBERT over conventional BERT for structured text understanding and SQL generation tasks. The study further highlights that the selection of an appropriate optimization strategy plays a critical role in improving model convergence and prediction accuracy.

Among the evaluated optimization techniques, the Lookahead optimizer achieved the best overall performance. The proposed TaBERT-SQLOVA architecture trained using Lookahead surpassed the baseline SQLOVA model by 0.2%, 0.5%, and 0.4% in where-num, where-col, and where-cond accuracy metrics, respectively.

REFERENCES

[1] Pengcheng Yin, Graham Neubig, Wen-tau Yih, Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data

[2] Wonseok Hwang, Jinyeong Yim, Seunghyun Park, Minjoon Seo. 2019. A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. NAACL, abs/1810.04805.

[4] Michael R. Zhang, James Lucas, Geoffrey Hinton, Jimmy Ba,2019. Lookahead Optimizer: k steps forward, 1 step back

[5] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In ICLR, 2016

[6] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. CoRR, abs/1709.00103.

[7] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In Advances in Neural Information Processing Systems, pp. 2692–2700, 2015a.

[8] Xiaojun Xu, Chang Liu, and Dawn Song. 2017. SQLNET: Generating structured queries from natural language without reinforcement learning. CoRR, abs/1711.04436

[9] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, Dragomir Radev. 2018.TypeSQL: Knowledge-based Type-Aware Neural Text-to-SQL Generation.

[10] Tianze Shi, Kedar Tatwawadi, Kaushik Chakrabarti, Yi Mao, Oleksandr Polozov, and Weizhu Chen. 2018.Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. CoRR, abs/1809.05054

[11] Chenglong Wang, Po-Sen Huang, Alex Polozov, Marc Brockschmidt, and Rishabh Singh. 2018a. Executionguided neural program decoding. In ICML workshop on Neural Abstract Machines and Program Inductionv2 (NAMPI).

[12] Tianze Shi, Kedar Tatwawadi, Kaushik Chakrabarti, Yi Mao, Oleksandr Polozov, and Weizhu Chen. 2018.Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. CoRR, abs/1809.05054