SpeakQL: SQL Generation from Natural Language

Madhu Damani
Student,
Thadomal Shahani Engineering College,
Bandra, Mumbai.

Gaurav Kamdar
Student,
Thadomal Shahani Engineering College,
Bandra, Mumbai

Laksh Jethani
Student,
Thadomal Shahani Engineering College,
Bandra, Mumbai.

Mukesh Israni, PhD
Professor,
Thadomal Shahani Engineering College,
Bandra, Mumbai.

ABSTRACT
In recent years, there has been growing interest in the complex task of converting natural language into SQL queries. This challenge typically involves using sequence-tosequence models, which require the serialization of SQL queries. However, a fundamental issue arises as a single SQL query can have multiple valid serializations, leading to the 'order matters' problem and making it difficult to train such models effectively. While existing state-of-the-art methods turn to reinforcement learning to address this issue, their success is limited. This paper presents SpeakQL, a novel approach tailored to scenarios where query order is not critical. SpeakQL adopts a sketchbased strategy, incorporating a dependency graph into its model architecture to consider the influence of prior predictions on current ones. Furthermore, SpeakQL utilizes GloVe embeddings and a column attention mechanism to enhance contextual comprehension, ultimately improving the query generation and result retrieval process.

Keywords
Machine Learning, Deep Learning, Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTMs), Global Vector Embeddings (GloVe), Word2Vec

1. INTRODUCTION
In today's landscape, relational databases house a wealth of information, and SQL query proficiency is essential for database navigation. NLP stands at the intersection of Human-Computer Interaction and AI, serving tasks like information retrieval and language analysis. NLP aims to enable seamless communication between humans and computers by eliminating the need for memorizing complex commands. Accessing data in databases via natural language offers a convenient solution, especially for users unfamiliar with SQL. This system addresses challenges in handling natural language text and speech, allowing users to articulate queries in plain language. NLP plays a pivotal role, striving to make computers understand and generate natural language. Our goal is to enhance decision-making through data use, simplifying interactions by converting natural language into SQL, much like Google Translate does for languages.

The Natural Language Interface (NLI), bridging natural language processing (NLP) and human-computer interaction, enables conversational interactions with computers. Our focus is on the automatic generation of structured query language (SQL) queries for relational databases. While SQL is the standard language for interacting with databases, it poses challenges due to its complexity. Our work introduces a natural language interface for relational databases, allowing users to communicate with databases in plain language, rather than relying on SQL.

In proposed approach, the objective is to use the disciplined approach to handle the problem of creating string sets. Our approach avoids the 'higher order' problems in the sequential model and thus eliminates the need to use reinforcement learning algorithms to handle the problem of sequential generation limited to some extent. To achieve a better performance than existing sequence-to sequence-based approaches using sketch-based approach and column attention and incorporate the ability to accept any kind of natural language input requiring some database related information from the user and provide output consisting of the user's required information with maximum accuracy.

2. LITERATURE REVIEW
Victor Zhong and his colleagues [1] employed a seq2seq (encoder-decoder) methodology, coupled with reinforcement learning, to provide incentives to the model based on its generated output. This augmented pointer network operates in a token-by-token manner, assembling the SQL query by selecting tokens from a concatenated input sequence. This input sequence comprises the column names essential for specifying the selection and condition columns within the query, the question pertinent to the query's conditions, and the restricted SQL language vocabulary, encompassing keywords like SELECT, COUNT, and so forth. The SQL query is constructed through the execution of three distinct functions: • [1] Aggregate Operation - The aggregation operation depends on the Question [2] Select Column - Depends on the Table Columns and Question Input and the generation of the SELECT [3] Where Clause - Augmented Pointer Network is used to train and generate the WHERE clause depending on the input question. Cross entropy loss is used to train the network but it fails to optimize in case of conditions being swapped and yet it generates the same result. This review encompasses various implementations involving the utilization of the WikiSQL dataset for natural language query processing. In the work by Tao Yu et al. [2] they present a type-aware model that categorizes words into entity types such as knowledge graph, column, or number. This approach treats the task as slot filling, grouping slots logically and capturing attribute relationships. Their model employs a two-bidirectional LSTM input encoder and operates on the WikiSQL dataset, comprising 87,673 examples from 26,521 tables. In practical scenarios, natural language queries often include rare, database-specific entities and numbers that lack accurate embeddings in pre-trained word embedding models. Furthermore, this model assumes exact column names and user query entries, which introduces privacy and security concerns. Yibo Sun et al. [3] focus on semantic parsing, translating natural language expressions into executable computer programs. They employ pointer networks to convert questions into continuous vectors, facilitating SQL query generation through three channels. This model determines when to generate column names, cells, or SQL keywords and incorporates column-cell relationships to enhance query structure. Pointer networks, originally introduced by Vinyals

et al. [4], have found success in various applications like reading comprehension, machine translation, and text summarization. However, the use of bidirectional RNNs with GRU units for question processing and column-cell relationship exploitation raises security and privacy considerations. Huang et al. [5] introduce a unique meta-learning approach for generating SQL queries from natural language inputs. Meta-learning leverages metadata from machine learning experiments and departs from traditional supervised training in NLP by advocating for task-specific models for groups of similar examples. To transition from conventional supervised learning to few-shot meta-learning, they introduce a relevance function that clusters examples into pseudo-tasks. This problem-specific function is explained within the context of the semantic parsing problem,

2.1 Standard Approaches Analysis

In natural language to SQL query conversion, a prevalent approach involves sequence-to-sequence models with reinforcement learning. An ongoing challenge is the importance of query order. To illustrate, consider sorting a set of random numbers by a specified criterion, generating n! valid permutations. When training with these uniformly selected permutations, the mapping assigns equal probability to all n! output configurations for a given input vector X, resulting in reduced statistical efficiency. Hence, constraining the output order as much as possible consistently yields better outcomes.

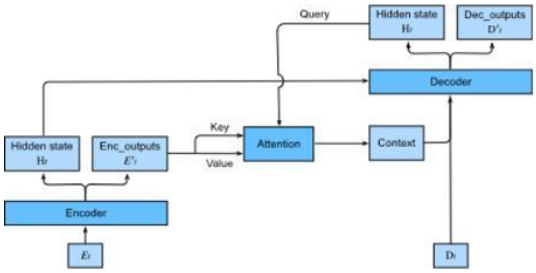
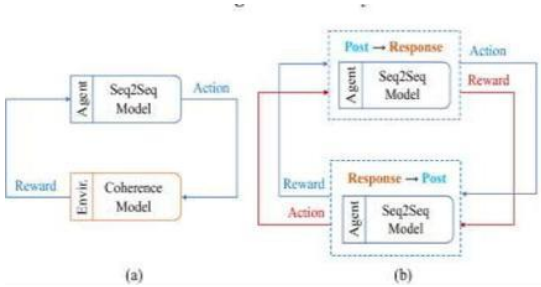


Fig 1. Seq-2-Seq with Column Attention

Fig 2. Seq2Seq with Reinforcement Learning

2.2 Sketch Based Approach Analysis

To address order-related challenges in sequence-to-sequence models, reinforcement learning is commonly used, but its impact can be limited. This study adopts a sketch-based approach to generate SQL queries, particularly focusing on the WHERE clause. This approach introduces two



mechanisms: sequence-to-set prediction to handle unordered constraints and column attention to capture dependencies defined in the sketch during prediction.

accompanied by an algorithm outline. Their work also leverages the WikiSQL dataset. Tong Guo et al. [6] adopt a distinct perspective by shifting their focus from SQL query generation to the extraction of data as answers to natural language questions. They employ semantic parsing within database-based question-answering systems, implementing this with BERT (Bidirectional Encoder Representations from Transformers). BERT, a contemporary language model, is pre-trained on extensive text data, rendering it adaptable for various tasks with the addition of just one output layer. This approach aims to directly retrieve answers from databases, thereby reducing the need for labor-intensive semantic parsing. Their implementation utilizing a BERT-based model produces baseline experimental results.

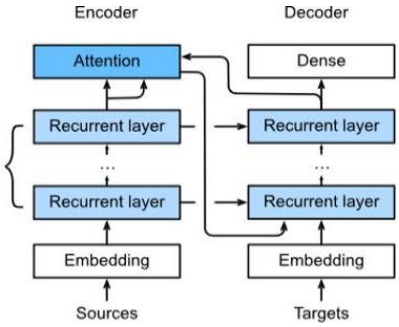
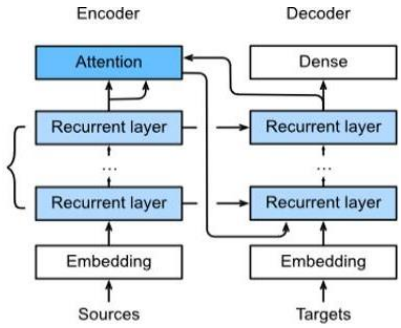


Fig 3. Encoder – Decoder Architecture

3. DESIGN AND IMPLEMENTATION

The design and implementation define the requirements along with the use case, architecture, functionality, techniques and the components.



3.1 Functional Requirements

Functional requirements encompass calculations, technical details, data processing, and other specific functionalities that delineate a system's objectives. Behavioral requirements, delineating system utilization based on these functional aspects, are encapsulated within use cases. [1] Dynamic Nature - The system must handle diverse, complex natural language statements, proficiently identifying the keywords. It should extract and store pertinent information to generate queries based on input statements. [2] Output Generation - The system should discern keywords in statements and endeavor to generate corresponding SQL queries and outputs. [3] Functionalities - The classifier must adeptly employ both local and global discriminators for keyword classification. Model training should not only facilitate query generation but also optimize the output.

3.2 Non - Functional Requirements

Non-functional requirements encompass aspects beyond system functionalities, addressing its overall performance and behavior. [1] Performance Requirements - Instead of sequentially generating column names, the model predicts the appearance of column names within a specified subset, computed as the probability Pwherecol [2] Reliability - Ensuring High Success Rate in Query Generation is imperative. The system must produce realistic outputs, even in complex inputs and adverse conditions, necessitating appropriate model training. [3] Security - Given the confidentiality of organizational data, stringent security measures must be in place. Data should be stored securely, adhering to principles of Confidentiality, Integrity, and Availability. [4] Scalability- The system must handle diverse inputs efficiently, including improper ones, while accommodating large datasets.

3.3 Design Architecture

System architecture defines the structure, behavior, and perspectives of a system. An architectural specification formalizes this, aiding understanding. It outlines the system's flow, from raw input to module output.
The system's model processes user input, extracting keywords for query formation. GloVe embeddings, trained using the Stanford NLP core library, map these keywords to corresponding column names. Subsequently, employing a sketch-based slot filling approach, the application generates the pertinent query. The ODBC library is utilized to construct the final query for execution against the database. The raw query is also provided as output for user and developer verification of query accuracy.

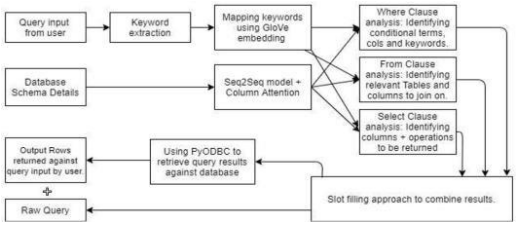


Fig 7. Architecture System Design

3.4 Sketch Based Approach Components

SpeakQL primarily focuses on slot filling within the sketch, eliminating the need to predict both output grammar and content. The sketch closely aligns with SQL grammar, and slot value prediction depends only on relevant slot values through directed dependencies in the sketch. Our model can be conceptualized as a graphical model based on this dependency graph, treating query synthesis as an interface problem on the graph. To handle more intricate SQL queries, we employ sketches supporting richer syntax.

3.4.1 Sequence-to-set

Our approach predicts the presence of column names in a subset, calculating the probability Pwherecol. It utilizes separate LSTMs for encoding column names and questions, with matching dimensions for vectors and embeddings.

3.4.2 Column Attention

In essence, deciding which column best fits the prediction of the actual column. For instance, the token "player" is more significant to predicting the column "player," but "number" is more relevant to predicting the column "no.

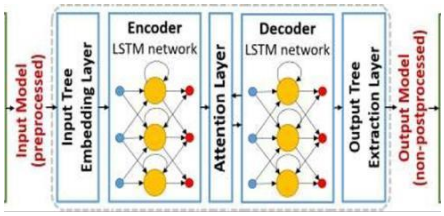
Fig 8. Encoder – Decoder with Column Attention Architecture

3.4.3 Selection of Column Slots

After calculating column inclusion probabilities, SpeakQL selects the top-K columns with the highest probabilities for the WHERE clause and classifies the operator slot for each column, choosing from =, ¡, ¿ with column attention.

3.4.4 Selection of Value Slots

In this scenario, we employ a sequence-to-sequence architecture to generate the Substring, and the order of tokens within the VALUE slot holds significant importance. The



procedure for selecting the column slot in the SELECT clause mirrors that of the WHERE clause, with the distinction that there is only one column to be chosen.

3.5 Input Encoding Details

In this process, the column name and the natural language description are regarded as a tokenized string. Every individual token is mapped to a distinct vector, which is then incorporated into a word embedding vector before being inputted into the bidirectional LSTM model. In this context, we make use of GloVe integration for word embeddings.

3.6 Training Details

The training details encompasses the input and the weight sharing details of the word integration.

3.6.1 Input Encoding Model Details

Column names and natural language descriptions undergo tokenization using the Stanford CoreNLP tokenizer, breaking them down into a sequence of tokens. Each token is then transformed into a one-hot vector before being passed through a word embedding vector and subsequently into the bi-directional LSTM. The 'GloVe' word embedding is employed in this process.

3.6.2 Weight Sharing Details

Only word integration is shared by SQL Net components. In order to forecast various positions in the sketch, the model additionally includes numerous LSTMs. Better performance can be achieved by using different LSTMs to predict distinct

time slots at the same time. Training with word embeddings produces better results.

3.7 Word Embedding Techniques

In NLP, extracting valuable information from text using machine and deep learning techniques involves converting words and sentences into numerical vectors, known as Word Embeddings or Word Vectorization. This process maps vocabulary words or phrases to real-number vectors, facilitating tasks like word predictions and semantic analysis. Word Embeddings play a vital role in various NLP applications.

[1] Compute similar words
[2] Text classifications
[3] Document clustering/grouping
[4] Feature extraction for text classifications
[5] Natural language processing.

There are two Embedding techniques which are of interest to in order to accomplish the task of mapping related words:

3.7.1 Word2Vector

Word2vec consists of a family of shallow neural network models designed for generating word embeddings. These models analyze large text corpora and create high-dimensional vector spaces where each word corresponds to a unique vector. The vectors are organized so that words with similar contextual usage in the corpus are positioned close together in the space.

3.7.2 GloVe Embeddings

GloVe, short for Global Vectors, is a model for word representation that transforms words into vectors based on their semantic relationships. This unsupervised learning approach uses co-occurrence statistics from a corpus to map words into a vector space, revealing meaningful linear structures.

Developed as an open-source project at Stanford, GloVe combines global matrix factorization and local context window methods for word representation learning.

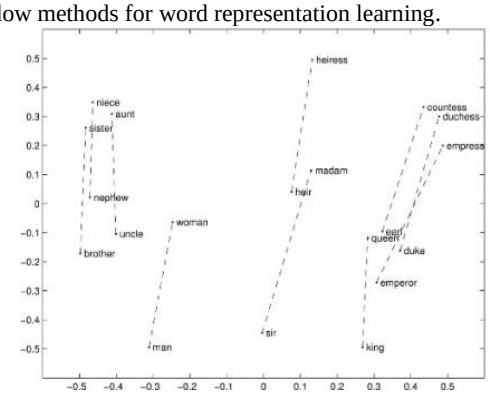


Fig 9. GloVe Word Embedding

3.8 Recurrent Neural Networks (RNN)

Recurrent neural networks (RNNs) preserve information through loops, considering past outputs up to ‘t-1’ in the current decision. This memory feature is beneficial for tasks requiring recollection of previous experiences. In the context of query interpretation, two notable networks are relevant.

3.8.1 Long Short-Term Memory Network

LSTMs track long-term dependencies by retaining contextual information for extended durations through input, forget, and output gates.
[1] **Input Gate**: The amount of information that we store in the current state is regulated by the input gate.
[2] **Forget Gate**: The amount of information (memory from the past experience) that we discard is regulated by the forget gate.
[3] **Output Gate**: The amount of information that should be exposed to the next state is regulated by the output gate.

3.8.2 Gated Recurrent Units (GRUs) Network

By storing contextual data for shorter periods of time and enabling information to go from one state to the next via the reset and update gates, the GRUs achieve the task of tracking long-term dependencies.
[1] **Reset Gate**: This mechanism controls how much of our stored memories (from prior experiences) are erased.
[2] **Update Gate:** The update gate controls how much data is revealed to the following state.

LSTMs and GRUs differ in information flow control. LSTMs excel with extensive sequential data requiring selective information exposure, while GRUs are faster for shorter sequences. GRUs are preferred for speed and shorter data, while LSTMs are accurate for lengthy sequences. In image captioning, GRUs suit the task due to limited information exposure. However, in visual dialog tasks, LSTMs are favored for retaining lengthy dialog histories.

4. RESULTS AND DISCUSSIONS

The results and testing strategies along with current loopholes are discussed below. This comprises of Evaluation Dilemma and strategy adopted for testing.

4.1 Testing

The database currently used comprises three tables: students, marks, subjects, with a provided schema. The system extracts query keywords and maps them to columns to generate SQL queries using a slot-based approach. It handles added complexities like comparisons, multiple conditions, and structured clauses, as well as aggregate functions (SUM, AVG, MAX, MIN). It can handle one layer of complexity for joins and nested queries and incorporates the ’distinct’ keyword for refined output.

4.1.1 Blackbox Testing

Black-box testing focuses on meeting standards while hiding an application's internal workings. When used in machine learning, it functions without regard to model specifics. Finding an appropriate test oracle for result validation is the main obstacle. The steps involved in black box testing are as follows: [1] First, requirements and specifications are reviewed [2] Testers identify valid and invalid inputs to evaluate accuracy [3] They create test cases using the selected inputs [4] The test cases are run, and the software testers compare the expected and actual outputs. [5] Any flaws are rectified and retested.

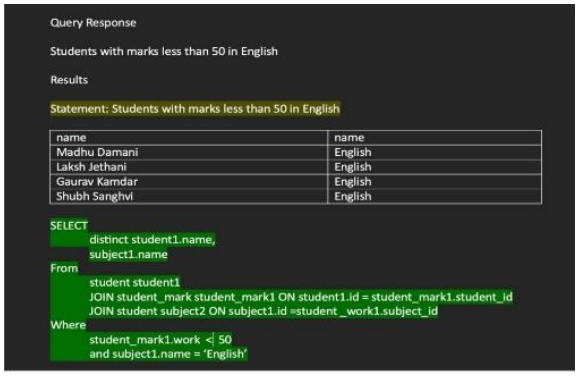


Fig 10. Testing Output

4.2 Evaluation Strategies

Three metrics are available to us for assessing our model's performance:

[1] **Logical-form accuracy:** To see if the generated SQL query and the ground truth match, we compare them.

[2] **Query-match accuracy:** We create a canonical representation from the synthesized SQL query and the ground truth, then we compare if two SQL queries match exactly. The ordering problem can be resolved by using this metric to remove false negatives.

[3] **Execution accuracy:** The degree to which the outcomes of the synthesis query and the ground truth query match.

4.3 Evaluation Dilemma

Due to the nuances in the nature of SQL queries, it is very common to witness two queries resulting in the same output.

Moreover, the use of aliases further makes it difficult to evaluate the correctness of a query syntax.

Lastly, complexities like JOIN A.id on B.id being as correct as JOIN B.id on A.id makes it difficult Thus, we stick evaluate the effectiveness of our model based on the correctness of the final set of tuples generated.

5. CONCLUSION

In conclusion, our innovative approach marks a significant stride toward democratizing database usage, bridging the gap for those unfamiliar with the intricacies of database queries. Leveraging processes like tokenization, parsing, and semantics, we seamlessly translate natural language queries into equivalent SQL commands. This predictive system not only discerns the user's intent but also rigorously validates and executes the resulting query. To maintain peak performance, continuous updates to the system's data dictionary, tailored to the specific domain, are imperative. Our system exhibits remarkable versatility, accommodating both straightforward and intricate queries. While it is a powerful tool for novice users to effectively manage databases, it's worth noting that further development is necessary to encompass the entire spectrum of SQL query types. With our solution, navigating the world of databases becomes accessible and effortless for users of all backgrounds.

6. FUTURE WORK

Some upgrades to this working project can be inclusion of insert and update statements. To implement these features, more focus must be given on the schema of all tables to maintain consistency throughout the database. In addition to that, the project can be made updated to work with NoSQL databases. In RDBMS the integrity of systems is of prime importance, and a well-defined schema is provided beforehand to the model. However, in case of NoSQL databases, the extreme flexibility in the structure can make it difficult to create well defined constraints while training the model. While handling of such databases is beyond the scope of this project, it definitely opens the door for future research opportunities for others to explore further down this path. Last but not least, developing standardized universal metrics which can handle the complexity brought in by the subjective nature of SQL queries would also prove to be helpful for several such related projects. In sum, the above-mentioned areas can be tapped upon by the community for further research.

7. REFERENCES

[1] Ayodele Adebiyi, Aderemi Adewumi, and Charles Ayo. "Stock price prediction using the ARIMA model". In: Mar. 2014. DOI: 10.1109/UKSim.2014.67. s

[2] Khalid Alkhatib et al. "Stock Price Prediction Using K-Nearest Neighbor (kNN) Algorithm". In: 2013. URL: https://api.semanticscholar.org/CorpusID:17150877.

[3] Vaishnavi Gururaj and R ShriyaV. "Stock Market Prediction using Linear Regression and Support Vector Machines". In: 2019 URL: https://api.semanticscholar.org/CorpusID:220725999.

[4] Chien-Feng Huang et al. "A comparative study of stock scoring using regression and genetic-based linear models". In: 2011 IEEE International Conference on Granular Computing. 2011, pp. 268–273. DOI: 10.1109/GRC. 2011.6122606.

[5] Zan Huang et al. "Credit rating analysis with support vector machines and neural networks: A market comparative study". English (US). In: Decision Support Systems 37.4 (Sept. 2004), pp. 543–558. ISSN: 0167-9236. DOI: 10.1016/S0167-9236(03)00086-1.

[6] Bast, H., & Haussmann, E. (2015, October). More accurate question answering on freebase. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (pp. 1431-1440). https://doi.org/10.1145/2806416.2806472

[7] Blunschi, L., Jossen, C., Kossman, D., Mori, M., & Stockinger, K. (2012). Soda: Generating SQL for business users. Proceedings of the VLDB Endowment, 5, 932-943. https://doi.org/10.14778/2336664.2336667

[8] Chang, S., Liu, P., Tang, Y., Huang, J., He, X., & Zhou, B. (2020, April). Zero-shot text-to-SQL learning with auxiliary task. In Proceedings of the AAAI Conference on Artificial Intelligence, (pp. 7488-7495). Association for the Advancement of Artificial Intelligence. https://doi.org/10.1609/aaai.v34i05.6246

[9] Clarke, J., Goldwasser, D., Chang, M. W., & Roth, D. (2010, July). Driving semantic parsing from the world's response. In Proceedings of the fourteenth conference on computational natural language learning (pp. 18-27). https://www.aclweb.org/anthology/W10-2903.pdf

[10] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. https://arxiv.org/abs/1810.04805

[11] Dong, L., & Lapata, M. (2018). Coarse-to-fine decoding for neural semantic parsing. In 56th Annual Meeting of the Association for Computational Linguistics, (pp. 731–742), Association for Computational Linguistics,

Melbourne, Australia. https://doi.org/10.18653/v1/P18-1068

[12] Ferré, S. (2017). Sparklis: An expressive query builder for SPARQL endpoints with guidance in natural language. Semantic Web, 8(3), 405-418. https://doi.org/10.3233/SW-150208

[13] Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J. G., & Liu, T. (2019). Towards complex text-to-SQL in cross-domain database with intermediate representation. In 57th Annual Meeting of the Association for Computational Linguistics, (pp. 4524-4535), Association for Computational Linguistics, Florence, Italy. https://doi.org/10.18653/v1/P19-1444

[14] Youssef Mellah et al. / Journal of Computer Science 2021, 17 (5): 480.489 DOI: 10.3844/jcssp.2021.480.489 488 He, P., Mao, Y., Chakrabarti, K., & Chen, W. (2019). X-SQL: reinforce schema representation with context. arXiv preprint arXiv:1908.08113. https://arxiv.org/abs/1908.08113

[15] Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, (pp. 328-339), Association for Computational Linguistics, Melbourne, Australia. https://doi.org/10.18653/v1/P18-1031

[16] Hwang, W., Yim, J., Park, S., & Seo, M. (2019). A comprehensive exploration on wikiSQL with tableaware word contextualization. arXiv preprint arXiv:1902.01069. https://arxiv.org/abs/1902.01069

[17] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). Albert: A lite bert for selfsupervised learning of language representations. arXiv preprint arXiv:1909.11942. https://arxiv.org/abs/1909.11942

[18] Li, N., Keller, B., Butler, M., & Cer, D. (2020). SeqGenSQL--A Robust Sequence Generation Model for Structured Query Language. arXiv preprint arXiv:2011.03836. https://arxiv.org/abs/2011.03836

[19] Liu, X., He, P., Chen, W., & Gao, J. (2019a). Multi-task deep neural networks for natural language understanding. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, (pp. 4487–4496), Association for Computational Linguistics, Florence, Italy.

[20] Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101. https://arxiv.org/abs/1711.05101 Lyu, Q., Chakrabarti, K., Hathi, S., Kundu, S., Zhang, J., & Chen, Z. (2020). Hybrid ranking network for textto-SQL. arXiv preprint arXiv:2008.04759. https://arxiv.org/abs/2008.04759