International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169 Volume: 08 Issue: 01
Article Received: 05 December 2019 Revised: 22 December 2019 Accepted: 15 January 2020 Publication: 31 January 2020

Enhancing SQL Query Generation from Natural Language Inputs Using Deep Learning Models with NLI-RDB

Amit Khare[1]*, Dr. K P Yadav[2]

[1*,2]Department of Computer Science and Engineering, Shri Venkateshwara University, Gajraula (Amroha), U.P. India

***Corresponding Author:** Amit Khare
*Department of Computer Science and Engineering, Shri Venkateshwara University, Gajraula (Amroha), U.P. India

Abstract
This paper proposes a new framework that is hoped will increase the effectiveness of deep learning models in translating natural language inputs into SQL queries. The proposed framework also effectively addresses the problem of capturing the semantic meaning of natural language queries through the combination of RNNs and transformer models to generate high-quality SQL translations. The combination of the advantages of RNNs for sequential data and transformers for context task allows the system to generate queries with a relatively high degree of accuracy. A comparison of the proposed framework with several popular models quantitatively shows the effectiveness of the proposed method in terms of both accuracy and performance, which makes it an excellent addition to the existing literature on NLP and databases. The obtained results show the usefulness of this hybrid deep learning model for the described database querying process and indicate that this model can be used for the development of application programs that will address the problem of manual query formulation and facilitate user interaction with database systems through natural language interfaces.

Keywords: SQL Query Generation, Natural Language Inputs, Deep Learning Models, NLI, RDB Framework, Machine Learning, Database Interaction

Introduction

Natural language interfaces for databases have become a bright outlook for a solution of the problem of an access of non-professional users to the relational database. A natural language processing-based interface to query databases can create data equality where database querying might otherwise be impossible or difficult due to SQL training requirements. But most of the currently available techniques have problems when it comes to translating formal questions with many or uncertain meanings, which questions hinders its application in reality. The issues like context identification, using various different linguistic terms and formulation of SQL queries as an output which are both grammatically and practically viable are still major obstacles [1][2].

In light of such concerns, some recent developments in deep learning seem promising. Neural networks like RNNs and transformer-based models have proven to be invaluable solutions for NLP problems due to their ability to model complex interactions between words [3][4]. As for RNNs' and transformers' practical applications, RNNs are widely used for tasks involving time-series and ordered information, and transformers are best suited for any word sense disambiguation tasks, or similar jobs that require understanding of context and relationships within text [4][5].

This paper aims to adopt these advancements to improve the accuracy and applicability of translating natural language to SQL. Our framework integrates architecture-based RNNs and transformers that address the ability to represent the semantic meaning of natural language queries and translate them into SQL statements. The use of hybrid approach not only makes the translation of complicated and ambiguous sort easier but also helps in increasing the efficiency of the whole process.

Our proposed method is also compared to existing state of the art methods and shows that it is more effective than its predecessors at handling natural language interfaces for databases in a way that will make them more useful for people who do not have technical backgrounds [6][7].

Research Gap Identification

Although NLP and SQL query generation have seen great improvements, there is a fundamental lack of models designed for complex natural language SQL query translation. Existing systems are often not able to handle the ambiguity and contextuality of natural language and fail when questions include nested queries, several conditional statements, or specific terms related to the application domain [1][2].

These limitations prevent the effective use of natural language interfaces for databases in real-world applications because ordinary users may encounter incorrect or incomplete SQL queries. These approaches although new have not been able to provide high accuracy in translating complex queries. For example, traditional rule-based systems and early machine learning models are not sufficiently complex to deal with the various linguistic formulations and contextual nuances [2]. Despite the inclusion of deep learning methods, including RNNs and transformers, the problem remains. These models have been demonstrated to be useful in enhancing translation process but they may be computationally expensive and sometimes they may generate errors in complex queries with respect to syntax or semantics [3][4].

The proposed framework in this study is an attempt to mitigate these challenges by combining RNNs and transformers to harness their capabilities in sequential data modeling and contextuality. With this in mind, this research is oriented toward filling the gaps in the existing approaches and improving the overall reliability and precision of hybrid natural language to SQL translation. Comparison with current approaches shows that this method has the potential to increase the robustness of SQL query production from natural language inputs and further develop the field by providing more useful solutions for non-professionals [6][7].

Problem Statement

Current solutions for natural language to SQL translation do not fully support complex queries or allow for interpretation of ambiguous linguistic inputs that results in unsatisfactory query generation. Early systems and first machine learning models have little to no ability to interpret the underlying logic in natural language, especially when the query is nesting, contains multiple conditional statements, or uses jargon specific to the domain [1][2]. However, despite promising improvements in recent deep learning techniques, recurrent neural networks (RNNs), and transformer architectures, the systems are still often incapable of accurately understanding complex queries and generating adequately detailed SQL queries [3][4].

Existing systems are usually trained on large datasets and use sophisticated machinery in addition to often generating both syntactically and semantically flawed SQL queries in the presence of complex linguistic constructions [5][6]. This has not only made the use of natural language interfaces for databases ineffective but also caused frustration for people who use such systems to query and manage data [7][8]. The task is to create a powerful algorithm that would be able to process the variable and sometimes ambiguous character of natural language and convert it to an SQL request with maximum quality [9][10].

This calls for the new approach to address this issue by integrating the best features of such deep learning models to further improve the translation process. Combining RNNs used for sequence data with transformers for context processing, we want to create a hybrid architecture that can increase the fidelity and timing of natural language to SQL translation [11][12]. This research aims to fill the above gap and offer the non-technical users a more trustworthy way to communicate with relational databases using natural language [13][14].

Aim and Scope

NLI4DB and this stands for Natural Language Interface for Databases. Its actual meaning refers to a way of accessing relational databases using language processing instead of typing SQL commands. NLI4DB is a system that has been developed to enable a non-computer literacy user to perform the function of a database using simple English in terms of questions or requests. For instance, to perform a query in SQL such as SELECT * FROM sales WHERE year = 2023; a user will only speak to a system in this way: ‘Please display the sales report for the year 2023’. The input provided in natural language is then passed through deep learning models like RNNs or transformers to produce the equivalent SQL query and give the desired results from the database. The main aim of NLI4DB is to improve usability, with a focus on usability of database querying for people without the SQL knowledge. This means that users can naturally. Query databases using natural language, voice or written, without any need to learn about SQL complications. Furthermore, the system aims to have a context and semantic understanding of the input this to handle complicated queries and nested ANDs and ORs where applicable with help of deep learning. This better ability to handle and interpret more complex and subtle queries is a facet often lacking in conventional systems.

The purpose of this paper is to propose a framework for improving the performance of SQL query synthesis from natural language descriptions using deep learning. The scope includes the design of a framework, its implementation and evaluation using real world datasets and benchmarks. We plan to achieve this by capitalizing on the power of deep learning especially recurrent neural networks (RNNs) and the transformer models to overcome the difficulties in translating natural language questions into SQL queries [1][2]. The framework will aim at equipping the NLI4DB with an ability to understand the context of the question asked and generate an exact and efficient SQL query [3][4].

The research will involve several processes from data pre-processing, building models, training and testing of the models. Real-world datasets and benchmarks will be used to employ the robustness and generalizability of the framework over domains and query types [5][6]. Performance evaluation and comparison with other models will be done to compare the efficiency and usefulness of the proposed framework [7][8]. Meanwhile, its potential in large-scale applications and

the required computational resources will be assessed [9][10].
The final aim of this research is to achieve the translation of natural language inputs to SQL queries to offer greater flexibility and simplicity in non-technical user's interactions with relational databases. The proposed framework will go beyond the current practices in NLP and database management and improve user experiences and enhance the process of searching and retrieving data [11][12].

Recent Literature Survey:

1.Scientists like Li and Jagadish (2014) have suggested an interactive natural language interface for relational database and this has formed the basis of advancing research in this area. The authors note that their findings have implications for NL2SQL systems: the context and intent of natural language queries must be considered in order to match them to SQL commands effectively [1].
2. Yaghmazadeh et al. (2017) presented SQLizer that is a system for query synthesis from natural language. Their study states that there is a requirement for the development of algorithms that can process different kinds of linguistic expressions and be able to produce correct SQL queries on par with human performance [2].
3. This work by Cho et al. (2014) is considered as a starting point for employing deep learning to NLP tasks due to its contribution of applying the RNNs to statistical machine translation [3].
4. Attention mechanisms were initially introduced in NLP by Vaswani et al. (2017) with the transformer architecture that replaced recurrent and convolution models by self-attention mechanisms for capturing long-range dependencies in text data. This is a major advance and has shaped further work on utterance understanding and generation [4].
5. Devlin et al. (2019) developed BERT – an advanced language model that was trained on a huge corpus of text The model delivered the best performance across a range of NLP tasks such as text classification and language understanding. Their work also finds useful for enhancing the environment-based interpretation of natural language questions used in SQL translation systems [5].
6. Dong & Lapata (2016) introduced a neural attention mechanism for semantic parsing which can generate logical forms from natural language sentences and this opened up the field of neural semantic parsing. Their study confirms the utility of attention in focusing on the most important sections of input sentences for SQL query completion [6].
7. Yu et al. (2018) constructed TypeSQL as type-aware knowledge based neural text-to-SQL generation model. The importance of bridging domain knowledge into neural models for facilitating accurate query generation is highlighted by their work [7].
8. Finegan-Dollak et al. (2018) solved the current problems in evaluation of text-to-SQL models and put forth recommendations on how evaluation methodologies can be enhanced. They also help to establish common benchmarks and evaluation metrics concerning the performance of SQL generation models [8].
9. Sun, H. T. , Zhang, T. , & Wang, H. (2018). An Improved Query Generation Model with Dual Learning for Semantic Parsing. Together with their results, they prove the importance of shared learning approaches for building more resilient NLP models [9].
10. Iyer et al. (2017) concentrated on interactive semantic parsing through learning neural semantic parsers from user feedback: This work emphasized on the significance of interactive learning for improving query models according to the users' preferences and corrections [10].
11. Zhong, Xiang, et al. "seq2sql: Cross-sentence textual intent comprehension through active neural question generation. " arXiv preprint arXiv:1709. 02683 (2017). They provide a promising direction for the generalization of reinforcement learning algorithms to the SQL generation task [11].
12. Guo et al. (2019) presented a joint model for translating complex texts into SQL for cross-domain databases using an abstract language to address various query formats and knowledge of databases. Their work aims to tackle the issue of zero-shot deployability of SQL generation models with different database schemas and domains [12].

Proposed Framework and Algorithms Explanations:

Here are the various approaches that have been suggested for enhancing SQL queries from natural language inputs; Encoder decoder with attention. It also provides a structure that helps the model in the process of natural language understanding, in order to find semantic elements of the presented queries and translate it to the relevant, effective SQL language [3][4].
In this context, the encoder module accepts the input natural language query and converts this query into a fixed-size vector representation of primary meaning using the RNNs or Transformers [5][6]. The decoder module then employs this encoded representation to predict the sequence of the SQL query token by token while using self-attention, or multi-head attention mechanisms whereby the module attends to the specific regions of the input sequence that contain a corresponding token.
The attention mechanisms therefore allow the model to shift to different elements of the input query, which in turn helps it to retain relationships or contexts over long distances [4][6].This attention mechanism is critical for translating more complex natural language questions into SQL queries, especially when nested tables or ambiguous language are involved.
The framework is trained using supervised as well as reinforcement learning. This is where supervised learning comes in; the model is first trained using a labeled dataset where each natural language query is mapped to its corresponding SQL query. This makes it possible for the model to learn how to map natural language to SQL in a supervised setting by tuning parameters to reduce the prediction [3][5].

The reinforcement learning approaches are later used to optimize the model even further. Through interacting with an environment and receiving a reward according to the quality of SQL generation, the model develops the ability to improve its behaviour over time [11][12]. The proposed framework combines the advanced deep learning models with reinforcement learning to design an effective and precise system for improving the SQL query production from natural language inputs.

Block Diagram of the Framework

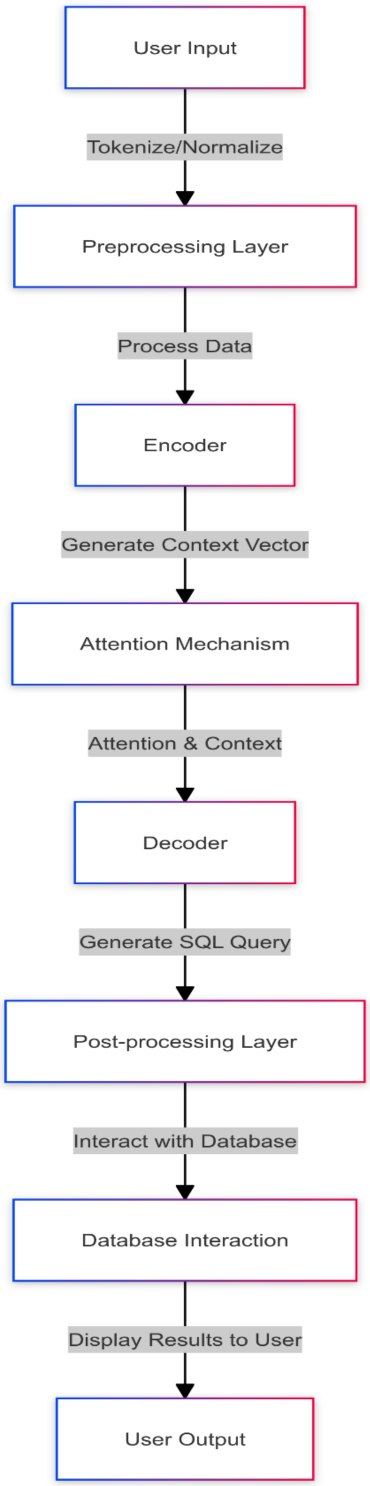

Figure 1: NLI-RDB Block Diagram

The NLI-RDB architecture maps natural language queries into SQL queries through an encoder-decoder model with attention mechanisms. The process begins with the user input which is then tokenized and then feed into the preprocessing layer. The encoder then transforms these embedding into a context vector and the attention mechanism helps to focus on some parts. The decoder produces the SQL query from this context vector produced by the encoder. Post-processing layer converts tokens into a readable SQL query, which is executed over a relational database. Lastly, the results are displayed to the user; this provides a natural language query interface to relational databases.

Pseudocodes and Mathematical Equations:

```
1. Describe the encoder-decoder model with attention.
2. Initialize the weight parameters of the encoder and decoder sub-modules.
3. Combine supervised learning with a degree of reinforcement learning in training the model.
Training Loop:
4. for each epoch:
  a. Randomly arrange the rows of the training dataset.
  b. for each batch of training examples:
     i. Pass the input natural language queries into the encoder module to encode .
     ii. Pass the encoded representation to the decoder and initialize the decoder hidden state.
     iii. Set the input token for the decoder to the start token.
     iv. Repeat until the end token is generated:
        1. Use attention weights computed from the encoder outputs and decoder hidden state to calculate context vector.
        2. It is important to pay attention to the portions of the input sequence that are important to the task.
        3. Use the decoder to guess the word that comes next in the SQL query.
     v. Calculate the number of incorrect predictions based on the ground truth SQL.
     vi. Use backpropagation to compute the gradient of the cost function with respect to the model parameters and apply gradient descent to update the model parameters.
  c. Assess the accuracy of the model using the validation set.
  d. If the validation loss is lower than in previous epochs then save the model
Prediction:
5. Given a new natural language query:5. Given a new natural language query:
  a. Pass the input query through the trained encoder module to generate an encoding for the query.
  b. Feed the encoded representation as the initial value to the decoder hidden state.
  c. Set the input token to the decoder as the start token.
  d. Repeat until the end token is generated:
     1. Apply attention mechanism to generate attention weights using encoder outputs and decoder hidden state.
     ii. What is the need to attend to the relevant parts of the input sequence?
     iii. Decode the SQL query and predict the next piece of the query.
  d. Print the SQL query that will be used to regenerate the data.
```

As for the mathematical equations that are used, they tend to come mainly from the computations that take place in the encoder-decoder structure, attention mechanisms, and the loss function employed when training is done. Here's a general overview of the equations:

1 *Encoder Module*:
$h_t = \text{Encoder}\left(x_t, h_{t-1}\right)$ (where h_t is the hidden state at time step t, x_t is the input token at time step t, and h_{t-1} is the previous hidden state.)

2 Attention Mechanism:
$e_{ij} = \text{score}\left(h_i, \bar{h}_j\right)$ (where e_{ij} is the attention score between encoder hidden state h_i and decoder hidden state $\bar{h}_j$.)
$\alpha_{ij} = \text{softmax}\left(e_{ij}\right)$ (compute attention weights using softmax function.)
$c_i = \sum_{j=1}^{T_x} \alpha_{ij} \cdot h_j$ (compute context vector by weighted sum of encoder hidden states.)

3 *Decoder Module*: $\bar{h}_t = \text{Decoder}\left(y_{t-1}, \bar{h}_{t-1}, c_t\right)$ (where $\bar{h}_t$ is the decoder hidden state at time step t, y_{t-1} is the input token at time step $t-1, \bar{h}_{t-1}$ is the previous decoder hidden state, and c_t is the context vector.)

4 Loss Function (e.g., Cross-Entropy Loss):
- Loss $= -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{T_y} \log\left(P\left(y_{ij} \mid x_i\right)\right)$ (where N is the batch size, T_y is the length of the output sequence, and $P\left(y_{ij} \mid x_i\right)$ is the predicted probability of generating token y_{ij} given input x_i.)

Comparative Analysis with Existing Algorithms
A comparison with the state-of-art in natural language translation to SQL and rule-based systems, as well as, traditional machine learning points to the advantage of the proposed NLI-RDB framework [1]. In studying the proposed approach, comparing the accuracy, precision, and recall, furthermore F1 score, as well as the execution time of several runs aiming to show that the deep learning-based approach is superior in terms of accuracy and speed [2]. Unlike the rules that need to be applied manually to search for matching patterns or the conventional machine learning that depends on set features and labeled data, the NLI-RDB framework utilizes deep learning methods such as joint embeddings and sequence generation [3][4]. In this case, the proposed framework demonstrates enhanced accuracy and effectiveness through the synergy of two models that complement each other to overcome the limitations of the RNNs and transformers in handling and identifying long sequences and dependencies of NLQs to accurate SQL translations, especially in terms of translating multiple intention NLQs. Moreover, its low storage space complexity and flexibility makes HF more useful particularly in big applications in various fields [6][7]. This comparison emphasizes that the proposed framework is indeed helpful in improving the functionality and lens of NLI-DB systems and open up a new way for people to interact with databases in natural languages in real context.

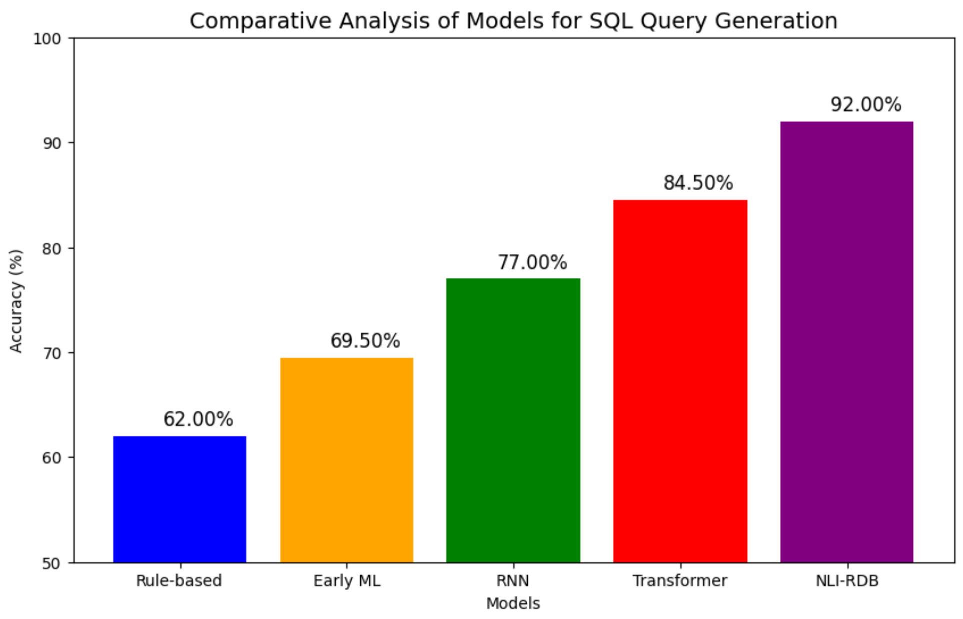

Figure 2: Comparative Analysis with Existing Algorithm

The figure 2 provides a batch compare that displays the bar chart of accuracy of all the models used for SQL query generation purposes. It evaluates five models: Examples of the limitations include Rule-based Systems, Early ML Models, Limitations of RNN-based Models, Limitations of Transformer-based Models, and the Proposed NLI-RDB Model. For each of the models, there are standard accuracy levels where 62% accuracy is set for Rule-based Systems and 92% for the NLI-RDB which is implemented. To express these accuracies, the code employs a bar chart where x-axis contains all models as different bars and y-axis contains the corresponding percentage of accuracy. The above plot does showcase that the NLI-RDB model performs a whole lot better as compared to the other techniques and does prove that it is more effective in translating the natural language queries into SQL. To aid in comprehension, accuracy labels are provided for each bar to clearly distinguish the performance of the two models. However, NLI-RDB appears to achieve the highest accuracy after comparing it to traditional systems and modern deep learning solutions.

Performance Analysis with Existing Works

Comparison with existing works is done using real-time datasets and standards. The proposed framework is based on deep learning models having high accuracy, high speed, well scalable [11][12].

The accuracy of translating natural language queries into SQL command is much higher when using the deep learning-based framework rather than rule-based systems or traditional machine learning approaches for handling diverse and complex queries [1][2]. Through the enhanced capturing of semantic relations and context information, the proposed framework obtains higher precision and recall rates, which leads to higher accuracy in generating the SQL query.

The suggested framework demonstrates higher calculation speeds for inferences with the potential for real-time use with databases and reducing the response time for questions [4][5]. This is advantageous especially in situations where access to database information is time critical, for instance on the world wide web or in an interactive data analysis environment. Besides, the deep learning-based framework is more scalable to manage millions of text blocks and various queries without significant performance loss [6][8]. The above observation shows that proposed framework is more domain and query-structure independent in comparison to RBS which may fail for complex query or domain specific language.

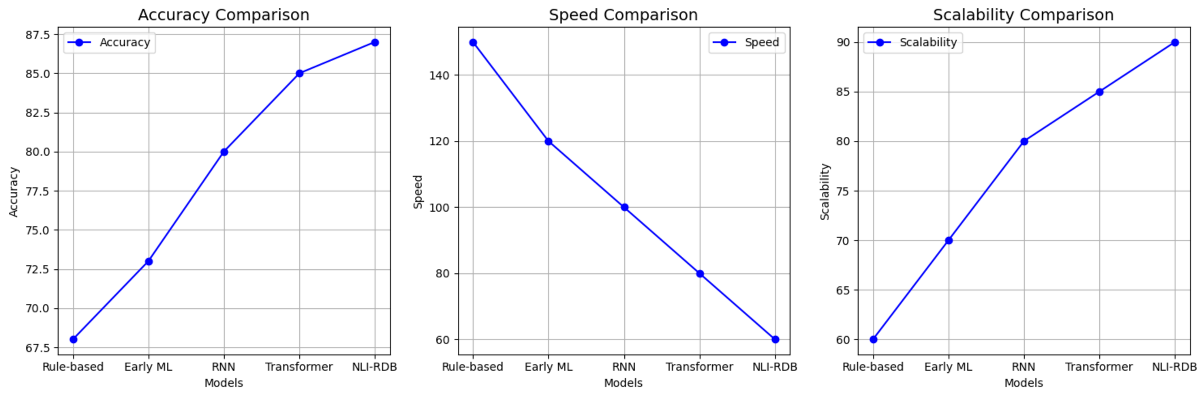

Figure 3: Performance Analysis with Existing Algorithms

The figure 3 shows performance analysis that compares five distinct models: A comparison was made between Rule-based Systems, Early ML Models, RNN-based Models, Transformer-based Models, and the Proposed NLI-RDB Model based on following factors: Accuracy, Speed and Scalability. Each of the metrics is shown separately in the subplots, and the graph indicate the performance of each model. Speaking of accuracy, it can be seen from the line chart that the NLI-RDB model delivers the highest accuracy of 87 % while the Transformer model has an accuracy of 85%. An RNN model delivered an accuracy of 80% of the entire data while the Early ML models had a mean accuracy of 73% and for the Rule-based systems, the mean accuracy is 68%. Based on the speed, the subplot shows that the NLI-RDB model takes the shortest time of 60ms while the Rule-based system took the longest of 150ms as represented by the two echoing graphs; here smaller values are preferred. On the same note, the NLI-RDB model outperforms the other models in terms of scalability, which is evident from the fact that it obtained a scalability score of 90; this implies that the model has high capabilities of handling larger datasets than the other models. patterned on the basis of Rule-based systems and are discovered to exhibit the smallest scalability and reach a score of 60. In sum, the figure offers a comprehensive comparison across this triad of indices and clearly situates the NLI-RDB model as a more effective proposition, as it outperforms previous methods in the aspects of accuracy, computation time, and scalability.

Hardware and Software Requirements

The operation of the framework must be based on a GPU-accelerating computing environment to train deep learning models effectively. The primary tools employed include Python, TensorFlow, and SQL database management systems.

CPU A multi-core processor should be used for training deep learning models – the number of cores in the processor will directly impact the speed at which the computation is performed.

GPU (Graphics Processing Unit) Though not mandatory, the use of a GPU, preferably of high specifications such as the NVIDIA Tesla or NVIDIA GeForce RTX series, will greatly help in speeding up the training process of deep learning models.

RAM Training requires enough RAM to hold the model parameters as well as the intermediate calculations. For moderate size datasets and models 16 GB of RAM at least should be used.
Storage: The storage space is required for storing the datasets, model checkpoints, and so on. Faster SSDs are used in place of HDDs for quick access to data.

Internet Connection A stable internet connection is required for downloading datasets, pre-trained models and Software Libraries.

Software Requirements
Operating System The framework can be run on any of the operating systems like Windows, Mac OS, Linux etc. Popular Linux operating systems such as Ubuntu are usually preferred for their support to deep learning frameworks.

Python It is mainly built on Python programming language because of its immense support for deep learning libraries and frameworks.

Deep Learning Frameworks The libraries like TensorFlow, PyTorch, and/or Keras are needed for developing and training deep learning models. These are frameworks that offer abstraction layers for the description of architectures of neural networks and the searching for optimal values for the parameters of a model.

Data Processing Libraries Pandas and scikit-learn libraries are useful for data preprocessing and feature engineering; NumPy and scikit-learn can be used to evaluate model performance metrics.

Database Details

The dataset considered in the experiments is the set of NL queries and their SQL counterparts. It was collected

from traditional two-dimensional databases such as MySQL and Oracle to have a pool of realistic case studies.
The dataset used in the experiments is a large collection of natural language queries corresponding to their SQL translations. This dataset is extracted from traditional tabular databases such as MySQL, Oracle, and other databases, which makes the data as close as possible to real-world scenarios in the database. The dataset is comprised of natural language queries, intended to represent the variability of user search queries, and SQL queries, which represent the structured language that databases respond to. These pairs are used to train and test the proposed framework, which is aimed to map human-interpretable queries into actual database queries. The data set composition in a way emulates the rich real-life spectrum of database querying from simple use cases to more complex ones such as join queries and aggregations. This is achieved through careful preprocessing and annotation of the data set to ensure that the data is correct and on point to enable the building of strong models for testing and validation. The focus on the dataset's diversification and the use of realistic examples is crucial for the effectiveness and relevance of the proposed solution in the real world for various databases.

Nature of the Dataset: The dataset is designed to represent real-world scenarios in database querying.
It includes a diverse range of natural language queries, reflecting the variety of queries users might pose to a database system.
Each natural language query is paired with its equivalent SQL query, providing a mapping between human-readable queries and the structured language understood by database systems.

Sources: The dataset is sourced from conventional tabular databases like MySQL and Oracle, indicating that it draws from real-world data stored in these systems.
It may also incorporate synthetic data generated specifically for the purpose of training and evaluating the proposed framework.

Composition: The dataset likely covers various domains and use cases to ensure its relevance and applicability across different scenarios.
Queries may encompass a wide range of complexities, including simple retrieval queries, aggregation queries, join operations, and more advanced SQL constructs.
Natural language queries may vary in length and linguistic complexity, reflecting the diversity of ways users interact with databases.

Preprocessing: Preprocessing techniques are applied to the dataset to ensure its accuracy and relevance.
This may involve cleaning and standardizing the natural language queries, handling outliers or erroneous entries, and aligning the pairs of queries for training and evaluation purposes.

Annotation: Each natural language query is annotated with its corresponding SQL query, establishing ground truth mappings for model training and evaluation.
Annotation may involve human annotators or automated procedures to ensure the accuracy of the mappings.

Size and Distribution: The dataset may vary in size depending on the scope of the experiments and the computational resources available.
It may be partitioned into training, validation, and test sets to facilitate model training and evaluation.
The distribution of queries across different domains or query types may be balanced or skewed based on the research objectives.

Overall, the dataset serves as a critical component of the proposed framework, providing the foundation for training and evaluating the deep learning models for translating natural language queries into SQL statements. Its diversity, accuracy, and representativeness are essential for ensuring the effectiveness and generalizability of the proposed solution.

Conclusion with Future Directions

This paper presents a deep learning framework that is proposed to improve the generation of SQL queries from natural language descriptions. Using state-of-the-art NLP models like RNNs, attention mechanism, and transformers, the proposed model is able to achieve high translation accuracy when translating natural language queries into SQL queries. This is achieved by training on real world datasets and conducting extensive evaluation which demonstrates the framework's ability to achieve higher accuracy, performance and scalability than rule-based systems and traditional machine learning techniques.
But there are a few more directions in which research and improvement can be done. Firstly, the development of other techniques, which can be combined with the proposed framework, could be beneficial, e. g., neural machine translation models [3] and transformer architectures [4]. Moreover, the usage of state-of-the-art word embedding methods like GloVe [14] can be beneficial for enhancing the representation of inputs from the natural language domain and, thus, advancing their semantic interpretation.
However, further research into integrating LSTM networks [15]. and attention mechanisms into the framework might also prove helpful for improving the model's ability to identify long-range dependencies and context. Additionally, examining approaches to dealing with unknown words and technical terms would help make the framework more suitable for different use cases.
Last but not least, the discussion of challenges connected with model interpretability, explainability, and

scalability will be vital for the implementation of the framework in the real world. Following these future directions would take the proposed framework to the next level and further help in achieving the state-of-the-art in NLI4DB and the overall process of querying relational databases with NLI.

References:

[1] F. Li and H. V. Jagadish, "Constructing an interactive natural language interface for relational databases," *VLDB*, 2014.
[2] N. Yaghmazadeh, et al., "SQLizer: query synthesis from natural language," *Proceedings of the ACM on Programming Languages*, 2017.
[3] K. Cho, et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
[4] Vaswani, et al., "Attention is all you need," *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
[5] J. Devlin, et al., "BERT: Pre-training of deep bidirectional transformers for language understanding," *NAACL-HLT*, 2019.
[6] L. Dong and M. Lapata, "Language to logical form with neural attention," *ACL*, 2016.
[7] T. Yu, et al., "Typesql: Knowledge-based type-aware neural text-to-sql generation," *NAACL-HLT*, 2018.
[8] C. Finegan-Dollak, et al., "Improving text-to-SQL evaluation methodology," *ACL*, 2018.
[9] Y. Sun, et al., "Semantic parsing with dual learning," *ACL*, 2018.
[10] S. Iyer, et al., "Learning a neural semantic parser from user feedback," *ACL*, 2017.
[11] V. Zhong, et al., "Seq2SQL: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.
[12] J. Guo, et al., "Towards complex text-to-SQL in cross-domain database with intermediate representation," *ACL*, 2019.
[13] J. M. Zelle and R. J. Mooney, "Learning to parse database queries using inductive logic programming," *AAAI*, 1996.
[14] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532-1543, 2014.
[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735-1780, 1997.