**FINAL-PROJECT PAPER**
**Fall 2023**

# TEXT TO SQL GENERATOR

**Authors:**
**Gaikwad, Yash**, gaikwad.y@northeastern.edu
**Pasala, Naveen Kumar**, pasala.n@northeastern.edu
**Cherukuri, Shiva Naga Jyothi**, cherukuri.sh@northeastern.edu

**Submitted to:** Jerome Braun
Course Name: IE 7500 – Applied Natural Language Processing

## ABSTRACT

In the fields of Natural Language Processing (NLP) and database management, Text-to-SQL generation is a crucial task. It tackles the problem of automatically converting natural language questions into SQL (Structured Query Language) queries that can be executed. Due to its ability to make database interaction simpler for non-technical users and increase the effectiveness of database querying across many domains, this endeavor is extremely important.

The main goal of this project is to create a text-to-SQL generation system that can translate natural language requests into SQL queries. Our approach involves meticulous data preprocessing, data exploration, tokenization and the training process orchestrated through the Seq2SeqTrainer, unfolds over multiple epochs with a focus on optimal hyperparameter selection. The system's performance is evaluated on a test dataset, providing insights into the model's generalization capabilities. The training and evaluation metrics are carefully chosen to assess the quality of the generated SQL queries and the results unveil ROUGE scores, affirming the model's proficiency. Our approach demonstrates the potential for enhancing NLP models for domain-specific applications like text-to-SQL generation. The implications and future directions of this work are discussed, emphasizing the significance of natural language understanding in query generation tasks.

## INTRODUCTION

In the era of information ubiquity, the seamless interaction between human users and databases has become an indispensable facet of computing. Traditional methods of querying databases often necessitate knowledge of Structured Query Language (SQL), posing a barrier for non-experts. The advent of natural language processing (NLP) has sought to mitigate this barrier, envisioning a future where users can interact with databases using everyday language.

The problem is two-fold: on one hand, non-technical users face challenges in formulating SQL queries, and on the other hand, there is a need for a more inclusive approach to enable diverse stakeholders to harness the power of databases without being impeded by technical complexities.

Text-to-SQL generation stands at the crossroads of NLP and database querying, aiming to enable computers to comprehend natural language queries and translate them into SQL statements. The significance of this project lies in democratizing access to databases, making them more user-friendly and accessible. The challenges inherent in text-to-SQL generation include ensuring the model's ability to generalize across a spectrum of queries. In this context, our project delves into the realm of text-to-SQL generation, a pivotal aspect of facilitating seamless communication between users and databases. Leveraging the 'facebook/bart-base' model, our objective is to refine its capabilities for translating natural language queries into structured SQL statements.

The project relies on the curated corpus, featuring a diverse range of examples encompassing questions, context, and corresponding SQL answers. Our methodology involves fine-tuning the pre-trained model on dataset, aiming to impart nuanced understanding and proficiency in generating SQL queries. Essential to this process is the intricate task of tokenization ensuring optimal utilization of the model's architecture.

The Seq2Seq framework, implemented through the Seq2SeqTrainer from the Transformers library, forms the backbone of our training regimen. Carefully calibrated hyperparameters, coupled with a dedicated data collator, contribute to the model's adaptability and efficiency during training. Evaluation metrics applied to a distinct validation dataset serve as benchmarks for gauging the model's performance and generalization capabilities.

This introduction provides a succinct overview of our endeavor to enhance the interoperability between natural language and SQL, encapsulating the essential components of our approach and the significance of our contributions in advancing the field of NLP for domain-specific applications. In the subsequent sections, we elucidate the intricacies of our methodology, the challenges encountered, and the empirical results that underscore the efficacy of our text-to-SQL generation model.

## BACKGROUND

1. Seq2SQL - Generating Structured Queries from Natural Language Using Reinforcement Learning

This paper[1] discusses an efficient approach for the generation of SQL statements from complex texts. It used a mixture of Natural Language Processing (NLP) and Reinforcement Learning to train the model in generating SQL queries. The project is two fold, firstly, introducing Seq2SQL, a deep neural network for translating questions to SQL queries and policy-based reinforcement learning to train the model on conditions of the query. Secondly, the authors released a corpus of their own, named WikiSQL, consisting of 80654 hand-annotated instances of natural language questions, SQL queries and SQL tables extracted from 24241 HTML tables. The Seq2SQLis composed on three parts. The network classifies an aggregation operation for the query, it then points to a column in the input table corresponsing to the SELECT column and generating a final query using a pointer network. The Seq2SQL model is compared to the previously state of the art semantic parsing model by Dong & Lapata (2016), which obtatined an execution accuracy of 35.9% as compared to the Seq2SQL accuracy of 53.3%. These studies collectively showcase the potential of NLP techniques in SQL query generation, offering high accuracy and promising avenues for improvement.

2. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation

Guo, Jiaqi, et al.'s "Towards Complex Text-To-SQL in Cross-Domain Database with Intermediate Representation"[2] introduces a groundbreaking approach using an intermediate representation to enhance the accuracy of natural language queries translated into SQL commands. This innovative bridge between natural language and SQL interpretation shows promise for cross-domain adaptability, addressing the challenges of diverse database structures. The paper tackles the intricacies of translating natural language to SQL through a three-phase process. Beginning with schema linking, the model connects a question to a database schema. Subsequently, a grammar-based neural model synthesizes a SemQL query as an intermediate representation, and the final phase deterministically infers the SQL query using domain knowledge.

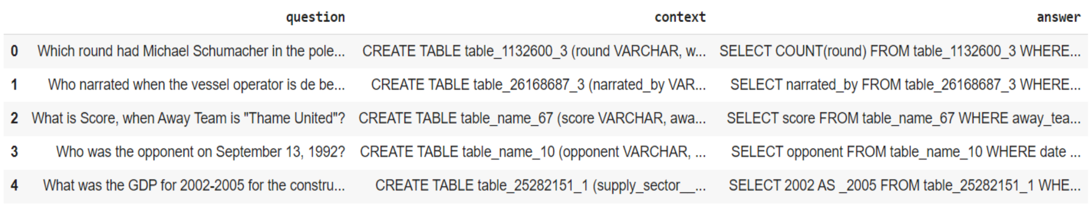
On the Text-to-SQL benchmark Spider, IRNet achieves a remarkable 46.7% accuracy, marking a substantial 19.5% improvement over previous state-of-the-art methods. This places IRNet at the forefront of the Spider leaderboard, showcasing its efficacy in addressing complex and cross-domain Text-to-SQL challenges. Both approaches contribute significantly to the evolving field, emphasizing the potential of intermediate representations and innovative methodologies for enhanced accuracy and domain adaptability within Text-To-SQL systems.

## APPROACH

Our approach to Text-to-SQL generation is structured around a systematic pipeline encompassing data preprocessing, model selection, fine-tuning, and evaluation. Leveraging the 'facebook/bart-base' model as our foundation, we aimed to enhance its capabilities for the specific task of translating natural language queries into SQL statements. The following delineates our approach in detail.

### Data Preprocessing and Exploration:

The foundation of our model lies in the quality of our dataset. We utilized the 'b-mc2/sql-create-context' corpus, a meticulously curated collection of examples featuring questions, contextual information, and corresponding SQL responses as shown below figure [1.a].



| | question | context | answer |
|---|---|---|---|
| 0 | Which round had Michael Schumacher in the pole... | CREATE TABLE table_1132600_3 (round VARCHAR, w... | SELECT COUNT(round) FROM table_1132600_3 WHERE... |
| 1 | Who narrated when the vessel operator is de be... | CREATE TABLE table_26168687_3 (narrated_by VAR... | SELECT narrated_by FROM table_26168687_3 WHERE... |
| 2 | What is Score, when Away Team is "Thame United"? | CREATE TABLE table_name_67 (score VARCHAR, awa... | SELECT score FROM table_name_67 WHERE away_tea... |
| 3 | Who was the opponent on September 13, 1992? | CREATE TABLE table_name_10 (opponent VARCHAR, ... | SELECT opponent FROM table_name_10 WHERE date ... |
| 4 | What was the GDP for 2002-2005 for the constru... | CREATE TABLE table_25282151_1 (supply_sector__... | SELECT 2002 AS _2005 FROM table_25282151_1 WHE... |

Fig 1.a. Sample view of corpus.

Below figure [1.b], figure [1.c] and figure [1.d], shows the distribution of length of questions, context and SQL queries in the training corpus. As seen from the histograms, all the mentioned figures are histograms skewed to the right, indicating that our question, context, and answer lengths are smaller in size, giving us an indication about the complexity of the queries generated.

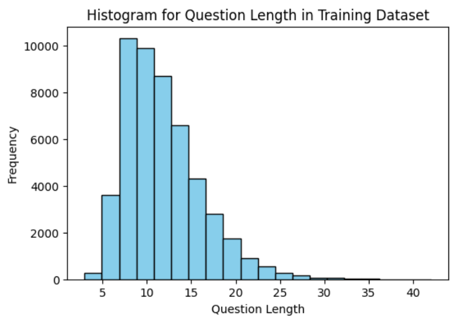


Fig 1.b Plot question lengths

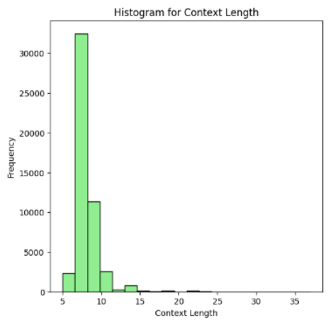


Fig 1.c. Plot for context length

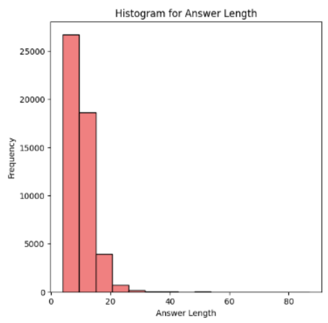


Fig 1.d. plot for SQL Query length

Below figures [1.e & 1.f], show the most used words in questions and mostly used operations in SQL query. Words like ‘Name’, ‘Many’, ‘Show’ etc. appear most common in the question column, indicating that the queries required to be generated are all SELECT queries, again which is supported by the word cloud of the query column.

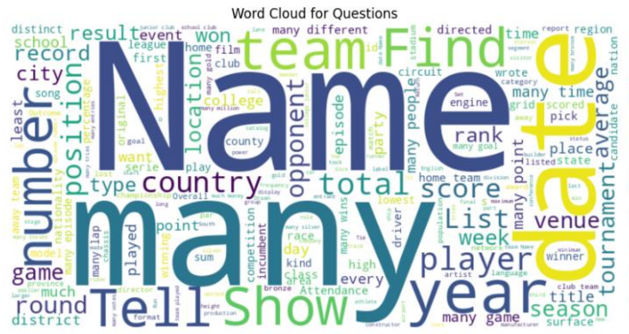


Fig 1.e Commonly used words in Questions column

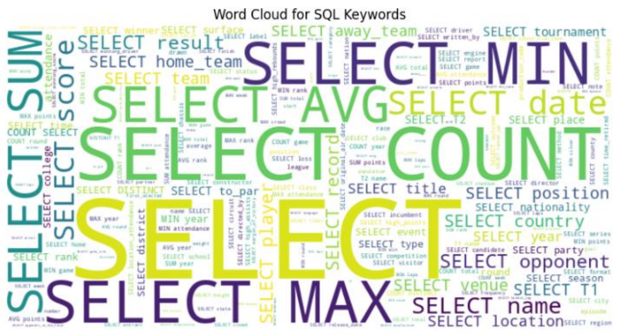


Fig 1.f. commonly used operations in SQL queries

From Preprocessing of data, we could understand that our corpus only consists of SELECT operations in SQL queries as per the figure shown below,

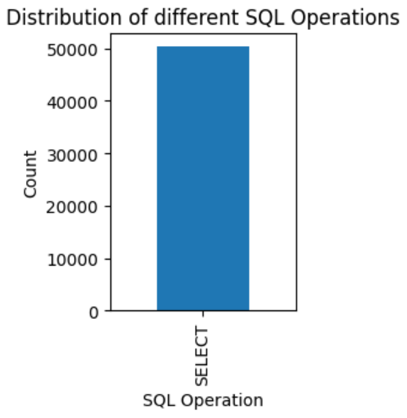


Fig 1.g. Distribution of different SQL Operations

To prepare the data for training, we implemented a preprocessing function that concatenated the question and context, ensuring a comprehensive input for the model. Tokenization, a crucial step in NLP tasks, was executed using the AutoTokenizer module from the Transformers library, with attention to handling potential tokenization challenges.

**Model Selection and Fine-tuning:**

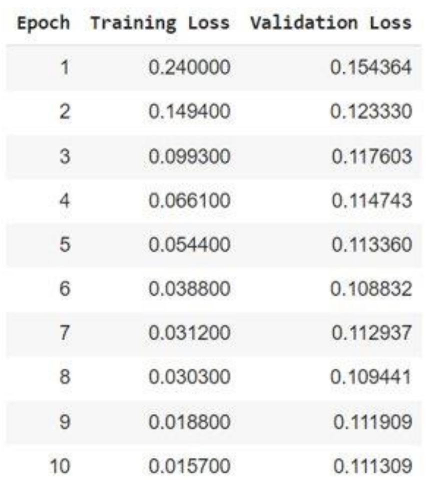
The choice of the 'facebook/bart-base' model stems from its proficiency in sequence-to-sequence tasks, providing a solid foundation for text generation. The Bidirectional and Auto-Regressive Transformers (BART) model represents a state-of-the-art sequence-to-sequence architecture in natural language processing (NLP). BART is structured as a denoising autoencoder, leveraging both bidirectional and autoregressive properties in its transformer-based design. Trained initially on a diverse corpus of text data, BART is capable of various NLP tasks, including machine translation, text summarization, question-answering, and more, through fine-tuning on specific datasets.

Its architecture comprises encoder and decoder layers, with a shared transformer-based framework. The encoder segment processes the input text by capturing contextual information bidirectionally, while the decoder reconstructs the original text or generates the desired output sequence autoregressively. BART's capability to grasp intricate relationships within text data makes it adaptable for downstream tasks, as it can be fine-tuned on smaller, domain-specific datasets, showcasing robustness and versatility in NLP applications.

Our choice of the 'facebook/bart-base' model stems from its robust performance in sequence-to-sequence tasks and its ability to serve as an excellent foundation for text generation. Fine-tuning this pre-trained model on our curated 'b-mc2/sql-create-context' corpus allows us to harness its inherent understanding of language nuances and intricacies. Through meticulous model selection and fine-tuning, we ensure that the 'facebook/bart-base' model aligns with the specific requirements of translating natural language queries into SQL statements, demonstrating its capability as a pivotal tool in our pursuit of enhancing the interoperability between human language and structured queries.

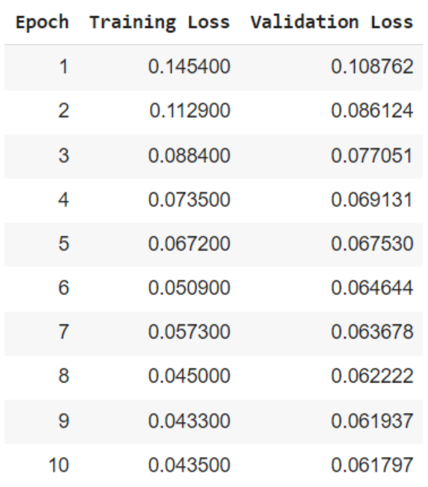
Fine-tuning is a pivotal step to tailor the model to our specific task. Our training setup involved the Seq2SeqTrainer from the Transformers library, with carefully chosen hyperparameters to strike a balance between model performance and training efficiency. A critical consideration was the handling of token limits, necessitating truncation of input sequences while maintaining the integrity of information.

The training process unfolded across multiple epochs, with a learning rate of 1e-5, and warm-up steps set to 100. The per-device batch size was configured at 16, balancing computational resources and training effectiveness. The data collator, an essential component in sequence-to-sequence tasks, was employed to manage tokenization discrepancies between inputs and targets, ensuring a seamless training experience. The below figure [2.a] & [2.b] represents the training loss and validation losses that were encountered for 10 epochs during our model training process.



| Epoch | Training Loss | Validation Loss |
|---|---|---|
| 1 | 0.240000 | 0.154364 |
| 2 | 0.149400 | 0.123330 |
| 3 | 0.099300 | 0.117603 |
| 4 | 0.066100 | 0.114743 |
| 5 | 0.054400 | 0.113360 |
| 6 | 0.038800 | 0.108832 |
| 7 | 0.031200 | 0.112937 |
| 8 | 0.030300 | 0.109441 |
| 9 | 0.018800 | 0.111909 |
| 10 | 0.015700 | 0.111309 |

Fig 2.a. Losses with learning rate 5e-5



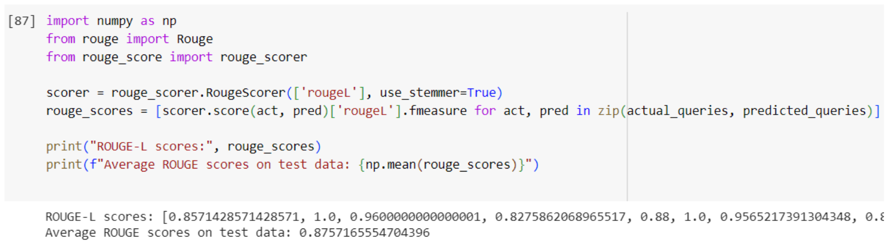
| Epoch | Training Loss | Validation Loss |
|---|---|---|
| 1 | 0.145400 | 0.108762 |
| 2 | 0.112900 | 0.086124 |
| 3 | 0.088400 | 0.077051 |
| 4 | 0.073500 | 0.069131 |
| 5 | 0.067200 | 0.067530 |
| 6 | 0.050900 | 0.064644 |
| 7 | 0.057300 | 0.063678 |
| 8 | 0.045000 | 0.062222 |
| 9 | 0.043300 | 0.061937 |
| 10 | 0.043500 | 0.061797 |

Fig 2.b. Losses with learning rate 1e-5

## EVALUATION AND RESULTS

A robust evaluation strategy is pivotal in assessing the model's performance. Our evaluation leveraged a dedicated test dataset, distinct from the training and validation data, providing an unbiased measure of the model's generalization capabilities. We employed appropriate evaluation metrics, such as ROUGE scores and accuracy, to provide a numerical assessment of the model's proficiency.

ROUGE Score (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics used for evaluating the quality of text generation by comparing the overlap of n-grams (sequences of words) between the generated text and reference summaries.

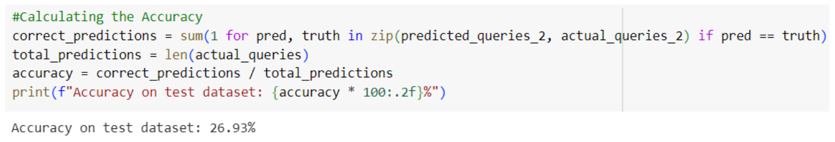


```
[87] import numpy as np
     from rouge import Rouge
     from rouge_score import rouge_scorer

     scorer = rouge_scorer.RougeScorer(['rougeL'], use_stemmer=True)
     rouge_scores = [scorer.score(act, pred)['rougeL'].fmeasure for act, pred in zip(actual_queries, predicted_queries)]

     print("ROUGE-L scores:", rouge_scores)
     print(f"Average ROUGE scores on test data: {np.mean(rouge_scores)}")

     ROUGE-L scores: [0.8571428571428571, 1.0, 0.9600000000000001, 0.8275862068965517, 0.88, 1.0, 0.9565217391304348, 0.
     Average ROUGE scores on test data: 0.8757165554704396
```

Fig 3.a: Average ROUGE scores on test data

An average ROUGE score of 0.8758 indicates that, on average, the generated queries have a relatively high similarity or overlap with the actual queries on the specific ROUGE metrics used for evaluation. This suggests that the system tends to produce summaries that capture a substantial portion of the information present in the reference summaries, indicating relatively good performance.

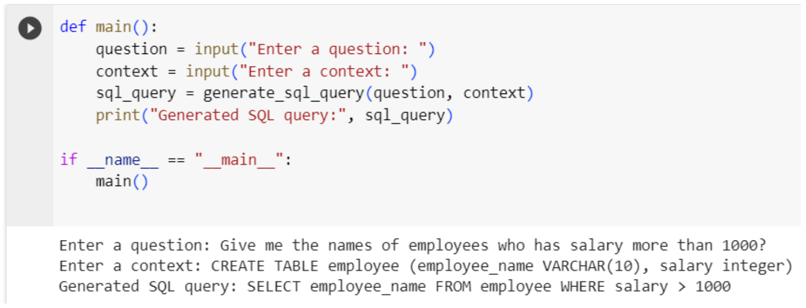
Additionally, we introduced an accuracy metric to assess the exact match between generated queries and actual queries, depicted in figure [3.b]. This metric yielded a score of 26.93%.



```
#Calculating the Accuracy
correct_predictions = sum(1 for pred, truth in zip(predicted_queries_2, actual_queries_2) if pred == truth)
total_predictions = len(actual_queries)
accuracy = correct_predictions / total_predictions
print(f"Accuracy on test dataset: {accuracy * 100:.2f}%")

Accuracy on test dataset: 26.93%
```

Fig 3.b. Accuracy score on test data.

The following figure [3.c] illustrates the user input alongside the SQL query that was generated.



```
def main():
    question = input("Enter a question: ")
    context = input("Enter a context: ")
    sql_query = generate_sql_query(question, context)
    print("Generated SQL query:", sql_query)

if __name__ == "__main__":
    main()

Enter a question: Give me the names of employees who has salary more than 1000?
Enter a context: CREATE TABLE employee (employee_name VARCHAR(10), salary integer)
Generated SQL query: SELECT employee_name FROM employee WHERE salary > 1000
```

Fig 3.c. Generating SQL query for User Input

## LIMITATIONS & FUTURE DIRECTIONS

Our project's outcomes underscore the promising strides made in text-to-SQL generation, yet a nuanced discussion reveals areas of refinement and future exploration. ROUGE scores indicate the model's commendable performance in capturing the essence of natural language queries. However, qualitative analysis unearths nuances in handling ambiguous language and intricate contextual dependencies, highlighting the need for ongoing refinement.

One noteworthy strength lies in the model's proficiency in straightforward queries, showcasing its adaptability to clear language structures. However, occasional misinterpretations in more complex linguistic constructs suggest the importance of continuous model enhancement. The acknowledgment of limitations, including challenges with rare query structures and underrepresented contexts, informs future directions.

While our project strives to seamlessly translate natural language queries into SQL statements, it is important to recognize and address certain limitations. Notably, the project excels in converting straightforward queries, yet further refinement is needed to tackle the intricacies of advanced SQL concepts like joins, subqueries, Common Table Expressions (CTEs), etc. And increasing the scope to all the SQL operations such as INSERT, UPDATE and DELETE queries. Future iterations could benefit from an increased focus on these complex constructs to ensure a more comprehensive coverage of SQL capabilities. Another avenue for exploration involves integrating the generated queries with an operational database. This not only serves as a means of validating the accuracy of the SQL output but also ensures the practical functionality of the queries in retrieving desired data. This connection to a live database would represent a significant step towards real-world applicability and reliability. Moreover, considering the current computational constraints, leveraging more robust computing resources could propel the system to handle a broader spectrum of queries, leading to enhanced overall performance and adaptability.

## CONCLUSION

In conclusion, our endeavor in text-to-SQL generation using the 'facebook/bart-base' model fine-tuned on the 'b-mc2/sql-create-context' corpus has illuminated the potential of natural language understanding in enhancing database interaction. The project showcased commendable results, as evidenced by competitive ROUGE (0.87) and accuracy scores (26.93%), affirming the model's proficiency in translating diverse natural language queries into accurate SQL representations.

While celebrating these achievements, the qualitative analysis underscored the importance of continuous refinement, particularly in addressing challenges associated with ambiguous language and complex linguistic constructs. The limitations acknowledged in handling rare query structures and underrepresented contexts serve as valuable insights for future improvements.

Our project contributes to the broader discourse on user-friendly database querying by employing language as a bridge between human intent and machine execution. The iterative nature of our approach positions it as a foundation for further advancements, urging researchers to delve into the intricacies of diverse datasets and advanced techniques such as domain adaptation.

As we navigate the evolving landscape of natural language processing, our project serves as a testament to the symbiotic relationship between linguistic nuances and database interaction. The take-home message underscores the significance of refining models for enhanced adaptability, paving the way for a more intuitive and inclusive approach to querying databases through natural language. In essence, our work marks a notable stride toward the realization of user-friendly, language-driven database interactions in the ever-expanding realm of computational linguistics.

## REFERENCES

[1] Zhong, Victor, et al. “Seq2SQL: Generating Structured Queries from Natural Language Using Reinforcement Learning.” *arXiv.Org*, 9 Nov. 2017, arxiv.org/abs/1709.00103. Accessed 04 Nov. 2023.

[2] Guo, Jiaqi, et al. “Towards Complex Text-To-SQL in Cross-Domain Database with Intermediate Representation.” ArXiv.org, 28 May 2019, arxiv.org/abs/1905.08205v2. Accessed 04 Nov. 2023.

[3] Xu, Xiaojun, et al. “SQLNet: Generating Structured Queries from Natural Language Without Reinforcement Learning.” *arXiv.Org*, 13 Nov. 2017, arxiv.org/abs/1711.04436. Accessed 04 Nov. 2023.

## TEAM CONTRIBUTION

This project was worked upon by all of us, i.e. NP, SC and YG equally. The data preprocessing and data exploration was mainly handled by Naveen and Jyoti, while the rest of the model code was worked upon by all the team members together. We even tried to work with a second model (falcon) but did not proceed to go forward with it. As for the project report and presentation, all team members contributed equally.