National College of Ireland

A Robust Text-to-SQL Parser With Optimized Pretraining Approach

MSc Research Project
Data Analytics

Kirubakaran Balaraman
Student ID: x19241658

School of Computing
National College of Ireland

Supervisor: Hicham Rifai

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Kirubakaran Balaraman
Student ID:	x19241658
Programme:	Data Analytics
Year:	2020/2021
Module:	MSc Research Project
Supervisor:	Hicham Rifai
Submission Due Date:	16/08/2021
Project Title:	A Robust Text-to-SQL Parser With Optimized Pretraining Approach
Word Count:	6345
Page Count:	19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	16th September 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A Robust Text-to-SQL Parser With Optimized Pretraining Approach

Kirubakaran Balaraman
x19241658

Abstract

Semantic parsing of natural language to Structured Query Language (SQL) has recently become a popular research topic with the release of the manually annotated WikiSQL dataset. Most recent research has used the encoder-decoder architecture with Bidirectional Input Representations for Transformers (BERT) for generating input embeddings. These models are made content-aware by passing table schema information coupled with the database contents as additional knowledge. Though BERT-based models have achieved superior performance over non-BERT variants, the model is undertrained. In the newer model like RoBERTa, hyperparameters are further optimized and trained on a bigger corpus. This research takes a novel approach by using RoBERTa to generate input embeddings for the decoder models like Bi-LSTM. A sketch-based slot filling approach is adopted for the Bi-LSTM decoders. The research also improved the size of the data and robustness of the model to diverse linguistic patterns by performing synonym-based paraphrasing. Considering the data privacy, this research omitted using the database contents as an additional input, thus making the model only schema-aware. The model is evaluated using the logical form and execution accuracy and showed better performance than the non-schema-aware counterparts. However, the model's performance is lower than those using table contents. The decrease in performance is acceptable owing to the privacy vs. performance tradeoff. The logical form and execution accuracy of the model with the test set are 68.8% and 76.6% respectively.

1 Introduction

In the modern world the amount of data stored in databases is increasing rapidly due to the invaluable insights that can be mined by deep analysis of the data. Businesses started relying on machine learning and deep learning algorithms to make predictions and make critical decisions. Relational databases are the most widely used database for storing and maintaining the data due to its simple, straight-forward approach of storing data in tables. The healthcare, retail and financial institutions rely on databases to store and retrieve the records. In order to use the full potential of these databases, users need a lot of technical knowledge about how database works, the underlying architecture, storage structure of the data and a language called SQL. This acts as a barrier for non-technical users such as nurses accessing patient records, retail workers, etc. So there is a need for an interface between such users and the database. This motivated several researchers to work on semantic parsing task to convert natural language text to SQL for building Natural Language Interface for Databases (NLIDBs).

Initially rule-based and grammar-based algorithms are created to perform the translation. However these NLIDBs are tailored for specific databases and organizations, and reusing it for a different database needed considerable changes. Models such as ATHENA (Sen et al.; 2020), SODA works based on this approach and are not able to handle the ambiguous nature of the natural language. With the advancement in natural language processing using deep learning models, the researchers started addressing the above problem by training neural networks to generate an executable SQL query. But the models performed with poor accuracy due to the insufficient data to train on. The release of the popular text-to-SQL datasets like WikiSQL and Spider gave rise to more research on this topic. Early researchers considered the semantic parsing as a sequence-to-sequence translation similar to language translation problem (Zhong et al.; 2017). But the problem with this approach is that improper ordering of the generated SQL will lead to syntactically incorrect queries throwing errors when executed in the database.

A hybrid approach of the rule-based and sequence-to-sequence models motivated researchers to consider text-to-SQL as a slot-filling problem. The underlying syntax of the SQL is kept constant by only predicting values for different clauses such as SELECT, FROM and WHERE. Researchers were able to achieve state-of-the-art results with the existing models by incorporating a pretrained model called BERT as a input layer to generate intermediate representations. Such representations provided the decoders with additional knowledge by using table schema and content. The pre-trained models released by (Liu et al.; 2019; Yin et al.; 2020; Herzig et al.; 2020) are trained on huge volumes of tabular and textutal data and are built on top of BERT. In this study we replace BERT by a more optimised version called RoBERTa, in which the hyperparameters of the BERT are tuned carefully and trained with more epochs.

1.1 Research Question

How data augmentation techniques with synonym-based paraphrasing coupled with rigorously optimised RoBERTa model improves the text-to-SQL parsers ?

1.2 Motivation

Previously released models such as the one in Guo and Gao (2019) relied on the contents of the data assuming that access is available. Models that makes use of content needs retraining when implemented in a new database. With industries having millions of records in its databases it becomes a overhead to use the whole database content as input to the model. Also such models used the initially released BERT encoder for generating knowledge representations which was outperformed by pretrained models like RoBERTa, XLNET, TaBERT, etc.,. The natural language is ambiguous and same sentence can be represented in many variations. For instance, "Show me the records of all the students?" can also be represented as "Fetch the details of all the pupils". In order to make the model effective, we need to consider such linguistic variations. To increase the robustness of semantic parsers when deployed on an unknown database, current models have begun to incorporate the underlying database structure in addition to the submitted natural language question (Wang et al.; 2019; Lin et al.; 2020).

1.3 Contributions of the research

More recent researches made use of the table schema to provide the model with the knowledge about the existing relationships between the question keywords and the table columns (Guo and Gao; 2019; Lin et al.; 2020). In this paper techniques to augment the training data with more diverse linguistic patterns are implemented. For this purpose, the Paraphrase Database (PPDB) is used to inject semantic variations in the data by replacing words with synonyms. The existing encoder architecture is tweaked by replacing BERT with RoBERTa (Liu et al.; 2019), and trained on the augmented data.

The remainder of this paper is organized as follows. First, in Section 2, similar works are critically evaluated to find the research gap. Section 3 describes the research methodology and details the steps taken as part of the research. In the next section, Section 4, the design specifications explain the novel approaches carried out, the model architecture, and the algorithms used. The Section 5 explains the actual implementation details and the tools used. The performance of the text-to-SQL model is evaluated with suitable metrics and discussed in section 6. In the end, section 7 concludes the research work with key points and future work.

2 Literature Review

In the field of natural language processing, semantic parsing is a hot area for research. The problem of converting natural language questions to SQL queries is being studied around the world. Researchers can accomplish this task with the help of state-of-the-art models and benchmark datasets like Spider and WikiSQL (Zhong et al.; 2017).

2.1 Different approaches for semantic parsing

2.1.1 Rule-based approach

In rule-based NLIDB systems, the database knowledge is used to interpret the provided NL question, and the intermediate query is converted to an actual SQL query. Such knowledge is acquired by following different strategies in different researches. Baik et al. (2019) developed an NL-SQL system that extracts domain knowledge from SQL query logs and utilizes it to map keywords in NL questions to database elements such as columns, table names, and values and determine whether a keyword represents a token. The model was validated using IMDB, MAS, and YELP (Baik et al.; 2019) datasets. When more complex and queries are submitted, it failed to detect the keyword. The future models aided in resolving the issue.

In a model such as ATHENA++, database information is defined as domain ontologies that are constructed manually for each domain. This results in a data model with real-world linkages by translating and mapping question tokens to columns, values, or tables (Sen et al.; 2020). ATHENA++, which is based on ATHENA, operates by interpreting natural language questions using a domain-based ontology and overcomes earlier systems' shortcomings in building complicated nested queries (Sen et al.; 2020). It determines whether a nested query is required from the NL query keywords using nested query detectors and builders. Even though the users have to follow a specific pattern of natural language question. In order to give more freedom for naive users submitting the queries to the system, NaLIR is introduced in Li and Jagadish (2014). The system

works interactively in parsing the questions submitted by the user and determining the key elements such as column names and where clauses. When the system couldn't parse a specific part of the question, the interactive communicator repeatedly asks the users' input until the final output is reached. The problem with this approach is the system is more dependent of the user in parsing complex queries.

2.1.2 Deep Learning approach

With the outstanding capabilities of machine learning and deep learning models, researchers are devising an improved solution to old problems on the planet. Researchers are utilizing neural networks to address a variety of issues in healthcare, technology, the military, and space science, among others. Recent research has attempted to solve the Natural language to SQL (NL-SQL) challenge using neural networks.

Models without database schema Usta et al. (2021) developed the Database Tagger (DBTagger) paradigm with the primary goal of optimizing keyword linkage during the encoding step. The researchers developed a keyword mapping method that uses the part-of-speech (POS) tags from the input NL query (Usta et al.; 2021). Here, a unique architecture is designed to extend the sequence tagging framework by utilizing database query logs. This model is schema-independent and is intended for use with relational databases. These methods struggled to identify the relationships between question keywords and the database elements when the words are paraphrased. For example, when presented with a question to a student database such as "Show the total number of pupils in the class", these models struggled to identify that "pupil" denotes "students". The following section will discuss how incorporating the database schema improves SQL creation.

Models including Database schema representation The supervised sequence-to-sequence models are trained on pairs of Natural Language (NL) and SQL queries. Recent research has concentrated on optimizing this by adding the underlying table schema and, in certain cases, actual data. By applying appropriate rules, the rule-based systems convert the NL query to an intermediate form.

The WikiSQL dataset used in this study is released as part of the research work conducted by Zhong et al. (2017). Seq2SQL is the first model trained with the WikiSQL dataset, which used a method similar to language translation using Bidirectional Long Short Term Memory (Bi-LSTM) networks. It uses reinforcement learning to predict correct SQL by providing rewards to the generated SQL. Although it achieved state-of-the-art performance compared to the previous models, it was below 50%, which is too low for implementation in the real world. And also, it struggled when the ordering of the where conditions in the question are too complex.

SQLova was designed by Hwang et al. (2019), which included table schema by concatenating the NL question with the table headers. In contrast to other encoder-decoder approaches, this research encoded the concatenated input using a BERT encoder and passed the resulting word embedding to the SQLova layer. The decoding process was inspired by the prior work of See et al. (2017); Zhong et al. (2017). Additionally, it contains a shallow layer that assists in confining the BERT output tokens to specific sketches such as select-column, select-aggregation, and so on.

Rather than concatenating the table structure, Bogin et al. (2019) first encoded the database schema using Graph Neural Networks (GNNs), which is provided along with the

input query. This encoded graph representation is then utilised by bi-directional LSTMs in both the encoding and decoding steps. This technique results in more accurate SQL query creation, as demonstrated by the researchers when comparing the WikiSQL and SPIDER benchmark to the prior state-of-the-art models(Bogin et al.; 2019). Using a similar GNN technique, Song et al. (2019) offers the Hierarchical Schema Representation Network as a schema-aware neural network with a decomposing design (HSRNet). It is capable of tackling the challenging and cross-domain process of text-to-SQL conversion. The HSRNet represents the database schema's relationships using a hierarchical schema tree and encodes them using GNN. Unlike sequence-to-sequence models, this process is separated into three parts (Song et al.; 2019). When both the query and database schema are supplied, column candidates are identified and a SQL query template is constructed. A detail completion module populates the data by utilizing the column candidates and the associated drawing. The approach is validated against a variety of baseline models and is found to improve accuracy. In contrast to the method described above, the authors employed a new way to encode the database schema in their research by dubbing the Relation-Aware self-Attention (RAT) mechanism (Wang et al.; 2019). The relational structure and relationships between the query keywords, such as column-table relationships, main and foreign keys, are stored in a single sequence (Wang et al.; 2019). Following that, schema linking is performed, which associates sequence tokens with database elements. The paper kept the model simple without using BERT to make the model comparable with other models.

Along with the database structure, some researchers leveraged the database's actual data to improve model performance. Li et al. (2020) developed the SeqGenSQL model, a weakly supervised model that enables direct translation. By modifying the architecture and performing question and data augmentation, a state-of-the-art pretrained model, T5, is used. Additionally, the model's input includes column headers and type information, which aided in achieving higher accuracy with the WikiSQL benchmark dataset (Zhong et al.; 2017). It made advantage of the database's information by sampling the table's first two or three examples. Additionally, the research addresses the issue of hallucination by utilizing a gated layer that assists in either extracting or producing column names from the query. In BRIDGE, a model architecture developed by Lin et al. (2020), a slightly different technique is used, in which the possible value sets for each field are inputted rather than the actual data. For example, the marital status field can have the following value sets:{"Single", "Married", "Divorced"}. It used a BERT encoder and a pointer-generator network to decode (See et al.; 2017). Guo and Gao (2019) utilizes the entire database's content and generates two representational vectors by matching the table values and column headings to the NL queries. These feature vectors are supplied to the decoder as extrinsic inputs to assist in query translation. In the publications mentioned above, keyword mapping was demonstrated to be effective when the database content was combined with the schema. But the main problem with the above approaches is that the model relies too much on the database contents. With companies having infinite records stored in their databases, inputting all that knowledge to the NLIDB system makes it an overhead. When handling with databases containing critical information like bank records and health data of patients in the hospitals will give rise to ethical issues such as privacy and confidentiality.

Another well-known neural network technique is slot filling, which utilizes a predetermined SQL sketch to fill the slots with expected columns and values from natural language questions. TypeSQL, as introduced by Yu, Li, Zhang, Zhang and Radev (2018),

organizes the slots and uses Bi-LSTMs to forecast the output for each slot. The model is type-aware, implying that it associates question keywords with certain types such as column, number, or person . This technique was also employed in Recursively Yielding Annotation Network for SQL (RYANSQL) by Choi et al. (2020), which provided sketches for tackling the SPIDER benchmark's difficult queries. Due to the fact that the sketches are customized for a single dataset, they may have an effect on the model's performance when confronted with a difficult query type not covered by WikiSQL. This sketch-based method is also used in the SQLova (Hwang et al.; 2019). The technique of fuzzy semantic to structured query language (F-SemtoSql) is based on a fuzzy choice during the decoding stage (Li et al.; 2019). It employs the same method of slot filling as SytnaxSQLNet (Yu, Yasunaga, Yang, Zhang, Wang, Li and Radev; 2018) and arranges the slots according to a dependency graph. It classifies the natural question into four types of events (atomic, aggregate, complex, and composite) and uses the attention mechanism at each stage of the decoding process. During the training phase, random masks are applied to the input natural question tokens to allow the machine to predict the masked values based on the context (Li et al.; 2019). In each preceding article, incorporating the database schema and content resulted in improved keyword linkage, which enriched the translation process by providing additional information about the underlying database.

2.2 Data augmentation

Data augmentation is a widely used technique for producing more data from existing data. Many contemporary state-of-the-art neural models use this technique to enhance their performance. While current benchmarks such as SPIDER and WikiSQL already contain a substantial amount of data, enhancing them will assist expand the dataset and enable varied natural queries for the same SQL. This information can then be utilized to train the models.

Weir et al. (2020) created a natural language pipeline for implementing the technique of data augmentation. It is constructed in such a way that any neural translator may be inserted into it to boost performance (Weir et al.; 2020). It accomplishes this improvement in accuracy by paraphrasing the natural language questions contained in the dataset to generate new data, hence making the plugged model more resistant to linguistic changes (Weir et al.; 2020). It validated the pipeline by bringing in cutting-edge models such as SyntaxSQLNet (Yu, Yasunaga, Yang, Zhang, Wang, Li and Radev; 2018), which boosted accuracy by approximately 7%. Additionally, the previously stated SeqGenSQL Li et al. (2020) model augments the data by employing a reverse training technique thus creating 250k fresh samples via NL questions generated by SQL citeli2020seqgensql. This is accomplished by inverting the input and output of the T5 transformer, which generates new NL queries from the SQL in the training set, as well as silver data (augmented data).

Yu et al. (2020) took a fresh strategy to augmentation in the research by using it during the pre-training phase rather than the end-of-training phase. This approach to pre-training is referred to as Grammar-Enhanced Pre-Training For Table Semantic Parsing (GRAPPA). It generates a grammar template for question-SQL combinations taken from the Spider dataset and then mixes it with the real table schema to generate new samples (Yu et al.; 2020). These samples are subsequently encoded and transformed to SQL queries using the transformer. Thus, data augmentation enhanced the accuracy of earlier state-of-the-art models when used with benchmark datasets.

2.3 Pretrained models for input encoding

The pretrained models are widely used in question answering, part-of-speech tagging and other similar NLP problems to easily identify the keywords and provide knowledge as hidden representations. Using these pretrained models for text-to-SQL task has proved to improve the accuracy. BERT is a pre-trained model for modeling language that was introduced by Devlin et al. (2018). The majority of state-of-the-art models acquired traction as a result of their use of the BERT encoder in their study. The Generalized Autoregression Pre-training Approach for Language Understanding (XLNet), the Robustly Optimized BERT Pre-training Approach (RoBERTa) (Liu et al.; 2019), and the DistilBERT (Sanh et al.; 2019) pre-trained models were all built on top of BERT to address the various BERT-related issues. Pre-trained models encoded both textual and tabular input.

Herzig et al. (2020) published a weakly supervised question-answering model that does not construct logical forms. As a result, Table Parsing (TAPAS) can derive operations from natural language rather than from formal definitions. It contained extra features to capture table structure and two new layers introduced for predicting the aggregator and to select cell. With extra embeddings that capture tabular structure on top of BERT and two classification layers for cell selection and aggregator operator prediction. Liu et al. (2019) introduced RoBERTa, an optimized BERT, by dynamically modifying the masking pattern used to mask the training data. Additionally, it trained for a longer period of time than BERT, resulting in state-of-the-art performance on the GLUE, RACE, and SQuAD benchmarks.

A different variant of BERT known as TaBERT incorporated textual and semi-structural tabular data (Yin et al.; 2020). The research proposed a method for converting a table structure to a linear format that is BERT compatible. Additionally, it offers a content snapshot of the table by capturing rows that match the question's keywords. It was trained on 26 million tables and paragraphs and is intended to be used in conjunction with any semantic parser (Yin et al.; 2020). It was evaluated on Spider and outperformed its competitors.

2.4 Summary and research gap

The text-to-SQL problem is approached using both deep learning and a rule-based technique. The majority of models were trained and evaluated using WikiSQL or the Spider dataset, which both give massive hand annotated datasets. After the introduction of BERT, the focus of the research shifted away from direct sequence-to-sequence models and moved towards the introduction of an intermediate representation between the BERT encoder and the final query generator. These logical representations are either SQL-constrained templates or graph networks. Using pretrained models such as BERT to complement existing models has significantly improved their performance. Following the publication of BERT, numerous optimized models such as RoBERTa, TaBERT, and GRAPPA were made available. The researchers then concentrated on enhancing the decoding mechanism, employing techniques such as execution-guidance and pointer generation networks. Several researchers merged the grammar-based and neural approaches through the use of sketch-based slot filling algorithms. Finally, several models experimented with training the model with more data through the use of data augmentation approaches, which increased the model's robustness.

According to the authors Liu et al. (2019), BERT is undertrained and training with

more data and changing the hyperparameters can improve the model. Employing improved models like RoBERTa will help improve the performance. Most of the state-of-the-art models rely on BERT for encoding. Augmenting the data with paraphrases helped to improve performance in a similar problem like question answering. In this research, these changes are applied to already existing model and experimented to achieve better performance.

3 Methodology

In order to carry out this research, the Knowledge Discovery in Databases (KDD) has been used with slight modification to fit the problem of semantic parsing as shown in Figure 1. The highlighted part of the process flow diagram shows major changes made and also the novelty part of this research.

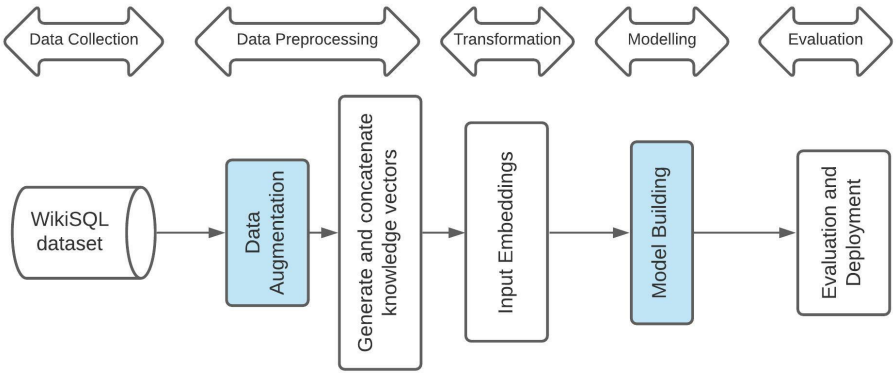

Figure 1: Process flow of the methodology followed

3.1 Data Collection

By reviewing the previous research papers in semantic parsing, it was observed that WikiSQL and Spider are the popular datasets, containing huge volumes of question-SQL pairs. After analysisng the datasets, it is found that the Spider dataset contains many databases from different domain areas in contrast to the single database of WikiSQL. Considering the limitations in time and computational resources and to reduce the complexity, the scope of the project is confined to single database and WikiSQL dataset has been chosen. WikiSQL is released as part of the study conducted by Zhong et al. (2017) and contains two JSON files: one with the natural language question and the SQL query and the other having the schema information of the tables. An annotated form of the dataset in which the natural language question is tokenized is also available to download.

3.2 Data Augmentation

The WikiSQL dataset is augmented to increase the size of the dataset and to inject variations in the natural language questions. This is a common technique in NLP problems in which the textual data is augmented by generating paraphrases of the existing English

language sentences. In this way, the model can be trained to learn different linguistic patterns and can understand that same question can be paraphrased in different ways. For example, two different questions can denote the same SQL query. This is done by doing synonym based paraphrasing with the help of synonyms available in PPDB.

3.3 Model Building

Model building involves two stages: encoding input tokens to generate knowledge embeddings using BERT and generating SQL based on the RoBERTa output along with the question and table schema.

3.3.1 Input Encoding

After augmenting the data, the next task is to create input embeddings using the RoBERTa pretrained model (Liu et al.; 2019). In order to provide to help the LSTMs in the next stage of the process to better identify the relationships between the question keywords and the column headers, this step is carried out. The two knowledge vectors which are the question knowledge and the header knowledge is generated and concatenated to every instance of data using the algorithm used in Guo and Gao (2019). These vectors carries additional information which will be fed as inputs the Bi-LSTMs in the step of decoding to identify the columns and numbers.

3.3.2 SQL Generation using Bi-LSTM

The Bi-LSTMs takes the input embeddings from the RoBERTa and passes it through six sub modules to predict SQL components. These SQL components are filled in SQL sketch like a slot filling problem to produce the final SQL query. This will then be compared with the ground truth query and submitted to the database to evaluate. We will discuss the evaluation methods followed in the next subsection.

3.4 Model Evaluation

After the model has been trained with the annotated training data, it is evaluated using two new measures that is specific for the text-to-SQL semantic problems. The papers reviewed in the related work section use such measures and it will be useful to compare our model's performance with previous baselines.

3.4.1 Logical Form Accuracy

The logical form accuracy is used to evaluate the semantic correctness of the SQL. This is done by comparing the ground-truth (from train data) query with the predicted query for each clause separately. We are not using the exact match accuracy of the predicted queries as the same query can be written in different ways. The match is checked within each clauses such as select, from and where. The ordering of the conditions in the where clause is ignored in this accuracy calculation.

3.4.2 Execution Accuracy

The execution accuracy is used to evaluate the syntactic correctness of the predicted SQL. This is done by executing the query in the SQLite database created using the table

schema data. The output is considered correct if it successfully executes without throwing an error. The functions for these evaluations are provided as part of the WikiSQL dataset page.

3.5 Deploying the model

The model checkpoint that was trained as part of the training process is used to generate SQL queries for a sample database. A sample SQLite student database is created with few records and the trained model is deployed to convert natural language questions submitted to the database to predict SQL queries. Multiple manually formed English language questions are inputted to the model along with the table schema to validate the predicted output.

4 Design Specification

4.1 Augmentation with Synonym paraphrasing

The paraphrasing of the annotated WikiSQL dataset is done with the help of synonyms in the PPDB database by following the algorithm 1. The algorithm iterates through each sentence in the data and replace synonyms of the words available in the PPDB database. Then the new sentences with slight synonym changes are added to the existing data and the question and header vectors are generated. By doing this, the model can be made more robust to linguistic variations and will help improve the model.

Algorithm 1 Synonym paraphrasing to augment data

```
function PARAPHRASE(Data)
    SynonymDatabase ← FetchPPDBSynonymData
    for every sentence in Data do
        for every word in sentence do
            if word has synonyms in SynonymDatabase then
                wordlist ← synonym
            else
                wordlist ← word
            end if
            newSentence ← sentence(wordlist)
        end for
        Add newSentence to ParaphrasedData
    end forreturn ParaphrasedData
end function
```

4.2 Knowledge vector generation

Additional knowledge to BI-LSTMs for SQL generation is provided by concatenating two knowledge vectors with the actual data called question vector and header vector. These knowledge carry information about the question keywords and the table headers by identifying the columns. The algorithm used in Guo and Gao (2019) is adopted to generate these vectors. It matches the question keywords with the column headers passed

```
{
 question: 'Tell me what the notes are for South Australia',
 sql: {'agg': 0, 'conds': [[3, 0, 'SOUTH AUSTRALIA']], 'sel': 5},
 table_id: '1-1000181-1',
 question_tok: ['Tell','me','what','the','notes','are','for','South','Australia'],
 header: ['State/territory','Text/background colour','Format','Current slogan','Current series','Notes'],
 types: ['text','text','text','text','text','text'],
 phase: 1,
 question_knowledge: [0, 0, 0, 0, 4, 0, 0, 0, 0],
 header_knowledge: [0, 0, 0, 0, 0, 1]
}
```


Figure 2: Contents of single processed data in the WikiSQL dataset

as input and uses the vectors to represent this by marking the index. For example, the question shown in Fig 2 contains a word *"notes"* which is present in the header vector. The algorithm identifies this and generates two vectors *question_knowledge* with *length = number of words in question* and *header_knowledge* with *length = number of headers*. These vectors contains 4 and 1 in the corresponding indices to denote the relationship.

4.3 RoBERTa model for input embedding

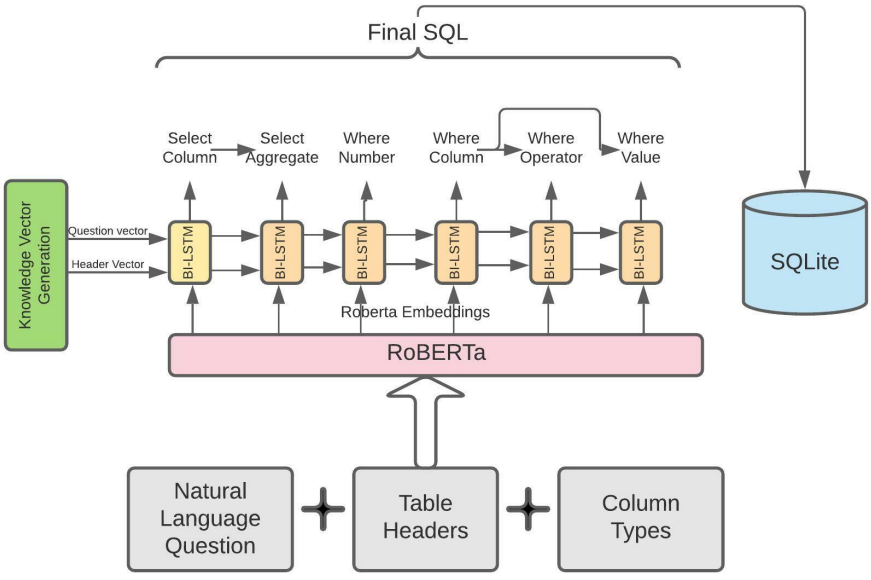

Figure 3: Architecture design of the model

The RoBERTa model introduced by Liu et al. (2019) is imported from the Transformers package in python. The previous work of SQLova (Hwang et al.; 2019) and Guo and Gao (2019), used BERT for tokenization and creating input embeddings. The

'roberta-base' pretrained model is imported and used to produce knowledge embeddings by passing natural language question concatenated with the table schema as shown in Figure 3. The input to the RoBERTa consists of tokens in input question along with headers merged with it as follows: $[CLS], Q_1, Q_2, Q_3, ...Q_{\mathrm{n}}, [SEP], H_1, [SEP], H_2, [SEP], ..., H_{\mathrm{k}}, [SEP]$, where $Q_{\mathrm{i=1,2,...n}}$ denotes the question tokens and $H_{\mathrm{i=1,2,...k}}$ denotes the table headers The hidden states from the last layer of the RoBERTa is passed into the LSTM models for each clauses.

4.4 LSTM models for predicting SQL

Syntax-guided slot filling approach used in previous researches such as Choi et al. (2020); Hwang et al. (2019) is used to predict values for separate modules such as *sel-agg, sel-col, where-col, where-op, where-val, where-num*. The sketch can be found below:

```
SELECT $sel-agg($sel-col) FROM $table
WHERE $where-col $where-op $where-val
```

In this modules, the *where-num* denotes number of where conditions needed. For each module separate Bi-LSTMs are used to predict the final column, value and operator and the slots are filled accordingly. The input dimensions for the Bi-LSTMs is the same dimensions of the RoBERTa hidden states. Each Bi-LSTMs perform different task.

select-column The first Bi-LSTM predicts the probability of the column that needs to be selected from the given natural language utterance and also using the knowledge vectors.

select-aggregate takes value among one of the following choices:{*MAX, MIN, COUNT, SUM, AVG, NONE*}. This is dependent upon the predicted select column value as well.

where-number takes an integer value and denotes the number of where conditions associated with the SQL query.

where-column is predicted by using the information from the table schema, the question and the RoBERTa embeddings. This prediction has a dependency on the where-number prediction as it denotes how many columns need to be predicted.

where-operator takes a value from the following set: {"=","<",">"}. This is the operator that acts on the where column and is directly connected to it.

where-value is extracted from the natural language question and the RoBERTa embeddings helps the LSTMs to identify this and predict.

5 Implementation

The implementation of this research is carried out using Python 3.6 programming language due to the wide range of libraries that it provides. Due to the computationally demanding nature of this research, the implementation environment is setup in Google

colab to train and run the models. The annotated version of the WikiSQL dataset is uploaded to the google drive and mounted to the google colab environment. The code adopted from previous researches are also uploaded along with the data.

The code is written in Pytorch as it is commonly used for NLP tasks involving tensors running on GPU. The dataset consists of train, test and dev sets First, the dataset is

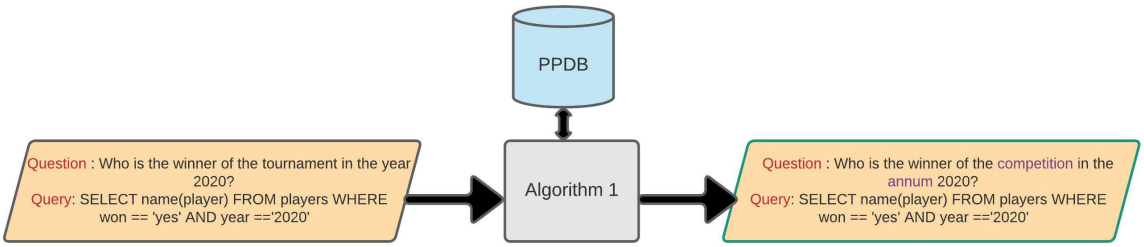

Figure 4: Generating new data by paraphrasing

preprocessed by tokenizing the natural language questions as shown in Fig 2. The word ans sentence tokenizers from the *nltk* library is used for this task. Then the knowledge headers are generated and concatenated with the actual data by using the question and header vector generation algorithms as discussed in the previous section. In order to increase the dataset size the data augmentation is carried out. First, The PPDB data, downloaded from their website[1], is used to extract synonyms. Then algorithm 1 explained in the design section is written by making considerable changes to the data preprocessing code adopted from the previous researches (Guo and Gao; 2019; Pal et al.; 2021). This helps generating new sentences with words replaced by synonyms present in the PPDB database. For example, the Fig 4 shows how the words 'tournament' and 'year' in the question is replaced with the synonyms 'competition' and 'annum' to form a new sentence. This will help the model learn the different words having similar meaning.

In the next step, we import the RoBERTa base model *(roberta-base)*, the tokenizer and the optimizer from the *Transformers (version = 3.4.0)* package. The adopted code is changed to use RoBERTa as an encode instead of the BERT model. Then both the RoBERTa and the Seq2seq model were loaded. The input for the RoBERTa is the preprocessed data containing the question tokens, corresponding table headers along with their type information. The RoBERTa generate embeddings which will then be fed to the next stage of Bi-LSTM models. The The input dimensions of the Bi-LSTMs are set to the dimensions of the hidden states of RoBERTa. The LSTMs consists of 2 hidden layers, with dropout rate being 0.3. The question knowledge and header knowledge dimensions are given as 5 and 3 respectively. Then the linear transformation is applied to the data followed by softmax activation.

The models are then trained for 5 epochs with the training set and validated with the dev sets. The size of the train, dev and test sets are 56355, 8421 and 15878 respectively The best model is saved in the project folder by comparing the logical form accuracy in each epoch. The saved models are used to evaluate using the test sets and accuracies are recorded.

In order to make inferences, a sample SQLite database containing student records is created and uploaded to the google drive. In order to connect to the database and submit

[1]http://paraphrase.org/#/download

query, the *sqlalchemy (version=1.3.23) and records* packages of python are imported. The model output is checked by inputting manually formed natural language questions along with the table schema information.

6 Evaluation

Model	Dev LF (%)	Dev EX (%)	Test LF (%)	Text EX (%)
Our model (Data Aug + RoBERTA + LSTMs)	69.7	77.4	68.8	76.6

Table 1: Logical Form and Execution accuracy for Dev and Test sets. LF and EX stands for logical form and execution accuracy respectively.

After building and training the model, evaluation is crucial for assessing the performance of the model. The evaluation will be based on logical form accuracy and the execution accuracy which are explained as part of methodology in section 3.

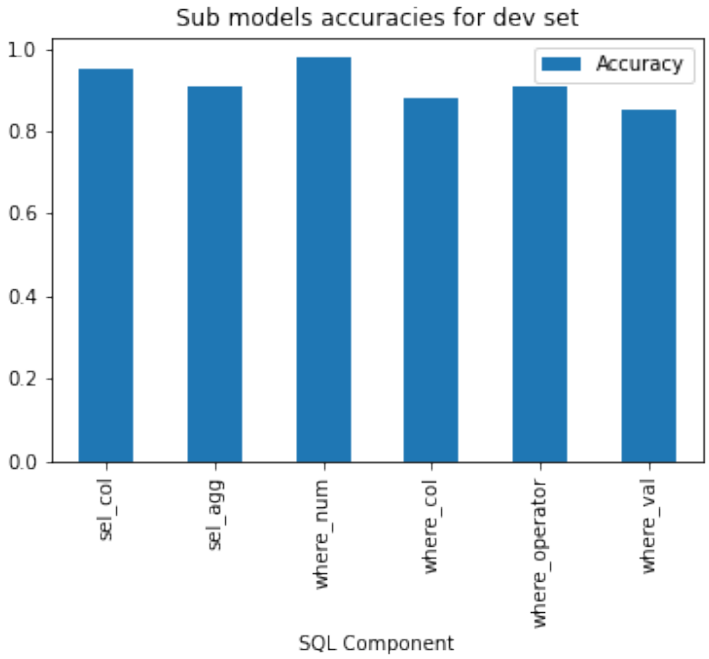

Figure 5: Dev Set accuracy for each sub models

From the table 6, we can see the logical form accuracy and execution accuracy for both dev and test sets. The previous models does not use any pretrained model to encode the input data to generate embeddings. They are mostly direct sequence to sequence translation. The recent models used BERT as a pretrained model and also made use of the contents of the table along with the table schema. None of the models performed data augmentation by using paraphrasing. For the model trained in this research, we achieved a logical form accuracy of 68.8% and an execution accuracy of 76.6%. This shows that our model shows improved performance when compared to the models without BERT encoder. The reason for the decreased accuracy in comparison with the BERT enabled models will be analysed in the discussion section.

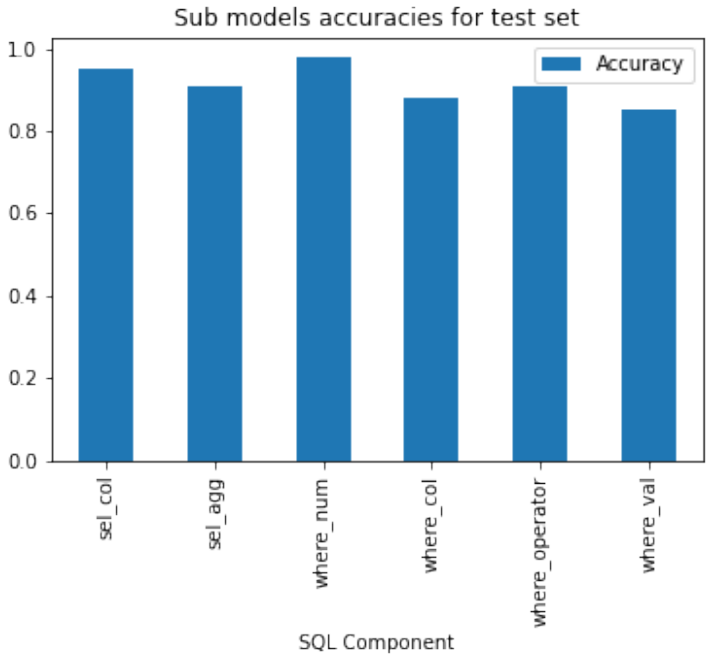

Figure 6: Test Set accuracy for each sub models

The figures 5 and 6 shows the logical form accuracies for each sub modules such as select-column, select-aggregate, where-number, where-column, where-operator and where-value. The accuracies for dev and test set are almost similar. In both the sets, the sub modules predicting select column and number of where conditions showed better performance. Although the individual model accuracies are higher, the overall logical form accuracy is only 68.8%. This is because, the logical form accuracies for individual modules are calculated without forming the whole query where as the final accuracy is the accuracy of the combined query. For example, if the select column for a query is predicted wrong, then the whole query is considered as wrongly predicted despite the fact that other components such as where and aggregate are predicted correctly.

6.1 Experiments

To experiment and analyse the model behaviour when presented with an unseen database not present in train, dev or the test set, a sample SQLite database with single student table is used. This table consists of few records of students details such as marks in different subjects, their gender and the birth date. A set of manually prepared natural language questions are prepared and passed as input to the model along with the table schema. It was observed that the model performed well most of the times, but failed to predict the select column correctly in some cases. The model was able to identify differences in the language and matched the question keywords with the correct columns. For example, when submitted with a question like "Provide names of all the pupils" the model correctly identified "pupil" denotes "student" and outputted the correct query.

6.2 Discussion

In this research, the augmented WikiSQL dataset is used to train the model. The paraphrasing of the dataset surely helped the model identify the linguistic variations better as discussed in the previous section. The usage of RoBERTa as a pretrained model also showed better performance than the baseline models. But our model achieved lower accuracy than the content-aware models introduced by Guo and Gao (2019) and Hwang et al. (2019). Providing the table contents along with the table schema surely helped these models learn better patterns. This decrease in accuracy is acceptable considering the risk of data privacy of highly sensitive data. When dealing with databases containing patient and bank information, privacy and confidentiality should be given more importance. With respect to the individual submodels performance, the Bi-LSTMs achieved higher accuracy in predicting the individual components. The logical form accuracies for the all the individual predictions are in the range of 88% to 99% which is really good.

In terms of the limitations, the WikiSQL dataset used in this study does not contain more complex queries involving joins and subqueries, which makes our model not capable of generating such queries. And our model is only capable of generating queries with single columns only due to the limitation of the dataset. By training the model with a complex dataset like Spider Yu, Zhang, Yang, Yasunaga, Wang, Li, Ma, Li, Yao, Roman et al. (2018) will improve its performance. This was not done as part of this research due to the time and computational limitations.

Submodule	Dev (%)	Test (%)
Select-column	95.21	95.00
Select-aggregate	90.54	90.61
Where-column	89.25	88.19
Where-number	98.24	97.66
Where-operator	91.55	90.79
Where-value	85.70	85.09

Table 2: Logical form accuracy for each submodule

7 Conclusion and Future Work

The research focused on building a neural parser that converts natural language question to SQL query by training using WikiSQL dataset. The primary goal was to find out the effect of synonym based data augmentation in the robustness of the model and the enhanced RoBERTa model in improving the performance. In order to achieve this, the architecture of the previously existing models was modified to perform the input embeddings that are fed to the Bi-LSTMs. These Bi-LSTMs are responsible for predicting the SQL query by taking the NL question, RoBERTa embeddings and the knowledge vectors as inputs. Then it fills the slots of the SQL by predicting output for the different SQL clauses. The model was trained with the WikiSQL dataset and evaluated using the logical form and execution accuracies. The model performed well with variations in linguistic

patterns due to the when experimented with sample databases. The model was able to surpass the performance of most of the previous models that failed to use the table schema information. But, ignoring the table contents considering privacy have decreased the performance of the model slightly.

In the future, the robustness of the model can be further improved by performing more rigorous paraphrasing and thus increasing the data size. As part of this research, only word-level paraphrasing has been done by changing synonyms using simple word replacement. Using a encoder-decoder model like BERT or RoBERTa for doing sentence-level paraphrasing can be done to observe the change in performance. Also, using a multi-domain dataset like Spider to train the model will help the model generate more complex queries.

References

Baik, C., Jagadish, H. V. and Li, Y. (2019). Bridging the semantic gap with sql query logs in natural language interfaces to databases, *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, IEEE, pp. 374–385.

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

Choi, D., Shin, M. C., Kim, E. and Shin, D. R. (2020). Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases, *arXiv preprint arXiv:2004.03125* .

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:1810.04805* .

Guo, T. and Gao, H. (2019). Content enhanced bert-based text-to-sql generation, *arXiv preprint arXiv:1910.07179* .

Herzig, J., Nowak, P. K., Müller, T., Piccinno, F. and Eisenschlos, J. M. (2020). Tapas: Weakly supervised table parsing via pre-training, *arXiv preprint arXiv:2004.02349* .

Hwang, W., Yim, J., Park, S. and Seo, M. (2019). A comprehensive exploration on wikisql with table-aware word contextualization, *arXiv preprint arXiv:1902.01069* .

Li, F. and Jagadish, H. V. (2014). Nalir: an interactive natural language interface for querying relational databases, *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 709–712.

Li, N., Keller, B., Butler, M. and Cer, D. (2020). Seqgensql–a robust sequence generation model for structured query language, *arXiv preprint arXiv:2011.03836* .

Li, Q., Li, L., Li, Q. and Zhong, J. (2019). A comprehensive exploration on spider with fuzzy decision text-to-sql model, *IEEE Transactions on Industrial Informatics* **16**(4): 2542–2550.

Lin, X. V., Socher, R. and Xiong, C. (2020). Bridging textual and tabular data for cross-domain text-to-sql semantic parsing, *arXiv preprint arXiv:2012.12627* .

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach, *arXiv preprint arXiv:1907.11692* .

Pal, D., Sharma, H. and Chaudhuri, K. (2021). Data agnostic roberta-based natural language to sql query generation, *2021 6th International Conference for Convergence in Technology (I2CT)*, IEEE, pp. 1–5.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, *arXiv preprint arXiv:1910.01108* .

See, A., Liu, P. J. and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Vancouver, Canada, pp. 1073–1083.
URL: *https://www.aclweb.org/anthology/P17-1099*

Sen, J., Lei, C., Quamar, A., Özcan, F., Efthymiou, V., Dalmia, A., Stager, G., Mittal, A., Saha, D. and Sankaranarayanan, K. (2020). Athena++ natural language querying for complex nested sql queries, *Proceedings of the VLDB Endowment* **13**(12): 2747–2759.

Song, M., Zhan, Z. and Haihong, E. (2019). Hierarchical schema representation for text-to-sql parsing with decomposing decoding, *IEEE Access* **7**: 103706–103715.

Usta, A., Karakayali, A. and Ulusoy, Ö. (2021). Dbtagger: Multi-task learning for keyword mapping in nlidbs using bi-directional recurrent neural networks, *arXiv preprint arXiv:2101.04226* .

Wang, B., Shin, R., Liu, X., Polozov, O. and Richardson, M. (2019). Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers, *arXiv preprint arXiv:1911.04942* .

Weir, N., Utama, P., Galakatos, A., Crotty, A., Ilkhechi, A., Ramaswamy, S., Bhushan, R., Geisler, N., Hättasch, B., Eger, S. et al. (2020). Dbpal: A fully pluggable nl2sql training pipeline, *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 2347–2361.

Yin, P., Neubig, G., Yih, W.-t. and Riedel, S. (2020). Tabert: Pretraining for joint understanding of textual and tabular data, *arXiv preprint arXiv:2005.08314* .

Yu, T., Li, Z., Zhang, Z., Zhang, R. and Radev, D. (2018). Typesql: Knowledge-based type-aware neural text-to-sql generation, *arXiv preprint arXiv:1804.09769* .

Yu, T., Wu, C.-S., Lin, X. V., Wang, B., Tan, Y. C., Yang, X., Radev, D., Socher, R. and Xiong, C. (2020). Grappa: Grammar-augmented pre-training for table semantic parsing, *arXiv preprint arXiv:2009.13845* .

Yu, T., Yasunaga, M., Yang, K., Zhang, R., Wang, D., Li, Z. and Radev, D. (2018). Syntaxsqlnet: Syntax tree networks for complex and cross-domaintext-to-sql task, *arXiv preprint arXiv:1810.05237* .

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S. et al. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task, *arXiv preprint arXiv:1809.08887* .

Zhong, V., Xiong, C. and Socher, R. (2017). Seq2sql: Generating structured queries from natural language using reinforcement learning, *arXiv preprint arXiv:1709.00103* .