Natural Language to SQL query

Sai Pranith Reddy Thirumala saipranithred.thirumala @stonybrook.edu
Shiva Sankeerth Yarradla shivasankeerth.yarradla @stonybrook.edu
Sai Bhavana Ambati saibhavana.ambati @stonybrook.edu

Abstract

The idea of this problem is to generate SQL queries given the natural language questions. The natural language sentence is usually in the form of a question, which is then converted to an SQL query and then executed on the database to give a result. The aim of this project is to to perform a comparative study of some existing approaches. The first question we are trying to answer is : the natural language queries are not very lengthy, so does employing GRU models instead of the existing models improve performance is terms of time, accuracy and F1 scores. The other questions are : Does employing sophisticated models like BERT, at the embedding layer and/or at the encoding layer improve the accuracy significantly ? BERT based models take significantly large time to train, which is not ideal when the schema is changing regularly. Does adding domain information enhance performance ?

1 Introduction

The challenge of translating natural language to SQL queries is gaining a lot of attention over the past few years. It is an exciting and important problem to solve as relational databases are being used in many real-world applications such as the financial, sales, and healthcare industries. Solving this problem can enable non-technical users in interacting with the databases with complex designs or schemas for the storage of information. The scope of this project is to generate SQL queries where only one table is referenced and only one column is selected. This problem is solved by adopting a template-based strategy. This is done by leveraging the structure of SQL queries. SQL queries contain fields like SELECT-column, SELECT-aggregation, WHERE-number, WHERE-column, WHERE operator, WHERE-value. The model thus divides the task into many sub-problems: the sub-problems being, the classification of the above categories. Also, the order of the WHERE conditions does not matter. Once the classification for each field is performed, the SQL query can be generated by just filling the slots in the following format :

SELECT aggregation SELECT-column
WHERE (column, operator, value) (number)
(conditions are separated by "AND")

Based on the existing research, we have come up with the following questions that we believe need to be addressed.

a) We observed that the natural language queries are quite brief and short, so can models simpler then LSTM be used ? b) Since users usually know the table name they need to look at, as it is evident in table names, so does incorporating the table name information as input enhance performance ? c) Is there a difference in performance when the 3 categories are predicted sequentially (giving the output of one category to the next) and when the categories are predicted independently. d) BERT based models take significantly large time to train, which is not ideal when the schema is changing regularly. Does employing BERT, at the embedding layer and/or at the encoding layer improve the accuracy significantly ?

The main algorithm to predict the queries uses classifier based models to predict the select column and the aggregation column. Then the where category is predicted by first predicting the number of conditions, then the columns. and operators present in each condition. We referred to 4 main models : a) SQLNet : Predicts the 3 categories sequentially and uses bi-directional LSTM for classification . Uses a sketch-based approach i.e., sketch contains a dependency graph so that one prediction can be done by taking into consideration only the previous predictions that it depends on. It uses glove embeddings and does not predict the order of the WHERE category. b) HydraNet : Model by Microsoft that uses Roberta as the first layer. c) Content Enhanced BERT-based Text-to-SQL Generation : The 6 categories in order, ensuring that each category predicted is added to the input passed to the classifier to predict the next category. It take the header of the table as input to predict the first category.

To explore if comparable results can be reached in lesser time, we use models with varying levels of sophistication. For this we used the architecture of SQLNet for classification of the categories. We modified the embedding layer of SQLNet to use BERT embeddings, then we used GRUs with the same hyper-parameters as the LSTM model. Then we added a pre-trained BERT model before the LSTM layer to increase the complexity. To observe the difference in performance when the query is predicted sequentially as opposed to independently, we modified The content-enhanced model (The ontology based model) to use GRUs. The models have been evaluated in terms of the time it takes to train, accuracies and the F1 scores. We used the WikiSQL dataset as it is one of the most comprehensive datasets which is needed for complex models.

Based on the experiments, we found that

1. The LSTMs are roughly 5.5 percent more accurate than GRUs for the models with SQLNet architectures and Ontology based architectures.

2. When BERT was added at the embedding layer, it increased the accuracy of SQLNet by roughly 3 per cent.

3. When pre-trained BERT was added to classifying the categories independently, the accuracy increased by an additional 3 per cent as compared to the LSTM model.

4. The performance in terms of F1 scores was consistent with accuracies.

5. GRUs took 25 per cent less time then LSTMs with the hyperparameters, but took more time than LSTMs to achieve the same accuracies.

2 Models

2.1 Baseline Model

The baseline model predicts the categories independently. The architecture is shown in the diagram.

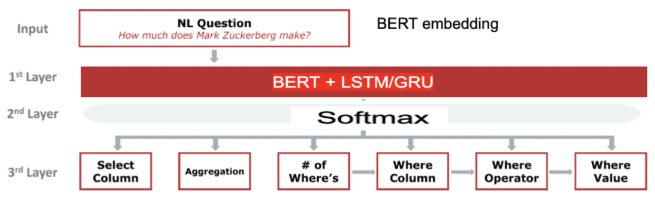


Figure 1: Baseline Architecture

In the baseline model, categories are predicted separately. The input is encoded using BERT embeddings. There are 3 variations to this model. The first one does not uses only bidirectional LSTM in the first layer, in the second variation, it uses GRUs and in the 3rd, the pretrained BERT model and ROBERTa model are used, followed by a bi-directional LSTM.

These are the hyperparameters :

2.2 SQLNet Model Variations

The following is the architecture for SQLNet :

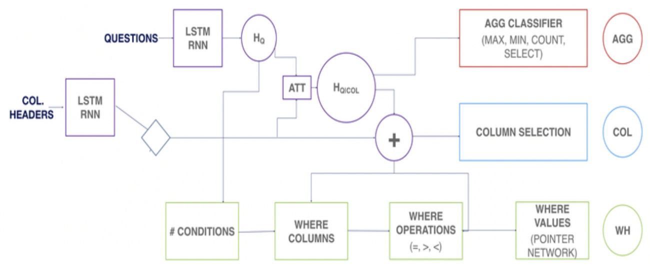


Figure 2: SQLNet Architecture

The SQLNet model uses column attention to enhance the performance . The categories are predicted sequentially, and for prediction of every category, different categories already predicted are used using an attention mechanism. It predicts the aggregator column first which does not use any other category information. It leverages the fact that the order of the conditions does not matter and hence, first predicts the number of conditions, then the list of columns, followed by the operators and then eventually the value for the column.

We made 2 variations to this model : replacing the Glove embeddings to use BERT embeddings, and replacing LSTMS with GRUs.

2.3 Ontology Based Model Variations

The following diagram shows the architecture of the model :

This model leverages the observations that some of the table content and headers are already

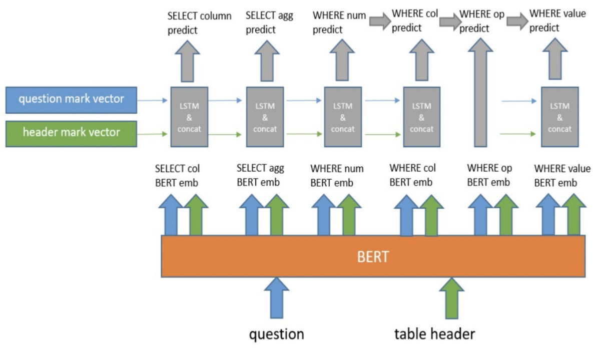


Figure 3: Content Enhanced BERT-based Text-to-SQL Generation

present in the input natural language query. These additional feature vectors are encoded and passed to the model. The feature vector is constructed by matching information of the table content and the natural language query, making sure that the length of the query and the question vector are of the same size. The addition of this vector improves the accuracy of the WHERE category variables as the table header and value go hand in hand. injects the knowledge that the answer cell and its corresponding table header are bound together. Hence locating one is tied to the other.

The advantage of using the table headers as input for the model is practical in the real world applications, but incorporating table values would not be practical as databases are huge.

This model was modified to use GRUs instead of LSTMs and also uses the same hyperparameters as the previous models.

3 Evaluation

State the purpose of your evaluation. How should one evaluate a system for your task. What are the questions to ask?

3.1 Dataset Details

For this task we used the WikiSQL Dataset : The dataset of 87,726 pairs of SQL queries and natural language questions. 61,297 samples, 9,145 development samples, and 17,284 test samples. Even though the scope of the dataset is very constrained, it is still very challenging because the tables and questions are very diverse containing 24K different sets of tables. We chose this dataset because it is an order of magnitude larger than semantic parsing datasets, which is needed for neural networks, and all tables belong to the same database. Also, the natural language questions are created by human beings (employing crowd-sourcing on Amazon Mechanical Turk).

The following image shows a sample entry in the database :

```
{
   "phase":1,
   "question":"who is the manufacturer for the order year 1998?",
   "sql":{
      "conds":[
         [
            0,
            0,
            "1998"
         ]
      ],
      "sel":1,
      "agg":0
   },
   "table_id":"1-10007452-3"
}
```

Figure 4: Data sample

3.2 Evaluation Measures

3.3 Baselines

We used the following metrics to evaluate the models :

1. the classification accuracy for each category

2. F1 score for the classifications

3. the time taken to train the model to achieve best results.

4. Query-match accuracy: The order of the conditions does not matter, for instance, ‘WHERE sports = ’Baseball’ AND city = ’New York” is the same as ‘city = ’New York’ and WHERE sports = ’Baseball’ ’.Since the order of the conditions do not matter, we extract the 6 categories from the ground truth and then match for each category.

5. Execution accuracy: Since there are multiply possible queries for the same operation, the generated query is executed along with the ground truth query and then the results of each query are compared.

The following are the hyperparameters that were used to train the models

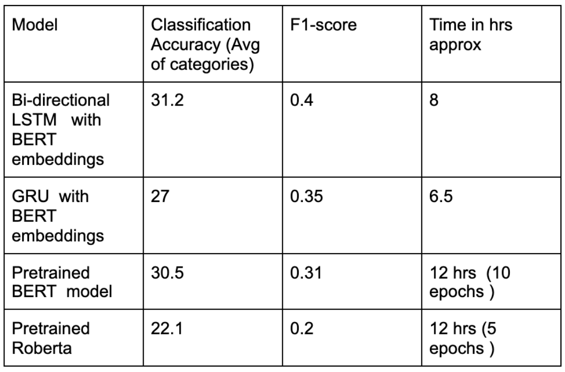
1. Learning rate: We used 103 for all the 3 baseline models, 103, for the ontology based model and 104, for SQLNet

2. All the models were trained for 100 epochs except the BERT based baseline model which was trained for 10- epochs,
3. Batch Size of 64 was used for all models
4. The ADAM optimizer was used for all the models.
5. Cross Entropy loss was used to train all the models

The batch size and the epochs were changed for the models as they have the most significant impact on the performances

3.4 Results

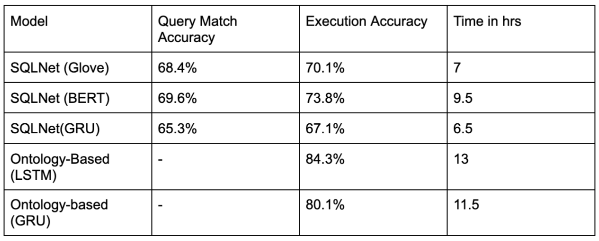
The following table shows the results for baseline models:



Model	Classification Accuracy (Avg of categories)	F1-score	Time in hrs approx
Bi-directional LSTM with BERT embeddings	31.2	0.4	8
GRU with BERT embeddings	27	0.35	6.5
Pretrained BERT model	30.5	0.31	12 hrs (10 epochs)
Pretrained Roberta	22.1	0.2	12 hrs (5 epochs)

Figure 5: Baseline results

The following table shows the results for the state of the art models



Model	Query Match Accuracy	Execution Accuracy	Time in hrs
SQLNet (Glove)	68.4%	70.1%	7
SQLNet (BERT)	69.6%	73.8%	9.5
SQLNet(GRU)	65.3%	67.1%	6.5
Ontology-Based (LSTM)	-	84.3%	13
Ontology-based (GRU)	-	80.1%	11.5

Figure 6: State of the art model variations

3.5 Analysis

1. Modifying the SQLNet model to use BERT word embedding improves the performance of the SQLNet model by 3 per cent in query-match accuracy and 2.5 in execution-match accuracy
2. The LSTMs are roughly 5.5 percent more accurate than GRUs (as GRUs use lesser gates) for the models with SQLNet architectures and Ontology based architectures.
3. When pre-trained BERT was added to classifying the categories independently, the accuracy increased by an additional 3 per cent as compared to the LSTM model.
4. The performance in terms of F1 scores was consistent with accuracies.
5. GRUs took 20-25 per cent less time then LSTMs with the hyperparameters, but took more time than LSTMs to achieve the same accuracies.
6. The average execution accuracy is higher than query-match accuracy by 6.5 per cent approximately.
7. The baseline model does not perform well as it does not perform semantic parsing to the question. It should learn to capture the structures of SQL queries itself
8. The accuracy of the where column is significantly lower than that of the other categories. This is because the process for generating the ’which’ category is more complicated.

3.6 Code

Google drive link for the code for this project : https://drive.google.com/drive/folders/1DWTaUN4ZSrBLLpZ8yv7NL0cAMpZi_oEg?usp=sharing

4 Conclusions

To solve the problem of converting natural language to sql query, we first modified the the SQL-Net model to use BERT embeddings to see that it improves performance. Then we tried to see if comparable accuracies could be achieved using GRUs. Our results show that GRU models have slightly lower accuracies then LSTMs for lesser time. We then tried to implement a model that uses pre-trained BERT model and bi-directional LSTM. The categories were predicted independently, so the model gives low accuracy, but gave better accuracies than select and where column. We can expect better results by further tuning parameters.

5 References

1. SQLNet : https://arxiv.org/abs/1711.04436
2. Seq2Sql : https://arxiv.org/abs/1709.00103
3. Hybrid Ranking Network for Text-to-SQL: https://www.microsoft.com/en-us/research/uploads/prod/2020/03/HydraNet_20200311-5e69612887fcb.pdf
4. Content Enhanced BERT-based Text-to-SQL Generation: https://arxiv.org/abs/1910.07179
5. Table2answer: Read the database and answer without SQL, https://arxiv.org/pdf/1902.04260.pdf
6. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. arXiv preprint arXiv:1809.08887, 2018.
7. Syntax tree networks for complex and cross-domaintext-to-sql task. arXiv preprint arXiv:1810.05237, 2018.
8. Attention is all you need. In Advances in Neural Information Processing Systems, pages 5998–6008, 2017.
9. v. Typesql: Knowledge-based type-aware neural text-to-sql generation. arXiv preprint arXiv:1804.09769, 2018.