Text to SQL Model

Vineesh
CSE
IIIT Sri City
Andhra Pradesh, India
vineesh.s17@iiits.in

Lushaank
CSE
IIIT Sri City
Andhra Pradesh, India
lushaank.k17@iiits.in

Phani
CSE
IIIT Sri City
Andhra Pradesh, India
Sriphanisainath.k17@iiits.in

Shiva Sai Charan Ruthala
CSE
IIIT Sri City
Andhra Pradesh, India
shivasaicharan.r17@iiits.in

***Abstract*—Text to SQL is a system that converts natural language statements to SQL queries that help in retrieving information stored in a databases by expressing commands in natural language. There are various domains like healthcare, customer support and search engines, which requires elaborating of structured data having information on the text where developing a systems for each use case is a tedious process so this is one thing where automation with generalisation has a huge potential over human effort in providing robust SQL synthesize systems.Our main focus is to leverage this systems to bridge the gap between non-technical users and database systems, as users do not require to understand the database schemes and query language syntax.**

1. Introduction

Semantic parsing is the tasks of translating natural language to logic form. Mapping from natural language to SQL (NL2SQL) is an important semantic parsing task for question answering system. Text to SQL can help in retrieving the information stored in database by expressing commands in natural language. In recent years, deep learning and BERT-based model have shown significant improvement on this task. It offers applications in various domains like healthcare, customer support and search engines, which requires elaborating of structured data having information on the text. This proves to be useful as it bridges the gap between non-technical users and database systems, as users do not require to understand the database schemes and query language syntax. SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data, i.e., data incorporating relations among entities and variables. SQL is one of the most used languages for querying the database of the retrieval of data. With the help of machine learning and knowledge-based resources, text language to SQL conversion is facilitated. Various other methods like rule-based syntax analysis, semantic matching, and pattern matching are included in the NoSQL category. However, we limit our study in this series on the conversion of natural language to SQL only. Bidirectional Encoder Representations from Transformers (BERT) is a Transformer-based machine learning technique for natural language processing (NLP) pre-training developed by Google. BERT was created and published in 2018 by Jacob Devlin and his colleagues from Google. As of 2019, Google has been leveraging BERT to better understand user searches. BERT is a deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus. Context-free models such as word2vec or GloVe generate a single word embedding representation for each word in the vocabulary, where BERT takes into account the context for each occurrence of a given word. For instance, whereas the vector for ”running” will have the same word2vec vector representation for both of its occurrences in the sentences ”He is running a company” and ”He is running a marathon”, BERT will provide a contextualized embedding that will be different according to the sentence. Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feed forward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems). LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

1.1. Motivation

Our main goal of the model is to use Google Programs to improve our models in better recognising the texts and try to convert them into a queried language namely, SQL. We wanted to implement the Text to SQL model which could be helpful in creating a platform for programmers for better understanding of the language. While looking from the user point of view, we could say that our model would be helpful for beginners who have just started to learn SQL and could also be useful for the experts in reducing the time in which they needed to code. Then, we were while searching for models which could be useful in Text to SQL, we could find several models that shows promising. But we finally came upon the BERT Model developed and published by Google, could be helpful in building our model because it uses Transfer Learning, which could be very useful in identifying the patterns and similarities in SQL. Along with these models, BERT is built on RNNs and LSTMs, which makes training the model easier than other models we have seen and could be useful in building our model.

2. State of the art/Background

2.1. SQLNet

[1]The model was developed with an aim to show that the use of reinforcement learning in Text2SQL tasks should be limited.Before SQLNet, all the models in the literature employed reinforcement learning to improve the decoder results when it generated any of the appropriate serializations.SQLNet overcomes the limitation by avoiding the seq2seq structure in cases when order is irrelevant.The model uses a sketch-based approach in which a sketch consists of a dependency graph so that previous predictions can be taken into consideration for making the predictions.incorporates column attention (attention refers to weights assigned to important words and phrases in sentences) mechanism for improving the results. The idea of sequence-to-set prediction includes the prediction of column names of interest instead of generating a sequence of column names.Column attention is a part of generic attention mechanism for computing the feature attention map on a question based on the column names.The most difficult part is pridicting WHERE clause in Text2SQL task.SQLNet finds the columns that appear in WHERE clause and for each chosen column it finds the constraints by predictions the slots of OP and value as shown in SQL sketch.OP slot is filled by one of the 3 classes:¡,¿,= and VALUE slot is predicted based on the sub string from the natural language question.SELECT clause consists of a column name and an aggregate function like count, sum, max, etc. The only difference between SELECT and WHERE is the column name selection. In SELECT, only one column is chosen.Accuracy of SQLNet on WikiSQL test set is 64.4

2.2. IRNet

[2]A neural approach called Intermediate Representation(IRNet) for complex and cross-domain Text-to-SQL aims to address 2 main challenges in Text2SQL tasks.The challenges include the mismatch between expressed intents in natural language and the challenge in predicting columns caused by a greater number of out-of-domain words.IRNet decomposes natural language into three phases instead of end-to-end synthesizing of SQL Query.During the first phase, the schema linking process over a database schema and a question is performed. IRNet employs SemQL domain-specific language, which serves as an intermediate representation between SQL and natural language.This model includes Natural Language (NL) encoder, Schema Encoder and Decoder.Every component of the model provides a different function for accomplishing Text2SQL task. NL encoder takes the input of natural language and encodes it into an embedding vector. These embedding vectors are there used to construct hidden states using a bi-directional LSTM. Schema encoder takes database schema as input and outputs representations for columns and tables.Finally, the decoder is used to synthesize SemQL queries using a context-free grammar. IRNet yields a significant improvement of 19.5percent over previous benchmark models and achieves 46.7percent accuracy on SPIDER dataset.Incorporating IRNet with BERT shows a substantial improvement in performance and yields 54.7 percent accuracy.

3. Dataset

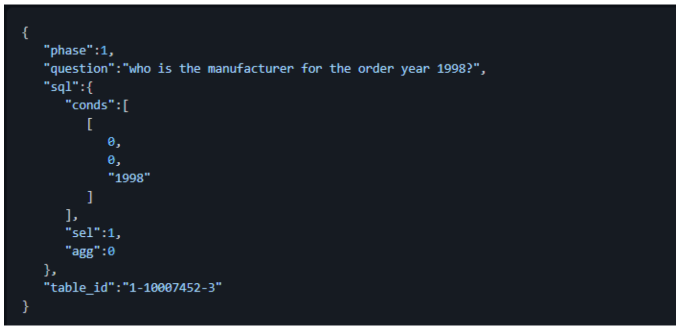
Along with other natural language processing tasks, Text2SQL highly relies on the type of dataset used. Different datasets with different structures, lengths, and queries have been created. There are a total of 9 datasets in the field of semantic parsing out of wikisql is one of the benchmark with lot of research done on it.

3.1. WIKISQL

WikiSQL is a massive semantic parsing dataset consisting of 80K+ natural language questions and respective SQL based queries on 24K+ tables extracted from Wikipedia.The database in the test set does not appear in the train or development sets its purely to know how our model is trained.This dataset curated with simplified assumptions about the SQL queries and databases. Thus, the dataset consists of SQL labels that only cover a single SELECT column and aggregation, and WHERE conditions. Moreover, all the databases only contain single tables and complex queries consisting of advanced operations like JOIN, GROUP BY, and ORDER BY, etc. are not included.

4. prepossessing

Data reprocessing is an essential step in building a Deep Learning model and depending on how well the data has been prepossessed; the results are seen. In NLP, text prepossessing is the first step in the process of buildings model. The various text prepossessing steps are: Handling stop words and synonyms, Tokenization, Stemming or Lemmatization.



```
{
    "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 1. Dataset

4.1. Handling stop words

Stop words refer to the most common words present in a language. Depending on the task, any group of words can be chosen as stop words and remove those from the question tokens.

4.2. Tokenization

Tokenization is used to split an input question into a varied list of tokens. It can be as simple as a separator on space but in most cases, it is based on multiple rules such as regex(regular expressions).It is important to perform tokenization during the prepossessing state as it will reduce the mismatches during the conversion.

4.3. Stemming

Stemming reduces correlated words to the same token by removing different endings of the words. Whereas, Lemmatization removes inflectional endings and gives a word corresponding to a base form of the word.One of the foundational algorithms for the stemming task is Porter's algorithm. It is based on simple heuristics that are applied to the longest suffix.However, there is a disadvantage of this technique. It is because it chops of words like "director" to "direct" which is erroneous.

5. Proposed System

[3]Our model goal is to predict six SQL components SELECT column, AGGREGATION(None, COUNT, SUM, AVG, MIN, MAX), Number of WHERE conditions(0, 1, 2, 3), WHERE column, WHERE OPERATORS(=, ¡, ¿), WHERE VALUE(a textspan).The typical architecture of our Text-to-SQL model has three layers with prepressed data is fed to the first layer.

5.1. Input and Feature Vector Encoding

We design the architecture so that it accepts the concatenation of both question and columns as an input. Some models take type embedding or positional embedding as an input as well. Column names can consist of several words. Here we used alredy esisting algorithms to encode knowledge from a structured data. The baseline of these alogoriths is to match words of table content and question tokens also table header and question tokens. Refer Figure 1 and Figure 2

Algorithm 1 The construction for question mark vector

```
vector = [0]*question_length
for cell in table do
    if contains(question,cell) then
        start_index = get_index(question, cell)
        vector[start_index:start_index+len(cell)] = 2
        vector[start_index] = 1
        vector[start_index+len(cell)] = 3
        break
    end if
end for
for one in header do
    if contains(question,one) then
        index = get_index(question, one)
        vector[index] = 4
    end if
end for
```

Figure 2. Embedding Process

Algorithm 2 The construction for table header mark vector

```
vector = [0]*header_length
index = 0
for one in header do
    if contains(question, one) then
        vector[index] = 1
    end if
    index = index + 1
end for
for cell in table do
    if contains(question,cell) then
        vector[get_column_index(table, cell)] = 2
    end if
end for
```

Figure 3. Embedding Process

5.2. First Layer

The input is fed to the first layer which is BERT to encode the input with the contextualized word representation, there are different options to choose for this layer like MT-DNN, ROBERTA etc. but we tryed with BERT. This layer helps to act as an attention for further tasks.

5.3. Second Layer

Question mark vector and header mark which is the output from BERT are concatenated and fed to the LSTMs to strengthen the encoder output. The goal of this step is to capture a larger context of an input. LSTM enables the model to access information about prior, current, and future input. The resulting embeddings don't merely capture the local context of a word but they can, in principle, capture the entire semantic of the input.

5.4. Third Layer

The final layer consists of classification modules over six different components of the final SQL as mentioned above. Each module has its own attention weight to align the hidden representation and the final label.

5.5. Training and Inference

During the training, we minimize the loss, a summation of six individual sub-task losses using cross-entropy.in this model inference is pretty straightforward. The highest-scoring labels will be returned from each module. If the highest-scoring column is [None], we ignore the output from the WHERE-number predictor and return zero WHERE condition.

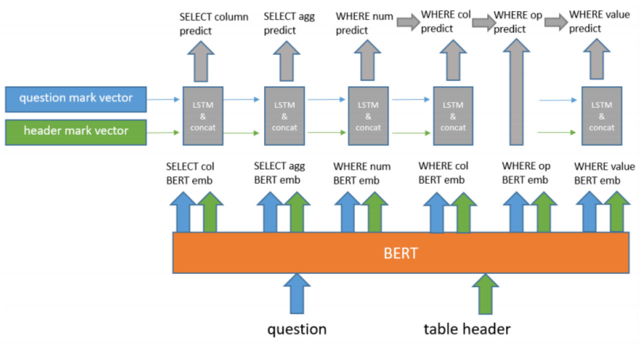


Figure 4. Embedding Process

5.6. SELECT column

We need to find the probability of select column given that question, table header, final question mark vector and table header mark vector that resulted from matching the columns according the above mentioned algorithms.

5.7. SELECT agg

we need to find the probability of select aggregate given the question and question mark vector.

5.8. WHERE number

We need to find the probability of where number given that question, table header, final question mark vector and table header mark vector that resulted from matching the columns according the above mentioned algorithms.

5.9. WHERE column

We need to find the probability of where number given that question, table header, final question mark vector and table header mark vector and where number probability.

5.10. WHERE op

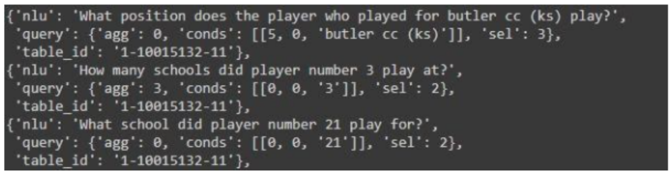
The occurrence of WHERE operator is given by the probability of WHERE operator given the question, table header, probability of WHERE column, probability of WHERE operator.

5.11. WHERE value

The occurrence of WHERE value is given by the probability of where value given the question, table header, question mark vector, table header mark vector, probability of WHERE number, probability of WHERE column, probability of WHERE operator.

6. Results

When a text question was given by the user, this model understands the query and then divides the new query according to our base model WikiSQL, and shows us the output. Each variable namely "agg", "conds", "sel" represents a unique set of rules to be followed in order to convert this kind of dictionary into an SQL query.



```
{'nlu': 'What position does the player who played for butler cc (ks) play?',
 'query': {'agg': 0, 'conds': [[5, 0, 'butler cc (ks)']], 'sel': 3},
 'table_id': '1-10015132-11'},
{'nlu': 'How many schools did player number 3 play at?',
 'query': {'agg': 3, 'conds': [[0, 0, '3']], 'sel': 2},
 'table_id': '1-10015132-11'},
{'nlu': 'What school did player number 21 play for?',
 'query': {'agg': 0, 'conds': [[0, 0, '21']], 'sel': 2},
 'table_id': '1-10015132-11'},
```

Figure 5. Results

7. Conclusion & Future Work

For a given text command, the model is able to correctly able to predict the appropriate SQL query, that could be able to distinguish different types of the query command. These types of models could be useful for creating appropriate queries in a short span of time when compared to average human, which might be the case of forgetting an important word in the query or not knowing a certain keyword that might shorten the code and many such cases could be avoided when using these types of models. It works fairly good for a model predicting SQL from the text but the training of the model to correctly fit the required setup might take a lot of time.

Discuss about [**?**] your system what you have developed, how it is better than the current state of the art [**?**] work etc... Also you can discuss about the possible future work plan in this area.

References

[1] D. S. Xiaojun Xu, Chang Liu, *SQLNet: GENERATING STRUCTURED QUERIES FROM NATURAL LANGUAGE WITHOUT REINFORCEMENT LEARNING*. arXiv:1711.04436v1, 13 Nov 2017.

[2] Y. G. Y. X. J.-G. L. T. L. D. Z. Jiaqi Guo, Zecheng Zhan, *Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation*. arXiv:1905.08205v2, 29 May 2019.

[3] H. G. Tong Guo, *Content Enhanced BERT-based Text-to-SQL Generation*. rXiv:Huilin Gao, 2020.