Automated Text to SQL Generation

April 28, 2022

Registration number:	2110376
Project:	Seq2seq translation
Link to GitHub:	https://github.com/panks11/CE888/tree/submit/assignment2/seq2seq

Executive summary (max. 250 words)	226
Introduction (max. 600 words)	620
Data (max. 300 words/dataset)	320
Methodology (max. 600 words)	559
Results and Discussion (max. 1000 words combined)	631
Conclusions (max. 500 words)	258
Total word count	2614

Contents

Abstract

Data has always played an important role in every organization and business sector. With advancement of technology, the data being stored and managed by the organizations have increased exponentially. This has given rise to various database management systems which secure and backup the data. The major business activities like recommendations, forecasting and decisions making require data extraction. This implies frequent use of SQL to interact with a database. However, writing SQL queries require technical expertise and understanding of database schemas. Due to the increased demand of usage of SQL, the automated Translation of a spoken question to a SQL query has been a popular research task. Due to the complexity involved in NL2SQL Translation, machine learning techniques are being applied. Deep learning using LSTM Layers, Seq2SQL [20], SQLNet [16] are few of the approaches. In this project, we have built Sequence to Sequence models with Encoder-Decoder RNN layers and Attention layer to capture the context of the sentence and translate them to SQL queries. We have evaluated the accuracy for these models and observed the Sequence to Sequence approach translates the English sentences with moderate accuracy and can be improved to achieve higher accuracy. Our experiments show that the syntactical information of the SQL query can be helpful in text to SQL translation. Furthermore predictors for Aggregator, Column Names and Condition values using classification could achieve better results.

1 Introduction

Automated Text to SQL Generation is an automated task which targets to convert Natural Questions into Structured Query Language. These queries are capable to retrieve data stored in Relational Database Management systems easily and efficiently. For several years, Relational Database management systems have been helping in storing and managing huge amount of data in various sectors and organizations. Retrieving and analysing the stored data to make better business decisions or recognizing emerging trends has gained huge importance recently. However, writing SQL statements can be tricky as it requires a background knowledge about the SQL syntax and database schemas. Additionally, development of diverse and sophisticated data storage systems to handle gigantic amount of data increase the complexity of translations.Natural Language to SQL conversion can play a vital role by bridging this technical gap between humans and machines.[6]. The Natural Language to SQL conversion frameworks are being developed and improved since 1970, with systems like LUNAR and LADDER used by non-technical users to pose questions about Moon rock samples and US naval ships respectively[15].Several solutions have been proposed to face this challenging problem [1]. The different approaches and architectures applied in the field of Natural Language Interface to Database systems(NLIDB) include [13]:

1. Pattern Matching systems: Early NLIDB system were based on pattern matching to provide answers. The user input was mapped to a query using a set of rules. The approach was simple, however, specific to a database schema. ELIZA is an example of this approach.

2. Syntax Based systems: These systems parses user question into a syntax tree. The tree describe the syntactic structure of queries and lexicon of words that may appear in user queries. The approach gives information about sentence structure, words, grammatical functions, relationship of words and sentence grouping. This approach is also application specific and cannot be generalized. LUNAR is an example.[15]

3. Semantic Grammar systems: These systems is similar to Syntax based, as the questions are parsed to a tree, with grammar categories not necessarily corresponding to syntactic concepts. The objective is to simplify the tree by reducing the number of nodes. The NLIDB systems like Planes, Ladder, Rel used this approach [14].

4. Intermediate Representation Languages: These systems attempt to gain cross-domain applicability or database independence. The user queries are transformed from parse tree to intermediate logic query and then translated to a query. PRECISE is a successful implementation of such systems [12].It combines semantic and statistical approaches to achieve database independence. It tokenize and lemmatize a natural language question, and looks up in the lexicon the set of database elements it could potentially participate in. The statistical parser extracts parts-of-speech tags and attachment relationships between words, which are fed back into the tokenizer to create multi-word tokens. A graph matching algorithm is used to find the most appropriate matching of tokens to database elements. These matches are then used by the query generator to formulate the SQL query.

Recently, deep learning techniques, involving convolutional and recurrent neural networks (CNN and RNN), the encoder-decoder model and semantic analysis are being researched for Text2SQL Translations [8]. Seq2seq is one of the approaches to process the Text-To-SQL task. Seq2SQL encodes the text using an encoder to get the semantic representation of the input text and decodes it using a decoder for generating the SQL statement [20]. Recently proposed architectures have achieved more than 80 percent accuracy on the well-known Text-to-SQL benchmarks such as WikiSQL and Spider dataset. Photon is an example system, employing deep learning at its core element and based on BERT DB Schema [18]. With our experiments we will study the utilization of the Encoder-Decoder Recurrent Neural Network technique to generate SQL queries given a Natural Language text. We intend to explore the efficiency and improvements of basic Sequence to Sequence Deep Leaning models.

2 Data

To address the need of a competent dataset for task of Natural Language to SQL generation Spider dataset is considered.[17]

2.1 Dataset Collection

The Spider dataset is downloaded from the link. The dataset consist of *train*, *dev* JSON files containing the English Question and respective SQL Query pairs categorized under 'db_id' see Fig 1. The dataset also includes the database schema information for each department,however,it is not considered for our experiments. We have selected 8659 questions from train_spider.json and train_others.json. The Easy Queries with single Table are filtered for the translation resulting in dataset containing 3530 English-SQL pairs. A stratified split of 80:20 on 'db_id' is performed on the dataset to obtain Train and Test data respectively. We have used 2800 (approx) queries to train the Translation model and 700 (approx) Queries for Testing it see Fig 1 and Fig 2.

2.2 Data Preparation

For NL2SQL translation we have selected English sentences under column 'question' as Input and SQL statements under column 'query' as the Target column. After data loading we will perform following steps for data preparation in a pipeline. [2]. Refer figure 3 :

1. Text Cleaning : First, special characters which are not part of the SQL syntax are removed for example, 'dollar' and double slash. The white space is added around the punctuation marks and extra white space is removed from the Input and Target sentences.

2. Standardization : Second, the input and target sentences are converted into lower case and normalized using NFKD method. This step will help in reducing the vocabulary for our model.

3. Append Sentence Start and End Tags : Third, each input and target sentence is appended with '[start]' and '[end]' tags to mark the start(SOS) and end(EOS) of the sentence respectively.

4. Text Vectorization : Fourth, the input and target sentences are tokenized. A word index, mapping each word to id and reverse word index, mapping id to word is created. Two Text Vectorization layer is created to learn the vocabulary of English sentences and SQL.

5. Text Padding : We use Text padding to create the input and target vectors of length equal to the length of the longest sentence. It is achieved by adding Zeroes to either start or end of the vector.

6. Word Embedding : Sixth, after encoding our input and target words to single integer representation, we will create word embedding to capture relationship of words in n-dimensional vectors.

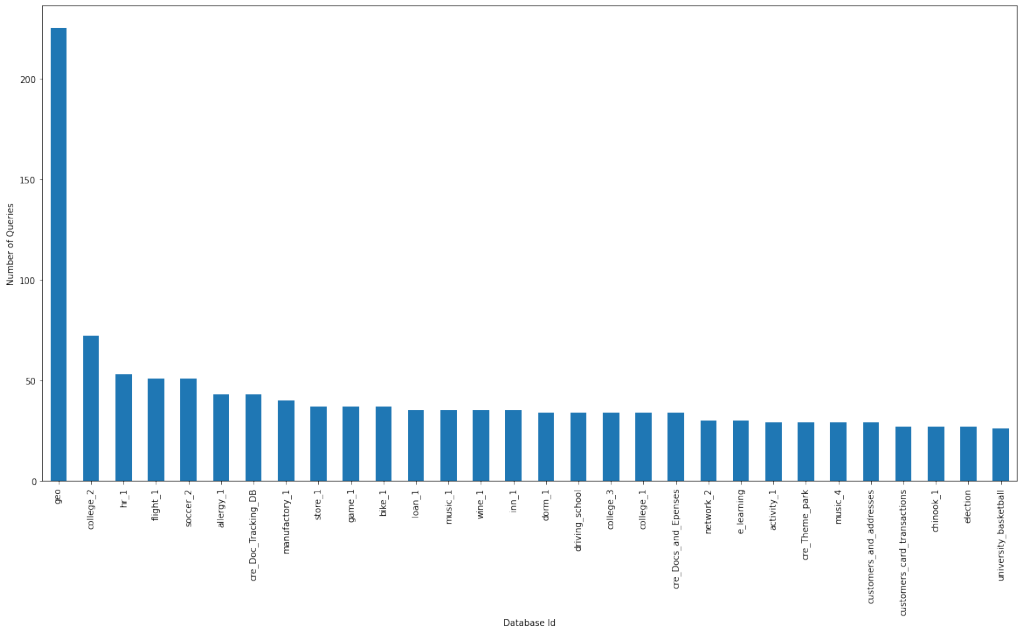

Figure 1: Training Set : Number of Queries vs Database Id

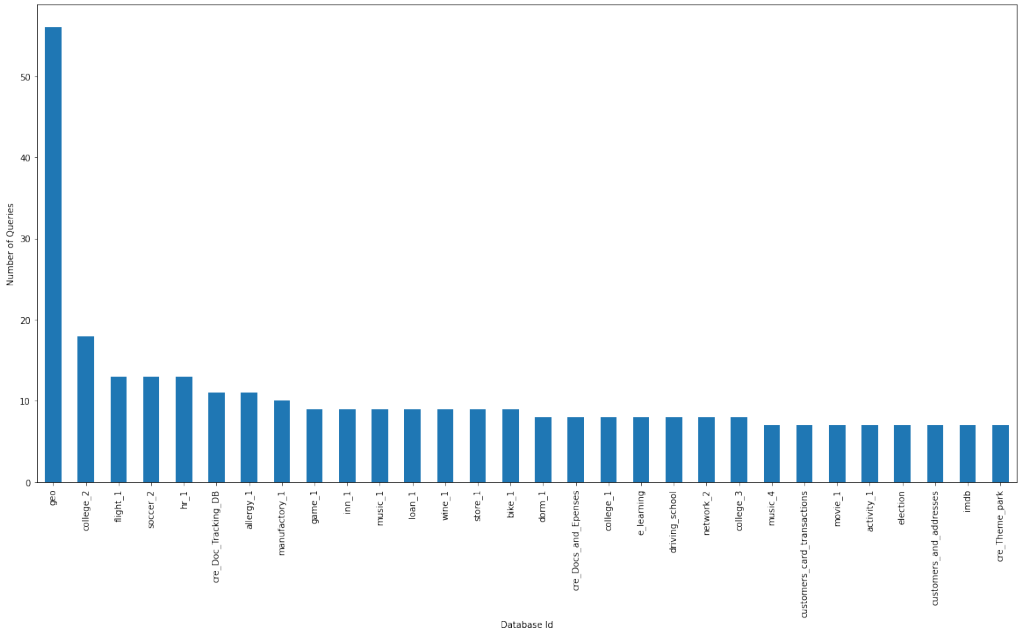

Figure 2: Testing Set : Number of Queries vs Database Id

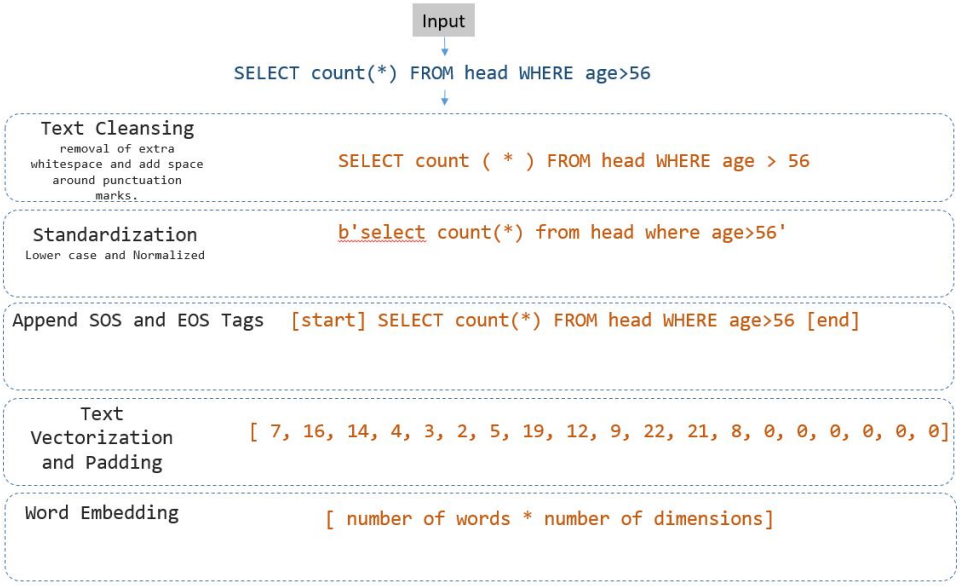

Figure 3: Sample Data Preparation

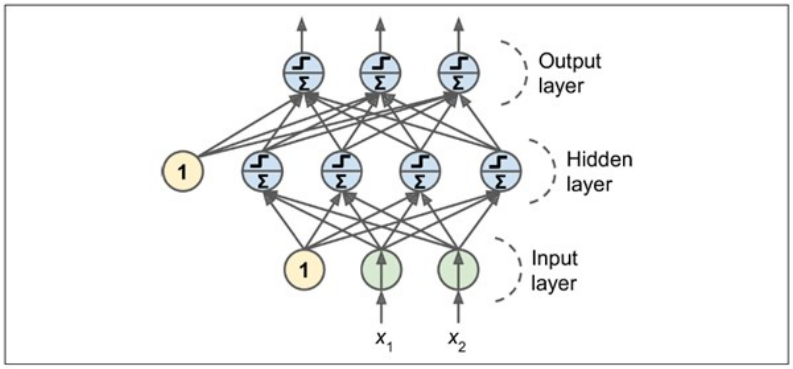

Figure 4: FeedForward Network

3 Methodology

In our study, we aim to utilize RNN based sequence-to-sequence neural machine translation models for the NL2SQL conversion. We simply view SQL as any other target language which need to be translated from English as the source language.

3.1 Encoder-Decoder Architecture

Recurrent Neural Network (Figure 5) are like feed forward network with a difference of connections pointing in backward direction as well. A recurrent neuron, at a given time t receives the input x(t) as well as the (t-1) output y(t–1). This allows them to involve working with sequence of data.[9]. The RNN can be used in different architectures, like Sequence-Vector model used in Sentiment analysis, Vector-Sequence Model used in Image captioning, Sequence to sequence models for text. To deal with our Translation task we will consider the RNN based Encoder-Decoder Architecture.[5]

3.2 Modelling Approach

We have implemented two neural network architectures to solve the NL2SQL translation problem. The first approach uses an Encoder-Decoder model built using LSTM layers, the second approach focuses on using Encoder-Decoder with Attention Layer to include context of the input sentence.

1. Encoder-Decoder Model : The approach is inspired by Encoder-Decoder Network for Neural machine Translation [7].The architecture consists of LSTM Encoder-Decoder where one

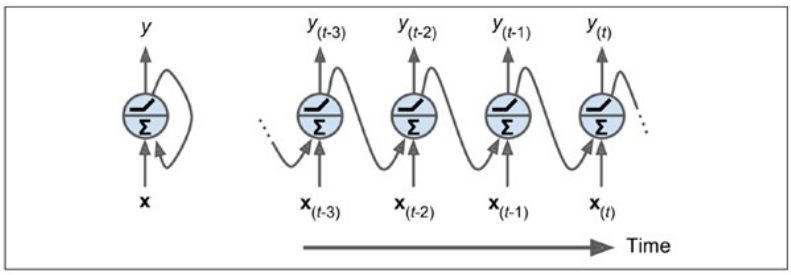

Figure 5: Recurrent Neural Network

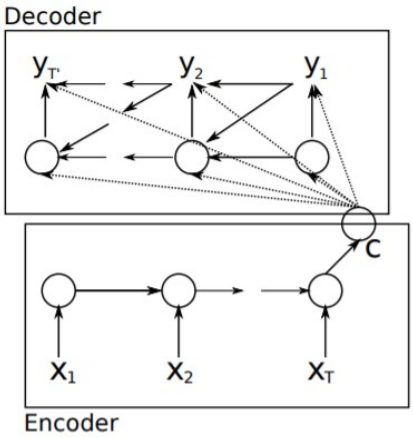

Figure 6: An illustration of the proposed RNN Encoder–Decoder

LSTM layer encodes a sequence of symbols into a fixed length vector, essentially encoding the meaning of the input sequence in a single vector, the other decodes the representation into another sequence of symbols, shown in Figure 6. The two networks target to maximize the conditional probability of output sequence given an input sequence during the training process.
For training the model, size of input and target sequences is fixed and equal to the length of the longest sentence. It uses Custom word embedding for the task. The English sentences are fed as input to the encoder, the encoder cell and hidden states along with SQL query appended with start-of-sentence token is fed to the decoder, the decoder output ending with end-of-sentence token is sent to a dense layer to make predictions. The model is fit by minimizing the *categorical cross entropy* loss in each iteration using *rmsprop* optimizer.
For prediction, the encoder remains the same. At time step t, the decoder accepts it's t-1 time step output, hidden and cell state to predict the next word. The process continues until end-of-sentence is encountered. The decoder output tokens are remapped to words for final output.

2. Neural machine translation with attention : The approach is inspired by Tensorflow Tutorial for Implementation of Sequence to Sequence Network and Attention model found at this link. The Tensorflow Dataset API is used to fetch the data and shuffle it in batches and apply the data preparation steps described in 2.2 Section. A variation of Custom and Glove word embedding is used for the task. A encoder-decoder model with Attention mechanism is created as shown in Figure 7. Reference [11].
For training, the encoder consists of an Embedding and GRU layer which encode the tokenized Input IDs into new sequences. The internal state of the encoder initializes the Decoder and the processed sequence is passed to attention head. The attention head is used to selectively focus on parts of the input sequence. We have used Bahdanau's additive attention to implement the attention layer [3]. The decoder receives output from encoder and use a RNN layer to keep track of what is generated so far. It generate the context vector using the RNN output as query to the attention over the encoder output. The RNN output and context vector generate the attention vector. Based on the generated attention vector logit prediction for next token is generated. The model is fit by minimizing the *SparseCategori-*

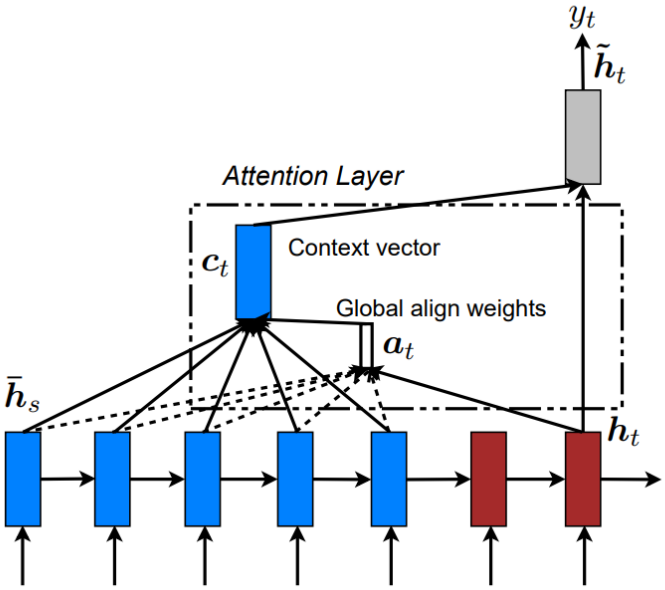

Figure 7: Model Architecture

calCrossentropy loss in each iteration using *Adam* optimizer.
For prediction, the model is similar to the Training, except the input to the decoder at each time step is a sample from the decoder's last prediction.The decoder output tokens are remapped to words for final output.

4 Results

The Sequence to Sequence models with RNN layers were trained on a subset of Spider dataset and evaluated on a separate Test set. Table 4 provide the hyper-parameter details employed by the Encoder-Decoder and NMT with Attention models.

Parameters	Enc-Dec	NMT(Attention)	NMT(Attention)
Word embedding:	Custom (256d)	Glove (100d)	Custom(256d)
Encoder layer:	LSTM	GRU	GRU
Decoder Layer:	LSTM	RNN	RNN
Learning rate:	0.01	0.01	0.01
Number of Epochs:	50	20	20
Optimization and regularization:	rmsprop	ADAM	ADAM
Loss functions:	Categorical CE	Sparse Categorical CE	Sparse Categorical CE

As Table 4 suggests the Encoder decoder models were trained on more epochs due to the simplicity of the model architecture.

The training loss converge to low values with increasing number of epochs. The batch and Epoch losses plotted using Tensor board callback for NMT Attention Models are shown in Figure 8 and 9. The batch and Epoch losses plotted using Tensor board callback for NMT Attention Models with Glove Embedding are shown in Figure 10 and 11. The epoch loss graphs show the NMT models achieve low loss values with 25 to 30 epochs. The Batch loss graphs show spikes which indicate fluctuation of loss values near the start and end of the epochs.

Model	Test Exact Match Accuracy
Encoder-Decoder	2.8%
NMT Attention(Glove 100d)	28.89%
NMT Attention(custom 256d)	33.56%

For evaluating the performance of our models,we have considered 'Exact Match Accuracy' as the **Evaluation Criteria**. The Exact Match is a condition when the strings match completely i.e.

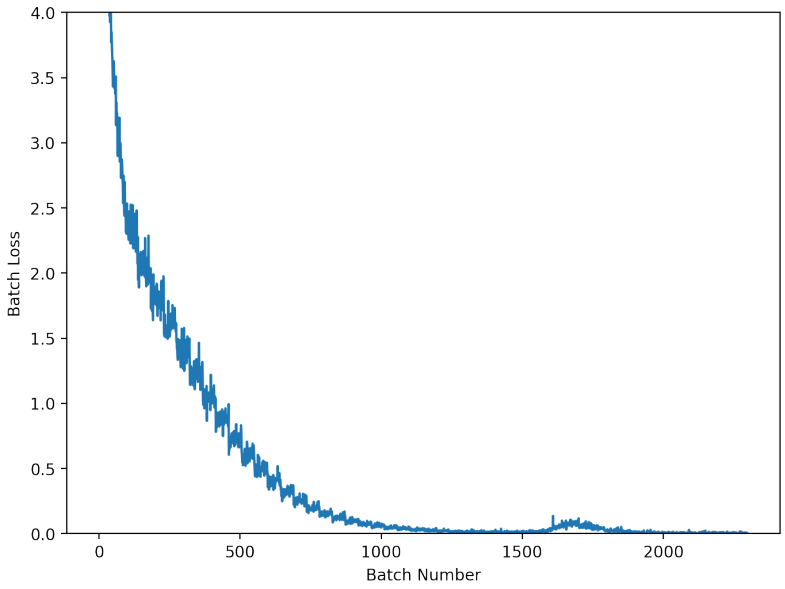

Figure 8: NMT Model :Batch Loss vs Number of Batches

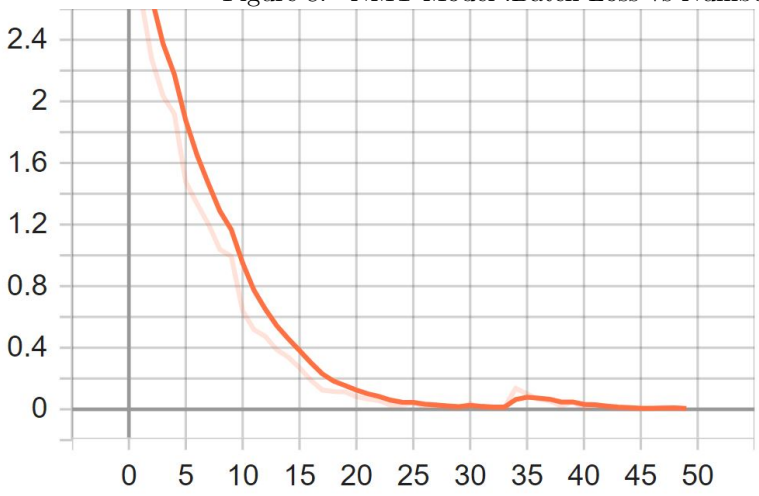

Figure 9: NMT Model :Epoch Loss vs Number of Epochs

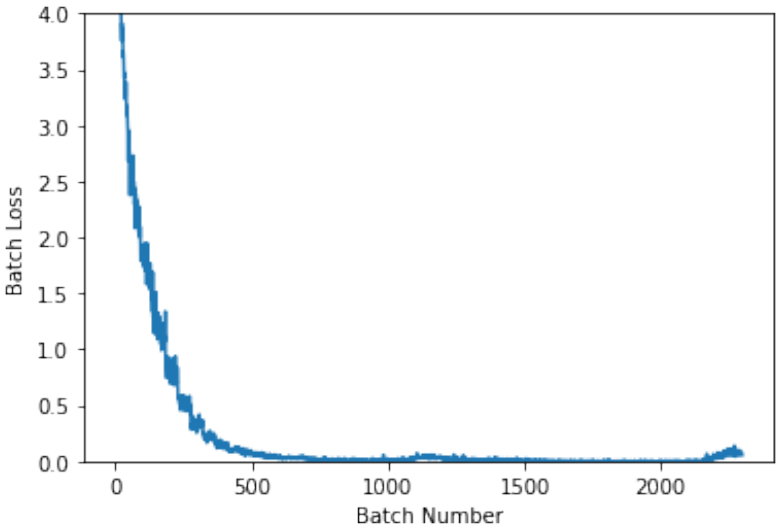

Figure 10: NMT(Glove) Model :Batch Loss vs Number of Batches

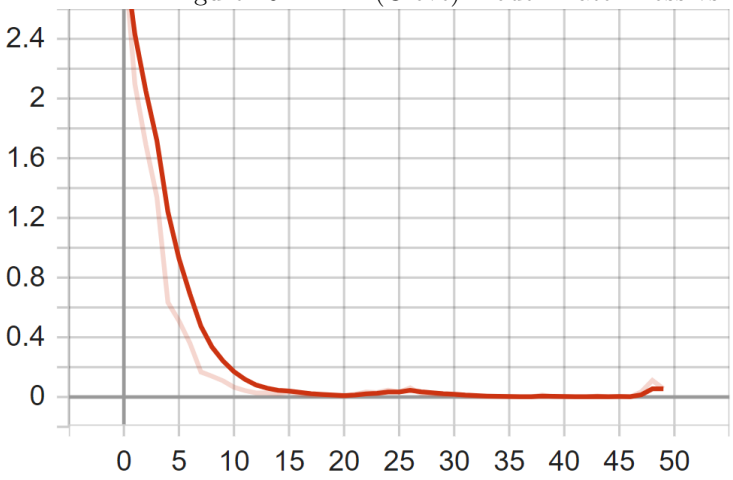

Figure 11: NMT(Glove) Model :Epoch Loss vs Number of Epochs

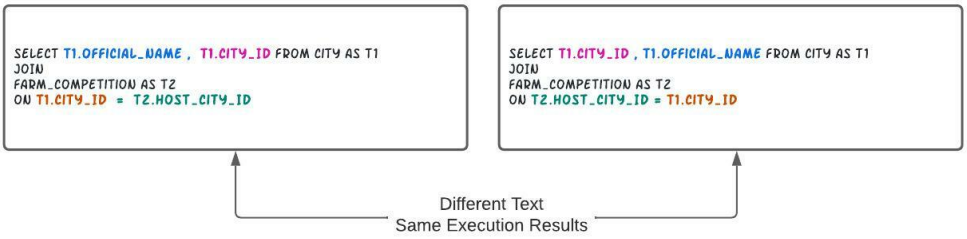

Figure 12: Matching Queries

the Target SQL Query and the Predicted SQL Queries match, ignoring space and case conversion. Based on 'Exact Match' criteria Table 4 summarize the accuracy for our models on Test Dataset.

As 4 Table suggests the highest accuracy achieved using the Sequence to Sequence models is 33.56% approximately on Test Data. The custom Embedding of NMT Attention model works better than the Glove embedding indicating the model predicts better with higher dimensions of word Embedding. Our results show the NMT models were able to learn the SQL translations for Easy queries and also formulate the SQL in correct format. We have observed the placement of keywords like 'select','from', 'where', 'count' appear correctly according to the User query. However, the NMT models are not much successful in interpreting the correct Table/Column Names and Values. The Encoder Decoder model architecture performance is very low due to the simplicity of model architecture. The models under study only consider Semantic parsing and do not take the SQL Query structure into consideration. This may indicate the models like Seq2SQL and SQLNet which take advantage of the structured pattern of SQL queries should be further analysed for the NL2SQL Translation task. We expect to do more in-depth study on the topic in the future.

5 Discussions

The model with Attention layer works much better on the dataset. The simple SQL queries are being successfully translated. The major difficulty in the translation is placing the correct Table names, Column Names and different Condition values. Since the model is not aware of the underlying Database schema the error to predict the correct database elements is high. It is also difficult to predict the Condition values as they not fixed and vary for each question. The encoder-decoder approach is not able to learn the prediction task effectively as seen in this example:

Input : *find the distinct number of president votes .*
Actual Query : *SELECT count(DISTINCT President_Vote) FROM VOTING_VoteRECORD*
Enc-Dec Model Predicted Query : *select distinct (distinct) from*
The NMT Attention Model Predicted Query : *select count (distinct president_vote) from voting_rd*

We would like to discuss some improvements for out study, like the Evaluation criteria can be improved from the 'Exact Match' criteria as the exact match scenarios will not take into consideration the sequence of columns or Joins appearing in a query. The queries, however, produce same execution results. Thus, the evaluation criteria should involve correct prediction of Selected Columns and the aggregation function , 'Where' Columns without values, as example illustrated in Figure 12.[19]. Second, we can include variations to our models like usage of Bi-directional LSTM layers and recently developed Transformer models to improve learning of Encoder-Decoder models. We can experiment with BERT Embedding which have been proved successful in Machine Translation tasks recently.

6 Conclusions

In this report we have tried to explore the text to SQL conversion task using a Sequence-to-Sequence Model. We built a simple Encoder Decoder RNN architecture to achieve the task. Our results show the model with Attention layer performs better but still has limitations. Also, we

have observed the model learnt better with higher dimension word Embedding. The NL2SQL conversion task is still being researched and many improvements have been introduced to achieve high accuracy. The limitations of the proposed model can be understood in the terms of the complexity of this task which is higher to any other Semantic parsing problem.[10] It requires a very high Translation accuracy as any small error in the query can invalidate the result . In current approach, database schema which is crucial for translating the SQL queries is not taken into consideration. It is important for the model to learn the database components along with the target SQL queries. The translation tasks can be ambiguous due to inter dependant columns like Last Salary and Current Salary which might look similar to models but store different information.

To address the aforementioned challenges our model require further improvements and analysis. The further recommendations would include [4] encoding the question and Schema information during Training to consider how words relate to each other within and across question and schema. Consider developing a classification mechanism to predict the SQL components in a SQL query like Columns to be Selected, Aggregation functions to be used (None, Count, Sum, Average etc), Columns in where conditions, Operators in where conditions.

References

[1] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. Natural language interfaces to databases– an introduction. *Natural language engineering*, 1(1):29–81, 1995.

[2] R. Arumugam and R. Shanmugamani. *Hands-On Natural Language Processing with Python: A practical guide to applying deep learning architectures to your NLP applications*. Packt Publishing Ltd, 2018.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Y. Cao. Why is cross-domain text-to-sql hard? *https://www.borealisai.com/en/blog/introduction-cross-domain-text-sql-semantic-parsing/*, 2021.

[5] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[6] H. S. Dar, M. I. Lali, M. U. Din, K. M. Malik, and S. A. C. Bukhari. Frameworks for querying databases using natural language: a literature review. *arXiv preprint arXiv:1909.01822*, 2019.

[7] A. Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* " O'Reilly Media, Inc.", 2019.

[8] J. Gu, Z. Lu, H. Li, and V. O. Li. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*, 2016.

[9] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[10] A. Kim. Text-to-sql learning to query tables with natural language. *https://towardsdatascience.com/text-to-sql-learning-to-query-tables-with-natural-language-7d714e60a70d*, 2020.

[11] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[12] A.-M. Popescu, A. Armanasu, O. Etzioni, D. Ko, and A. Yates. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 141–147, 2004.

[13] E. Reshma and P. Remya. A review of different approaches in natural language interfaces to databases. In *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, pages 801–804. IEEE, 2017.

[14] K. J. Sathick and A. Jaya. Natural language to sql generation for semantic knowledge extraction in social web sources. *Indian Journal of Science and Technology*, 8(1):1, 2015.

[15] W. Woods. The lunar sciences natural language information system. *BBN report*, 1972.

[16] X. Xu, C. Liu, and D. Song. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*, 2017.

[17] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, et al. 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.

[18] J. Zeng, X. V. Lin, C. Xiong, R. Socher, M. R. Lyu, I. King, and S. C. Hoi. Photon: a robust cross-domain text-to-sql system. *arXiv preprint arXiv:2007.15280*, 2020.

[19] R. Zhong, T. Yu, and D. Klein. Semantic evaluation for text-to-sql with distilled test suite. In *The 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2020.

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

Acronyms

LSTM Long Short-Term Memory. 2, 5

NL2SQL Natural Language to SQL Conversion. 2, 5, 10, 11

RNN Recurrent Neural Network. 5