A LSTM based novel approach to convert Natural Language into Structured Query Language

by

Examination Roll: 160537, 160549, 160551

A Researech Project Report submitted to the
Institute of Information Technology
in partial fulfillment of the requirements for the degree of
Bachelor of Science in
Information Technology

Supervisor: Manan Binth Taj Noor, Lecturer

Institute of Information Technology
Jahangirnagar University
Savar, Dhaka-1342

December 2020

DECLARATION

We hereby declare that this thesis is based on the results found by ourselves. Materials of work found by other researcher are mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Syed Mohammad Rakib	Md. Rafi Mahafid	Md. Rasheduzzaman Riad
Roll:160537	Roll:160549	Roll:160551

CERTIFICATE

This is to certify that the thesis entitled **Converting Natural Language into SQL** has been prepared and submitted by **Syed Mohammad Rakib**, **Md. Rafi Mahafid** and **Md. Rasheduzzaman Riad** in partial fulfilment of the requirement for the degree of Bachelor of Science (honors) in Information Technology on July 20, 2020.

Manan Binth Taj Noor
Supervisor

Accepted and approved in partial fulfilment of the requirement for the degree Bachelor of Science (honors) in Information Technology.

Prof. Dr. Md. Abu Yousuf Chairman	Dr. Shahidul Islam Member	Dr. Rashed Mazumder Member

Prof. Dr. Md. Hasanul Kabir
External Member

ACKNOWLEDGEMENTS

We feel pleased to have the opportunity of expressing our heartfelt thanks and gratitude to those who all rendered their cooperation in making this report.

This thesis is performed under the supervision of Manan Binth Taj Noor, Lecturer, Institute of Information Technology (IIT), Jahangirnagar University, Savar, Dhaka. During the work, she has supplied us a number of books, journals, and materials related to the present investigation. Without her help, kind support and generous time spans she has given, we could not perform the project work successfully in due time. First and foremost, we wish to acknowledge our profound and sincere gratitude to her for her guidance, valuable suggestions, encouragement and cordial cooperation.

We express our utmost gratitude to Dr. M Mesbahuddin Sarker, Director, IIT, Jahangirnagar University, Savar, Dhaka, for his valuable advice that have encouraged us to complete the work within the time frame. Moreover, we would also like to thank the other faculty members of IIT who have helped us directly or indirectly by providing their valuable support in completing this work.

We express our gratitude to all other sources from where we have found help. We are indebted to those who have helped us directly or indirectly in completing this work.

Last but not least, we would like to thank all the staff of IIT, Jahangirnagar University and our friends who have helped us by giving their encouragement and cooperation throughout the work.

ABSTRACT

Databases store a huge amount of data from all around the world. So extracting data from these huge databases is quite complicated. It also requires complex SQL queries to retrieve data from databases. When queries are joined to retrieve data from large databases, it becomes very difficult for the people from non-technology background. It would be quite easier if natural language could be used to retrieve data from databases without using any query language at all. Moreover, it will be time saver and simpler. So we propose Seq2SQL, a deep neural network model that will first convert Natural Language into SQL queries and then provide the result. JSON dataset which is imported in this proposed work is based on seq-2-seq semantic parsing. No grammar is required for semantic parsing, which will eventually, be classified into aggregation, OP, select, col, where etc clause. This form will be considered as augmented inputs for this work which will be in tokenized form. Embedded inputs are then put for further LSTM encoding. 2 layers of bidirectional LSTM will be applied and data will be trained by encoder using policy gradient. After that, pointer network is used for decoding LSTM. LSTM decoder will predict final SQL sequence based on the input sequence, aggregation and table data. Use of transfer learning in training the dataset will make training more efficient and time convenient.

Keywords: NLP, Seq2SQL, RNN, LSTM, WikiSQL and Semantic Parsing.

LIST OF ABBREVIATIONS

NLP	Natural Language Processing
SQL	Structured Query Language
LSTM	Long Short Term Memory

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER I

Introduction

1.1 Background

With the quick improvement of data engineering, industry engineers frequently perform data queries for data analyses and/or obtaining online reports. SQL is a popular and flexible language for querying data. NLP(Natural Language Processing) is a part of computer science and artificial intelligence that deals with human languages. Despite large amount of data can be fetched efficiently and accurately from Relational Databases whoever it requires one to master formulate the formal queries. As for SQL, it is a Structured Query Language. Query languages are used in programming and designed for managing data held in a Relational Database Management System (RDBMS). Sometimes it becomes quite complex when the database is very large. As for Natural language, we use it in our everyday life. That is why, researches are going on for the development in human computer interaction for a long time. In this regard, there was an early natural language understanding computer program, developed by Terry Winograd in 1968. Related work is in [1] After that in 1978, Hendrix built the first good database with Natural Language Processing where human language can interact with database. The journey of NLP and SQL is still going on.

1.1.1 NLP and SQL

There are different categories in NLP. One of them is Natural Language Understanding, which can convert text into more formal representations such as first-order logic structures that are easier for computer programs to manipulate. Natural language understanding involves the identification of the intended semantic from the multiple possible semantics which can be derived from a natural language expression which usually takes the form of organized notations of natural language concepts

A relational database is a digital database based on the relational model of data. A software system used to maintain relational databases is a relational database management system. We used Natural Language to convert Structured Query Language to retrieve data from databases [2]

1.2 Overview

Nowadays, databases are getting more and more complex. Relational databases are getting much bigger. It has become harder to retrieve data from the database using structured query language. So we have proposed Natural language to SQL query conversion system where necessary information can be found faster from any complex relational database in an automated way using only natural language query.

We applied JSON dataset for the Natural Language input. From the Natural Language input, we did semantic paring; which showed the result seq-2-seq natural semantic parsing without any requirement of grammar. It classified Aggregation, OP, select, col, and where clause. We have formatted the data in tokenized structure. From semantic paring, we produced Augmented Inputs. After that, we have used 2-layer bidirectional LSTM network for encoding which encoded word by word. Then we trained with Policy Gradient. We used pointer network in the trained dataset to decode the LSTM. From the decoded LSTM, we received new state and different scores. From various score, only the highest scored token was chosen to generate the SQL queries from Natural Language.

1.3 Problem Statement

Due to rapid computing scenario, information retrieval methods are being highly accustomed to help organizations, education institutions, on demand information which is required while communicating with users and client meeting to present them with accurate data. Despite large amount of data can be fetched efficiently and accurately from Relational Databases whoever it requires personnel to master the semantics to formulate the formal queries. Linguistics (Natural Language processing) and Artificial Intelligence capabilities can form an association to understand information in natural language and generate database query language such as NoSQL and fetch information from selected database.

1.4 Motivation

In modern science, data is one of the most important elements. In every day life we store data and use data in various field. Each data is related to other data as well. Here comes the relational database. However, retrieving data from Relational database becomes quite troublesome. And those who don't understand SQL queries, it becomes impossible for them to retrieve data from databases. But things will get easier if we use natural language to generate SQL queries and use the SQL queries to extract data from the database. Even if it's in relational form, we can still use Natural Language to receive our valuable data from any relational databases with the help of Deep Learning.

In our research we first received the Natural Language from the users.We used the input and takenized the form. After that we used 2-layer bidirectional LSTM network and Pointer Network to encode and decode the input. As a result, tokens with the highest score are chosen to generate the SQL query. This query can retrieve data from any database.

1.5 Objective

- To use Query language to fetch data.
- To store the data in heterogenous form.
- To visualize the fetched data in specified format.
- To demonstrate complex queries from multiple normalized database tables.

1.6 Research Outline

Rest of the report is structured as follows: In **Chapter II** a literature study on related work is given including explanations for the most important terms used in this thesis-basic concept. **Chapter III** introduces system model including system architecture. Architecture of Proposed Diagram, Dataset Pre-processing, Predict Aggregation, column, selection and Accuracy of the model through this chapter. In **Chapter IV** we can decide from our model that using this model becomes very convenient in everyday life.

CHAPTER II

Literature Review

2.1 Prior Works

There has been a huge advancement in the field of Natural Language Processing (NLP) in recent years, particularly because of the recent progresses in Deep Learning. However, the idea of word representation is not a recent idea at all.

2.1.1 Early Efforts

Being introduced at [3], it has gone through a number of development stages. These earlier efforts focused mainly in using feature representations to quantify (semantic) comparison used hand-crafted characteristics. Word embeddings are bottomed on the scheme that contextual information can singly constitute a feasible portrayal of linguistic objects. When it comes to Machine Language Translation, we also find some works in the last years in last century [4] [5] [6]. They mostly relied on different Time-Delay Neural networks which were based on two important features : 1. A 3-layer positioning of trivial computing units, a hierarchy that permitted the formation of nonlinear decision surfaces using error backpropagation. 2. The time-delay arrangement. While they have been the pioneer, there are some recent approaches that are more convenient to our work.

2.1.2 Semantic Parsing

Semantic parsing is another approach that has been studied extensively in the NLP community [7] [8] [9] [10] [11] [12]. There are three distinct works found among them, which includes (1) the language of the logical form; for example : first order logic, lambda calculus, lambda dependency based compositional semantics (lambda DCS) and structured query language (SQL); (2) the language of the knowledge based form; for example: facts from large collaborative knowledge bases, semi-structured tables and images; and (3) the supervision used for learning the semantic parser, e.g. question-denotation pairs and question-logical form pairs. Generating sequence in the means of linearizing trees [13] [14] [15] [16] [17] is another buzz subtopic in this era . They aimed at mapping a natural language utterance to a logical form. Recent success rate of these works shows how efficient neural networks can be in this field.

2.1.3 Grammar-based Methods

Grammar-based methods to convert natural language into machine language have been adapted in some works [18] [19]. This method was applied in SQL generation as it had shown some remarkable developments in other semantic parsing tasks. They introduced techniques to make a schema-dependent grammar with minimal actions taken. These techniques were then analyzed on some difficult text-to-SQL datasets such as ATIS and SPIDER and had been able to reduce errors at significant rate.

2.1.4 Long Short Term Memory

Long Short Term Memory (LSTM) [20] inherits the same architecture as vanilla Recurrent Neural Network (RNN). Many LSTM based text generation models have been proposed [21] [22] [23]. The memory units in LSTMs, which are referred to as cells, take the combination of previous state and current input as input. These cells actually decide what to keep in memory and what to eliminate. It only learns to keep relevant information to make predictions and forgets all the non-relevant data. As a result of this, LSTM deals with very less redundant works than all the other methods mentioned above.

2.2 Limitations of Prior Works

Firstly, neural network based methods that involve semantic parsing are more or less data hungry, which results in closely correlate with the capacity of training data. Secondly, in order to have meaningful interactions, knowing when the model is uncertain in its predictions, semantic parsing method still needs refinement for better user experience. Thirdly, most of the researches regarding grammar-based method were worked on WikiSQL, which is a collection of simple queries and can be parsed be using only 4 grammar rules. So they cannot be expected to generate queries that are comparatively complex.

For a good training model, we will have to train a large dataset which generally takes a long time to proceed. In contrast with the prior works, we are using transfer learning which will not only make the data training smoother, but also will make it easier and time convenient. Again We have chosen Semantic Parsing over Grammar-based method so that our system can deal with a wide range of queries. Unlike most of the grammar-method works, our system can generate nested queries and queries that include aggregate functions. To overcome the limitations of semantic parsing method, we have considered LSTM with attention to have ease with the training data and also to enhance user experience. Combining semantic parsing and LSTM with attention, it is clear to have better performance and more accuracy than other existed works.

CHAPTER III

Methodology

Natural Language Processing is a part of computer science and artificial intelligence. SQL, it is a Structured Query Language. SQL language is used in programming and designed for managing data held in a relational database management system (RDBMS). First, we need to import JSON dataset in our model. After that, we use semantic paring technique to classify aggregation, operation, select, col, and where clause. It gives us a tokenized form. We have applied 2 layers of bidirectional LSTM and trained the encoding using Policy Gradient. After that, we use Pointer Network for decoding LSTM. We fit our model to generate the best weight value and scores to predict more accurate SQL query.

3.1 Proposed System

In our model, first we pre-process our dataset which gave tokenized data. After receiving tokenized data we extract the natural language query and target SQL query. Then we need to clarify semantic relationship between the words in a sentence. For that we use Word Embedding. Using LSTM we can predict Aggregation, Column and Selection clause. LSTM encoder will encapsulate hidden states for input data, then it will be provided to the decoder as input. Decoder will predict our target SQL sequence word by word. To process large datasets with about 56 thousands lines of natural language question and query, we will divide the data into smaller chunks and train these smaller datasets with transfer learning. Best fitted model with best scores will predict our target SQL query.

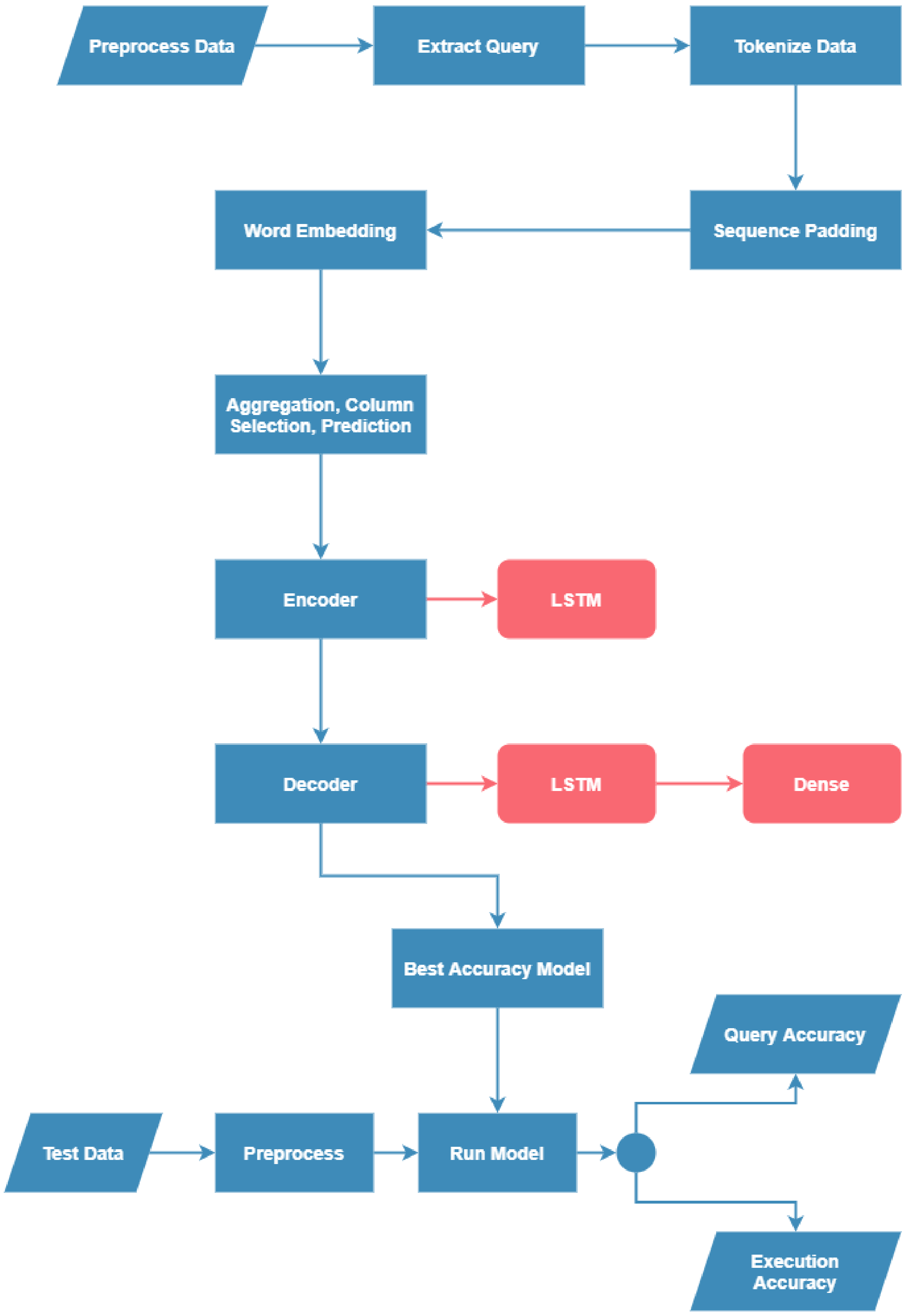

Figure 3.1: System Diagram

3.2 System Implementation

3.2.1 Dataset Pre-processing (Train, Test, Dev Data)

The data collected for this program are usually in the form of table and natural language query. But to train our NLP conversion program, semantic parsing is required. Deep semantic parsing is the most suitable for this because of the fact that meaning of every word carries different level of significance and it can influence the accuracy of the finally generated query in a significant way. Data pre-processing consists of some steps where tokenization, modification of table data and query data will be done. This pre-processing needs to be done on all of train, test and validation datasets.

3.2.1.1 Tokenize data and query

The first step here is tokenization, which is one of the most common task when the work is based on natural language. Tokenization means splitting every part of a text document. The smallest unit of the splitted document is called token. Tokenizing can be done by splitting text document into sentences or splitting sentences into word and so on. We need to split every word from dataset document and store tokens individually. This tokenization step is required because to generate a more accurate SQL query, training program needs to understand the meaning of every word and process those words based on their impact on the actual query the user intends to make.

After tokenizing the dataset, natural language query, tokenized query, table and column value are concatenated together to hold necessary column, operator values together.

3.2.1.2 Sequence Padding

Length of all the natural language input or SQL sequences output will not be same most of the time. To make training model efficient, input vector of the model should have same length. For this purpose, padding with extra dummy value is done at the beginning or at the end of the sequence to make all of them of same length. Most of the time 0 is used as the dummy value. Adding extra 0 at the beginning is called pre-sequence padding, and adding at the end is called post-sequence padding. If length for any sequence gets too large than the expected length, we might need to truncate the sequence as well.

3.2.1.3 Word embedding for dataset

Word embedding plays an important role in natural language processing and deep learning. It helps to maintain the semantic relationship between the words of the sentences. Words of similar type of meaning can have similar representation with the help of word embedding. In this technique, every word is represented as vector in a broad vector space based on their meaning. Their might be a lot of dimensions in a vector space of word embedding. Learning process of word embedding technique is similar to neural network. Large scale corpus is used to define meaning and vector value of each word in word embedding. Various word embedding methods such as word2vec, glove, embedding layer are used to define vector value. Word2vec, which was invented by google presumes that words which are closer in a sentence tends to have semantically same value. Word2vec deduces vector value based on distributional hypothesis. Glove was developed at Stanford where global corpus statistics are captured by this technique.

3.2.2 Network Model Architecture

Best fitted training model for aggregation, select, column prediction are stored to test accuracy. Test dataset of table and query is preprocessed like training dataset. Whiles testing, previously stored models are used to generate SQL query from natural language query of test dataset. Comparison with the output query and actual SQL query provides an accuracy value. Execution of the output query can generate a query solution which will be again compared for accuracy check. Execution accuracy tends to provide slightly better result than the query accuracy as comparison of SQL query sentences may show bigger difference just because of change of a single word than the actual query.

3.2.2.1 Aggregation Predictor

Our Aggregation predictor does the job of a 6-way classification problem. The 6 possible outcomes are : MAX, MIN, COUNT, SUM, AVG and no aggregator. The input question word vector goes through the LSTM encoder first to produce a hidden state embedding. The hidden state is then fed towards a fully-connected layer and a softmax layer to make a probability distribution for all the six possible outcomes.

3.2.2.2 Column Predictor

There is a difference between Aggregation Predictor and Column Predictor. Unlike Aggregation Predictor, Column Predictor doesn't have a limited range of value. So instead of a classification problem, here we had to deal with an information retrieval problem. Thus, Transformer architecture was introduced.

Our Column predictor produces a probability distribution for the given column list from the existing tables. It assumes that the table is visible to the predictor and we need to make sure the column names are just correct so they are exposed to the model. It first encodes the question and the column names to word embeddings. Then the distribution is calculated using the column names, the input question, trainable matrices and the total number of columns from the table associated with the input question.

3.2.2.3 LSTM

During back propagation through many nodes and layers, Recurrent Neural Network or RNN suffers from vanishing gradient problem while making predictions. Gradient shrinks exponentially through time. Besides contribution from starter layer shrinks too. Due to this problem, earlier layers can't learn efficiently for long sequence. To solve this short term memory problem, LSTM or Long Short Term Memory will be used as it will help to carry the value of all of the cell state knowledge. Various cell states and gates in LSTM decide which data to store and which data to erase in the training process.

3.2.2.4 Encoder

As our working neural network model is based on seq2seq model, main priority of our model is to predict SQL query sequence from natural language question sequence. But an important issue here is that length of natural language question and SQL query will not be same always. Length of these two types of sequence will be different most of the times. That's why we need to implement our LSTM model with encoder and decoder where encoder encapsulates the meaning and relationship of the input vectors in hidden state and decoder predicts the output from encapsulated hidden state. Encoder is made of many layers of recurrent unit. It will contain LSTM cell in in every layer of the stack. It collects input from input sequence word by word, process these inputs , train model based on their meaning and pass the output to the next layer as part of the input for next layer. It will compute hidden state, cell state

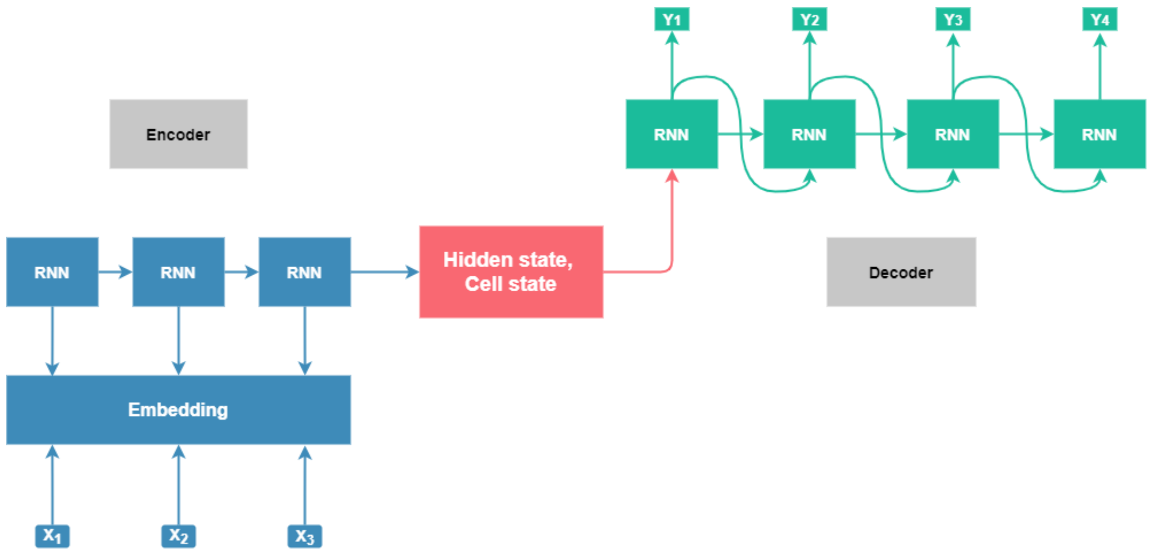

Figure 3.2: Encoder-Decoder Model

and outputs in every layer. Encoder encapsulates the knowledge on input sequence and produces encoder vector which helps the decoder to make better predictions for SQL query.

3.2.2.5 Decoder

Only final hidden state and cell state of encoder are fed forward to the decoder, outputs are not used here. Decoder starts prediction with the "Start of Sentence" or "<SOS>" token and the final hidden state, cell state of encoder. In every unit of decoder, it receives the hidden state from previous layer and predicts output as well as the hidden state for current unit. It stops providing predictions when it encounters "End of Sentence" or "<EOS>" token.

$$h_t = f(W^{(hh)}h_{t-1} + W^{(hx)}x_t) \tag{3.1}$$

3.2.2.6 Transfer Learning

While processing and training a large dataset, sometimes it takes a lot of time and consumes large processing power. To overcome this issue, use of transfer learning plays a very important role. Generally it is used to transfer the knowledge of an existing model which is trained previously with same types of data or different types of data. To make the training programme easier, smoother and time convenient, transfer learning will be a great help. Basically we will split our large dataset to

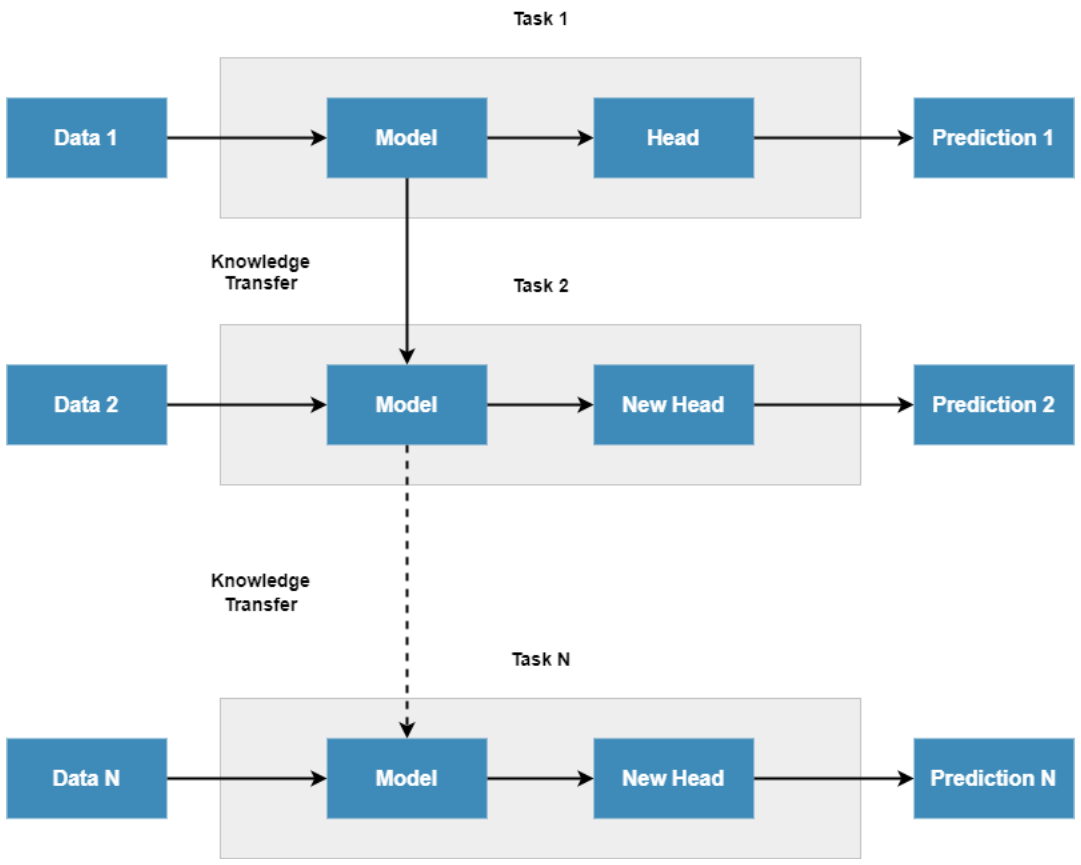

Figure 3.3: Transfer Learning

smaller chunks of data, then every chunk will be trained with our network model. When we train our model with smaller dataset for the first time, trained model will be stored in device. After that, whenever we train our model with new smaller part of the dataset, pre-trained model will be loaded from device and that existing model will be trained all over again. As a result we are never losing knowledge on already trained model and we don't need to train with full dataset altogether at a time. This method will save both time and processing memory. Besides it will increase performance of the network model too.

3.2.2.7 SQL Query Generation

Here, all the prior predictions will be merged and complete queries will be generated. Some of the sample queries in the natural language and their converted form into SQL are given below :
Text query: Which department did Rashed go to?
Corresponding SQL query: SELECT department WHERE studentName = 'Rashed'

Text query: How many tutorials were taken in the class of Mr. Mahafid ?
Corresponding SQL query: SELECT COUNT(tutoralID) where teacherName = Mahafid
Text query: How many exams did Rakib take?
Corresponding SQL query: SELECT SUM(tutorialID) where studentName = Rakib

3.2.3 Test with test data

Testing the data is a vital part, once the queries are generated. We uesd two approaches to check the accuracy of our generated queries in the prior steps: checking query accuracy and checking execution accuracy.

3.2.3.1 Checking Query Accuracy

Here, we observed the matches between the ground truth queries and our generated queries in percentage. However, strings were not compared directly. To check the similarity, we first divided the queries into number of parts, such as : the column in SELECT clause, the aggregator in SELECT clause, and each condition in 'WHERE' clause. This is done so that the negative effect of condition ordering is minimized. For example, 'WHERE id = 2 AND course = LSTM' should be equal to 'WHERE course = LSTM AND id = 2'. This is achieved by transferring the WHERE clause into a set of conditions and comparing each element in the set.

3.2.3.2 Checking Execution Accuracy

In this section, we observed the matched execution results between the ground truth queries and our generated queries in percentage. There can be more than one SQL queries that both satisfy some specific condition. Therefore, we had to use this accuracy checking to calculate the accuracy of our queries.

3.3 Gantt Chart

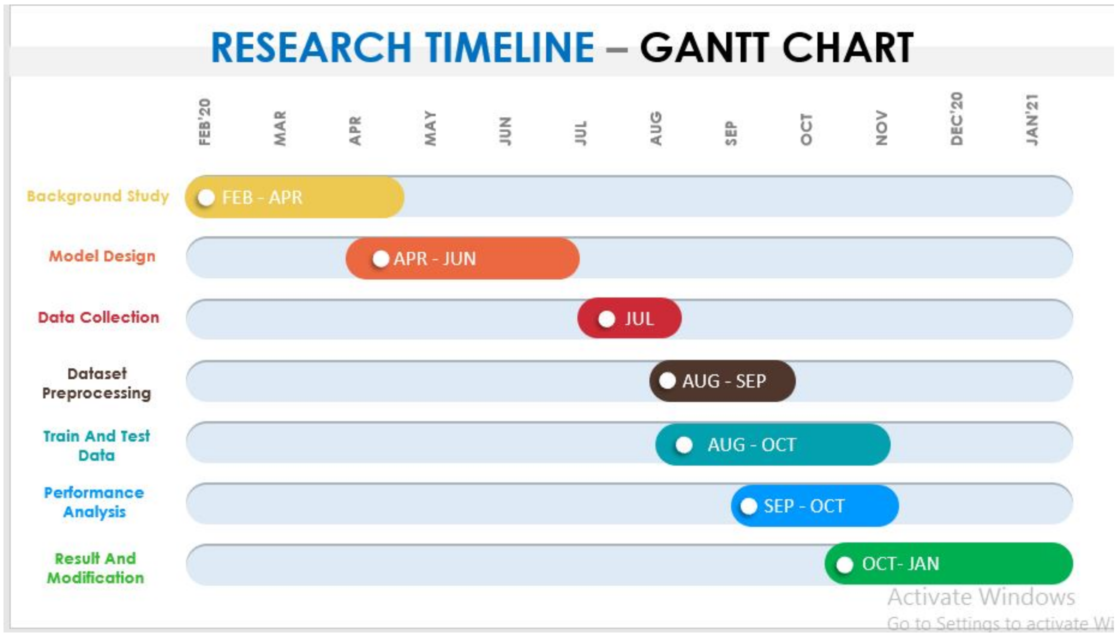

Figure 3.4: Gantt Chart of Research Timeline

CHAPTER IV

Conclusion

In this research paper we have discussed our proposed system for converting Natural Language to SQL queries. This model gives us a more efficient way to use Natural Language. Some techniques on preprocessing data was discussed with our research model. In modern world, processing data and fetching required information from data is a great challenge. In this regard our NLP to SQL model will add more contribution. People who don't understand Structured Query Language will be greatly benefited as they can easily retrieve data from database. We expect that our model will provide better performance with some more work on our model.

References

[1] J. Allen, *Natural language understanding.* Pearson, 1995.

[2] A. R. Sontakke and A. Pimpalkar, “A rule based graphical user interface to relational database using nlp,” *interaction*, vol. 5, no. 6, 2014.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[4] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, “Phoneme recognition using time-delay neural networks,” *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[5] N. Koncar and G. Guthrie, “A natural language translation neural network,” in *New Methods in Language Processing*, pp. 219–228, 1997.

[6] M. A. Castano, F. Casacuberta, and E. Vidal, “Machine translation using neural networks and finite-state models,” *Theoretical and Methodological Issues in Machine Translation (TMI)*, pp. 160–167, 1997.

[7] L. S. Zettlemoyer and M. Collins, “Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars,” *arXiv preprint arXiv:1207.1420*, 2012.

[8] P. Liang, M. I. Jordan, and D. Klein, “Learning dependency-based compositional semantics,” *Computational Linguistics*, vol. 39, no. 2, pp. 389–446, 2013.

[9] J. Berant, A. Chou, R. Frostig, and P. Liang, “Semantic parsing on freebase from question-answer pairs,” in *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1533–1544, 2013.

[10] J. Krishnamurthy and T. Kollar, “Jointly learning to parse and perceive: Connecting natural language to the physical world,” *Transactions of the Association for Computational Linguistics*, vol. 1, pp. 193–206, 2013.

[11] P. Pasupat and P. Liang, "Inferring logical forms from denotations," *arXiv preprint arXiv:1606.06900*, 2016.

[12] S. Iyer, I. Konstas, A. Cheung, J. Krishnamurthy, and L. Zettlemoyer, "Learning a neural semantic parser from user feedback," *arXiv preprint arXiv:1704.08760*, 2017.

[13] L. Dong and M. Lapata, "Language to logical form with neural attention," *arXiv preprint arXiv:1601.01280*, 2016.

[14] R. Jia and P. Liang, "Data recombination for neural semantic parsing," *arXiv preprint arXiv:1606.03622*, 2016.

[15] C. Xiao, M. Dymetman, and C. Gardent, "Sequence-based structured prediction for semantic parsing," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1341–1350, 2016.

[16] K. Guu, P. Pasupat, E. Z. Liu, and P. Liang, "From language to programs: Bridging reinforcement learning and maximum marginal likelihood," *arXiv preprint arXiv:1704.07926*, 2017.

[17] L. Dong, C. Quirk, and M. Lapata, "Confidence modeling for neural semantic parsing," *arXiv preprint arXiv:1805.04604*, 2018.

[18] P. Yin and G. Neubig, "A syntactic neural model for general-purpose code generation," *arXiv preprint arXiv:1704.01696*, 2017.

[19] K. Lin, B. Bogin, M. Neumann, J. Berant, and M. Gardner, "Grammar-based neural text-to-sql generation," *arXiv preprint arXiv:1905.13326*, 2019.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] D. Pawade, A. Sakhapara, M. Jain, N. Jain, and K. Gada, "Story scrambler-automatic text generation using word level rnn-lstm," *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 10, no. 6, pp. 44–53, 2018.

[22] X. Chen, Y. Li, P. Jin, J. Zhang, X. Dai, J. Chen, and G. Song, "Adversarial sub-sequence for text generation," *arXiv preprint arXiv:1905.12835*, 2019.

[23] W. Wang, Z. Gan, H. Xu, R. Zhang, G. Wang, D. Shen, C. Chen, and L. Carin, "Topic-guided variational autoencoders for text generation," *arXiv preprint arXiv:1903.07137*, 2019.