



Dissertation on
**“Question and Answering System Based on
Text to SQL Generation”**

Submitted in partial fulfilment of the requirements for the award of degree of

**Master of Technology
in
Computer Science & Engineering**

UE20CS970 – Project Phase – 2

Submitted by:

Ragul G S G

PES2PGE20DS011

Under the guidance of

Dr. Narayana D

Prof. Sudha B.G

January - March 2022

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY**

**(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India**



PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

‘Question and Answering System Based on Text to SQL Generation’

is a bonafide work carried out by

Ragul G S G

PES2PGE20DS011

In partial fulfilment for the completion of Fourth Semester Project Phase - 2 (UE20CS970) in the Program of Study - Master of Technology in Data Science and Machine Learning under rules and regulations of PES University, Bengaluru during the period Jan 2022 – March 2022. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 4th semester academic requirements in respect of project work.

Signature
Dr. Arti Arya
Prof. CSE

Signature
Dr. Sandesh B J
Chairperson

Signature
Dr. B K Keshavan Dean
of Faculty

External Viva

Name of the Examiners

Signature with Date

1. _____

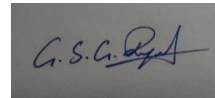
2. _____

DECLARATION

We hereby declare that the Project Phase - 2 entitled **“Question and Answering System Based on Text to SQL Generation”** has been carried out by us under the guidance of Dr. Narayana D and Prof. Sudha B.G, Dept of CSE and submitted in partial fulfilment of the course requirements for the award of degree of **Master of Technology in Data Science and Machine Learning** of **PES University, Bengaluru** during the academic semester January – May 2021. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES2PGE20DS011

Ragul G S G



ACKNOWLEDGEMENT

I would like to express my gratitude to Dr. Narayana D and Prof Sudha B.G, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE20CS970 - Project Phase – 2.

I am grateful to the project coordinators, Dr. Saravathi V, for organizing, managing, and helping with the entire process.

I take this opportunity to thank Dr. Sandesh B J, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT :

The relational database contains a significant amount of data generated all around the world. However, getting knowledge and business insights from those data needs separate knowledge about any of the query languages. To make this process simpler, this work is based on how to query a data or table in a plain English language rather than using any of the query language. This model is built with the help of encoder-decoder with attention mechanism and deep neural networks to query the answer through the plain English text by semantic parsing. Deployment is also planned for the same.

Table of Contents

1 INTRODUCTION	9
1.1 Background	9
1.2 Problem Statement	9
1.3 Existing Approach	9
2 LITERATURE SURVEY	10
2.1 Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task	10
2.2 Attention is all you need	10
2.3 SQLNet: Generating structured queries from natural language without reinforcement learning	10
2.4 SEQ2SQL: Generating structured queries from natural language using reinforcement learning	10
2.5 Content Enhanced BERT-based Text-to-SQL Generation (NL2SQL)	11
2.6 Learning a Neural Semantic Parser from User Feedback Network	11
2.7 SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task	11
2.8 Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation	11
2.9 Grammar-based Neural Text-to-SQL Generation	12
2.10 Model-based Interactive Semantic Parsing: A Unified Framework and A Text-to-SQL Case Study	12
3 SYSTEM REQUIREMENTS SPECIFICATION	12
3.1 Functional Requirements	12
3.2 Non-Functional Requirements	13
3.3 Hardware Requirements	13
3.4 Software Requirements	14
4 PROPOSED APPROACH	14
5 DATASET DESCRIPTION	14

6 MODEL ARCHITECTURE	16
6.1 Tokenization	16
6.2 Embeddings	17
6.3 Encoder-Decoder Network	17
6.3.1 Encoder	17
6.3.2 Decoder	18
6.3.3 Attention Mechanism	19
7 WORKING MODEL	23
8 IMPLEMENTATION	24
9 EVALUATION METRICS	25
10 RESULTS	26
11 CONNECTING TO THE DATABASE	27
12 CONCLUSION AND FUTURE WORKS	28
13 REFERENCES	29

List of Figures

Fig 4.1: Block Diagram of proposed methodology	14
Fig 5.1: Spider Dataset Overview	15
Fig 6.1: Model Architecture	16
Fig 6.2: Basic sequence to sequence encoder-decoder model	18
Fig 6.3: Encoder-Decoder with attention mechanism	20
Fig 7.1: Working model of Seq2SQL	23
Fig 8.1: Luong-attention	24
Fig 9.1: Evaluation Metrics	25
Fig 10.1: Results from the model	26
Fig 11.1: Output from the database	27

1.INTRODUCTION:

Text to SQL has been studied as one of the prominent research areas by many researchers from the past decade. Since we need to query and database some people are having lack of knowledge in query languages. Due to this an idea has been arised that why can't we query a database in normal human language itself. This gives the rise of the concept text to SQL. The main motto of this idea is about people who are lacking in query languages, they can get insights from the database without knowing them.

1.1 Background:

The problem we discuss here is a branch of broader problem; natural language to machine language. SQL is known for its unique, high-quality language and close communication with the underlying data on the rdbms. We will apply these features to our project. SQL is a data acquisition tool. To create a system that can generate a SQL query from natural language text we need to create a system that can understand the natural language. Most of the research done so far solves this problem by teaching a system of identifying parts of speech for a particular word and using named entity recognition in a natural language called tagging. After this the system is made to understand the meaning of the natural question in which all words are combined which is called parsing. If the parsing is done successfully the system will generate a SQL query using the appropriate MySQL syntax. Here in this approach we use the context vector associated with the attention mechanism to better understand the contextual meaning of the words.

1.2 Problem Statement:

This project makes use of natural language processing techniques to work with text data to form SQL queries with the deep learning techniques. Input will be in plain English language query text and it will be generated into relevant query. Then the generated query is pipelined to the database to fetch the relevant answer for the plain query text with the help of the generated query.

1.3 Existing Approach:

The existing approach is to generate the query are of sketch based where it contains dependency graph so that one prediction can be done by taking into consideration only the previous conditions that it depends upon. This approach is named as SQLNet and it is performed with the wikisql dataset. Another method is a simple method to leverage the table content for the BERT-based model to solve the text-to-SQL problem. Based on the observation that some of the table content match some words in question string and some of the table header also match some words in question string, it encodes two addition feature vector for the deep model. This method is known as NL2SQL. These above models are trained with the help of WIKISQL dataset where it consists only of single table queries.

2. LITERATURE SURVEY :

2.1 Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task

The spider dataset is presented as a large scale, complex and cross domain semantic parsing text to SQL dataset annotated by 11 college students. Consists of 10,181 queries and 5,693 complex SQL database questions with 200 multiple tables, covering 138 different domains. It explains complex semantic domain classification and text-to-SQL function where complex queries and detailed information appear. Spider differs from some of the previous semantic analytical operations in that it all uses the same site and exactly the same programs in the train set and in the test set. They test a wide range of high quality models and the best model achieves only 12.4% of the exact match accuracy on database split setting. This shows that Spider presents a major challenge in future research.[1]

2.2 Attention is all you need

It propose a new simple network architecture, the Transformer, based solely on attention mechanisms. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. This model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, this model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs.[2]

2.3 SQLNet: Generating structured queries from natural language without reinforcement learning

A sketch-based approach where the sketch contains a dependency graph so that one prediction can be done by taking into consideration only the previous predictions that it depends on. In addition, it propose a sequence-to-set model as well as the column attention mechanism to synthesize the query based on the sketch. By combining all these techniques, it showed that SQLNet can outperform the prior art by 9% to 13% on the WikiSQL task.[3]

2.4 SEQ2SQL: Generating structured queries from natural language using reinforcement learning

A deep neural network for translating natural language questions to corresponding SQL queries. This model uses rewards from in-the-loop query execution over the database to learn a policy to generate the query, which contains unordered parts that are less suitable for optimization via cross entropy loss. By applying this query execution environment to WikiSQL, Seq2SQL outperforms a state-of-the-art semantic parser, improving execution accuracy from 35.9% to 59.4% and logical form accuracy from 23.4% to 48.3%.[4]

2.5 Content Enhanced BERT-based Text-to-SQL Generation (NL2SQL)

It is a simple method to leverage the table content for the BERT-based model to solve the text-to-SQL problem. Based on the observation that some of the table content match some words in question string and some of the table header also match some words in question string, it encodes two additional feature vector for the deep model. It is tested on the WikiSQL dataset and outperforms the BERT-based baseline by 3.7% in logic form and 3.7% in execution accuracy and achieves state-of-the-art.[5]

2.6 Learning a Neural Semantic Parser from User Feedback

The approach is to rapidly and easily build natural language interfaces to databases for new domains, whose performance improves over time based on user feedback, and requires minimal intervention. To achieve this, it adapts neural sequence models to map utterances directly to SQL with its full expressivity, bypassing any intermediate meaning representations. Experiments suggest that this approach can be deployed quickly for any new target domain, as it shows by learning a semantic parser for an online academic database from scratch.[6]

2.7 SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task

In this paper they propose SyntaxSQLNet, a syntax tree network to address the complex and cross-domain text-to-SQL generation task. SyntaxSQLNet employs a SQL specific syntax tree-based decoder with SQL generation path history and table-aware column attention encoders. They evaluate SyntaxSQLNet on the Spider text-to-SQL task, which contains databases with multiple tables and complex SQL queries with multiple SQL clauses and nested queries. They use a database split setting where databases in the test set are unseen during training.[7]

2.8 Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation

In this paper they present a neural approach called IRNet for complex and cross-domain Text-to-SQL. IRNet aims to address two challenges: 1) the mismatch between intents expressed in natural language (NL) and the implementation details in SQL; 2) the challenge in predicting columns caused by the large number of out-of-domain words. Instead of end-to-end synthesizing a SQL query, IRNet decomposes the synthesis process into three phases. In the first phase, IRNet performs a schema linking over a question and a database schema. Then, IRNet adopts a grammar-based neural model to synthesize a SemQL query which is an intermediate representation that we design to bridge NL and SQL. Finally, IRNet deterministically infers a SQL query from the synthesized SemQL query with domain knowledge.[8]

2.9 Grammar-based Neural Text-to-SQL Generation

This work is about sequence-to-sequence paradigm employed by neural text-to-SQL models typically performs token-level decoding and does not consider generating SQL hierarchically from a grammar. Grammar-based decoding has shown significant improvements for other semantic parsing tasks, but SQL and other general programming languages have complexities not present in logical formalisms that make writing hierarchical grammars difficult. They introduced techniques to handle these complexities, showing how to construct a schema-dependent grammar with minimal over-generation. They analyzed these techniques on ATIS and SPIDER, two challenging text-toSQL dataset.[9]

2.10 Model-based Interactive Semantic Parsing: A Unified Framework and A Text-to-SQL Case Study

In this paper, they proposed a new, unified formulation of the interactive semantic parsing problem, where the goal is to design a modelbased intelligent agent. The agent maintains its own state as the current predicted semantic parse, decides whether and where human intervention is needed, and generates a clarification question in natural language. A key part of the agent is a world model it takes a percept and transitions to a new state. Then they propose a simple yet remarkably effective instantiation of our framework, demonstrated on two text-to-SQL datasets (WikiSQL and Spider) with different state-of-the-art base semantic parsers. Compared to an existing interactive semantic parsing approach that treats the base parser as a black box, their approach solicits less user feedback but yields higher run-time accuracy.[10]

3. SYSTEM REQUIRMENTS:

3.1 Functional Requirements:

The functional requirements for a system is to describe what the proposed system can do.

1. The developed system should recognize normal language query as an input.
2. System should allow the input to pass through the various stages of the model.
3. System must provide the quality of service to user.
4. System must provide opt output from the database to the user.

3.2 Non-Functional Requirements:

These are the requirements that are not directly interacted with specific functions delivered by the system.

Performance: The system should perform well as the output should be free of syntax error.

Functionality: This product will deliver on the functional requirements.

Availability: This system will retrieve the relevant output only when the query is a exact match.

Flexibility: It provides the users to load the NL queries easily.

Learnability: The software should be easy to use and reduces the learning work.

3.3 Hardware Requirements:

The minimum requirements to run the software developed as part of this study is summarized in Table 3.1

Processor	Intel Core-i7, 2.8GHz, Windows 11
No of CPU's	4
RAM	16GB

Table 3.1 Hardware requirements for the proposed system

3.4 Software Requirements:

The platform and software frameworks used are tabulated in Table 3.2

Operating system	Windows 11
Development Language	Python 3.7
Deep Learning Framework	TensorFlow
Language processing utilities	NLTK
Third-party libraries	Numpy, Pandas, SciKit

Table 3.2 Software requirements for the proposed system

4. PROPOSED APPROACH :

The proposed approach follows by considering only the query and question column and those are tokenized and let into the word embeddings. These embeddings are then passed to the encoder and decoder layer which also contains attention mechanism. The attention mechanism used here is to provide the help of context vector to the model. The proposed methodology is shown in fig 4.1

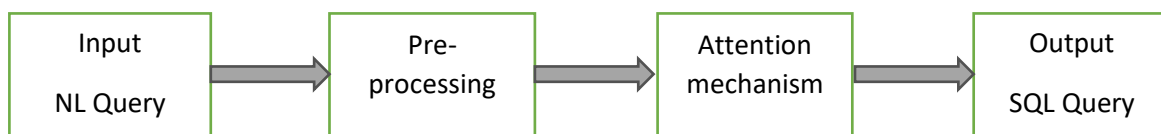


Fig 4.1 Block Diagram of proposed methodology

5. DATASET DESCRIPTION:

The dataset used here is Spider is a large-scale complex and cross-domain semantic parsing and text-to-SQL dataset annotated by 11 Yale students. It consists of 10181 questions and its relevant SQL queries on 140 databases on different domains and their tokens. For our purpose we are only going to use the queries and questions

Dataset	Questions	Databases	Domain	Table/DB
ATIS	5,280	1	1	32
GeoQuery	877	1	1	6
Yelp	131	1	1	7
WikiSQL	80,654	26,521	-	1
Spider	10,181	200	138	5.1

Table 5.1: Comparisons of text-to-SQL datasets.

Spider is the only one text-to-SQL dataset that contains both databases with multiple tables consists from different domains and complex SQL queries. It was well created to test the ability of a system to generalize to not only new SQL queries and database schemas but also new domains. Shown in Table 5.1

[] df							
	db_id	query	query_toks	query_toks_no_value	question	question_toks	sql
0	department_management	SELECT count(*) FROM head WHERE age > 56	[SELECT, count, (, *,), FROM, head, WHERE, ag...	[select, count, (, *,), from, head, where, ag...	How many heads of the departments are older th...	[How, many, heads, of, the, departments, are, ...	{'from': {'table_units': [['table_unit', 1]], ...
1	department_management	SELECT name , born_state , age FROM head ORD...	[SELECT, name, ,, born_state, ,, age, FROM, he...	[select, name, ,, born_state, ,, age, from, he...	List the name, born state and age of the heads...	[List, the, name, ,, born, state, and, age, of...	{'from': {'table_units': [['table_unit', 1]], ...
2	department_management	SELECT creation , name , budget_in_billions ...	[SELECT, creation, ,, name, ,, budget_in_bill...	[select, creation, ,, name, ,, budget_in_bill...	List the creation year, name and budget of eac...	[List, the, creation, year, ,, name, and, budg...	{'from': {'table_units': [['table_unit', 0]], ...
3	department_management	SELECT max(budget_in_billions), min(budget_i...	[SELECT, max, (, budget_in_billions,), ,, min...	[select, max, (, budget_in_billions,), ,, min...	What are the maximum and minimum budget of the...	[What, are, the, maximum, and, minimum, budget...	{'from': {'table_units': [['table_unit', 0]], ...

Fig 5.1: Spider Dataset Overview

6. MODEL ARCHITECTURE:

In this project the text to sql model is designed with the combination of encoder and decoder layers with attention mechanism supported by GRU layer in both encoder and decoders.

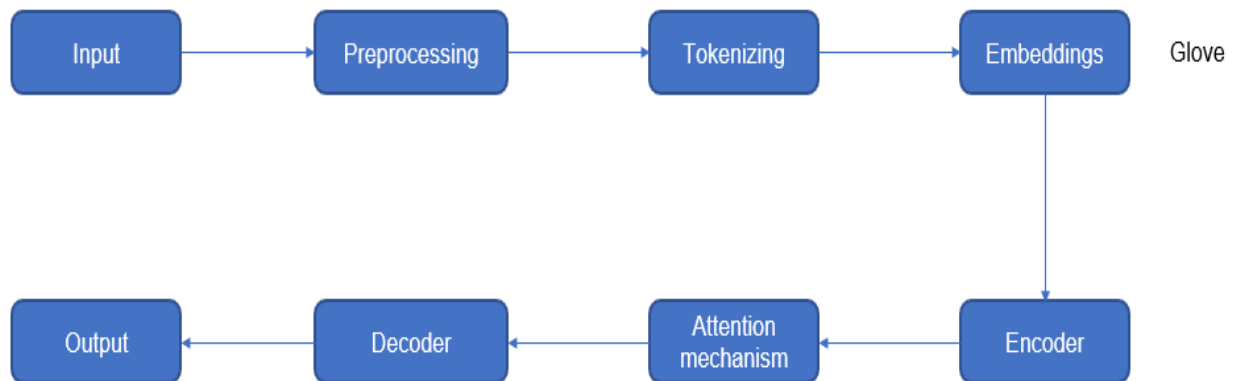


Fig 6.1: Model Architecture

6.1 Tokenization:

Tokenization is one of the most common and important functions when it comes to working with text data. Token making is the division of sentence, paragraph or whole document into smaller units, such as individual words or phrases. Each of these sub-terms is called a token.

For Eg:- Natural language processing is a sentence gets tokenized into ('Natural', 'language', 'processing').

Different methods are used to perform this process such as using Python split() function, Regular expressions, NLTK, Spacy, Gensim, Keras.

In our method we will use the tokenizer from the keras. This class will allow us to vectorize the chorus of text, by changing each text into consecutive numbers (each number is a token index in the dictionary) or a vector where the coefficient of each token can be binary, based on the word count and tf-idf.

Tokenization is an important step in all NLP pipelines. We can't just jump into the model building part without first cleaning the text.

6.2 Embedding:

Numerical representations is the essential one for most of the machine learning models or algorithms. Embedding methods also known as vectorizing converts the symbolic or word representations into a meaningful numerical values that will capture the underlying semantic relations between the words which means that two identical words are represented by almost identical vectors placed very close to the vector area. These are important in solving many of the problems of Natural Language Processing.

Embedding also can be done using different methods namely TF-IDF, bag of words, word2Vec, continuous Bag of words (CBOW), skip-gram, glove and bert embeddings.

In our method we will use the glove embeddings with 6 billion tokens with 50 dimensional vectors. Glove (Global Vectors for Word Representation) is another way to create embedding. Based on matrix factorization techniques in word content matrix. A large matrix of information on co-occurring events is formed and counts for each word and how often we see this word in some cases in a large corpus. Generally, we scan our corpus as follows: in each term, we look at contextual terms in the area defined by the window size before term and the window size after term. Also, we offer a small weightage for very distant words.

The number of "contexts" is large, as it is equal in size. We therefore combine this matrix to form a low-sized matrix, in which each line now produces a vector representation of each word. Typically, this is done by minimizing "loss of reconstruction". These losses attempt to obtain low dimensional presentations that may explain many of the variations in high dimensional data.

6.3 Encoder-Decoder network:

The structure of an encoder-decoder with recurrent neural networks has become the most effective and common way these days to solve many NLP tasks. The encoder-decoder model is a way of arranging common neural networks for sequence-to-sequence prediction problems or sequence based input to provide more detailed predictions.

6.3.1 Encoder:

Encoder is a type of network that encode, that is, extracting features from the given text data. It reads the input sequence and summarizes the information in the so-called context vector, in this GRU network mode, these are called hidden state and cell state vectors. We usually do not consider the output of the encoder and only save internal conditions. This contextual vector aims to contain information for all input features to help the decoder make accurate predictions. The encrypted and cell status of this network is transmitted to the decoder as input.

6.3.2 Decoder:

Decoder as the word means decode, which translates the context vector found in the encoder. The context vector of the encoder codec cell is inserted into the first cell decoder network. Using these initial conditions, the code generator begins to produce an output sequence, and these effects are also considered to predict the future. Stack of a few GRU units where each predicts the output in step-time t . Each recurring unit accepts a hidden state from the previous unit and generates its output and its hidden state to pass through the additional network.

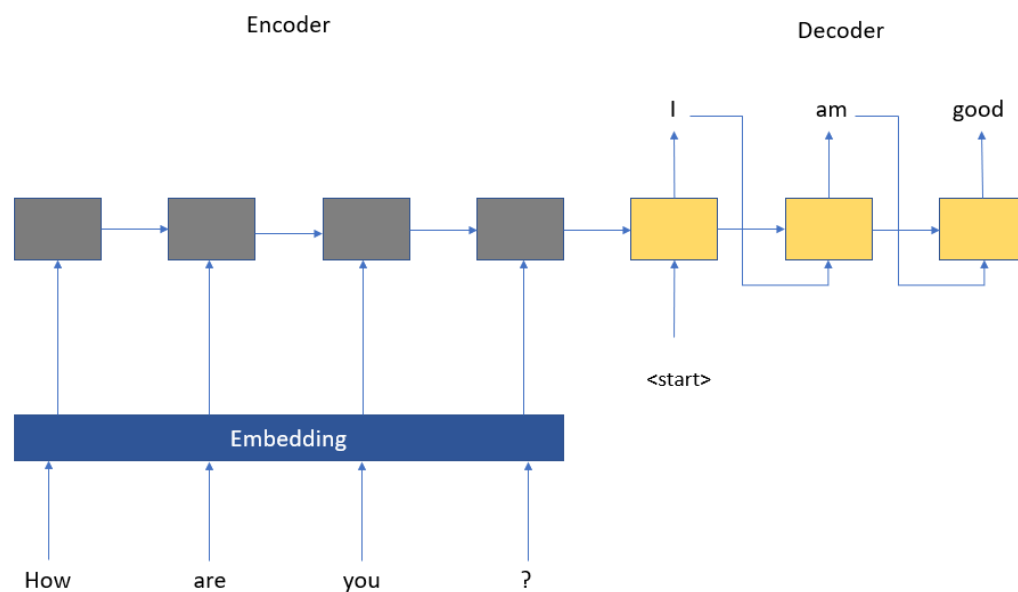


Fig 6.2: Basic sequence to sequence encoder-decoder model.

The main disadvantage for this network is its inability to extract strong contextual meaning and relationships in long semantic sentences, i.e. if a particular piece of long text has a specific context or relationship between its sub-string, the basic seq2seq model with encoder-decoder cannot detect those conditions, therefore it reduces the performance of the model and ultimately tend to decrease accuracy.

Keeping this in mind, further improvements are needed to the existing network model, so an important context definition and relationship can be analyzed and our model can produce

and provide better predictions. There is an approach that will play a major role in determining the context vector.

6.3.3 Attention Mechanism:

Attention is the development of an existing network of sequence-sequence models that address this limit. The simple reason why it is called ‘attention’ is because of its ability to find value sequentially based on contextual meaning and relationships.

First, it works by providing a weighted or multi-character context from the encoder to the decoder and the learning method where the decoder can interpret it was really paying more attention to the following encoding network as it predicts the output each time step in the output sequence.

We can assume that by using the attention method, there is this idea of freeing the existing architecture of the encoder-decoder from the internal representation of the unchanging short length of text. This is achieved by keeping the output in the middle of the GRU network encoder associated with a certain level of importance, from each step of the input sequence and at the same time trains the model to learn and give special attention to these central elements and then associate them with the elements in the output sequence.

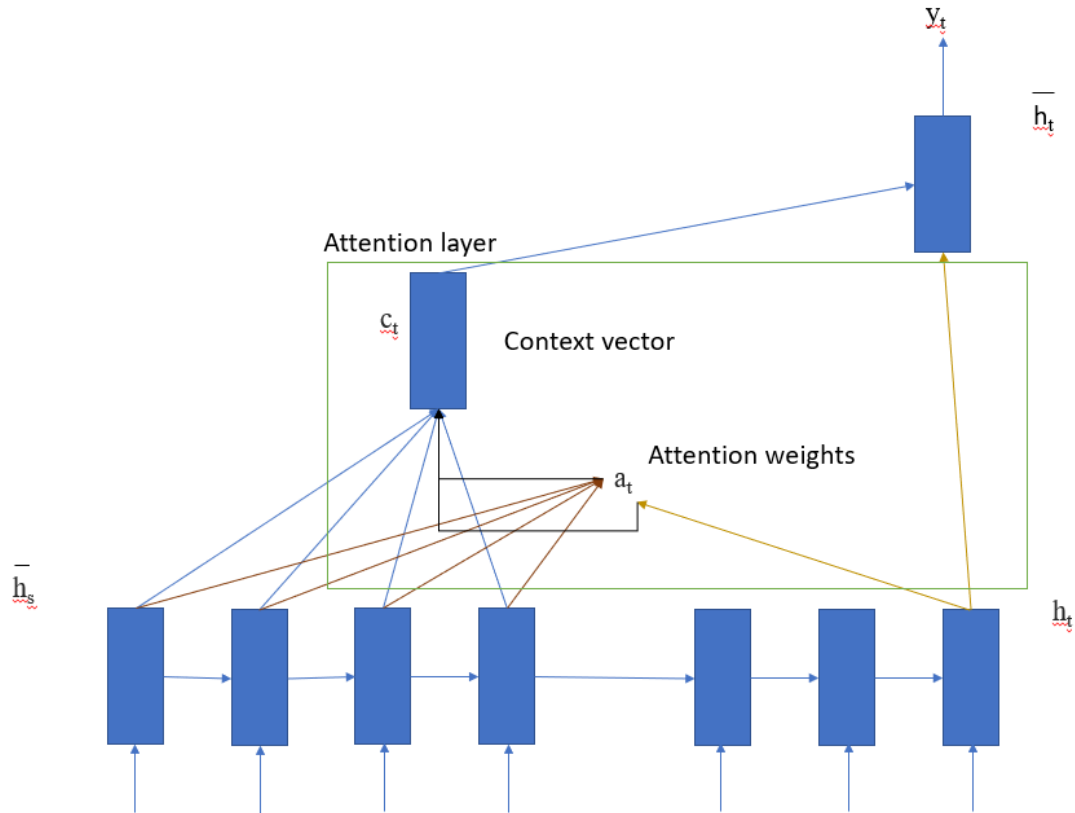


Fig 6.3: Encoder-Decoder with attention mechanism

In simple terms, because of the few factors selected in the input sequence, the output sequence becomes conditional, i.e., combined with weighted issues based on contextual meaning and relationships. These conditions are those contexts, which receive attention and therefore, are constantly trained and predict the desired results.

Sequence to sequence models based on attention require good power of computing resources, but the results are much better compared to traditional sequences to sequence model. In addition, the model is also able to show how attention is paid to the input sequence when predicting the output sequence. This can help in understanding and diagnosing exactly what model is being considered and how much of a certain input output pairs.

Instead of encoding input sequences into a single context vector to move forward, the attention model is trying a different approach. This model strives to develop a context-sensitive vector by selecting each step of the output, so that it can focus and generate specific scores for those relevant and appropriate filters, training our decoder model in full sequence and especially those filters to get predictions.

Let us consider the following points to make this idea clearer.

Attention is proposed as a way of aligning and interpreting a certain long piece of sequential information, which does not need to be of a fixed length. Here, the alignment is a problem in machine translation that identifies which parts of the input sequence are appropriate for each word in the output, while translating is the process of using the correct information to select the relevant output. With the help of attention models, these problems can be easily tackled and provide flexibility in translating long sequence of information.

Let's try to look at the sequence of these processes in the following steps:

1. In encoder-decoder model, the input sequence will be encoded as a single contextual value vector. We will find a context vector that includes the encrypted state and the cell state of the GRU network.
2. The attention model needs access to the output, where it is the context vector from encoder for each time step of input.
3. Decoder input requires to be specified with some initial and ending tags like <start> and <end>. These markers will help the decoder know when to start and stop producing new predictions, while training our model in each timestep.
4. The decoder releases one value at a time, which extends to deeper layers further, before finally providing a prediction (say, \hat{y}) of the current output timestep.

5. The alignment model determines (e) how good each encoded input (h) corresponds to the current output of the decoder (s). Scores calculation requires output from decoder from the previous output time stamp, e.g. $S(t-1)$. If you hit the first output of the decoder, this will be 0.
6. Scoring is done using a function, say, $a()$ known as alignment model.
7. One of the basic methods of this network is to have a single layer network in which each input ($s(t-1)$ and h_1, h_2 , and h_3) is measured. With the help of the mat conversion function, output is also estimated.
8. After getting weighted results, alignment scores are normalized using the softmax function. The creation of standard points helps them to be treated as opportunities, indicating whether each encoded time step (annotation) is likely to correspond to the current output time step or not. These common points are called annotated weights.
9. After getting the weight of an annotations, each annotation, (h) multiplies by the weight of the annotation, say, (a) producing a new contextual vector in which the current output time step could be decoded. The context vector received thus is a weighted amount of annotations and standard alignment scores.
10. Finally, the decoding is done according to the output encoder decoder model, using the destination context vector for the current time step. This context vector can be embedded into deep neural networks to effectively learn and extract additional features, before receiving final predictions

Unlike the seq2seq model without attention, we have used a contextual value vector for all decoder timesteps but in the case of the attention mode, it generates a context vector at all times with a timeline of words sorted by their respective scores.

Because of the attention-based models, contextual relationships are more widely used in and the performance of the model appears to be much better compared to the basic seq2seq model, given the high power computing capacity.

Using word embedding may help the seq2seq model achieve some improvement with limited computational power, but long sequences with heavy contextual information may not be well trained.

7. WORKING MODEL:

The model comprises of three layers which are listed below.

1. Encoder layer
2. Attention layer
3. Decoder layer

Example : How many students are there? -----> select count(*) from students

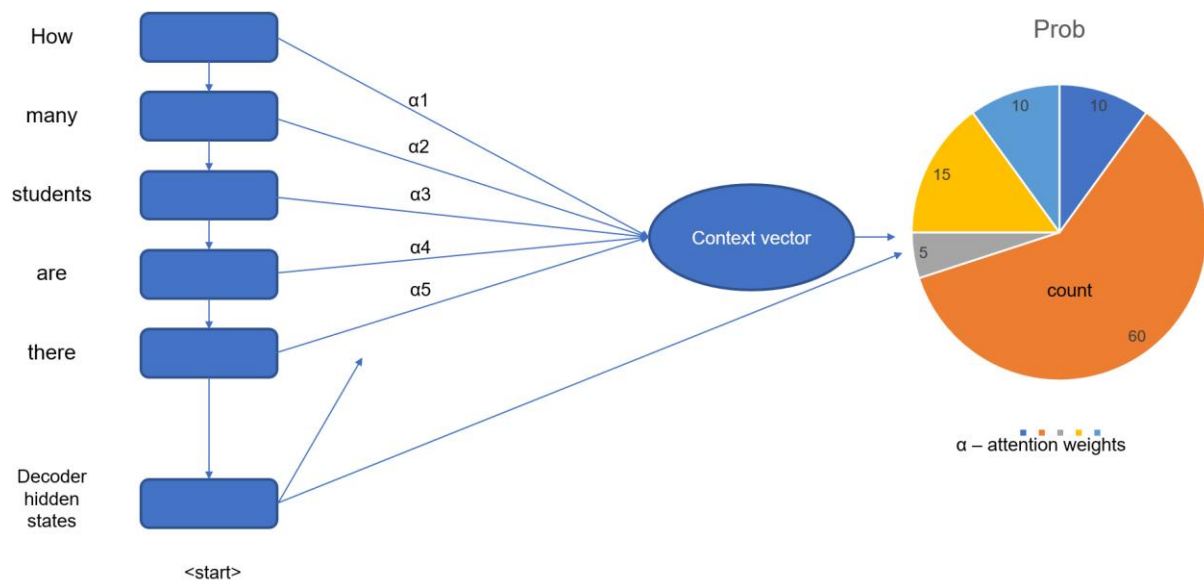


Fig 7.1: Working model of Seq2SQL

8. IMPLEMENTATION:

The attention algorithm of Luong et al. performs the following operations[11]:

1. The encoder generates a set of annotations, $H = \mathbf{h}_i, i = 1, \dots, T$, from the input sentence.
2. The current decoder hidden state is computed as: $\mathbf{s}_t = \text{GRU}_{\text{decoder}}(\mathbf{s}_{t-1}, \mathbf{y}_{t-1})$. Here, \mathbf{s}_{t-1} denotes the previous hidden decoder state, and \mathbf{y}_{t-1} the previous decoder output.
3. An alignment model, $a(\cdot)$ uses the annotations and the current decoder hidden state to compute the alignment scores

$$e_{t,i} = a(\mathbf{s}_t, \mathbf{h}_i)$$

4. A softmax function is applied to the alignment scores, effectively normalizing them into weight values in a range between 0 and 1

$$\alpha_{t,i} = \text{softmax}(e_{t,i})$$

5. These weights together with the previously computed annotations are used to generate a context vector through a weighted sum of the annotations

$$\mathbf{c}_t = \sum_{i=1}^T \alpha_{t,i} \mathbf{h}_i$$

6. An attentional hidden state is computed based on a weighted concatenation of the context vector and the current decoder hidden state

$$\tilde{\mathbf{s}}_t = \tanh(\mathbf{W}_c[\mathbf{c}_t; \mathbf{s}_t])$$

7. The decoder produces a final output by feeding it a weighted attentional hidden state

$$\mathbf{y}_t = \text{softmax}(\mathbf{W}_y \tilde{\mathbf{s}}_t)$$

8. Steps 2-7 are repeated until the end of the sequence.

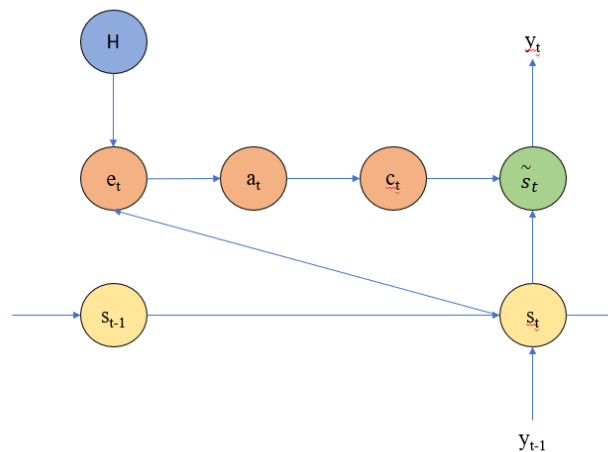


Fig 8.1: Luong-attention

9. EVALUATION METRICS:

The evaluation metrics used are component matching and exact matching. The reason for selecting this metric is that the baseline model uses this metric and hence better comparison can be provided. We compute the F1 scores for each component separately. F1 score combines precision and recall and can be calculated as

$$F1 = 2 \times (Precision \times Recall) / (Precision + Recall)$$

Where,

$$Precision = True\ Positive / (True\ Positive + False\ Positive)$$

$$Recall = True\ Positive / (True\ Positive + False\ Negative)$$

Precision is a good metric to use when the cost of false positives is high. It tells how many are actually positive from the predicted positive. On the other hand Recall is a good metric to be used when the cost of false negatives is high. F1 score tries to balance the two and hence is a better metric to be used when the classes are imbalanced. F1 score ranges from 0 to 1, with 1 being the score for a model with perfect accuracy. One of the disadvantages of F1 Scores is that the scores for each component are computed independently. It does not consider the mutual information between these components. Thus, we use a second evaluation i.e Jaccard similarity matching score which compares the entire translation with the label. If all the components match then then score is 1, otherwise the score is 0. The Jaccard similarity is also called as the Jaccard index and the Intersection over Union. The Jaccard similarity metric used to determine how similar the words between two text documents and closer together in their context i.e. how many common words exist on top of the total text. One more Performance metrics we are adding here is Rouge1 score. Rouge measures with the help of recall: how much the words and/or n-grams in the human reference summaries appeared in the machine generated summaries.

---Evaluation Metrics on Training Dataset: ---

Precision: 0.7929

Recall: 0.8192

F1_score:0.8059

Rouge1 Similarity Score: 0.8563

Jaccard Similarity: 0.8525

---Evaluation Metrics on Validation Dataset: ---

Precision: 0.7924

Recall: 0.8213

F1_score:0.8066

Rouge1 Similarity Score: 0.8561

Jaccard Similarity: 0.8562

Fig 9.1: Evaluation Metrics

10. RESULTS:

```
translate("How many students are there?", preprocess=True)
```

```
'select count ( * ) from student'
```

```
translate('What are the ids of the students who registered for course 301?', preprocess=True)
```

```
'select student_id from student_course_attendance where course_id = 301'
```

```
translate('Find the name of all students who were in the tryout sorted in alphabetic order.', preprocess=True)
```

```
'select t1 . pname from player as t1 join tryout as t2 on t1 . pid = t2 . pid order by t1 . pname from player as t1 join tryout as t2 on t1 . pid = t2 . pid order by t1 . pname from player as t1 join tryout as t2 on t1 . p  
id = t2 . pid order by t1 . pname from player as t1 join tryout as t2 on t1 . pid = t2 . pid order by t1 . pname from player as t1 join tryout as t2 o  
n t1 . pid = t2 . pid order by t1 . pname from player as t1'
```

Fig 10.1: Results from the model

11. CONNECTING TO THE DATABASE:

SQLite3 can be integrated with Python using the `sqlite3` module. We don't need to install this module separately because it is shipped automatically with Python version 2.5.x and up. To use this `sqlite3` module, you must first create an interactive representation of the database and then optionally create a cursor object, which will assist you in executing all SQL statements.

`sqlite3.connect()`:

This API opens a connection to the SQLite database file. When the site is successfully unlocked, it returns the connection object.

```
[ ] model_output = translate("How many students are there?", preprocess=True, plot=False)
    model_op2 = model_output
    model_op2
```

```
'select count ( * ) from student'
```

```
[ ] my_conn = create_engine("sqlite:///content/drive/MyDrive/text2sql-main/text2sql-main/spider/spider/database/dorm_1/dorm_1.sqlite")
```

```
[ ] r_set = my_conn.execute(model_op2)
    for row in r_set:
        print(row)
```

```
(34,)
```

```
[ ] model_output = translate("How many acting statuses are there?", preprocess=True, plot=False)
    model_op = model_output
    model_op
```

```
'select count ( distinct temporary_acting ) from management'
```

```
[ ] from sqlalchemy import create_engine
```

```
[ ] my_conn = create_engine("sqlite:///content/drive/MyDrive/text2sql-main/text2sql-main/spider/spider/database/department_management/department_management.sqlite")
```

```
[ ] r_set = my_conn.execute(model_op)
    for row in r_set:
        print(row)
```

```
(2,)
```

Fig 11.1: Output from the database

12. CONCLUSION AND FUTURE WORKS:

In this project, we have documented our implementation for sequence to SQL using the Model built with encoder-decoder with attention layer, the best results of which were obtained by using Luong attention. We constructed a model and each module was an independent decoder. The decoding process can be viewed as recursive calls to these modules. The dataset used was Spider.

Since the process is in research stage many adaptations and further experiments can be performed to improve the performance of this model. A pre-trained word embedding such as BERT can be used instead of training the embedding with Glove. Future work concerns constructing syntax trees for nested complex queries and still have more room for improvement. The model must also be tested on some of the complex text to SQL datasets like Atis, Geoquery, etc.,.

13. REFERENCES:

- [1] Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S. and Zhang, Z., 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.
- [2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- [3] Xu, X., Liu, C. and Song, D., 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.
- [4] Zhong, V., Xiong, C. and Socher, R., 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.
- [5] Guo, T. and Gao, H., 2019. Content enhanced bert-based text-to-sql generation. *arXiv preprint arXiv:1910.07179*.
- [6] Iyer, S., Konstant, I., Cheung, A., Krishnamurthy, J. and Zettlemoyer, L., 2017. Learning a neural semantic parser from user feedback. *arXiv preprint arXiv:1704.08760*.
- [7] Yu, T., Yasunaga, M., Yang, K., Zhang, R., Wang, D., Li, Z. and Radev, D., 2018. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. *arXiv preprint arXiv:1810.05237*.
- [8] Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J.G., Liu, T. and Zhang, D., 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. *arXiv preprint arXiv:1905.08205*.
- [9] Lin, K., Bogin, B., Neumann, M., Berant, J. and Gardner, M., 2019. Grammar-based neural text-to-sql generation. *arXiv preprint arXiv:1905.13326*.
- [10] Yao, Z., Su, Y., Sun, H. and Yih, W.T., 2019. Model-based interactive semantic parsing: A unified framework and a text-to-SQL case study. *arXiv preprint arXiv:1910.05389*.
- [11] Luong, M.T., Pham, H. and Manning, C.D., 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.