SPEECH TO VISUALIZATION USING TRANSFORMER BASED DEEP LEARNING MODEL

MSc Data Analytics
Research Project

Vishal Gajanan Patwardhan
Student ID: X18190839

School of Computing
National College of Ireland

Supervisor: Jorge Basilio

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vishal Gajanan Patwardhan		
Student ID:	X18190839		
Programme:	Msc. Data Analytics	**Year:**	2021-2022
Module:	Research Project		
Supervisor:	Jorge Basilio		
Submission Due Date:	14th Sep 2022		
Project Title:	SPEECH TO VISUALIZATION USING TRANSFORMER BASED DEEP LEARNING MODEL		
Word Count:	**9761**	**Page Count: 22**	

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Vishal Gajanan Patwardhan

Date: 14th September 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

SPEECH TO VISUALIZATION USING TRANSFORMER BASED DEEP LEARNING MODEL

Vishal Gajanan Patwardhan
X18190839

Abstract

Nowadays, so many studies are being carried out to ease the use of the systems to make them touchless. In this research, a speech-driven system called Speech-to-Visualization draws the visualizations to gather insights from the specified database by speaking natural language questions with the chart's information. To support this research, the nvBench dataset supports the natural language to visualization tasks. Using the Google speech recognizer, speech is converted to text and processing it to train the transformer-based Seq2Seq model with the forcing of attention and SQL-Aware translation module to help predict the SQL query components (select column, aggregate function, aggregate column, table name, where condition, order by column, order by type, group by column and limit value) and form the SQL query and also the information of chart types like a bar graph, scatterplot, line graph and pie. The predicted query is then executed on the SQLite3 database to get the desired structured data, i.e., rows and columns and transformed to feed to the matplotlib library of python's visualization library to draw the chart. The quantitative evaluation shows that the Seq2Seq model with the forcing of attention achieves the overall accuracy of 77% using the nvBench dataset.

1 Introduction

There are countless numbers of researchers working towards the goal of learning how to teach machines to think and react like humans. Voice assistants like Google, Apple's Siri, Amazon's Alexa and many other apps are a few fantastic examples of software that performs tasks by speech. This shows how the deep learning approach has progressed to build such systems. Speech-based systems enable disabled people, sometimes known as users without hands, to use the system in the same manner as regular users, with no discernible change in the system's usefulness for either group of users.

Almost all organisations have databases to store their daily business data, which is a considerable amount, including data like customers, employees, transactions and other data, which are very important for a business analyst to identify their business trends. This management of databases needs a strong knowledge of SQL. SQL, Structured Query Language, is a language to talk to the database. In relational databases, data is stored in a structured format, for instance, tables, rows, and columns. Such information needs to be correctly extracted, transformed, and loaded into visualization tools which draw the visualization just in a second. This states that drawing visualization requires good database understanding and solid skill to write SQL queries, which is not viable for a layman and disabled person. This inspires the fact to put into action a system that can translate the natural language question posed to the machine into valuable insights gleaned from visualizations.

This research uses Python's SpeechRecognition library to use the Google Speech recogniser. Google speech recognizer is a free API, easy to incorporate in the code, which enables accurate recognition of text derived from the speech, which is passed to the Seq2Seq model to predict the SQL components.

In the past, some systems were used to convert Text to SQL, called Natural Language Interfaces for Databases (NLIDB) systems. These systems were built using a rule-based methodology. In addition, the processing of natural language became one of the most intriguing areas of study. This resulted in the development of many advanced deep learning models including Seq2SQL (He, Bai and Jiang, 2019) and (Zhong, Xiong, and Socher, 2017), BERT, which was utilized by (Hui et al., 2021), (Zheng et al., 2022), (Hui et al., 2020), (Chen et al., 2021), (Guo and Gao, 2020), and (Kim and Lee, 2021) and Sequence-to-Sequence, i.e. Seq2Seq was employed by (Luo et al., 2022), (Xu and Singh, 2017), (Utama et al., 2018), and (P. Wang et al., 2020) demonstrate successful results in the process of transforming natural language to SQL query generation. The performance of these models is compared, and it is determined that Seq2Seq performs better than the other models, making it the most potent model in the context of SQL. nvBench is an extensive dataset specially created to support Natural Language to Visualization tasks. This dataset is used to train the Seq2Seq model. For this research, a code was adopted by (Luo et al., 2022), which supported natural language in visualization. Still, the visualization was performed using the Vega Zero query; for this research, the code was modified to get the SQL query as the output and to tune the hyperparameters carefully for better results.

1.1 Motivation

The development of an intelligent virtual assistant is the goal of this line of research. It will be implemented to aid the impaired user with hand-related disabilities and layman users to draw the visualization using their speech most straightforwardly and productively. This is a challenge or creates a significant challenge in the organisation because there are very few layman users or employees with disabilities who do not know database and visualization tools, so it is necessary to research this topic. Also, there are no such software or systems with simple accessibility features that would help these users learn. So, this research project aims to understand these challenges and allow these users to draw the visualization just by understanding the user's natural language question.

1.2 Research Question

How can a system help interpret a disabled or layperson's question in the form of speech to draw the visualization from the database data by executing a synthesized SQL query?

1.3 Contribution of the Research

There is much recent research where there has been text-to-SQL and text-to-visualization translation, but there is minimal research where speech-to-visualization is implemented. In this paper, an approach to training the machine learning model with a cross-domain nvBench dataset to formulate the complex SQL query using the sketch technique to execute on the SQLite3 database is performed. It further transforms the database output to draw the visualization using python's matplotlib library for visualizations.

The structure of the paper is outlined as follows: In section 2, the literature review explains various models by categorizing them by rule-based and deep learning methods. Section 3 describes the research methodology. Section 4 presents the design specification of the speech-to-visualization architecture. Section 5 explains the implementation of this research. Section 6 defines the evaluation criteria of the research study. Section 7 states the conclusion & future work of the research.

2 Literature Review

Natural Language Processing (NLP), a booming topic, inspires many researchers and data enthusiasts to experiment with and practice fantastic novel ideas that will contribute to society and minimize the need for a workforce. Natural language processing is a machine learning task which involves the processing of audio or text gathered from any environment to learn the different characteristics of the environment to think like a human and react to it. There are vast numbers of experiments or research being conducted in this field. Some research in natural language processing involves domain-specific tasks involving domain-specific semantic parsing, such as translation tasks. To support this project, some previous researchers conducted the study to convert text-to-SQL, but minimal research was performed to correct speech to visualization. So, breaking the literature review into two parts, i.e., End-to-End Speech-to-SQL models and fundamentals of Speech-to-Visualization.

2.1 Rule-Based Speech-to-SQL Models

There are three critical components to implementing Speech-to-SQL: 1) Method to convert speech to text, 2) parsing the text and mapping it to the database schema to create SQL, and 3) finally getting the data from the database using the generated SQL queries. Classical and Deep Learning methods can be used to accomplish this task.

2.1.1 Automatic Speech Recognition (ASR)

To convert speech to text, a system needs an Automatic Speech Recognition (ASR) engine (Kumar, Kumar, Mitra and Sundaram, 2021). Researchers improved the quality of their ASR by using both an acoustic and a language model. This allowed them to transform speech more accurately into the appropriate text. The Hidden Markov Model (HMM) was the source of inspiration for the acoustic model, which can also be referred to as a phonetic model in more straightforward terminology. HMM can identify the proper Y with the outcome of X by watching the state of X in any generic representation and using the information. The smallest unit of sound that makes up a word and differentiates between two terms of varying lengths when it comes to pronunciation is called a phoneme. For example, the sound /f/ in fire and the sound /w/ in the wire have a reasonably similar accent, yet there is a slight distinction between them because of the /f/ and /w/ sounds. During the training phase, the supervised learning technique was employed to help construct the acoustic model, which was then utilized to detect the proper probable spoken word readily. The Language Model and the Hidden Markov Model (HMM) were merged to improve the accuracy of the spoken word (Kumar, Kumar, Mitra, and Sundaram, 2021). At the same time, Microsoft's advanced search and dictation model uses Azure's Custom Speech Service, which provides a dataset of spoken phrases that help to train their ASR engine. This was accomplished by using Azure's Custom Speech Service. (Shah, Li, Kumar, and Saul, 2020) were able to recognize and maintain the proper grammar sequence in the speech using a language model. Following the processing of the speech signals by the ASR, these are then sent on to the subsequent model, which is responsible for the tokenization and translation procedures.

2.1.2 Input Tokenization: The Conversion of NL to Machine Language

Input Tokenisation is a method of converting a word into machine language to aid in comprehending complex sentences. To fully grasp the meaning of the text, it is necessary to translate and analyze the native language's syntax and semantics. NL (Hiregoudar, K. G., 2018) was turned into a stack of tokens unneeded ones like "an," "a," and "the" were omitted from the list of tokens. The synonym database technique was used to map the remaining tokens to the database schema's table names and characteristics. Tokens are divided into three categories: keywords, special characters (Spl-Chars) and literals, according to findings by (Shah et al. 2020). SELECT, FROM, and similar expressions are the most commonly used keywords. In the SQL syntax, Spl-Char comprises a limited number of elements. Also included in Literals are the table's name, an attribute, and any associated values. Using a grammar-based approach, the authors (Kumar, Kumar, Mitra, and Sundaram, 2021) processed English-based text for the Text Translator. Their solution included a lexical analyzer, parser, and table finder that enabled them to transform Natural language queries into SQL queries. As a result of Lexical Analyzer, they accomplished two key activities: mapping source words to target phrases and sending them to the ground truth phrase. By eliminating the meaningless words, the lexical analyser could focus on translating tokens for the essential and relevant target words.

2.1.3 Translation Techniques in Text-to-SQL

The process of converting the generated text into a semantically meaningful language is known as text-to-SQL translation. To generate a syntactically valid SQL query, Structure Determination (Shah, Li, Kumar, & Saul, 2020) created an intermediate model that takes input from the ASR engine and uses a placeholder for the literal while preserving Spl-Char and keywords. We can break down the Structured Determination into its two constituent parts. Literal masking and SplChar handling. Words like "select", "from", "where", "order", "group", "natural", "and", "or", "not", "limit", "between", "in", "sum", "avg", and "min" are all examples of dictionary names that may be found in SplChar Handling. Characters like [* = > ().,] and [= > ().,] can be found in SplCharDict, and vice versa. These enhancements made it simpler to convert text into SQL. Translation helped the tokenization's improved input using clause extractor & Mapper (Hiregoudar, Gonal, & K. G., 2018). Clause Extractor takes in tokenized text and generates an initial query. Afterwards, the Mapper employs the query to match the table names and attributes of the database structure. After tokens have been mapped to database schemas, the output text is sent to SQL queries.

2.1.4 Structuring of SQL query

An SQL query must be a grammatically correct statement to succeed. The issue may occur if the SQL queries are incorrectly spelt. (Shah, Li, Kumar, and Saul, 2020) A valid SQL query is generated at the end of the process using SQL Grammar and SplChar Handling. In Structure Determination, the Literal Determination feature is used to substitute the actual table name and characteristics for the query templates. English text queries are checked to see if they are syntactically correct using a tool called the Syntactical Analyzer (Kumar, Kumar, Mitra, and Sundaram, 2021). The grammar rules, also known as production rules, are used to construct a parse tree to accomplish this. A method of syntax-directed translation is used to help with semantic analysis when implementing the tree. To generate appropriate SQL queries, the English query is transformed into an intermediate representation using syntax-directed translation. Assess the need for natural joins when constructing a JOIN query using a tool like a Mapper (Hiregoudar, Gonal, & K. G., 2018). Data stored in tables is retrieved from the database via a SQL query.

2.2 Basic Concepts of Speech-to-Visualization Using Deep Learning Approaches

Speech-to-Visualization being almost new research, can be implemented by breaking the research into small components such as 1) Building automatic speech recognition, 2) Parsing text into semantic SQL queries, and 3) Drawing the visualization by fetching the data from the databases based on SQL query generated in the second component. The below sub-points explain various literature reviews on the same.

2.2.1 Speech Recognition

A computer program that can understand spoken language and transform it into text is referred to as "Speech Recognition," which is frequently abbreviated as "Automatic Speech Recognition" (ASR). According to (Basystiuk et al., 2019), new voice recognition techniques and approaches will become one of the finest revolutionary technologies that will minimize a lot of communication time that is presently not in texting but may be done through audio or speech. This will make speech recognition among the best emerging solutions that will become one of the best innovative technologies. (Basystiuk et al., 2019) explores the many advantages and disadvantages connected with well-known chatbots. This study compares machine learning Python libraries, such as Keras and PyTorch. The libraries are evaluated on several criteria, including their learning times, their loss efficiencies, and their translation accuracy. To accomplish the same results, they use a model called Sequence-to-Sequence, which is founded on the RNN. RNN achieves the greatest results when applied to data in which the sequence is a primary distinguishing characteristic. They described the method in terms of its temporal complexity in comparison to the one that was already accessible. And it was believed that RNN development will be one of the top models capable of detecting human voices and carrying out tasks based on the instructions that they provide. According to the results, the Keras Library is more efficient on the dataset that they used.

Traditional automatic speech recognition (ASR) systems are replaced by attention-based encoder-decoder designs such as Listen, Attend, and Spell (LAS) (Chiu et al., 2022) a group of Google scientists. However, it wasn't apparent if such architectures could be used effectively for tasks like voice search or dictation in the past. This study investigates a wide range of structural and optimization enhancements to their LAS model with the goal of achieving much better results. They demonstrate that word piece models can be employed in place of graphemes in the structural context. A multi-head attention architecture is also introduced, which improves upon the standard single-head attention. There are several techniques for improving performance, including synchronous training, planned sampling, label smoothing, and minimization of the word error rate. For streaming recognition, they used a unidirectional LSTM encoder. This model's WER is 4.1% on dictation tasks compared to 5.0% for conventional systems, which was tested over 12,000 hours of voice search time. The proposed adjustments improved the WER from 9.2% to 5.6 percent whereas the best conventional system only managed 6.7%. This show that ASR is one of the common approaches while converting the speech-to-text.

2.2.2 Text Translation to Semantic SQL Queries

2.2.2.1 Sequence-to-Sequence (Seq2Seq) Models

Seq2Seq models are a subset of Recurrent Neural Network designs that are commonly used to address complex language issues such as machine translation, text summarization, developing Chatbots, question answering, and so on. NL2VIS (Luo, Tang, and Li, 2021) is one of these systems that introduced a novel approach for translating natural language to visualization through the construction of a SQL query. To perform well in the cross-domain natural to Visualization job, nvBench was developed from the (NL, SQL) benchmark, and it has 25000 (NL, VIS) pairings spread over 750 tables covering around 100 domains and a total of seven different kinds of charts. The results of the nvBench dataset training show that this will catapult NL2VIS to new heights (Luo, Tang, and Li, 2021). In their recent study, show how the NL2VIS benchmark may be implemented with the help of the nvBench dataset. The self-attention mechanism block is placed in between the encoders and decoders in the old transformer based Seq2Seq model. This model is implemented using encoders and decoders. Encoders take natural language as their input and break it down into a more understandable representation known as h. But decoders take h as input and provide a string of results. There was a lot of training data used to teach ncNet, and it was all presented as input-output pairs. This model performs exceptionally well in natural language to visualization task.

(Utama et al. 2018) developed a sophisticated Natural Language Interface called DBPal that makes use of the Seq2Seq model and is better able to paraphrase and different linguistic variants when translating natural language text to SQL. It is also the goal of DBPal to rephrase a user's query regardless of the database's architecture. Researchers were interested to look at pre-existing text-to-SQL models after learning that many of them are LSTM-based and have attention mechanisms (He, Bai, & Jiang, 2019). After developing Seq2SQL, SQLNet, and SQLNet with BERT word embedding, they compared their respective levels of accuracy. After discovering that medical professionals have challenges when trying to obtain patient data from EMRs, researchers developed a deep learning-based TRanslate-Edit Model for Query-to-SQL (TREQS) that can rapidly translate the question into a SQL database. This is according to (Wang, Shi, & Reddy, 2020). In order to determine which Seq2Seq learning model was most effective in a real-world database environment, a team of researchers (Xu and Singh, 2017) examined many of them. The Seq2SQL method was created by employing in-loop query execution for both data learning and SQL query construction (Zhong, Xiong, and Socher, 2017).

One was the DBPal system's Neural Query Translation, while the other was its Interactive Auto-Completion (Utama et al., 2018). When it comes to Neural Query Translation, DBPal conducts some pre- and post-processing by managing all parameters, modifying SQL queries, and expanding the query language to include JOINS. The Seq2Seq model uses interactive auto-completion to suggest the next sequence word as the user formulates their question in the interface. The aggregate operator, SELECT, and WHERE clauses were the three key components of SQL in the Seq2SQL model used by (He, Bai, and Jiang, 2019), which was inspired by (Zhong, Xiong, and Socher, 2017). The table query and SQL vocabulary are encoded using a bi-LSTM encoder and fed into a two-layer bi-LSTM model as input. These three parts were predicted using an LSTM network, a linear network, and a softmax network, with a cross-entropy loss applied to the first two and a policy gradient added to the third. The researchers also used SQLNet (He, Bai, and Jiang, 2019) to help with the slot filling method. Using WikiSQL's SQL queries, they were able to foresee the select column, aggregator, operation, number of where conditions, and value in the condition. (Wang, Shi, and Reddy, 2020) used the publicly available MIMIC dataset to create the first MIMICSQL question-SQL pair because no such dataset existed in the health sector. When the TREQS model was run on MIMICSQL, the recover approach was able to construct the desired SQL query. For this task, (Xu and Singh, 2017) used the WikiSQL dataset and implemented pre-processing, the vanilla Seq2Seq model, the Bidirectional Sequence 2 Sequence model, the Reversed Sequence 2 Sequence model, the Pointer Attention model, the Attention Seq2Seq Model. Bidirectional Seq2Seq was shown to be the most accurate in training and validation (Xu and Singh, 2017) among these models, with a score of 81% and 56% respectively. The Seq2SQL model also achieved state-of-the-art performance as a semantic parser on the WikiSQL dataset (Zhong, Xiong, and Socher, 2017).

2.2.2.2 BERT Based Models

BERT, or Bidirectional Encoder Representation from Transformers, is a deep learning model that uses bidirectional encoders. For every input and its corresponding determined weight, BERT indicates that all output results are connected. The Text-to-SQL schema dependency study was built with the help of the Python machine learning tool PyTorch (Hui et al., 2021). Tokenization was performed first using CoreNLP. Language annotations may be added to text, tokens, parts of speech, numbers, and timestamps with the aid of Core NLP. Immediately after CoreNLP tokenization, WordPiece tokenized the text. WordPress is a subword segmentation technique primarily used in natural language processing (NLP). To make input more readable, the BERT-large-uncase version was utilized, and the learning rate was adjusted to 1e-5. An incompatibility between the logic of plain language and the logic of SQL has prevented prior attempts to incorporate context-dependent information from interaction history or previously planned SQL queries (Zheng et al., 2022). The solution they came up with to this problem was a text-to-SQL model known as History Information Enhanced (HIE-SQL), which could incorporate context information from earlier utterances as well as the most recent project SQL query. Four parts make up the HIE-SQL system. For NL and SQL, we have the Multimodal Encoder. A pre-trained bimodal SQLBERT encoder encodes normal language as well as structured data from SQL servers. schema-linking relations in HIE-Layers are used to encode all of the language's output elements. The SQL query is generated here. There are challenges in implementing standard semantic parsing of natural language into SQL while setting up text-to-SQL across domains. In order to circumvent this issue, ShadowGNN suggests doing a more abstract and semantic examination of database structures (Chen et al., 2021). Because the entities of the semantic objects in the database are eliminated in the graph-based projection of query and schema, the schemas are well-designed. Then, within the framework of domain independence, a relation-aware transformer is used to build the logical connections between the query and the schema. We use the SQL decoder to piece together the actual SQL statement. Using unlabeled database contents, Text-to-SQL models may be trained to solve the header-column alignment difficulty (Kim and Lee, 2021). This provides important insight on the header span alignment. This is evident when comparing their model's performance to that of baseline models. Despite the fact that the Graph-based encoder was employed by ShadowGNN (Chen et al., 2021), the syntax produced by (Hui et al., 2020) was not satisfactory. For Text-to-SQL, a new model called as S2SQL, or Syntax to Question-Schema Graph Encoder, was developed with the goal of improving efficiency by collecting the syntactic information included in the query. Using the dataset, they observed that S2SQL + RoBERTa performed 2.8% better than RAT + RoBERTa in terms of accuracy, according to their research.

2.2.3 Conversion of Structured Data to Visualizations

Accessing the data in an RDBMS requires valid SQL queries. Depending on the level of filtering, it takes very little time to return the structured data requested by the user after running a valid query. Rows and columns define the structure of database-stored data. Depending on the user's needs, this data can be used in a different way.

Data from the COVID-19 database was utilized to create a use-case, and four features were considered (Luo et al., 2021). Kevin, who had previously worked on COVID-19's dashboards, was enlisted to help them test their use case and see how the user produces visualizations from the data. ncNet was imported first, and then the model parameters were passed in. He then went through the dataset by executing show_dataset() to gain some insights. They utilized the natural language question and looked at the visualization from the ncNet object using the nl2vis(nl question) method. A length of time was spent rephrasing his natural language as a result of the erroneous conclusions, and he eventually learned to correctly visualize with the aid of the Vega-Lite code2. Data processing, analysis, and visual transformation are all made possible using Vega-client-side Lite's application. It also lets you combine charts based on your decisions. Natural language to visualizations (NL2VIS) has been shown to perform best when trained on the nvBench dataset (Luo, Tang, and Li, 2021) by their use cases.

The data retrieved from the database using the prepared SQL query should be visualized in accordance with the research topic. Visualization speeds up the process of grasping and making sense of large amounts of data. If you don't have access to a visual representation of your data, you may miss out on patterns, correlations, and trends (J and Wijk, 2018). There are several Python tools available for use in making visualizations, including Matplotlib and Seaborn. Many other kinds of charts, including bar charts, line charts, pie charts, and scatter plots, may be made with the help of these libraries. (J and Wijk, 2018), making it simpler, more quickly and effectively to gain knowledge of the data. Every chart type has its own built - in functions in the visualisation library. In addition, the data for each predefined function must be provided in accordance with the type of input variable data. Any CSV, Excel, XML or any other organized or unorganized type of data can be used as an input parameter to the function. If you want to work with data like spreadsheets, SQL tables, or lists of series objects, then a DataFrame is what you need. If you want to generate DataFrame objects, you'll need to use a python package called Pandas. Many various visualizations were possible utilizing the structured data as a result of these methods (J & Wijk, 2018).

2.3 Research Niche and Contribution

The task of speech to visualization can be accomplished with either the rule-based method or with deep learning mthods. Evaluation and precision in creating a SQL query show that the deep learning method is superior. Besides (Luo, Tang, and Li, 2021) there wasn't much written on Speech-to-Visualization, therefore it's a promising area for future study. After doing some preliminary study on the matter, I became interested in the Text-to-SQL method, in which a question posed in plain language is converted into a SQL query and then executed to get the relevant database knowledge. Two more questions arose from the study topic itself: 1) can the question asked in a natural language be transformed into text, and if so, how? and 2) the means by which the tabular information stored in a SQL database may be represented graphically. Based on the literature study, we learned that most deep learning research and implementations have trained their models on the WikiSQL and Spyder Benchmark dataset or the nvBench dataset. Each of these data sets has a corresponding SQL query that is meant to be used with Natural Language queries. In contrast, WikiSQL's offered dataset only includes SQL queries that return a scalar result, rather than a table of values. Because a single number output isn't enough information for data visualization, the dataset must be more comprehensive. The nvBench dataset, which includes scalar-valued and table-valued output for the SQL queries available, will be evaluated as a possible solution to this problem.

It demonstrates that Google's voice recognition service may be used to transform speech signals to text with the use of an Automatic Speech Recognition engine. When compared to other models across the literature study, the Seq2Seq model with attention mechanism displays the highest accuracy and has the best performance (Luo et al., 2022). Predicting the Select column, Aggregate Function, Aggregate column, Table name, where condition, Order by column, Group column, Limit value, and other SQL query components, as well as the bar, scatterplot, line graph, and pie chart types, is where the attention mechanism shines (Luo et al., 2022). Following this, we will discuss the study's methodology.

3 Research Methodology

The Knowledge Discovery Database or KDD is performed with some key alteration in the methodology. Figure 1. Illustrates how this research process flows in the KDD methodology. Data Collection, Data Pre-processing, Data Transformation, Data Mining, Evaluation, and Knowledge are the components that make up this methodology. Following is a detailed explanation of this process.

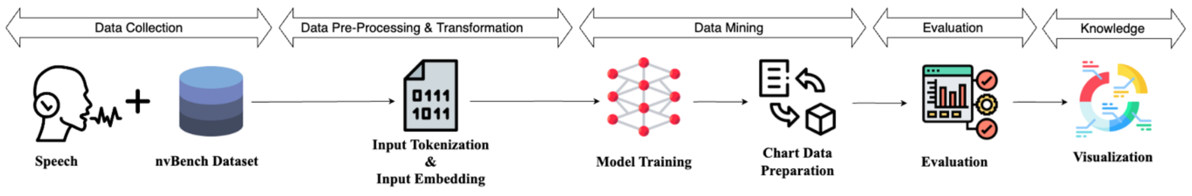

Fig. 1. Process Flow of Methodology

3.1 Data Collection

Data collection is just the process of acquiring the training data necessary to train our deep learning models. There are currently no speech datasets available for the subject of natural language. In accordance with the suggested approach, voice recognition will be used to turn spoken words into text. Therefore, only text-based data will be used to train the model. WikiSQL's dataset only includes SQL queries that produce scalar-valued results; in contrast, nvBench[1] dataset produces table-valued results from its SQL queries. Therefore, for the purposes of this study, will be utilizing the nvBench[1] dataset. NvBench is specifically created to support the cross-domain natural language question to visualization tasks. It has 105 domains, approximately seven different types of visualizations, and 25000 different (NL, VIS) pair combinations. This repository consists of a single JSON file. In addition, example databases as well as schema files that are capable of handling these JSON file data can be found. Separate training, testing and validation dataset was creating using the same JSON dataset. This csv dataset contains natural language questions, database name, columns, values, type of chart, resulting SQL query, and a special column called as 'source' which contains the original input combined of natural language question, Vega-Lite query, table name, column name and values into one sentence. source data with enclosed into *NL*, *C*, and *D* tag. *NL* contains natural language question along with the visualization user wants to draw, *C* contains the Vega-Lite Query (Luo et al., 2022) which is the replaced by the SQL sketch to support this research and *D* contains Columns *COL* and Values *VAL* from the specific database which supports the mapping of columns and values with the natural language question *NL*. The processing of these data will take place in the following section, which is "Data Pre-processing."

3.2 Data Pre-processing & Transformation

Data Pre-processing is a procedure involving the processing of data prior to the subsequent step or method. In this step, the 'source' columns present in the dataset is converted into the vocabulary by adding the initialization tokens also known as "start of sentence" <sos> and "end of sentence" <eos> which is very essential to train the natural language model.

Data transformation is the process of changing data into a form or a shape that may be used as an input to the following technique or procedure. In this research context, Input tokenization and Input Embedding is performed on the input source (Luo, Tang and Li, 2021).

3.2.1 Input Tokenization

In this step, tokenization process is performed on the input source data as already explained in 3.1 which contains natural language question along with type of chart, SQL Query sketch, and database table, columns and values information are categorized separately with the tokens such as *NL*, *C* and *D* (Luo et al., 2022). Natural language question which contains the details of the type of chart user wants to visualize is enclosed in a start token <NL> and end token </NL>, SQL Query is enclosed in start token <C> and end token </C> and Database information of table is enclosed in a start token <D> and end token </D>, and inside of *D* token columns and values are enclosed with start token <COL> and end token </COL>; and start token <VAL> and end token </VAL> respectively.

3.2.2 Input Embedding

Input Embedding is the step which translates the input source into the machine understandable vectors. As inspired by (Luo, Tang and Li, 2021), three types of embedding performed are token embedding, type embedding, and position embedding. Input tokens are converted into input embedding which results in vectors. Type embedding is also performed on the input

[1] https://sites.google.com/view/nvbench/

tokens which helps to identify the type of tokens such as nl, template, table, col, and value. Finally, position embedding is also performed on the input tokens which helps in identifying the position of each token in the sequence of the input source.

3.3 Data Mining

In this stage, Sequence-to-Sequence (Seq2Seq) model is employed to predict the SQL query. Seq2Seq is an advanced Recurrent Neural Network (RNN) having encoders and decoders as an additional component. In this research, encoder and decoder both are stacked with the number of self-attention blocks (Vaswani et al., 2022) meaning that encoders consist of multiple encoders and same for the decoders as well containing multiple decoders.

3.3.1 Encoder: Feeding Input Embedding

Encoder takes a sequence of words or letters in the form of input vectors and pass it to the decoder to predict the output sequence of the input. Each input embeddings such as token embedding, type embedding and position embedding formed in the data transformation step is passed to encoders containing <sos> identifier which helps encoder to identify the start of the input embeddings (Petrovski et al., 2022).

3.3.2 Decoder: Generation of Output Sequence

Decoder takes an input from the encoder and predicts the next sequence of words or letters. The input from the encoders containing <sos> identifier and the embeddings are processed to predict the next target vector. This process is followed in recurrent manner until the <eos> token is found which is also called as end of sentence (Petrovski et al., 2022). Attention Forcing mechanism (Luo, Tang and Li, 2021) and SQL-aware translation helps to identify the SQL components such as select column, aggregate function, aggregate column, table name, where condition, group by column, sorting type and limit index and chart type to form a SQL query.

3.3.3 Chart Data Preparation for Visualization

After generating the valid SQL Query, it is then executed on the SQLite3 database to check the ground truth result and to get the structured output in the form or rows and columns. Output from the database is stored in the pandas dataframe. Both the dataframe and the chart type predicted from the natural language question by the decoders are passed to the respective function i.e., draw_bargraph(*dataframe*), draw_linegraph(*dataframe*) etc. These defined functions contain a built-in method of matplotlib package of python for creating visualizations.

3.4 Evaluation

In this stage, Evaluation of the results are measured using evaluation techniques for the problem. In this research, Quantitative Evaluation and Execution Accuracy are the two measures used to assess the correctness of SQL query.

3.4.1 Quantitative Evaluation

Quantitative Evaluation focuses on the accuracy as its measure of evaluation metrics (Luo, Tang and Li, 2021). This measures how much the SQL query matches the ground truth SQL query. Accuracy can be defined as:

$$Accuracy = P/Q$$

Where P indicates the number of correct SQL query, and Q is the number of testing instances

3.4.2 Execution Accuracy

Execution Form Accuracy helps to evaluate the SQL query in terms of syntactically (Zhong, Xiong, and Socher, 2017). If the SQL query is not syntactically correct it can pose error during its execution on the SQLite3 database. This evaluation is done by matching the output SQL query with the ground truth SQL query.

3.5 Knowledge

Knowledge is gathered by drawing the visualization user has asked in the natural language question (J and Wijk, 2018). In this step, a dataset from SQL is created for the chart display as described in section 3.4.2. Pandas dataframe is the structured data in the form of rows and columns similar like a spreadsheet. Using the user-requested chart as a guide, respective chart method is called to draw the visualizations i.e., draw_ bargraph (*dataframe*), draw_linegraph(*dataframe*) etc. by passing the dataframe as an input parameter. These functions are formed on the top of the matplotlib python package for visualizations.

4 Design Specification

Design specification explains the overall architecture of the Speech to visualization. Fig. 2. shows the architecture.

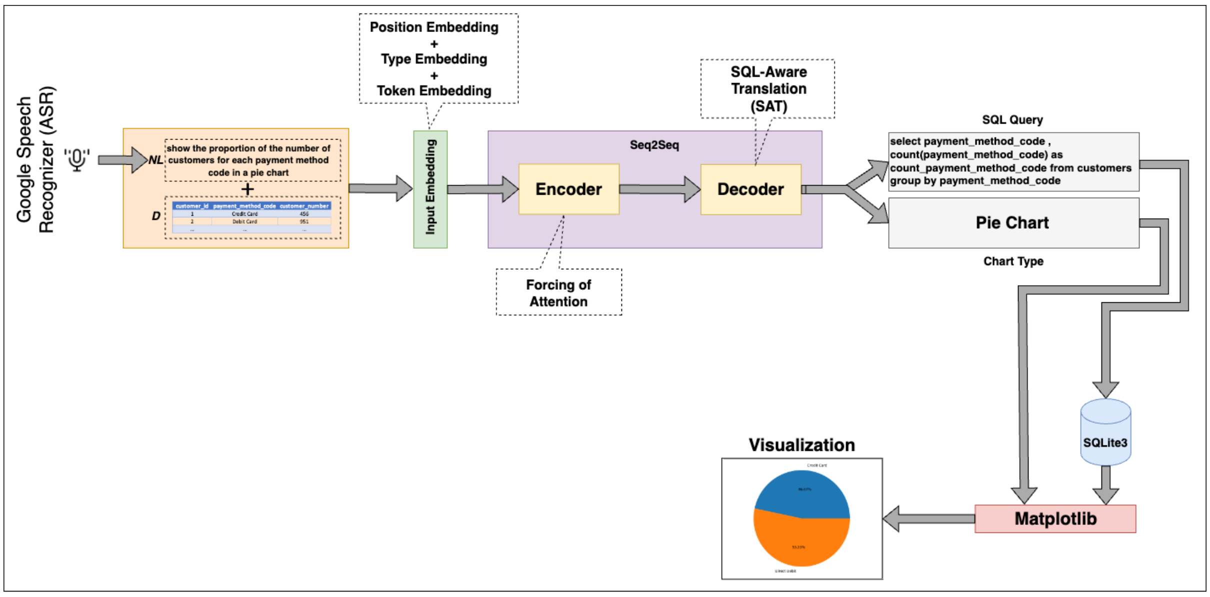
Google Speech Recognizer (ASR)
NL
show the proportion of the number of customers for each payment method code in a pie chart
+
D
customer_id payment_method_code customer_number
1 Credit Card 456
2 Debit Card 951
Position Embedding + Type Embedding + Token Embedding
Input Embedding
Seq2Seq
Encoder
Decoder
SQL-Aware Translation (SAT)
Forcing of Attention
SQL Query
select payment_method_code , count(payment_method_code) as count_payment_method_code from customers group by payment_method_code
Pie Chart
Chart Type
SQLite3
Matplotlib
Visualization

Fig. 2. Speech to Visualization Architecture

4.1 Automatic Speech Recognition (ASR)

Our hypothesis was that this research would benefit impaired (handless) users and layperson who might gain insights from data visualizations simply by speaking to a machine. The first step to accomplishing this is to build Automatic Speech Recognition (ASR). Using SpeechRecognition, you may leverage Google's voice recognition API (Chiu et al., 2022), which offers an ASR engine for your application code to implement. There is no cost or obligation for using Google's voice recognition API, and it's easy to get started. This helps to convert the speech to text which is the natural language question containing the information of the visualization chart.

4.2 Input tokenization & Embedding

With the help of input tokenization, the natural language question and database information such as table name, column name and values are combined into one input sentence by adding start and ending tokens for each entity (<NL></NL><D><COL></COL><VAL></VAL></D>). And then the sentence has the <sos> and <eos> to identify the start and end of the input sentence. Sentence formed after the input tokenization is converted into three different input embedding techniques such as

4.2.1 Token Embedding

Token embedding is generated by converting each input tokens into vectors.

4.2.2 Type Embedding

Each input tokens are converted into type tokens by adding different types of tokens to distinguish between natural language question, SQL query temple, and data tokens.

4.2.3 Position Embedding

Position embedding maintains the position information of each input tokens.

These three combined embeddings are passed as the single input to the encoder of the Seq2Seq model.

4.3 Sequence-to-Sequence Model

Sequence-to-Sequence learning is a model which is also called as Seq2Seq translates the one language context of fixed-length to another language context of fixed-length. A seq2seq model is made up of two neural networks: an encoder and a decoder. The encoder is responsible for understanding the input sequence and creating a hidden state representation. The decoder is responsible for producing a target output from a hidden representation. In this research, encoder and decoder both are employed with multiple self-attention layers meaning encoder contains many encoders and decoder contains many decoders. This architecture makes the Seq2Seq a transformer-based model (Luo, Tang and Li, 2021). The reason using the architecture is that transformer-based encoder and decoder does not recurrently pass the input token at each time unlike RNN which is the main benefit of using in this research context of converting natural language to visualization.

4.3.1 Forcing Attention

One of Transformer's most important new features is attention (Vaswani et al., 2022), which lets the model pick out the most important parts of the input sequence to focus on. Below Fig. 3. shows how the forcing of Attention impacts the model by forming the best correlation of the tokens using one of the sample tokens from the training data.

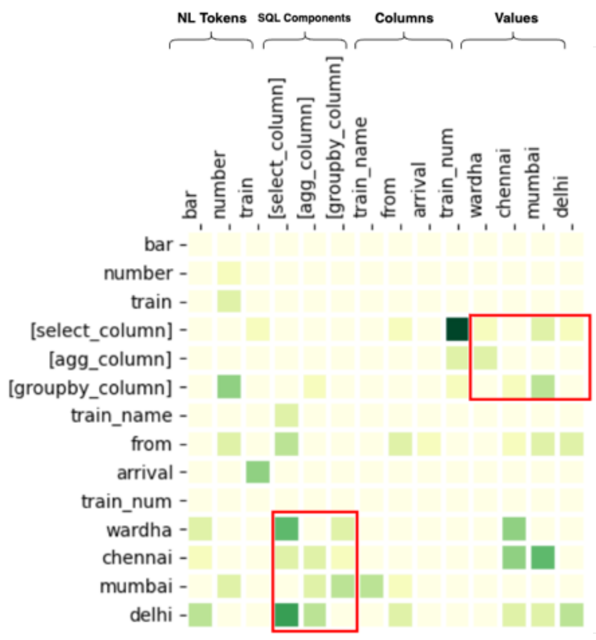

Fig. 3(a). Attention without forcing

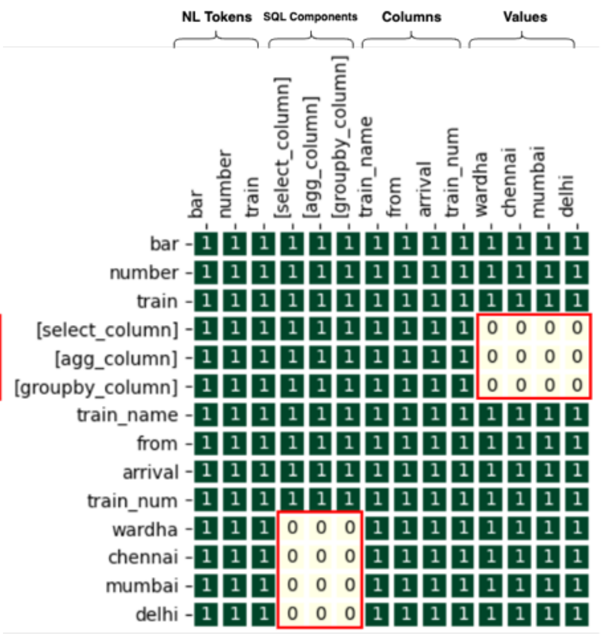

Fig. 3(b). Attention with matrix

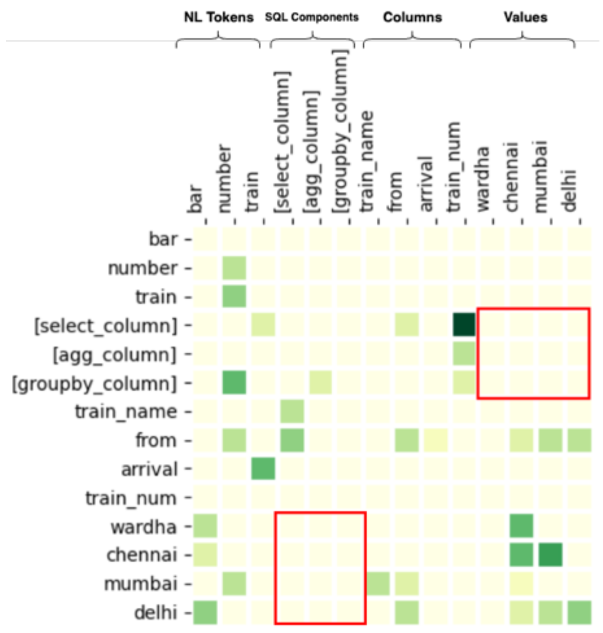

Fig. 3(c). Forcing Attention with matrix

Fig. 3(a) shows the attention matrix without forcing attention meaning and illustrates that there is a correlation of the [select_column], [agg_column] and [groupby_Column] with the cell values (highlighted in red) which is not true because the cell values can only be associated with the [where_condition] along with the where condition operator of SQL query.

Fig. 3(b) shows the forcing of Attention matrix which explicitly specifies the Boolean value '1' and '0' stating correlation and no correlation respectively. So, in this research context [select_column], [agg_column] and [groupby_Column] values need to be identified by finding correlation between columns token and not the values so forcing the '0' in the cell values which is highlighted in the red. This states that attention unit is informed to attend only important tokens.

Fig. 3(c) shows how forcing of attention in the attention matrix impacts the vector making the model perform more better. Red highlighted rectangle illustrates that how the instructions of the forcing attention matrix inform that which correlation should not be attended.

4.3.2 SQL-Aware Translation (SAT)

During the decoding stage, we cannot ensure enough that model always make a true prediction and it can predict false because of any reasons. This reason can be due to natural language query's intrinsic vagueness and other can be due to poor prediction of the encoding tokens or SQL components. For an instance, decoder will predict cell values as the token of the encoding tokens [*select_column*] instead of columns tokens. To overcome these issues, a novel approach called SQL-Aware translation (SAT) method is incorporated in decoder. SAT is responsible correct the incorrect output tokens until it finds the correct token. SAT aims to correctly identify the best predicted tokens and place them in the correct place holder of the SQL query sketch (Wang et al., 2018). To predict the best token, SAT holds the top three possibility tokens out of the output tokens generated by the decoder and then based on the highest probability value, selects the top token for all SQL components and the chart type. The SQL query sketch can be seen below:

Visualize [*chart_type*]
SELECT [*select_column*], [*agg_function*] ([*agg_column*])
FROM [*table_name*]
WHERE [*where_condition*]
GROUP BY [*groupby_column*]
ORDER BY [*orderby_column*] [*orderby_type*]
LIMIT [*limit_value*]

[*chart_type*] – This part is the type of chart such as pie chart, bar chart, scatter plot and line graph identified by the model using SAT.

[*select_column*] – This part is the SELECT column identified by the model is the highest probability of all the columns of the table.

[*agg_function*]- Aggregate function is the function which defines the aggregation of the values present in the [*agg_column*] column. Basis on the natural language question, model identifies the aggregate function out of the multiple choices such as SUM, MAX, MIN, AVG and COUNT.

[*agg_column*]- This part is the column on which the aggregation is performed using the [*agg_function*].
[*table_name*]- This part is the table name which is already identified before the prediction as user has to specify the database name and table name on which they have to draw the visualization.

[*where_condition*] – This part is the filtering of the data which consists of three different components such as column name (can be [*select_column*] or [*agg_column*]), operator sign (<, >, =, !=) and the value (can be any value containing in the [*select_column*] or [*agg_column*]).

[*groupby_column*]- This part is the clause of the SQL query which identify the grouping of the columns (can be [*select_column*] or/both [*agg_column*]).

[*orderby_column*] – This part is the clause of SQL query which identify the ordering of the data for the column (can be [*select_column*] or/both [*agg_column*]).

[*orderby_type*]- This part is used on the top of [*orderby_column*] to determine the type of ordering such asc (ascending) or desc (descending).

[*limit_value*]- This part defines the number of rows which needs to be displayed in the data.

4.4 Visualization

Once the valid SQL query is formed, it is then executed on the SQLite3 database to get the output in the row and columns manner. This output is then converted into the pandas dataframe which transforms the SQL output into the labelled axis with the index number. Using this dataframe, passing the individual column data to chart attributes according to its axis to plot the graph or chart. To achieve this method, matplotlib, a pythons visualization package is used to draw the user asked visualization by passing the relevant data to its chart plotting function.

5 Implementation

5.1 Development Environment

Development of this research was done using the Python programming language (version 3.6) which provides the best support in terms of the libraries and packages for performing various machine learning tasks with easy and fast. Due to high computational requirement for the training of the model, Google colab is used to train the model and save the best model. Whereas the prediction tasks are performed on the Visual Studio Code (VS Code) because Google Colab is not convenient to run the entire project (containing .py files) because it is more convenient using notebook files (.ipynb). During training all the train, test and validation dataset is uploaded on the Google drive which is further mounted in the Google colab to train the Seq2Seq model. After the training and testing, the final and best model is saved in the Google colab's local repository. This model file is downloaded at the local machine and saved in the project directory of the adopted code[2] of the researchers (Luo, Tang and Li, 2021) and prediction is performed.

5.2 Seq2Seq model

Implementation of the Seq2Seq model is done using the PyTorch, which is the machine learning framework for the task such as natural language processing and computer vision with incorporating the transformer-based encoder and decoder. Multiple encoder and decoder blocks are stacked with the self-attention blocks making it a multi-head attention layers by setting 8 as a number of heads. Dimension of the input embedding such as token embedding, position embedding, and type embeddings is set to maximum of 256. Input length token is specified with the limit of 512 and if the length of the input token is greater than 512 that it is truncated. Adam optimizer is used with the learning rate of 0.005 (Luo, Tang and Li, 2021). The dropout rate is set to 0.1 for both the encoder and the decoder. The model is trained for 10 epochs along with the batch size of 64 on the training data, test data and validation with and

[2] https://github.com/Thanksyy/ncNet

the best model with respect to best accuracy in the epochs is saved in the project directory folder. Saved model is then tested using the testing dataset and performance evaluation is done.

6 Evaluation

6.1 Evaluation of Training vs Validation Loss & Accuracies

Training of the Seq2Seq model is done with the training dataset of around 20,500 pairs of natural language question and the SQL query along with the chart information in both pairs. And the validation dataset consists around 1,150 pairs which was created on basis of the nvBench dataset. Model was run on 10 epochs by tuning its hypermeters. Fig. 4. below shows the loss of the training and validation in each of 10 epochs. And it illustrates that after each epoch, the loss during the training is gradually decreasing making the model to perform better. The overall accuracy of the model with inclusive of different difficulty level such as easy, medium, hard and extra hard SQL query it gives the accuracy of 76%.

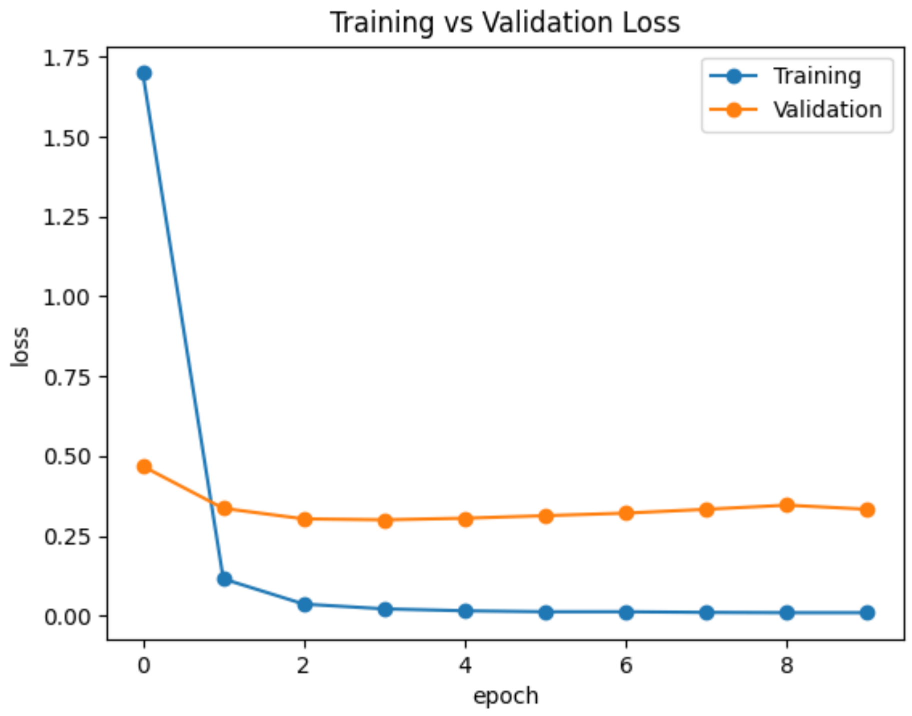

Fig. 4. Training vs Validation Loss

6.2 Quantitative Evaluation

Quantitative Evaluation compares the model's output with the ground truth in terms of the training data and its results by considering the measure as accuracy. It does this by looking at how accurate the model's output is. In this evaluation, as already explained, the data is spread across different levels of difficulty, such as easy, medium, hard, and extra hard and different types of charts such as pie chart, bar graph, line graph and scatter plot Fig. 5. shows the percentage accuracy for the same. In which highest accuracy of the model was achieved for the extra-hard and scatter plot chart with 85.2%. Whereas 21.7% for the medium difficulty SQL query and scatter plot.

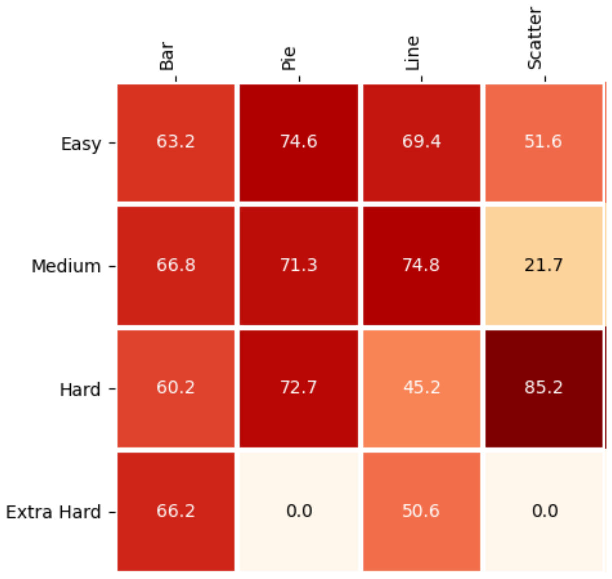

Fig. 5. Accuracies of SQL query difficulties vs type of chart

6.3 Execution Accuracy

Execution Accuracy is a measure to check if the SQL query is syntactically correct and valid for the execution on the SQLite3 database (Zhong, Xiong, and Socher, 2017). If the SQL query is not syntactically correct, database engine throws the exception by which it helps to understand the problem with the SQL query statement. If the SQL query is correct and valid it produces the output with result set which can be used to generate the chart.

6.4 Discussions

In this section, evaluation errors and limitations of the research is addressed in detail.

6.4.1 Evaluation Errors

There are various failed cases found during the research which affects the overall accuracy of the model. Some of the failed cases are mentioned below:

6.4.1.1 Incorrect Prediction of Columns

Prediction of incorrect columns is the one of the most common problem in this research. This incorrect prediction problem is because of the desired column spoken by the user in the natural language is not able to relate to the column of the table. For example, if the user asks or mentions the word “Invoice ID” in the natural language question but the actual column name in the table header is “InvID”, this leads to incorrect prediction of the column name during the prediction. This issue can be tackled by training the model with the table schema information or training it more using some pre-trained language-specific models such as BERT (Guo and Gao, 2022).

6.4.1.2 Incorrect Prediction of Where Conditions

Incorrect prediction of where conditions also lead to evaluation errors. This error occurs when the natural language is mis-interpreted by the model to correctly identify the logical operations. For an instance, if user asks, “draw bar chart showing the top 5 highest price of product greater than 100…” and suppose there are columns such as “net_price”, “gross_price” or another column which is highly correlated. And in this natural language question user is expecting the “net_price” but model tends to misinterpret and predicts where condition as “gross_price > 100”. The same problem can be resolved by training the model with more recently developed language models.

6.4.1.3 Misinterpretation of Spoken Words

There are some chances that Automation Speech Recognition (ASR) engine of Google speech recognition can misinterpret pronunciation of the user because of the accent which can lead to undesired word prediction by the ASR engine leading to wrong interpretation of the natural language query. To overcome this challenge, there is a provision to specify the accent such as Australian, American, British and many more during the implementation of the Google Speech Recognizer in the python code.

6.4.2 Limitations

6.4.2.1 Lack of Training Data

There is very limited training dataset available to train the deep learning model in the context of natural language to visualization task. Apart from nvBench dataset, there are some of the publicly available datasets such as WikiSQL, Spyder (Guo and Gao, 2020), (Kim and Lee, 2021), (Hui et al., 2021), (Wang et al., 2018) supports only natural language to SQL query but to use these data to support this research needs ample of time to convert these data to visualization information rich data.

6.4.2.2 Inability of Generating Complex Queries

This research is inability of generating the complex query containing the joins between the number of database tables, nested query, and sub query because of lack of such data to train the deep learning models. To train the deep learning model it needs enough amount of such data.

7 Conclusion & Future Work

This research aims to build the speech-driven moreover touchless-based system for the disabled and layperson in the field of data visualization analysis who can draw the visualization to find insights and trends in the data. This has been implemented using the Google speech recognizer which primarily converts speech to text, a Seq2Seq model with attention forcing making it the transformer based deep learning model and novel SQL-aware translation module which is responsible to generate the valid SQL query which can be used to execute on the SQLite3 database to get the result set to draw the visualizations using the matplotlib python’s library for creating the visualization. The deep learning model is trained using the large-scale nvBench dataset containing the data across 100 domains. By the state-of-the-art SQL-aware translation module and tuning the hyperparameters of the model the overall accuracy of the model achieved is 77%.

In the future, an attempt would be made to create more dataset which would support this research to train the deep learning model as well as to support the complex SQL queries mainly consisting of joins, nested or subquery which will make it more powerful system in the natural language to visualization tasks. And the need of user-friendly interface to see the information of the required inputs and outputs of the system in more easy and organised way.

Acknowledgments

I would like to convey my sincere appreciation to my supervisor, Jorge Basilio for his assistance and direction in finishing this research study. It would have not been possible to complete this research without his exceptional support. Every weekly calls with him always gave me the correct path and reminded me about positive or negative possibilities or cases in my approach. The kindness and expertise of him have improved this research in numerous ways and prevented me from errors.

References

Anguera, X., Luque, J. and Gracia, C., 2014. Audio-to-text alignment for speech recognition with very limited resources. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Abbas, S., Khan, M.U., Lee, S.U.J., Abbas, A. and Bashir, A.K., 2022. A Review of NLIDB with Deep Learning: Findings, Challenges and Open Issues. *IEEE Access*.

Basystiuk, O., Shakhovska, N., Bilynska, V., Syvokon, O., Shamuratov, O. and Kuchkovskiy, V., 2021. The Developing of the System for Automatic Audio to Text Conversion. In *IT&AS* (pp. 1-8).

Chang, S., Liu, P., Tang, Y., Huang, J., He, X. and Zhou, B., 2020, April. Zero-shot text-to-SQL learning with auxiliary task. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, No. 05, pp. 7488-7495).

Chiu, C.C., Sainath, T.N., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., Kannan, A., Weiss, R.J., Rao, K., Gonina, E. and Jaitly, N., 2018, April. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4774-4778). IEEE.

Guo, T. and Gao, H., 2019. Content enhanced bert-based text-to-sql generation. *arXiv preprint arXiv:1910.07179*.

He, Y., Bai, D. and Jiang, W., Text-to-SQL Translation with Various Neural Networks CS224N Project Final Report.

Hiregoudar, S., Gonal, M. and Karibasappa, K.G., Speech to SQL Generator-A Voice Based Approach.

Hui, B., Geng, R., Wang, L., Qin, B., Li, B., Sun, J. and Li, Y., 2022. S $^ 2$ SQL: Injecting Syntax to Question-Schema Interaction Graph Encoder for Text-to-SQL Parsers. *arXiv preprint arXiv:2203.06958*.

Hui, B., Shi, X., Geng, R., Li, B., Li, Y., Sun, J. and Zhu, X., 2021. Improving text-to-sql with schema dependency learning. *arXiv preprint arXiv:2103.04399*.

J.J.,Van Wijk, 2005, October. The value of visualization. In *VIS 05. IEEE Visualization, 2005.* (pp. 79-86). IEEE.

Kate, A., Kamble, S., Bodkhe, A. and Joshi, M., 2018, March. Conversion of natural language query to SQL query. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)* (pp. 488-491). IEEE.

Kim, D. and Lee, S., 2021. Self-supervised Text-to-SQL Learning with Header Alignment Training. *arXiv preprint arXiv:2103.06402*.

Luo, Y., Tang, N., Li, G., Tang, J., Chai, C. and Qin, X., 2021. Natural Language to visualization by neural machine translation. *IEEE Transactions on Visualization and Computer Graphics*, *28*(1), pp.217-226.

Wang, P., Shi, T. and Reddy, C.K., 2020, April. Text-to-SQL generation for question answering on electronic medical records. In *Proceedings of The Web Conference 2020* (pp. 350-361).

Petrovski, B., Aguado, I., Hossmann, A., Baeriswyl, M. and Musat, C., 2018. Embedding individual table columns for resilient SQL chatbots. *arXiv preprint arXiv:1811.00633*.

Qi, J., Tang, J., He, Z., Wan, X., Zhou, C., Wang, X., Zhang, Q. and Lin, Z., 2022. RASAT: Integrating Relational Structures into Pretrained Seq2Seq Model for Text-to-SQL. *arXiv preprint arXiv:2205.06983*.

Song, Y., Wong, R.C.W., Zhao, X. and Jiang, D., 2022. Speech-to-SQL: Towards Speech-driven SQL Query Generation From Natural Language Question. *arXiv preprint arXiv:2201.01209*.

Sen, J., Lei, C., Quamar, A., Özcan, F., Efthymiou, V., Dalmia, A., Stager, G., Mittal, A., Saha, D. and Sankaranarayanan, K., 2020. Athena++ natural language querying for complex nested sql queries. *Proceedings of the VLDB Endowment*, *13*(12), pp.2747-2759

Uma, M., Sneha, V., Sneha, G., Bhuvana, J. and Bharathi, B., 2019, February. Formation of SQL from natural language query using NLP. In *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)* (pp. 1-5). IEEE.

Utama, P., Weir, N., Basik, F., Binnig, C., Cetintemel, U., Hättasch, B., Ilkhechi, A., Ramaswamy, S. and Usta, A., 2018. An end-to-end neural natural language interface for databases. *arXiv preprint arXiv:1804.00401*.

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*.

Wang, C., Tatwawadi, K., Brockschmidt, M., Huang, P.S., Mao, Y., Polozov, O. and Singh, R., 2018. Robust text-to-sql generation with execution-guided decoding. *arXiv preprint arXiv:1807.03100*.

Xu, R. and Singh, A., Neural Semantic Parsing Natural Language into SQL.

Zheng, Y., Wang, H., Dong, B., Wang, X. and Li, C., 2022. HIE-SQL: History Information Enhanced Network for Context-Dependent Text-to-SQL Semantic Parsing. *arXiv preprint arXiv:2203.07376*.

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